Dictionary/3000

HP 3000 MPE/iX Computer Systems

Edition 4

Manufacturing Part Number: 32244-90001
E1287

U.S.A. December 1987

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for direct, indirect, special, incidental or consequential damages in connection with the furnishing or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19 (c) (1,2).

Acknowledgments

UNIX is a registered trademark of The Open Group.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Contents

6. The Dictionary DICTPDE Utility

Contents

Contents

Contents

Preface

This manual describes the Dictionary/3000 software system that operates on HP 3000 computers. It is the reference document for all persons involved in creating and maintaining a dictionary database and an IMAGE/3000 data base. It assumes a working knowledge of the HP 3000 computer system, including the IMAGE/3000 subsystem.

This edition of the manual has been expanded to include the new Dictionary extract utilities for COBOL II/3000 and PASCAL/3000 data declarations, and for the new VPLUS forms file definition utility. In order to use these utilities, the manual assumes a working knowledge of COBOL II/3000, PASCAL/3000, and VPLUS/3000. Also included in this manual is information for the Inform Security feature.

In addition, use of the Dictionary audit, load, and unload utilities can improve IMAGE/3000 response and allows faster application run time and increased throughput. Each utility provides benefits which are not included in IMAGE. For example, the primary benefit of the load and unload utilities, DICTDBL and DICTDBU, is that they allow selective loading and unloading of the data sets within a database. These can be used as tool to maintain the primary path sequence of the data in detail data sets. (DICTDBU and DICTDBL do not, however, provide a dynamic restructuring capability.) For full database loads and unloads, the DBLOAD and DBUNLOAD facilities provided with IMAGE are more efficient.

DICTDBA, the audit utility, gives you a quick way to find out the synonym chain count and average chain length. By maintaining chain counts as low as possible, database performance is improved.

In addition to this manual, you may need to consult the following manuals and self-paced courses:

Manual and Manual Part Number

MPE Command Reference Manual	30000-90009
MPE Error Messages and Recovery Manual	30000-90102
Using Files	30000-90015
IMAGE/3000 Reference Manual	32215-90003
VPLUS/3000 Reference Manual	32209-90001
KSAM/3000 Reference Manual	30000=90079
30000-90079 Transact/3000 Reference Manual	32247-90001
HP Inform/3000 User's Guide	32246-90001
PASCAL/3000 Reference Manual	32106-90001
COBOL II/3000 Reference Manual	32233-90001

Self-Paced Course and Course Product Number

Programming in Transact/3000	22842A
Using Dictionary/3000	22843B

WARNING **The Data Dictionary integrity can be destroyed by using a utility that is not listed below. Therefore, only the following utilities will be supported by Hewlett-Packard when used against the Data Dictionary:**

DICTINIT
DICTDBM
DICTDBM,UTIL
DICTDBD
DICTDBA
DICTDBC
DICTVPD
DICTCDE
DICTPDE
DBSTORE/DBRESTOR
***STORE/RESTORE**
***SYSDUMP**

Hewlett-Packard will investigate and attempt to resolve problems resulting from the use of utilities that do not appear on the above list. This service is not provided under HP's standard support agreements, but is available on a time and material basis.

HEWLETT-PACKARD IS NOT RESPONSIBLE FOR ANY LOSS OR DAMAGE RESULTING FROM CUSTOMER'S USE OF THE DATA DICTIONARY WITH UTILITIES NOT LISTED ABOVE.

---

*Please note that STORE/RESTORE and SYSDUMP will be supported only when used on the Data Dictionary in its entirety (e.g., the root file and all the data sets).

1 Introducing Dictionary/3000

Overview

Dictionary/3000 is a comprehensive set of programs that you can use to:

- Create any number of data dictionaries
- Create and maintain entries in the dictionary
- Create and maintain an IMAGE/3000 database
- Create HP Inform/3000 groups
- Generate COBOL and PASCAL data definitions

What is a Data Dictionary?

A data dictionary is a directory of information about the definition, structure, and usage of data. It does not contain the data itself. The data dictionary contains the name of each data item (element), its definition (size and type), where and how it is used, and its relationship to other data.

The entries in Dictionary/3000's data dictionary define and describe an organization's structure, identify the data used by an organization, specify where the data is stored, identify what programs generate the data, and define and describe HP Inform/3000 groups to report the data. These entries can also be used to generate data definitions for COBOL and PASCAL programs.

Thus, Dictionary/3000's data dictionary is a central repository for information about an organization and its data processing environment.

How to Use Dictionary/3000

Dictionary/3000 includes an easy-to-use interactive program called DICTDBM for the creation and maintenance of entries in the Dictionary. The DICTDBM program provides a set of commands that allow you to create and maintain entries, and to establish and define the relationships between entries in the Dictionary. Through a combination of commands, you control the information in your Dictionary for the following:

- Data
- IMAGE databases
- MPE, MPER and KSAM files
- HP Inform/3000 groups

- Physical locations
- Organizational structures
- Application programs
- Security for databases, files, and for HP Inform/3000
- Relationships between the entries
- Formats used for COBOL programs

DICTDBM is discussed in detail in Sections II and III of this manual. The DICTDBM commands and subcommands are discussed in Section IV.

Dictionary/3000 also includes a comprehensive set of utilities. These utilities allow you to perform various functions for the Dictionary, an IMAGE database, a VPLUS forms file, and for PASCAL and COBOL programs. The Dictionary/3000 utilities can be used as follows:

- Create and initialize the Dictionary itself
- Create an IMAGE database from the entries in the Dictionary
- Transfer the definition of an existing IMAGE database into the Dictionary
- Audit a database for synonym, chain, and linkage usages
- Unload an IMAGE database to tape or disk
- Load data from tape or disc into IMAGE database files
- Perform maintenance on the Dictionary
- Generate PASCAL declarations from the definitions in the Dictionary
- Generate COBOL source code from the definitions in the Dictionary

Table 1-1 shows all the Dictionary/3000 utilities, their function, and where you can find the information about the utility in this manual.

Table 1-1. Dictionary/3000 Utilities

Utility	Function	Manual Reference
Dictionary Initialization (DICTINIT)	Creates and initializes the Dictionary.	Appendix C
Database Creation (DICTDBC)	Creates an IMAGE database from the entries in the Dictionary.	Chapter 5
Database Definition (DICTDBD)	Transfers the definition of an existing IMAGE database into the Dictionary.	Chapter 5
Database Audit (DICTDBA)	Audits an IMAGE database for synonym, chain and linkage usages.	Chapter 5
Database Unload (DICTDBU)	Unloads the data files of an IMAGE database. DICTDBU allows selective unloading of the data files.	Chapter 5

Table 1-1. Dictionary/3000 Utilities

Utility	Function	Manual Reference
Database Load (DICTDBL)	Loads data from tape or disk into IMAGE database files. DICTDBL allows selective loading of data set files.	Chapter 5
Data Dictionary Cleanup (DICTDBM, UTIL)	Performs periodic maintenance on the Dictionary. DICTDBM, UTIL removes superfluous information that may exist as a result of extensive DICTDBM use.	Chapter 5
VPLUS Forms File Definition (DICTVPD)	Transfers the definition of an existing VPLUS forms file into the Dictionary.	Chapter 5
COBOL Definition Extract (DICTCDE)	Extracts data definitions from the Dictionary and generates corresponding COBOL source code for the definitions.	Chapter 7
PASCAL Definition Extract (DICTPDE)	Extracts data definitions from the Dictionary and generates corresponding PASCAL declarations for the definitions.	Chapter 6

How to Use this Manual

This manual is divided into the following sections:

Chapter 1	Introduces Dictionary/3000.
Chapter 2	Describes the concepts behind and the use of DICTDBM — a program designed to create and maintain entries in the Dictionary.
Chapter 3	Provides an explanation of the commands used to create and maintain entries for various structures.
Chapter 4	Describes the DICTDBM commands used to create and maintain entries in the Dictionary.
Chapter 5	Gives an overview of each utility, and describes how each is used.
Chapter 6	Describes the PASCAL Definition Extract Utility (DICTPDE).
Chapter 7	Describes the COBOL Definition Extract Utility (DICTCDE).
Appendix A	DICTDBM Error Messages.
Appendix B	Dictionary/3000 Utility Error Messages.
Appendix C	Initialization Procedures and Errors.
Appendix D	Describes how HP Inform/3000 links files to generate reports.
Appendix E	Glossary of Terms.
Appendix F	DICTDBM Quick Reference.

2 How to Run Dictionary/3000

Overview

Dictionary/3000 provides a Database Maintenance program (DICTDBM) to create and maintain entries in the Dictionary. This section describes how to run the program and use its HELP facility, describes the syntax for the commands in the program, defines the options that can be used with the commands, and lists all the commands and subcommands available in the DICTDBM program.

Running the DICTDBM Program

To use the Dictionary, you must first have it loaded and initialized onto your system by your system manager (see Appendix C).

After logging on, you are ready to run the Dictionary program. DICTDBM assumes the Dictionary exists in the PUB group of your log-on account. If the Dictionary you wish to use exists in a different MPE group and/or account, you must first identify these by issuing a file equation before running DICTDBM. The file equation should be as follows:

```
FILE DICT.PUB=DICT.group.account
```

Next, run the Dictionary program:

```
RUN DICTDBM.PUB.SYS
```

Then press the RETURN key.

The Dictionary prompts you to enter a password as follows:

```
PASSWORD FOR DICT.PUB>
```

The password will not be displayed as you enter it. You must enter the password exactly as it appears in the system - that is, all caps, all lower case, initial cap, or some combination of the three. The individual responsible for the data dictionary can give you the password information.

The Dictionary (DICTDBM) supports two types of field entry. One, form entry, has entry fields in enhanced video. The other, normal terminal display, uses a less-than sign (<) to delimit entry fields. Both types of field entry indicate the maximum number of characters you can enter. In this manual, all examples are shown in normal terminal display.

After the correct password is entered, the Dictionary lets you choose the type of field entry you wish to use with the following prompt:

```
FORMS ENTRY(Y/N)?>
```

A response of “Y” indicates that you want to use the enhanced video type of field entry.

Whenever a choice between two options is given within parentheses, the first choice listed is the DEFAULT choice, unless otherwise specified; thus, just pressing RETURN is the same as entering that choice.

After responding to the FORMS ENTRY prompt, DICTDBM issues a command prompt (>). You are now ready to use DICTDBM commands to create, manipulate, or maintain entries in the Dictionary. To terminate execution of DICTDBM, enter the EXIT command in response to a command prompt.

Entries are created in the Dictionary through a series of command strings in response to a command prompt (>). The action of the command string can be modified or enhanced by including one or more of the available command options.

Command Syntax

A DICTDBM command string has the following syntax:

Syntax

```
[option] COMMAND subcommand
```

option	A list of one or more command options which enhance or modify the action of the command. For example, REPEAT, SORT, etc. (See Options Used With Commands, described next, for an explanation of each option.)
COMMAND	Specifies the command action. For example, ADD, RELATE, etc.
subcommand	Specifies the target of the command's action. For example, ELEMENT, GROUP, etc.

For each command except the EXIT and HELP commands, you must enter both the command and the subcommand. Optionally, each command string can be preceded by one or more options. Only one command string is allowed for each command prompt.

The appropriate abbreviations may be used for the commands and subcommands.

Options Used with Commands

The following is a list of options which enhance or modify the action of the command. When used, they must appear first in the command syntax and can not be abbreviated.

PRINT	Directs the display to the line printer rather than to the user terminal.
TPRINT	Formats the terminal display in line printer format.
SORT	Sorts the display listing alphabetically before output.
REPEAT	Repeats a command until the termination character (]) or the RETURN key is entered in response to a prompt.

A single command string can have multiple options that are separated by commas.

When to Use the Options

The PRINT option is useful when you are displaying information. For example, use PRINT with the LIST, DISPLAY, or SHOW command.

The TPRINT option is useful before directing output to the line printer. The listing will be displayed on your terminal in line printer format. Line printer format includes a banner line which shows the date and time of the report and where page breaks occur including page numbering.

The SORT option is useful when you want an alphabetical listing from the LIST and DISPLAY commands. The REPORT command automatically displays the list in alphabetical order.

When used, the REPEAT option causes the prompts for the command and/or subcommand to be repeated until terminated by the RETURN key or the termination character (]). One

use for the REPEAT option is with the CREATE command when more than one data element is to be created. For instance, if you enter REPEAT CREATE ELEMENT, the prompt for ELEMENT is repeated, allowing you to create many data elements without re-entering the command. The ELEMENT prompt is repeated until you enter the terminating character.

Special Character Options

The following characters have special meaning when used in response to a DICTDBM prompt:

!	In response to a prompt, terminates the prompting cycle within the command. It is equivalent to pressing RETURN to all subsequent prompts.
]	In response to a prompt, if the REPEAT option is used with the command, prompts for one level are terminated and reissued for the next level up; if the REPEAT option is not used terminates the command.
]]	In response to a prompt, terminates the command.

Commands and Subcommands

The following is a list of the DICTDBM commands and subcommands, and their abbreviations, that can be used with the Dictionary. In Section IV, which describes the commands, each description includes a list of the subcommands that can be used with that command (see Section IV).

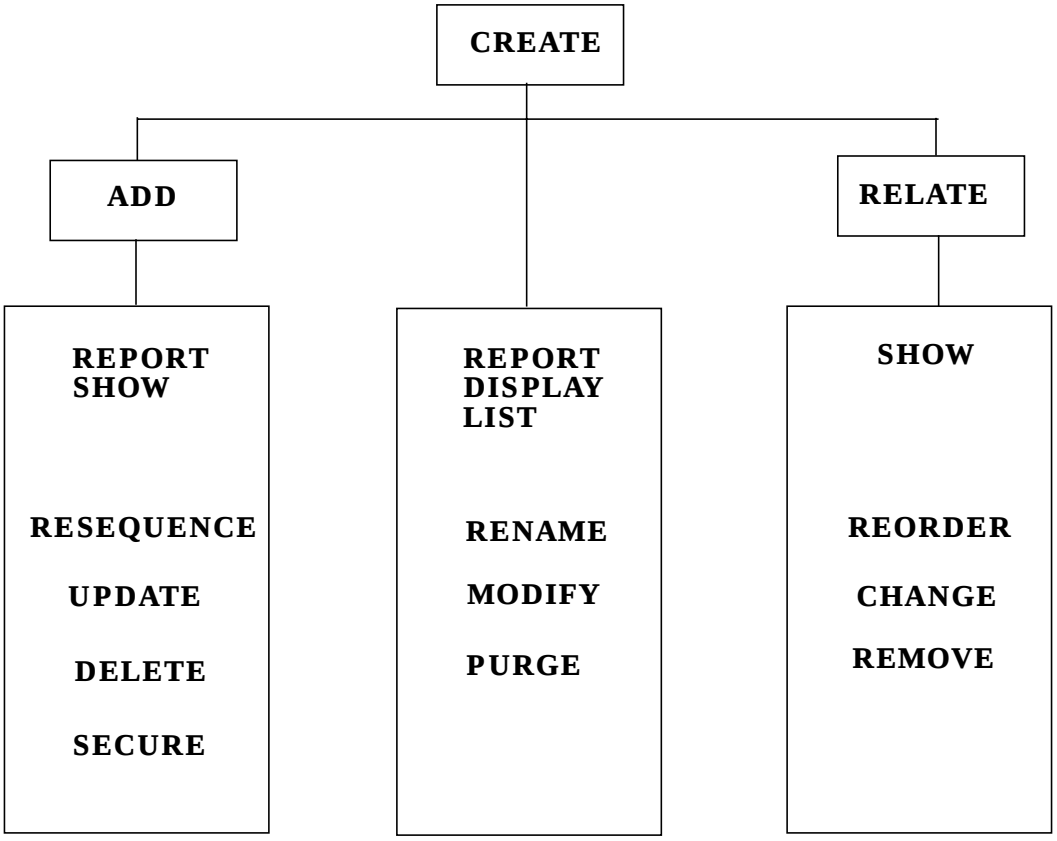
Command	Abbreviation	Subcommand	Abbreviation
ADD	A	CATAGORY	C
CHANGE	CHG	CLASS	CLS
CREATE	C	CLASS-FILE	CLF
DELETE	DEL	CLASS-GROUP	CLG
DISPLAY	D	ELEMENT	E
EXIT	E	FILE	F
HELP	H	FILE-LOC	FLC
LIST	L	GROUP	G
MODIFY	M	LOCATION	L
PURGE	P	PROCEDURE	P
RELATE	REL	PROCEDURE-LOC	PLC
REMOVE	REM	*	
RENAME	REN		
REORDER	REO		
REPORT	R		
RESEQUENCE	RES		
SECURE	SEC		
SHOW	S		
UPDATE	U		

* The subcommands for the HELP command are the command names.

Note that each abbreviation is either a single or a three letter character string.

Command Usage Chart

The following chart indicates the order in which the Dictionary commands can be used. The HELP command can be issued any time during a command sequence as long as the command prompt (>) appears.



The chart shows that entities must first be created in the dictionary through the CREATE command. Next, entities can be added to other entities through the ADD command, related to other entities through the RELATE command, or displayed or modified through the commands listed in the center box (REPORT, DISPLAY, LIST, RENAME, MODIFY, or PURGE). Following the ADD command, the REPORT, SHOW, RESEQUENCE, UPDATE, DELETE, or SECURE commands can be used. Following the RELATE command, the SHOW, REORDER, CHANGE, or REMOVE commands can be used. The EXIT command, not shown in the above chart, can be used at any time to exit the data dictionary program (DICTDBM). Looking at the three large boxes shown in the chart, note that each horizontal row of commands are similar. For example, UPDATE, MODIFY, and CHANGE are similar. The difference between such commands is the command it follows: ADD, CREATE, or RELATE.

The commands which follow the ADD command (REPORT, SHOW, RESEQUENCE, UPDATE, DELETE, and SECURE) can be used to display or modify entities which have been ADDed to other entities. For example, these commands can be used to display or modify an element which has been added to a file. Entities are always added to unlike entities - that is, an element is added to a file, but an element can not be added to another element. Hence, REPORT and SHOW can be used to display the association between unlike entities. Similarly, RESEQUENCE, UPDATE, DELETE, and SECURE are used to modify the association between unlike entities.

Similarly, the commands which follow the RELATE command (SHOW, REORDER, CHANGE, and REMOVE) can be used to display or modify entities which have been RELATEd to other entities. Entities are always related to like entities - for example, a file can be related to another file, but an element can not be related to a file. Hence, SHOW can be used to display the relationship between like entities; REORDER, CHANGE, and REMOVE are used to modify the relationship between like entities.

The commands which follow the CREATE command (REPORT, DISPLAY, LIST, RENAME, MODIFY, and PURGE) affect the entities themselves. They can be used to display or modify any entities which have been created, whether they have been added or related to other entities or not.

3 Using the DICTDBM Commands

Overview

The DICTDBM commands are used to create and maintain entries in the Dictionary. These entries define and describe data elements and structures. Each entry in the Dictionary is unique.

The following structures can be defined in the Dictionary:

- IMAGE databases and security
- Other file structures (MPE, MPER, KSAM, and VPLUS)
- Application programs
- Physical locations
- Organizational structures
- HP Inform/3000 groups

This section gives an overview of how to use the commands to create and maintain each structure.

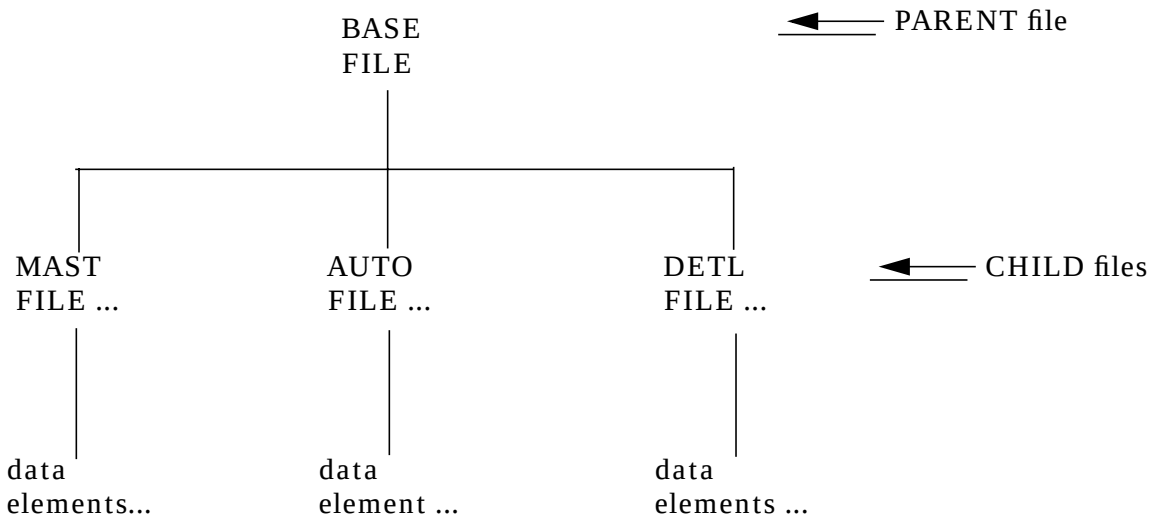
Using Commands to Define a Database

An IMAGE database can be defined in the Dictionary in two ways. For an existing IMAGE database, use the Database Definition utility program (DICTDBD) to transfer the database definitions from the database root file to the Dictionary. Refer to Section V of this manual for information on this utility.

The second way is to use the commands provided by the DICTDBM program. The database and the data sets are created in the Dictionary as files. The database is a BASE file; the manual master sets are MAST files; the automatic master sets are AUTO files; and the detail sets are DETL files. The data items in the database are created as elements in the Dictionary.

The definition for a database is completed by establishing the relationship between the BASE file and the data sets. This relationship is a hierarchical relationship and is defined using the RELATE command. The elements are then added to the appropriate data sets with the ADD command. (Refer to Section IV of this manual for a description of these commands.) The hierarchical structure for a database is shown in Figure 3-1

Figure 3-1. Summary of Hierarchical Structure for a Database



The BASE file is the top of the structure and is the PARENT file. The data sets are related to the BASE file and are CHILD files. In this structure, the PARENT file cannot have data elements directly associated with it. The following description outlines the steps involved when you use the DICTDBM commands to define a database. A complete description, including examples for all the commands is given in Section IV of this manual.

Creating Elements and Files for a Database

An element or a file must be defined in the Dictionary before it can be associated or related to another entry. The CREATE command is used to define an entry for a data element or a file.

Either the command string CREATE ELEMENT or CREATE FILE is used to create the entry in the Dictionary. The name for the element or the file must be unique. Use the REPEAT option before the command string to create definitions for more than one element or file without having to repeat the command. An entry for the BASE file must be created later to establish the hierarchical relationship between the data sets and the database.

If you are defining an IMAGE database, the IMAGE/3000 rules for defining a master data set and a detail data set apply. For example, if the data set is a detail set, you must indicate which data elements are the search items, the name of the master set each search item is related to, the data set's capacity and so forth.

After the entries are created, they may be displayed by the LIST or DISPLAY commands. If you want to delete the entry, use the PURGE command. If you want to change the entry, use the MODIFY command or the RENAME command.

The RELATE ELEMENT command string can be used to define CHILD elements (fields) within a PARENT data element. This allows access to parts of a larger data element. For example, in the case of an IMAGE database which contains a compound data item, CHILD elements can be used to provide access to the sub-items of the compound item. To do so, respond to the BYTE POSITION prompt by specifying the starting location of the sub-item within the compound item.

Relating Data Sets to a BASE File

The hierarchical relationship between the data sets and the BASE file is established with the RELATE command. The command string RELATE FILE is used to define this relationship. This command allows for more than one data set to be related to a BASE file during the command cycle. If relationships for more than one database are to be established, use the REPEAT option with the command string.

Relationship entries can be displayed by using the SHOW command. All the data sets and, optionally, all the data elements associated with the set, are displayed for a BASE file. For a data set, the elements directly associated with it are displayed. The elements are displayed in the physical order of their association with the file. The SHOW command indicates key elements with an asterisk, identifies the chain paths, and much more. The REPORT command displays an alphabetical list of the elements associated with a data set, or an alphabetized schema-type listing of elements for a database file.

You can delete the relationship entry with the REMOVE command and change the entry with the CHANGE command. The REORDER command is used to change the physical order in which the files are related to the BASE file.

Adding Data Elements to Data Sets

The ADD command is used to associate the data elements with the data sets. Use the command string ADD FILE to define this association. An entry is defined that associates an element with a data set. Elements may not be associated with a BASE file directly. DICTDBM will not accept such an entry and will issue an error message.

To display the association entries, use either the REPORT or SHOW commands. The SHOW command uses an asterisk to indicate which element is the key element in the file and to display the elements in the order in which they are associated with the file.

To delete an association entry, use the DELETE command. To change an association entry, use the UPDATE command. To change the physical order in which the elements are associated to the data set, use the RESEQUENCE command.

Securing a Database

The security that may be defined is the same security defined for an IMAGE database. IMAGE allows the database designer to control access to specific data sets and data items by defining up to 63 user classes. The classes are associated with data sets and data items in read or write class lists. This association determines which user classes may access which data elements and the type of access that is granted. For a complete description of how to protect an IMAGE database, see the *IMAGE/3000 Reference Manual*.

In the Dictionary, user security classes are identified by an integer from 0 through 9999 and are associated with a password defined by the database designer. Each security class number is unique and can be created only once.

When the Dictionary/3000 Database Creation utility program (DICTDBC) is used to create the database, the security class numbers are reassigned by the utility to conform to the numbering scheme used in IMAGE. The CREATE command must be used to define a user security class entry. You use the command string CREATE CLASS for this entry. During the command cycle, the prompt RESTRICT CLASS TO A FILE (N/Y) is issued which allows you to restrict the use of the class to a particular database. You may choose to respond Y to this prompt if the sets and elements you will add to this class belong to more than one database but you want DICTDBC to include this class in the schema of only one database. When DICTDBC retrieves the security classes associated with the sets and elements of a database for which it is to generate a schema, it will skip those classes restricted to another database.

After creating user security classes, either the LIST or DISPLAY command can be used to display the entry. If you want to delete an entry from the Dictionary, use the PURGE command. To change the entry, use either the MODIFY command or the RENAME command.

The association between the security classes and the data sets or the elements is established using either the SECURE or ADD command. The SECURE FILE and ADD CLASS command strings allow you to specify protection at the element level, while ADD CLASS-FILE specifies protection at the file level.

SECURE FILE is used to specify protection that applies to all the data elements in the file. In this way, it is similar to using repeated ADD CLASS command strings. SECURE FILE can also be used to specify protection for the file itself.

Creating a Database

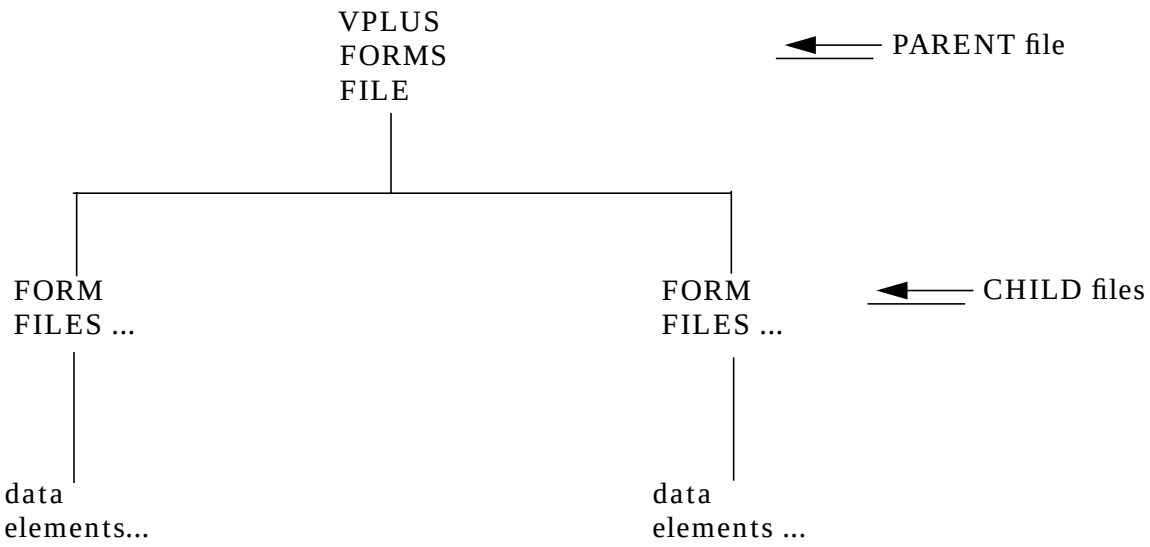
The Database Creation utility program (DICTDBC) can be used to create a schema file and a root file from the definition in the Dictionary. Note that the definition, associations, and relationships must already be defined in the Dictionary to use this utility. See Section V of this manual for a detailed description of this utility.

Using Commands to Define Other File Structures

MPE files, MPER files, KSAM files and VPLUS forms files can also be defined in the Dictionary. The DICTDBM commands are used to create and define these files in the Dictionary.

For MPE, MPER, and KSAM files, you define the record layout for the file. In the Dictionary, each field in the record is created as a data element with characteristics such as type, size, and so forth. For the VPLUS forms file, each form is created as a file, and the fields within each form are created as data elements. The definitions for a VPLUS forms file is completed by establishing the relationship between the VPLUS forms file and each form in the forms file. This relationship is also a hierarchical relationship and is established using the RELATE command. (Refer to Section IV of this manual for a description of this command.) This hierarchical relationship is summarized in Figure 3-2.

Figure 3-2. Summary of Hierarchical Structure for a VPLUS Forms File



The VPLUS forms file is the top of the structure and is the PARENT file. Each FORM file is related to the VPLUS file and is considered a CHILD file. In this structure, the PARENT file cannot have data elements directly associated with it.

The following description outlines the steps involved when the DICTDBM commands are used to define other file structures. A complete description including examples for all the commands is given in Section IV of this manual.

Creating MPE, MPER, KSAM, and VPLUS Elements and Files

Creating files and data elements for these structures is similar to creating them for a database. An element or a file must be defined in your Dictionary before you can associate or relate it to another entry. To define the entries, use the command string CREATE ELEMENT or CREATE FILE. The name used for the element or the file must be unique. Use the REPEAT option before the command string to create entries for many elements or files without having to reenter the command. A definition for the VPLUS forms file must be created before the hierarchical relationship between the forms file and the forms can be created.

After the entries are created, use the LIST or DISPLAY commands to display the entry. Entries can be deleted by using the PURGE command, and can be changed by using the MODIFY or RENAME command. To define the CHILD elements (fields) within a PARENT data element, use the RELATE ELEMENT command string.

Adding Data Elements to Files

The ADD command is used to associate data elements to files. Use the command string ADD FILE to define this association.

Elements may not be directly associated to the VPLUS forms file. DICTDBM will not accept this and will issue an error message. The elements must be associated with the file according to the record layout or form design. When elements are associated with a KSAM file, you must identify which element or elements are the keys for the file.

To display an alphabetized listing of association entries, use the REPORT command. For a form, the list shows the elements associated with the form; for a VPLUS forms file, the list shows all the elements associated with the CHILD files. The SHOW command is used to display each file and the elements associated with it. It also gives a complete listing of the forms in a VPLUS file and optionally, the elements within each form. The forms are displayed in the physical order of their relationship to the VPLUS forms file, and the elements are in the physical order of their association with the form. For KSAM files, the SHOW command indicates key elements with an asterisk, “*” .

An association entry can be deleted with the DELETE command or changed with the UPDATE command. If you want to change the physical order in which the association entries were made, use the RESEQUENCE command.

Relating Forms to a VPLUS Forms File

The hierarchical relationship between the forms and a VPLUS forms file is established using the RELATE command. The command string RELATE FILE is used to define this relationship. This command allows more than one form to be related to the VPLUS file during the command cycle. Use the SHOW command or the REPORT command to display the relationship entries.

The relationship entry is deleted by using the REMOVE command and is changed by using the CHANGE command. Use the REORDER command to change the physical order in which the CHILD files are related to the PARENT file.

Security for MPE, KSAM, and VPLUS Files

Security as defined and used for IMAGE databases is not applicable to MPE files, KSAM files, and VPLUS forms files. For these files "lockwords" are used for protection. The CREATE command is used to define a security class for the lockword. In the Dictionary, the lockword for the file is called the PASSWORD. After a class is created for the lockword, the ADD CLASS-FILE command string is used to associate the security class with the file.

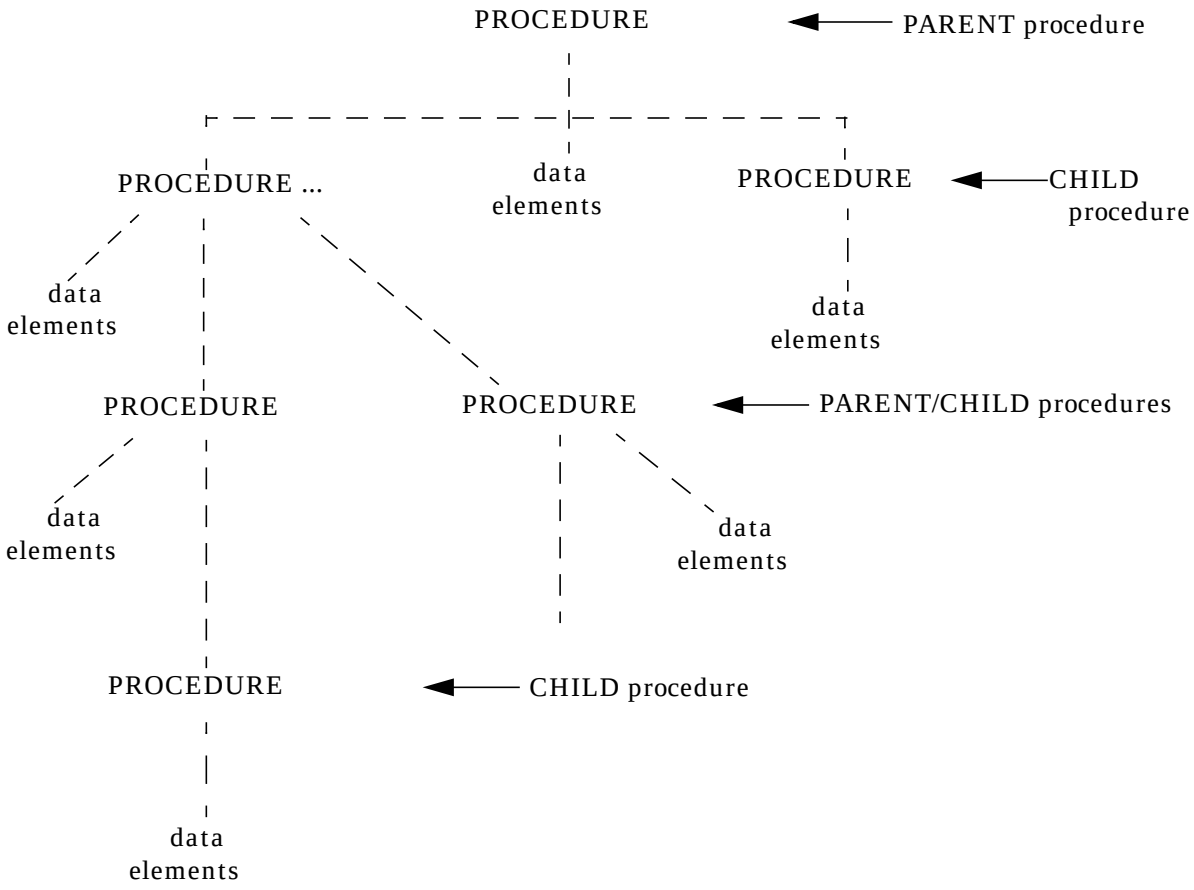
To display the security class entries, use either the LIST or DISPLAY command. If you want to delete an entry from the Dictionary, use the PURGE command. To change the entry, the MODIFY command or the RENAME command is used.

Using Command to Define Procedures

The Dictionary can be used to capture the data processing environment for your company by defining all the application programs, the routines, the subroutines, and the variables used in the Dictionary. The Dictionary uses PROCEDURES to document this information. Each program, routine, or subroutine is a procedure in the Dictionary. The variables used by each are data elements with characteristics such as type, size, and so forth.

A program and its routines or subroutines can be described in the Dictionary by defining a hierarchical relationship between them. The RELATE command is used to establish this hierarchical relationship. (Refer to Section IV of this manual description of this command.) The hierarchical relationship for procedures is summarized in Figure 3-3.

Figure 3-3. Summary of Hierarchical Structure for a Procedure



The procedure that uses or calls other procedures is at the top of the structure and is a PARENT procedure. The routines and/or subroutines used or called by the PARENT procedure are CHILD procedures. A CHILD procedure can in turn be a PARENT because it may use or call other routines or subroutines. This hierarchical structure allows data elements to be associated with a procedure regardless of their level in the structure.

Note that Dictionary/3000 will not allow you to document recursive procedures. That is, if a procedure has been documented as a CHILD of another procedure, Dictionary/3000 will not allow you to specify that other procedure as a CHILD of the first one.

The following description outlines the steps involved when you use the DICTDBM commands to define procedures. A complete description including examples for all the commands is given in Section IV of this manual.

Creating Data Elements and Procedures

The method for creating procedures and data elements is similar to creating files and data elements for a database. To define them, use the command string CREATE PROCEDURE or CREATE ELEMENT. An element or a procedure must be defined in your Dictionary before you can associate it with or relate it to another entry. The name used for the procedure or the element must be unique. Use the REPEAT option before the command string to create definitions for more than one procedure or element without having to reenter the command.

After the entries have been created, you can use the LIST or DISPLAY commands to display those entries. If you want to delete an entry, use the PURGE command. To change an entry, use either the MODIFY command or the RENAME command.

Relating Procedures to Procedures

Procedures may be related to other procedures by establishing a hierarchical relationship between the procedures with the RELATE command. Use the command string RELATE PROCEDURE to define this relationship. This command allows more than one CHILD procedure to be related to a PARENT procedure during the command cycle. If relationships are to be established for more than one PARENT procedure, use the REPEAT option with the command string.

Relationship entries can be displayed by using the SHOW command. For a PARENT procedure, the SHOW command displays the data elements associated with it, the CHILD procedures related to it, and optionally, the data elements associated with each CHILD. For a CHILD procedure, only the data elements associated with it are displayed. The elements are displayed in the physical order of their association with the procedure, and the CHILD procedures are displayed in the physical order they were related to the PARENT procedure.

The REPORT command can be used to display an alphabetized list of the elements used by a procedure. It can also be used to show the CHILD procedures used by the PARENT procedure.

To delete the relationship between entries use the REMOVE command. To change the relationship, use the CHANGE command. To change the physical order in which the CHILD procedures are related to the PARENT procedure, use the REORDER command.

Adding Data Elements to Procedures

The ADD command is used to associate data elements with a procedure. Use the command string ADD PROCEDURE to define the association. An entry is defined that associates the element with the procedure. To display the entries, use either the REPORT command or the SHOW command.

An association entry can be deleted by using the DELETE command or it can be changed by using the UPDATE command. If you want to change the physical order in which the associated entries were made, use the RESEQUENCE command.

Using Commands to Define Locations

The Dictionary can also be used to document the physical locations where procedures and files are used. DICTDBM uses LOCATIONS to capture this information. The specific MPE group and account, and the machine on which the procedure or data file is used, can be identified. This information is helpful when a change to a procedure or file occurs because it tells you where the procedure or data file resides. This information can also be used to generate file equations for the referenced files by HP INFORM/3000.

Hierarchical structures for locations cannot be defined in your Dictionary.

The following description outlines the steps involved when the DICTDBM commands are used to define locations. A complete description including examples for all the commands is given in Chapter 4 of this manual.

Creating Locations

Use the CREATE LOCATION command string to define an entry for a location. The name used for the location must be unique. Use the REPEAT option before the command string to create definitions for more than one location without having to reenter the command.

Information identifying the MPE group, account, and machine is not required for the entry. However, when changes are made to the files and procedures, you can save time and effort by knowing where the updates need to occur.

After the entries are created, use the LIST or DISPLAY commands to display the entry. If you want to delete an entry, use the PURGE command. To make changes to the entry, use either the MODIFY command or the RENAME command.

Adding Files and Procedures to Locations

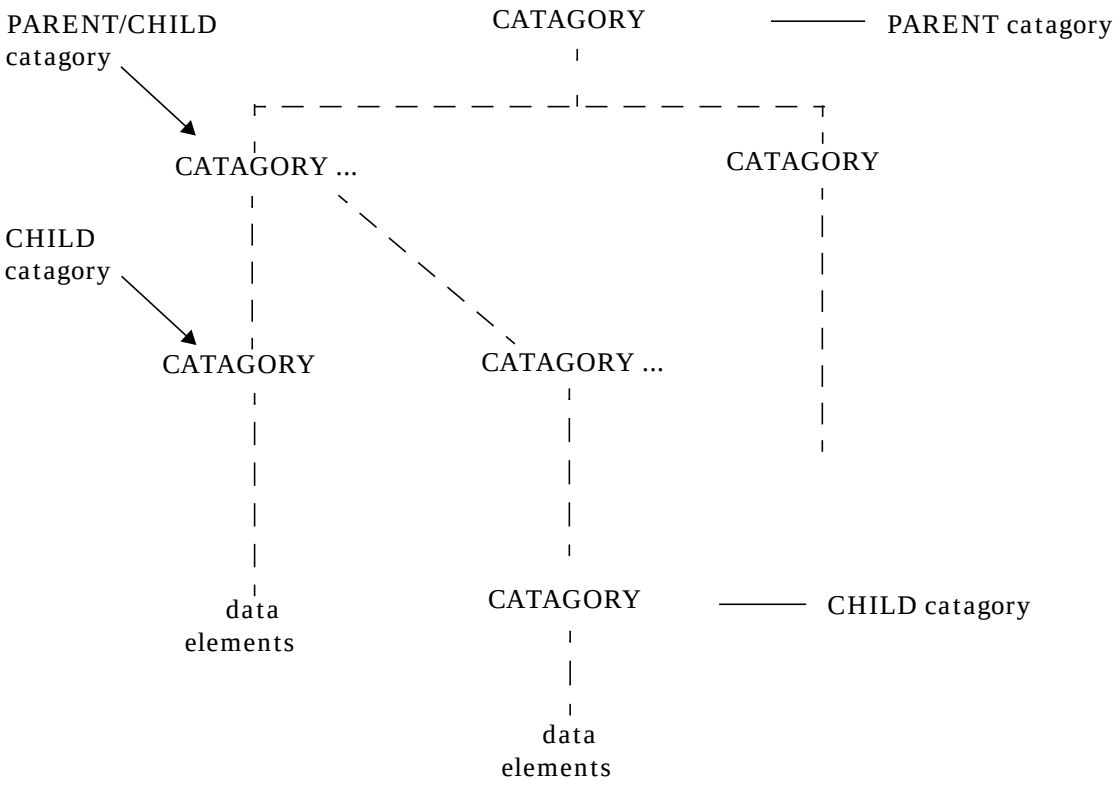
Before a file or procedure can be associated with a location, it must be defined in the Dictionary. The method for associating files and procedures with a location differs slightly from previous methods. You use the ADD command to define the association, but the command string is ADD FILE-LOC or ADD PROCEDURE-LOC. An entry is defined that associates the file or procedure with the location. To display the association entries, the command strings DISPLAY LOCATION or SHOW LOCATION are used. An association entry is deleted by using the command strings DELETE FILE-LOC or DELETE PROCEDURE-LOC.

Using Commands to Define Categories

The Dictionary provides a way for you to identify functional areas in your business and to show how information flows from area to area. The Dictionary uses CATEGORY to define this information.

Each functional area is a category in the Dictionary. The information may consist of departments, people, functions, documents, and so forth. The organizational flow chart for the functional area can be documented by defining a hierarchical relationship between them. This relationship is established by using the RELATE command. (Refer to Chapter 4 of this manual for a description of this command.) This hierarchical relationship is summarized in Figure 3-4.

Figure 3-4. Summary of Hierarchical Structure for a Category



The category at the top of the structure is a PARENT category. Other categories are related to the top category and are CHILD categories. A CHILD category can in turn be a PARENT category to other CHILD categories. In this hierarchical structure, a PARENT category cannot have data elements associated with it.

The following description outlines the steps involved when the DICTDBM commands are used to define categories. A complete description including examples for all the commands is given in Section IV of this manual.

Creating Elements and Categories

To create data element and category entries, use the CREATE command. You use the command string CREATE ELEMENT or CREATE CATEGORY to define the entry. An element or a category must be defined in the Dictionary before it can be associated or related to other entries. The name you use for the category or the element must be unique.

The created entries can be displayed using the LIST or DISPLAY commands. Changes can be made to the entry by using the MODIFY command. The entry can be renamed by using the RENAME command. To delete the entry, use the PURGE command.

Relating Categories to Categories

The hierarchical relationship between categories is established with the RELATE command. Use the command string RELATE CATEGORY to define this relationship. This command allows more than one category to be related to the PARENT category during the command cycle.

To display the relationships, use the SHOW command. For a PARENT category, this command displays each CHILD category. It also displays the data elements associated with the CHILD category. The elements are displayed in the physical order in which they were associated with the category, and the CHILD categories are displayed in the physical order in which they were related to the PARENT category.

The REPORT command can be used to display an alphabetical listing of the elements associated with a category. For a PARENT category, REPORT displays all the elements associated with the CHILD categories.

The relationship entry can be deleted with the REMOVE command, and it can be changed with the CHANGE command. Use the REORDER command to change the physical order in which the CHILD categories are related to the PARENT category.

Adding Elements to Categories

The ADD command is used to associate data elements to categories. Use the command string ADD CATEGORY to define this association. An entry is defined that associates an element with a category. Elements cannot be associated to a PARENT category. DICTDBM will not accept this and will issue an error message if it is tried.

To display association entries, use either the REPORT command or the SHOW command. If you want to delete the association entry, use the DELETE command. To change the entry, use the UPDATE command. To change the physical order of the association, use the RESEQUENCE command.

Using Commands to Define HP Inform/3000 Groups

The Dictionary must be properly set up in order to produce the desired reports using HP Inform/3000. Usually this is the responsibility of the Database Administrator (DBA). HP Inform/3000 enables the user to define and create reports from data contained in IMAGE databases, MPE files, and KSAM files. (For specific information on how to run HP Inform/3000 and produce reports, see the HP Inform/3000 User's Guide.)

HP Inform/3000 makes the distinction between two methods of organizing data: databases and HP Inform/3000 groups. In order for a user to define a report, HP Inform/3000 presents a Data Organization Menu from which one of these two methods must be selected. Which method is appropriate depends on whether the data elements can all be found in one data set, or whether they have been organized into groups.

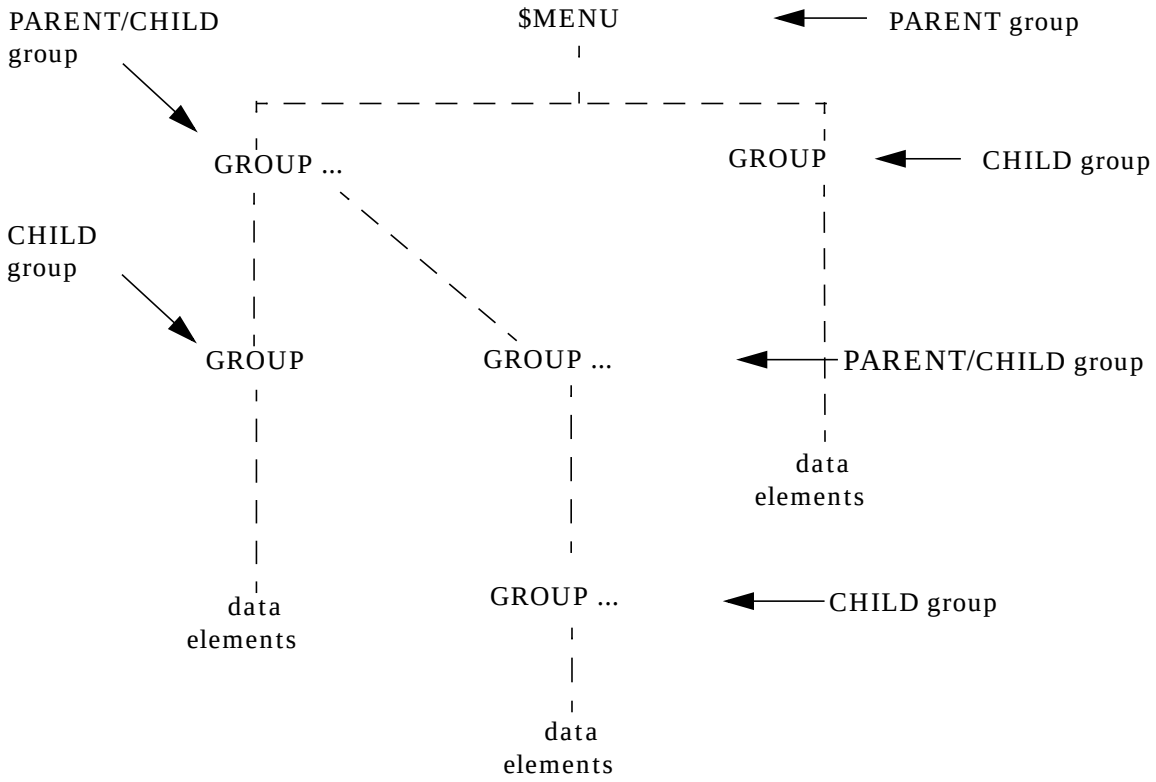
HP Inform/3000 Groups

HP Inform/3000 groups are logical groupings of data elements which have been defined in the Dictionary. Before defining HP Inform/3000 groups, the DBA needs to gather and organize all the information required by all the HP Inform/3000 users. For example, some users will want to produce marketing reports, some manufacturing reports, others personnel reports, etc. The groups can be defined according to these identified categories of information which have logically connected data elements - "data names" as they are called in HP Inform/3000.

By organizing data elements into groups, the DBA is taking advantage of HP Inform/3000's ability to access elements from multiple and different types of data files. Elements can be accessed from more than one data set. KSAM and MPE files can also be accessed. In addition, an element can be assigned an alias name which is especially meaningful to the HP Inform/3000 user. This name will then appear on HP Inform/3000's Data Names Menu.

A hierarchical relationship between groups is established by using the RELATE command (refer to Section IV of this manual for a description of this command). This relationship is summarized in Figure 3-5. It is reflected in the hierarchy of HP Inform/3000 Group Menus and Subgroup Menus as seen by the HP Inform/3000 user.

Figure 3-5. Summary of Hierarchical Structure for Groups



$MENU is the top of the structure and is the PARENT group. Other groups are related to $MENU and are CHILD groups. A CHILD group can in turn be a PARENT group of other CHILD groups (which are called "subgroups" in HP Inform/3000 and appear on HP Inform/3000's Subgroup Menu). This hierarchical relationship can exist up to 10 levels deep, starting with $MENU. A PARENT group cannot have data elements associated with it. Note that $MENU is created automatically when the Dictionary is initialized and cannot be altered or purged.

Each group in this hierarchy must be defined in the Dictionary. The data elements you wish to associate with a group and the files those data elements are associated with must first be defined in the Dictionary. Preceding topics in this section provide the details for defining databases and other file structures. The following description outlines the steps involved when you use the DICTDBM commands to define groups. A complete description for all the commands is given in Section IV of this manual.

Creating Elements and Groups

To create data elements and group entries, use the CREATE command. The CREATE ELEMENT or CREATE GROUP command string is used to define the entry. An element or group must be defined in your Dictionary before you can associate or relate it to other entries. The name entered for the element or group must be unique among elements and groups, respectively. The remaining prompts allow the user to provide useful information, but are not required. You can create many elements or groups in a command cycle by using the REPEAT option.

After creating entries in the Dictionary, you can use the LIST or DISPLAY commands. If you want to delete an entry, use the PURGE command. To change an entry, use either the MODIFY or RENAME commands.

The RELATE ELEMENT command string can be used to define CHILD elements (fields) within a PARENT data element. Note, however, that to report on a CHILD data element from HP Inform/3000, that data element must have been added to the group (the PARENT element may or may not be in the group).

Relating Groups to a Group

The hierarchical relationship between groups is established with the RELATE command. Use the command string RELATE GROUP to define this relationship. When relating groups to the top of the hierarchical structure, enter $MENU in response to the prompt for PARENT GROUP; in response to CHILD GROUP, enter the name of the group to be related to $MENU. The groups related to $MENU will all appear on HP Inform/3000's Group Menu.

A relationship for other levels is defined in a similar manner - by entering the name of the higher level group for PARENT GROUP, and the lower level group for CHILD GROUP. If the PARENT group is not $MENU, the CHILD groups related to that PARENT group will appear on an HP Inform/3000 Subgroup Menu.

The DESCRIPTION prompt allows you to supply useful information but is not required to define the relationship.

The relationship between groups can be displayed by using the SHOW command. For a PARENT group, this command displays each CHILD group, if any, and the data elements associated with each CHILD. For a CHILD group, the data elements associated with it are displayed. The elements are displayed in the actual order they were associated with the group.

The REPORT command can be used to display an alphabetical listing of the elements associated with a group or with related CHILD groups.

You can delete the relationship entry by using the REMOVE command and can change the entry by using the CHANGE command. The REORDER command is used to change the actual order in which the CHILD groups are related to the PARENT groups.

Example

```
>RELATE GROUP
            PARENT GROUP> $menu              <
            CHILD GROUP> marketing           <
            DESCRIPTION>
            CHILD GROUP> shipping            <
            DESCRIPTION>
            CHILD GROUP> personnel           <
            DESCRIPTION>
            CHILD GROUP>                     <
```

Adding Elements to Groups

The ADD GROUP command string associates data elements with previously created groups. The data element must already exist in the data Dictionary. The Dictionary will prevent you from associating elements with a PARENT group by issuing an error message and reissuing the prompt for GROUP. (Note that an element need not be associated with a file at the time the ADD GROUP command string is executed; however, the element must have been added to a file before HP Inform/3000 is executed.)

In response to the prompt for ELEMENT, enter the name of the data element. Responding to the ELEMENT ALIAS prompt is optional. HP Inform/3000 uses the element alias name in its Data Names Menu; if no element alias is given, the data element name (primary name) is used. Note that this element alias name should be meaningful to the HP Inform/3000 user since it will appear on HP Inform/3000's Data Names Menu. It may or may not be the same as the element alias name specified when the element is added to a file.

NOTE The element alias name is only used by HP Inform/3000 on its Data Names Menu. The data element name (primary name) will be used as the report heading for that element unless a heading has been specified in response to the HEADING TEXT prompt of the CREATE ELEMENT command string. In addition, the data element name (primary name) will be used in HP Inform/3000's "SELECTION CRITERIA FOR data name(s) " prompt unless you have responded to the ENTRY TEXT prompt of the CREATE ELEMENT command string. If you wish this to match the element alias name, you must specify the element alias name as the HEADING TEXT and the ENTRY TEXT on the CREATE ELEMENT command string.

The prompt for FILE allows you to designate the name of the specific file which should be used to retrieve the values of an element when a report is being created from this group. For example, the data element "date" may be the date a product was purchased in one file, and the date the product was shipped in another file. Thus it is important to specify from which file you want the values for "date" to be taken. This would apply even if the data element "date" had been given different alias names when it was added to the different files. (This particular situation could be avoided if two separate data elements, such as "pur-date" and "ship-date" , are created in the Dictionary and added to the appropriate file.) Note that when a file name is specified in response to the FILE prompt, the element must already be associated with that file.

If, on the other hand, it is not important which file is used (for example, suppose the data element "soc-sec-no" existed in more than one file), it is better not to respond to the FILE prompt. This allows HP Inform/3000 to retrieve the values in a more efficient manner.

If a file specified in response to FILE is an IMAGE data set with more than one PARENT file (BASE), the prompt PARENT FILE is issued next and must be answered.

The next prompt, VALUE AS A LINK, can be used to specify which elements should be selected to link the desired elements into a logical record for reporting:

- If each of the elements being associated with a group can link all the files, do not respond to the VALUE AS A LINK prompt for any of them.

- If only one of the elements can link all the files, then give it a high priority link value (1) and let the link values of the others default (0). * If more than one element is needed to link the files, give a link value of 1 to the one that should be tried first, 2 to the one that should be tried next, and so on. Use -1 if you never want the element to be used as a link and 0 (the default) if the element could be used if needed.

NOTE An element can link two files only if it exists in both files and if certain additional requirements, depending on the file being linked to, are met. See Appendix D, How HP Inform/3000 Links Files to Generate Reports, for more information on linking.

Elements assigned a positive link value form a prioritized list to be used when it is necessary to link files. The lower the positive integer assigned to an element, the higher the likelihood that it will be used as a link. In general, you should assign a link value of 1 to the element that will link the most files containing elements for the group, 2 to the element that will link the next most, and so on. This will ensure that the reports an HP Inform/3000 user requests from a particular group will be generated efficiently.

Table 3-1 summarizes the priorities of the possible link values:

Table 3-1. Link Value Priorities

Link Value	Priority
1	Highest priority
2	Next highest priority after link value of 1
3	Next highest priority after link values 1 and 2.
.	.
.	.
n	Next highest priority after link values 1, 2, ...n-1
-1	Never used as a link

Note that if an element has different meanings in different files (for example, the data element "date" may have different meanings in different files), you will not want it to be used as a link even if it links all the files containing elements for the group. Give such a data element ("date") a link value of -1. (Note that in this case, simply answering the FILE prompt, without specifying a link value of -1, will cause the correct values for "date" to be printed in the report but will not help to link the files correctly.)

By both responding to the FILE prompt for an element and assigning a positive link value to the element, you are designating a potential "driving file" . The driving file is the file that is accessed first. Since at least one line of the report will be generated for each record retrieved from the driving file, the contents of a report could be quite different if the driving file is different. By designating a potential driving file, you are specifying that file should be accessed first whenever:

- any element from the group which exists in that file is requested for the report, and

- no other potential driving file has been designated which has a higher priority link value specified and contains an element requested for the report.

Hence, if you respond to the FILE prompt and designate a link value of 1 for a particular element, the specified file will be accessed first whenever any element which exists in that file is requested in a report. If you respond to the FILE prompt and designate a link value of 2 for another element, the file specified in response to this FILE prompt will be accessed first whenever any element which exists in that file is requested in a report and no element which exists in the first file is requested.

NOTE The linking process determines whether it is possible to generate the requested report and, if so, what specific data will be printed in the report. For more information about linking, link values, and driving files, see Appendix D of this manual, How HP Inform/3000 Links Files to Generate Reports.

The DISPLAY ELEMENT ON MENU prompt allows the user to determine whether an element is to be displayed on HP Inform/3000's Data Names Menu and used when generating reports. By not displaying an element on the menu, you can include an element in the group for linking and not allow it to be used for reporting.

Note that a CHILD data element can not be used in linking files. CHILD elements can, however, be requested in an HP Inform/3000 report as long as they have been added to the group; the PARENT element alone in the group is not sufficient. If a data element which is a CHILD is needed for a report, HP Inform/3000 will try to find the PARENT element. Since a CHILD element could have more than one PARENT, it will first use a PARENT that has been added to the group. If no PARENT of the CHILD element exists in the group, HP Inform/3000 tries to find any PARENT of the CHILD element it can find. In order to ensure that the correct CHILD element is retrieved, you should add the PARENT element to the group (in addition to the CHILD element); if you do not wish users to report on the PARENT element, add it to the group and respond "NO" to the DISPLAY ELEMENT ON MENU prompt. (You may also wish to specify a link value of -1, if you do not want the PARENT element to be used to link files.) HP Inform/3000 retrieves the value of the entire PARENT element, but only displays the value of the appropriate CHILD element in the report.

You can display association entries by using the REPORT or SHOW commands. To delete the association, use the DELETE command; to change it use the UPDATE command. The order in which the elements are associated with the group are the order they will appear on HP Inform/3000's Data Names Menu; this can be changed by using the RESEQUENCE command.

Example

```
>ADD GROUP
                      GROUP> marketing           <
                    ELEMENT> name                <
               ELEMENT ALIAS> employee name      <
                       FILE> !                   <
                    ELEMENT> date                <
               ELEMENT ALIAS> date of hire       <
                       FILE> empfile             <
            VALUE AS A LINK>                     <
     DISPLAY ELEMENT ON MENU (Y/N)?n
                DESCRIPTION>
                    ELEMENT>                     <
```

Using Commands to Define HP Inform/3000 Security

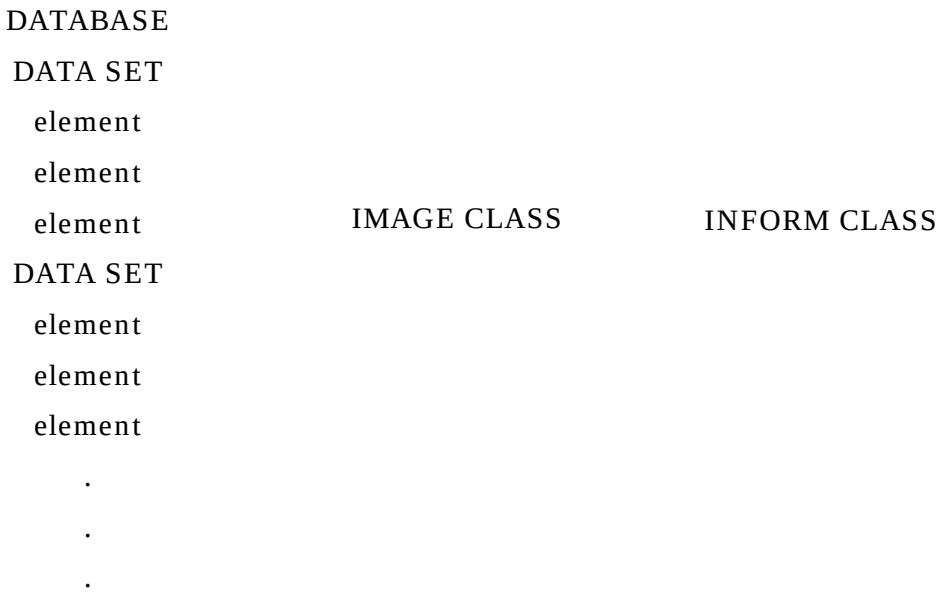
HP Inform/3000 security prevents unauthorized persons from gaining read access to the Dictionary by limiting the names of the database files and Inform groups displayed on the Inform menus. When Inform security is set up in the Dictionary, Inform menus will display only the information granted a user via an Inform password.

To establish Inform Security, the database administrator must first determine who should have access to what data. By using the DICTDBM commands, Inform classes are created and passwords are assigned to each Inform class. Database files and elements are associated to IMAGE classes which are in turn related to the Inform classes. Inform groups are directly associated to the Inform classes. When Inform executes, it looks for any Inform class relationships defined in the Dictionary and prompts you for an Inform password if any such relationships are found. If no Inform class relationships are found, you are not prompted for an Inform password and all the database files and Inform groups defined in the Dictionary will be displayed on the Inform menus. You will have to enter the necessary passwords/lockwords when you produce the report. The Inform password determines what Inform class you belong to. Only the database file names and Inform group names belonging to that Inform class will be displayed on the Inform menu.

If you use either the Dictionary manager or programmer level password as the Inform password, then you are granted access to all Inform groups and database files defined in the Dictionary. (See Appendix C of the manual for more information on the access levels to the Dictionary.)

Database and the Inform Class Relationships - Figure 3-6 shows the relationships between a database and an Inform class. Each line represents a relationship.

Figure 3-6. Relationships Between the Database and the Inform Class

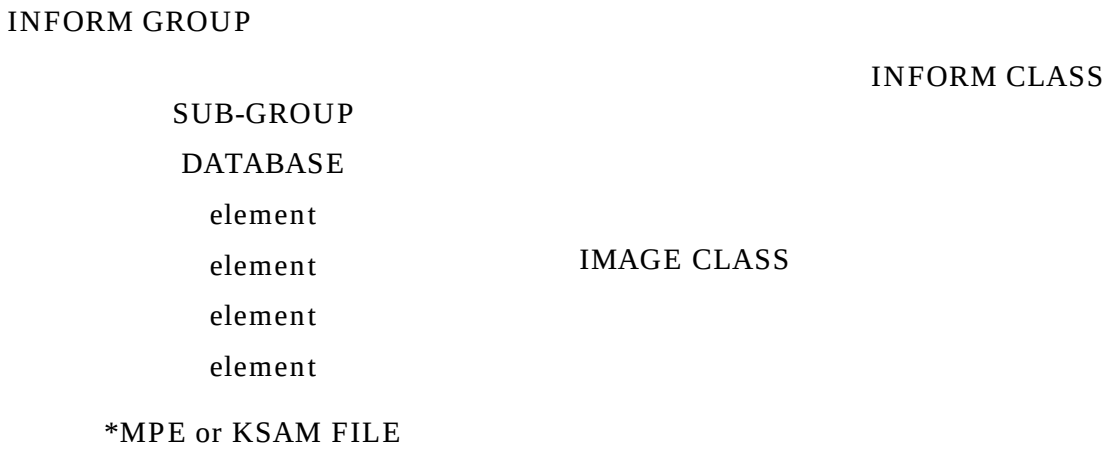


Note that the database file must be directly associated with the IMAGE class or Inform will not give you access to any of the database sets or elements. Each data set and element in the database must also be directly associated with the same IMAGE class as the database file or Inform will not give you access to that data. NOTE: Inform Security requires element (item) level access to be defined in the dictionary before the element (item) can be displayed on the Inform Menu. Inform opens a database in open mode 5 which requires security to be defined at the data item level. (See the IMAGE/3000 Reference Manual for more information.)

Depending on how security is to be implemented, a database, data set, or element can belong to more than one IMAGE class. An IMAGE class can belong to more than one Inform class.

Inform Groups and Inform Class Relationships - Figure 3-7 shows the relationships between an Inform group and an Inform class.

Figure 3-7. Relationships between an Inform group and Inform Class



*Relating an MPE or KSAM file to an IMAGE class is only necessary if the file has a lockword.

Inform groups and subgroups are directly associated to the Inform class. However, in order to access any elements in the Inform groups, the elements must be associated to an IMAGE class and the IMAGE class must be related to the same Inform class as the Inform group. Otherwise, Inform will not allow access to these elements. The database files in which these elements are contained must also be associated to an IMAGE class. It is necessary to associate an MPE or KSAM file to an IMAGE class if the file has a lockword. When Inform opens an MPE or KSAM file and that file has been added to an IMAGE class which has a password, Inform uses that password as a lockword when opening the file. That means that if you want Inform to open an MPE or KSAM file that has a lockword and you don't want to be prompted for the password, you must add the file to an IMAGE class whose password matches the file's lockword. In doing this, it is recommended that a different class be assigned to each file or database, and that CLASS-TYPE (MPEF, MPER, KSAM, or BASE) be used to distinguish them for documentation purposes.

Depending on how Inform security is to be implemented, an Inform group or sub-group can belong to more than one Inform class. An element can also belong to more than one IMAGE class.

Creating Inform Classes

In the Dictionary, Inform classes are created as INFO type classes and are identified by an integer from 0 through 9999. Use the CREATE CLASS string to create an Inform class. Enter the identifying class number and INFO as the class type. Enter the password to be assigned to the Inform class. If the password in the dictionary is entered in upper case then Inform also expects the password to be entered in upper case. If the case does not match Inform will issue an invalid password error message. Note that the Dictionary will not prevent you from creating duplicate passwords for Inform classes. If duplicate passwords are created, Inform will use the class belonging to the first matching password that it finds in the Dictionary.

The remaining prompts for this command string allow you to provide useful information, but are not required.

After an Inform class is created, you can use either the LIST or DISPLAY command to display the entry. If you want to delete the entry from the Dictionary, use the PURGE command. To change the entry, use either the MODIFY or the RENAME command.

Relating IMAGE Classes to Inform Classes

The hierarchical relationship between the Inform class and the IMAGE class is established with the RELATE command. Use the RELATE CLASS string to define this relationship. Enter the identifying number of the Inform class in response to the prompt for PARENT CLASS. Enter the identifying number of the IMAGE class in response to the prompt for the CHILD CLASS. The description prompt allows you to supply useful information but is not required to define this relationship.

The relationship between the classes can be displayed by using the SHOW command. For the Inform class, this command displays each CHILD class. For the CHILD class, this command displays the files and elements associated with the CHILD class in the order in which they were associated with the CHILD class. To display an alphabetized listing of the elements associated with a CHILD class, you can use the REPORT command. Reporting an INFO class will display "No elements found" because elements are not and cannot be directly related to an INFO class. To display the attribute information and related PARENT classes for a CHILD class, you can use the DISPLAY command.

The relationship between an Inform class and an IMAGE class can be deleted by using the REMOVE CLASS command string. The description of the relationship can be changed by using the CHANGE CLASS command string.

Adding Entities to an IMAGE Class

The association between the IMAGE classes and the different entities (databases, data sets, elements and files) is established using either the ADD command or the SECURE command. Remember that before Inform can access these entities, they must belong to an IMAGE class and that IMAGE class must be related to the Inform class.

To associate an element with an IMAGE class, use the ADD CLASS command string. To associate a file to an IMAGE class, use the ADD CLASS-FILE command string.

You can also use the SECURE command to assign all the elements associated with a file to an IMAGE class. By using the SECURE FILE command string, you can avoid adding each individual element, one at a time, to an IMAGE class. The SECURE FILE string can also be used to secure a database, data set, or a file and any child files it may have, to that same IMAGE class.

The association between an element or file and the IMAGE class can be deleted by using the DELETE command. You can display the association entries by using the REPORT or SHOW commands.

Adding Inform Groups to Inform Classes

For Inform security by groups to be implemented, an Inform group is directly associated to an Inform class. However, any elements belonging to the Inform group are not added to that Inform class. They are associated to an IMAGE class. That IMAGE class is then related to the same Inform class as the Inform group which the element belongs to. (See Figure 3-7 for a diagram of this relationship.)

The association between the Inform group and the Inform class is established using either the ADD command or the SECURE command. To associate an individual Inform group to an Inform class, use the ADD CLASS-GROUP command string. Enter the identifying number of an existing Inform class in response to the prompt for CLASS. Enter the name of an existing Inform group in response to the prompt for GROUP. To add an Inform group plus all of its child groups (if any) to an Inform class, use the SECURE GROUP command string. Enter the name of the Inform group in response to the prompt for GROUP. Enter the identifying number of an existing Inform class in response to the prompt for CLASS. Note that no elements belonging to the Inform group or to its child groups will be added to the Inform class. Again, these elements must first be associated to an IMAGE class. That IMAGE class must then be related to the same Inform class as the Inform group which the element belongs to or Inform will not allow access to those elements. To add a file to a class use the ADD CLASS-FILE command string.

You can display the attribute information and any associated Inform classes for an Inform group by using the DISPLAY GROUP command string. You can change the description of an Inform group to Inform class association by using the UPDATE CLASS-GROUP command string. The association itself can be deleted by using the DELETE CLASS-GROUP command string.

Example

```
Defining Inform Security for a database
             >REPEAT CREATE CLASS
                          CLASS> 101              To CREATE the INFO class
                           NAME>
                           TYPE> INFO
                       PASSWORD> BOSS
                 RESPONSIBILITY>
                    DESCRIPTION>
                           CLASS> 10              To CREATE the IMAGE class
                           NAME>
                           TYPE>
                       PASSWORD> VP
                 RESPONSIBILITY>
                    DESCRIPTION>
                           CLASS>
              >RELATE CLASS                       To RELATE the INFO class to
                                                  the IMAGE class
                    PARENT CLASS> 101
                     CHILD CLASS> 10
                     DESCRIPTION>
                     CHILD CLASS>
             >SECURE FILE                         To SECURE the database, data
                                                  sets, and elements to the IMAGE
                                                  class
                           FILE> BASE
                          CLASS> 10
            ACCESS CAPABILITY> R
        ELEMENTS WILL BE SECURED TO CLASS.
        SECURE FILE(S) TO CLASS (N/Y)?> Y
         FILE ACCESS CAPABILITY> R
                   OR
         >ADD CLASS-FILE                          To ADD specific data sets to the
                                                  IMAGE class
                         CLASS> 10
                          FILE> BASE
                   DESCRIPTION>
```

```
                FILE> DATASET1
ACCESS CAPABILITY> R
      DESCRIPTION>
                FILE> DATASET2
ACCESS CAPABILITY> R
      DESCRIPTION>
  >ADD CLASS                    To ADD elements to the IMAGE
                                class
            CLASS> 10
          ELEMENT> E1
ACCESS CAPABILITY> R
      DESCRIPTION>
          ELEMENT> E2
ACCESS CAPABILITY> R
      DESCRIPTION>
          ELEMENT> E3
ACCESS CAPABILITY> R
      DESCRIPTION>
```

Defining Inform Security for Groups: Sales Group and Account Orders

The elements for the Sales Group are Account, Sales Rep, and Address and are contained in the Salestat (KSAM) file which has a lockword of KEEP. The elements for the Account Orders group are Prod-No and Owner and are contained in the Warranty (MPE) file which has a lockword of SAFE.

```
>REPEAT CREATE CLASS
                CLASS> 100          To CREATE the INFO class
                 NAME> <cr>
                 TYPE> info
             PASSWORD> MICHAEL
       RESPONSIBILITY> <cr>
          DESCRIPTION> <cr>
                CLASS> 301           To CREATE the IMAGE classes
                 NAME> <cr>
                 TYPE> <cr>
             PASSWORD> KEEP
       RESPONSIBILITY> <cr>
          DESCRIPTION> <cr>
                CLASS> 302
                 NAME> <cr>
```

```
                TYPE> <cr>
            PASSWORD> SAFE
      RESPONSIBILITY> <cr>
         DESCRIPTION> <cr>
 >RELATE CLASS                    To RELATE the IMAGE class to the
                                  INFO class
      PARENT CLASS> 100
       CHILD CLASS> 301
       DESCRIPTION> <cr>
       CHILD CLASS> 302
       DESCRIPTION> <cr>
       CHILD CLASS> <cr>
>ADD CLASS                        To ADD elements to the Image
                                  class
             CLASS> 301
           ELEMENT> ACCOUNT
ACCESS CAPABILITY> R
       DESCRIPTION> <cr>
           ELEMENT> SALES REP
ACCESS CAPABILITY> R
       DESCRIPTION> <cr>
           ELEMENT> ADDRESS
ACCESS CAPABILITY> R
       DESCRIPTION> <cr>
           ELEMENT> <cr>
             CLASS> 302
           ELEMENT> PROD-NO
ACCESS CAPABILITY> R
       DESCRIPTION> <cr>
           ELEMENT> OWNER
ACCESS CAPABILITY> R
       DESCRIPTION> <cr>
           ELEMENT> <cr>
             CLASS> <cr>
>REPEAT ADD CLASS-FILE        To ADD a file to the IMAGE class
                              (Needed only if the file has a
                              lockword)
            CLASS> 301
```

```
              FILE> SALESTAT
       DESCRIPTION> <cr>
              FILE> <cr>
             CLASS> 302
              FILE> WARRANTY
       DESCRIPTION> <cr>
              FILE> <cr>
             CLASS> <cr>
>ADD CLASS-GROUP                    To ADD groups to the INFO class
             CLASS> 100
             GROUP> SALES GROUP
       DESCRIPTION> <cr>
             GROUP> ACCOUNT ORDERS
       DESCRIPTION> <cr>
```

- If you define a password for Inform Security in upper or lower case characters, Inform expects it to be entered in the exact same way. For example, if you defined BOSS in all upper case characters and then ran Inform and entered the password in lower case, Inform would not accept the password as a valid password.
- Dictionary does not check for duplicate passwords, so avoid assigning two different Inform classes the same password. Inform will use the first class it finds and from there use all the classes related to the Inform class to determine access. The end result is the user will not necessarily have access to the data they want.
- Inform Security requires element (item) level access to be defined in the dictionary and database before that item will be displayed on the Inform menu or accessed at report generation time.
- Avoid relating more than one IMAGE class associated with a particular database to the same Inform class. In the figure below, ORDERS is related to three IMAGE classes. You should choose one of those IMAGE classes to relate to one Inform class to establish access to ORDERS. If two IMAGE classes (1 and 2 in this example) are related to 101, Inform may find IMAGE class 1's password first or IMAGE class 2's password to use to open the database, the results are unpredictable. In this example it would be better to relate IMAGE class 1 to Inform class 101, 2 to 102, and 3 to 103. You may want to have different Inform classes for the database to allow access at multiple levels.

INFORM CLASS	101	102	103
IMAGE CLASS	1	2	3
DATABASE	ORDERS		

Using Selection Criteria

When using the DISPLAY or the LIST command string, you can dictate the listing you want to see by entering one of the Selection Criteria values in response to a prompt. A listing can be chosen based on character values, numeric values, or an expression of these values. In all cases, the type of listing is based on the subcommand used.

You can enter a string of characters with a caret character (^) at the end, beginning, or in the middle of the string. Depending on where used, the caret has the following meaning:

- If used at the beginning of a string, it means that all values ending with the string are to be listed.
- If used at the end of a string, it means that all values beginning with the string are to be listed.
- If two carets are used at the end of a string, it means that all values containing the string are to be listed.
- If embedded between characters in a string, it means that values that match wherever characters appear in the string are to be listed.
-

You can also enter an expression of these values using the following format:

```
[relationship ] value connector  [relationship ] value  ...
```

where:

relationship is an optional field that can contain one of the following two letter codes:

NE	not equal to
LT	less than
LE	less than or equal to
GT	greater than
GE	greater than or equal to

If a relationship is omitted, the default "equal to" is used.

value connector is the Boolean connection between the values. The following terms can be used:

TO	the selected field must contain one of a range of values where the low end of the range precedes this connector and the high end of the range follows this connector.
AND	the selected field must contain both the value preceding and the value following this connector.
OR	the selected field must contain one or both of the values.

The order of precedence is TO, then AND, then OR.

If the value to be selected contains one or more blanks, then the value

must be enclosed in quotation marks as follows.

If you need more than one line for a selection expression, end the first line with a connector, and another selection line prompt is issued.

Examples

The following are examples of using carets and connectors:

PROD^ and ^NO	Values that begin with PROD and end with NO are selected.
ACCT^ or PROD^ and ^NO	Values that begin with ACCT or values that begin with PROD and end with NO are selected.
^FDCUST^ and ^NAME or ACCT^ and ^NO	Values that begin with CUST and end with NAME, or values that begin with ACCT and end with NO are selected.
PROD^ or ACCT^ or CUST^	Values that begin with PROD or ACCT or CUST are selected.
"PROD NO"	Values that equal PROD NO are selected.

The following are examples of using relationship operators:

LT B	Values which are less than values beginning with B are selected.
GE R and LT V	Values which are greater than or equal to values beginning with R, and values which are less than values beginning with V are selected.
GE M or I^ or LT C	Values which are greater than or equal to values beginning with M, or values which are equal to values beginning with I, or values which are less than values beginning with C are selected.

Using Edit Description Commands

Three commands allow text editing of a description: MODIFY, CHANGE, and UPDATE. The MODIFY command allows you to alter a description of an entity you entered with the CREATE command; the CHANGE command allows you to alter a description you entered with the RELATE command; the UPDATE command allows you to alter a description you entered with the ADD command.

You indicate that you want to edit a description by responding with a "Y" or RETURN to the EDIT DESCRIPTION prompt. The description is edited by entering one of the following EDIT commands:

LIST or L	list one or more description lines
ADD or A	add description line or lines
DELETE or D	delete description line or lines
REPLACE or R	replace a description line
RENUMBER or REN	renumber description lines
HELP or H	displays available EDIT commands
EXIT or E	exit EDIT command

Example

```
>MODIFY CATEGORY
                  CATEGORY> FINANCE       <
    EDIT DESCRIPTION(Y/N)?>
    EDIT COMMAND>> ADD                          EDIT command ADD used.
              LINE NUMBER> 2             <     line number to be added.
                    2.000 > This category identifies the finance     <
                    3.000 > department for the Supplies Division.    <
                    4.000 >                                          <

                                               RETURN ends the a
                                               of lines.
    EDIT COMMAND>> EXIT                        EXIT to terminate EDIT.
    EDIT ATTRIBUTES(Y/N)? N
    >                                          RETURN to command prompt.
```

Using Edit Masks

You can define the output format for data elements in reports generated by HP Inform/3000, Report/3000, and Transact/3000. You do this by defining an edit mask for a data element to be formatted in a report.

An edit mask is a string of characters describing how the data element should look when it appears in a report. Characters such as dollar signs and decimal points can be inserted, and leading zeros suppressed before the data element is displayed in reports. You can define edit masks with either the CREATE or MODIFY commands.

Edit masks can be defined with up to 30 characters. All characters except the following, are treated as insert characters:

^ means that the next character in the source field is to be inserted into the display field.

Z means that leading zeros are to be suppressed.

$ means that leading zeros are to be suppressed and that the left-most zero is to be replaced with a dollar sign.

* means that asterisks are to replace leading zeros.

. means that the implied decimal point defined for the element is to be aligned in the position where this one appears.

! means that a decimal point for a data element is to appear in the position specified by “!” ; use of the "!" character overrides an implied decimal point for the data element.

If the last character of the edit string is a negative sign, a negative data element is displayed with a trailing minus sign (-). If the last characters of the edit string are CR (for CREDIT) or DR (for DEBIT), the data element is displayed with a trailing CR or DR, respectively. Table 3-2 shows how edit masks affect the displayed value for different data elements.

Table 3-2. Using Edit Masks with Elements

Data Element Value	EDIT MASK	Displayed Value
1234	$$,$$$.^^	$12.34
123456	$$,$$$.^^	$1,234.56
123456	***,**$.^^	*$1,234.56
000009	ZZZZ.^^	.09
-123456	$$,$$$.^^CR	$1,234.56CR
810807	^^/^^/^^	81/08/07
-123	$$,$$$.^^-	$1.23=
1234	ZZZZ!	1234.

Table 3-2. Using Edit Masks with Elements

Data Element Value	EDIT MASK	Displayed Value
-012345	none	-012345

4 DICTDBM Commands

Overview

A set of commands is provided by the Dictionary's DICTDBM program, which are used to create and maintain entries in a data Dictionary. These commands are used with a subcommand to identify the entity that receives the action of the command. Each command-subcommand used, generates a series of interactive prompts which allows you to enter the necessary data for each entry. This section contains the reference specifications for each command, along with the subcommands and prompts used with the command.

Table 4-1 gives a summary of the DICTDBM commands with a brief description of their function.

Table 4-1. Summary of DICTDBM Commands

COMMAND	FUNCTION
ADD	Adds an association entry.
CHANGE	Changes attributes of entity within a relationship.
CREATE	Creates an entry for a new entity.
DELETE	Deletes an association entry.
DISPLAY	Displays entry information.
EXIT	Terminates the DICTDBM program.
HELP	Briefly describes the functions of DICTDBM commands and subcommands.
LIST	Lists entries.
MODIFY	Modifies an entry.
PURGE	Purges an entry.
RELATE	Establishes a hierarchical relationship entry.
REMOVE	Removes a hierarchical relationship entry.
RENAME	Renames an existing entity.
REORDER	Reorders entities within a relationship entry.
REPORT	Generates a sorted listing of data element entries.
RESEQUENCE	Resequences entities within an association entry

Table 4-1. Summary of DICTDBM Commands

COMMAND	FUNCTION
SECURE	Assigns security for a file's data elements, for the file, or for a group.
SHOW	Shows hierarchical relationship and association entries.
UPDATE	Updates an association entry.

ADD

Adds an association between unlike entities.

Syntax

```
[option] ADD subcommand
```

Use the ADD command to associate an element, a file, or a procedure with an unlike entity. The association is one of ownership. The subcommand identifies the type of entity for which an association is made. For example, ADD CATEGORY associates a data element with a category. In other words, the data element now “belongs” to that category. Similarly, ADD FILE-LOC associates a file with a physical location. The file now “belongs” to that location.

Like entities (such as file to file) can not be associated through the ADD command. (See the RELATE command to relate like entities.)

Data elements can be added to a file's primary format or as secondary format. The primary list of elements associated with a file defines the primary format for that file. This is the default format and is used by the Dictionary utilities and HP Inform. Any number of elements can be associated with the primary format of a file.

A secondary format is an alternate format for a file's storage area (which contains the elements associated with the file). It is used by the DICTCDE utility to generate the REDEFINE clause in a COBOL II data division. (For more information on the DICTCDE utility, see Section 5 of this manual.) A file may have multiple secondary formats, each one represented by a single data element. Each element associated with a secondary format should reflect the alternate format for the file. (For instance, that element could be a parent element with several child elements related to it.)

Elements cannot be associated with a parent entity (an entity that has subordinate entities related to it) unless the parent entity is a procedure. Elements may be associated with parent procedures which have child procedures related to them.

When data elements are associated with an entity, they are added to that entity's list of associations in the order in which they were associated. (See the RESEQUENCE command to change this order.)

Subcommands

Any of the following subcommands can be used with the ADD command:

CATEGORY	adds a data element to a category
CLASS	adds a data element to a user security class
CLASS-FILE	adds a file to a user security class
CLASS-GROUP	adds a group to a user security class
FILE	adds a data element to a file
FILE-LOC	adds a file to a physical location

GROUP	adds a data element to an HP Inform group
PROCEDURE	adds a data element to a procedure
PROCEDURE-LOC	adds a procedure to a physical location

Example

```
>ADD FILE
         FILE> Orders                  Existing file.
 PRIMARY/SECONDARY(P/S)?&gt; P         Indicates element is to be in
                                       primary format.
        ELEMENT> Customer1             Adds an element to a file.
  ELEMENT ALIAS>
     DESCRIPTION>         Customer-account information used by Finance.
                >
        ELEMENT>
>                                      New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string. Two asterisks (**) next to a prompt indicate that this prompt and those which follow are repeated until [[RETURN]] is pressed in response to this prompt.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
ADD	CATAGORY	*	CATAGORY	
		**	ELEMENT	
			ELEMENT ALIAS	
			DESCRIPTION	
ADD	CLASS	*	CLASS	
		**	ELEMENT	The following prompts are issued only if CLASS is not an INFO type; otherwise CLASS is reissued.
		*	ACCESS CAPABILITY	
			DESCRIPTION	
ADD	CLASS-FILE	*	CLASS	The following prompts are issued only if CLASS is not an INFO type.
		**	FILE	
		*	ACCESS CAPABILITY	Issued only if file type is MAST, DETL, or AUTO.
			DESCRIPTION	
ADD	FILE	*	FILE	
			1. If FILE is MAST file	

COMMAND	SUB-COMMAND	PROMPT	
		** ELEMENT	Prompt is KEY ELEMENT if first element added to MAST.
		ELEMENT ALIAS	
		DESCRIPTION	
		2. If FILE is MPEF or MPER file	
		PRIMARY/SECONDARY (P/S)	
		** ELEMENT	
		ELEMENT ALIAS	
		DESCRIPTION	
		3. If FILE is AUTO file	
		* KEY ELEMENT	
		ELEMENT ALIAS	
		DESCRIPTION	
		4. If FILE is DETL file	
		** KEY ELEMENT	
		ELEMENT ALIAS	
		PATH MASTER FILE	* If element is search item.
		PATH SORT ELEMENT	Issued if PATH MASTER FILE responded to.
			* If element's chain is to be sorted.
		PRIMARY PATH (N/Y)	Issued if PATH MASTER FILE responded to.
			* If element defines the primary path
		DESCRIPTION	
		5. If FILE is KSAM file	
		PRIMARY/SECONDARY (P/S)	
		** ELEMENT	
		ELEMENT ALIAS	
		KEY ELEMENT (N/Y)	Issued if element is being added to the Primary format and element TYPE is not B, S or *.
			* If element is KEY element.
		PRIMARY KEY (N/Y)	Issued if Y entered for KEY ELEMENT and no primary key exists.
		DUPLICATES (N/Y)	Issued if Y entered for KEY ELEMENT or for PIMARY KEY.
		DESCRIPTION	
		6. If FILE is FORM file	

COMMAND	SUB-COMMAND		PROMPT	
			ELEMENT	
			ELEMENT ALIAS	
			FIELD NUMBER	
			DESCRIPTION	
ADD	FILE-LOC	*	LOCATION	
		**	FILE	
			FILE ALIAS	
			FILE SIZE	Issued only if file type is MPEF, MPER or KSAM.
			DESCRIPTION	
ADD	GROUP	*	GROUP	
		**	ELEMENT	
			ELEMENT ALIAS	
			FILE	
			PARENT FILE	Issued only if file belongs to more than one database.
			VALUE AS A LINK	
			DISPLAY ELEMENT ON MENU (Y/N)	
			DESCRIPTION	
ADD	PROCEDURE	*	PROCEDURE	
		**	ELEMENT	
			ELEMENT ALIAS	
			DESCRIPTION	
ADD	PROCEDURE-LOC	*	LOCATION	
		**	PROCEDURE	
			PROCEDURE ALIAS	
			DESCRIPTION	

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

ACCESS CAPABILITY Enter the type of capability to be assigned to the user security class for the file or element as follows:

If the subcommand is:

CLASS-FILE - enter one of the following:

R = read only

W = read, write, modify

X = null read/write list “(/)”

CLASS - enter one of the following:

R = read only

U = read and update only

M = read, write and modify

X = null read/write list “(/)”

Refer to the IMAGE Reference Manual for a detailed explanation of the user class access capabilities.

CATEGORY Enter the name of an existing category (20 characters maximum).

CLASS Enter a numeric value which is the identifying number of an existing user security class (0 to 9999).

To establish INFORM security, enter a numeric value which is the identifying number of an existing INFO type security class.

DESCRIPTION Enter the textual description which describes the association between the entities (50 characters per line maximum, unlimited number of lines).

DISPLAYELEMENT ON MENU (Y/N Enter a response to specify whether you want the element to be displayed for reporting on HP Inform's Data Names Menu.

DUPLICATES (N/Y)? This prompt generates data that will be used by the DICTCDE utility. Enter a Y or YES if this key may have duplicate key values for this KSAM file. Enter a N or NO if a duplicate key value is not allowed.

ELEMENT Enter the name of an existing data element which is the primary element (20 characters maximum).

ELEMENT ALIAS Enter the name by which the data element is known within the file. The maximum characters allowed are:

15 for FORM

16 for MAST, AUTO or DETL

20 for MPEF, MPER or KSAM

60 for CATEGORY

20 for an HP Inform GROUP

16 for PROCEDURE

DEFAULT: If [[RETURN]] is pressed in response to this prompt for an element, the alias becomes the primary element name.

Note that HP Inform and HP Report will be unable to retrieve the values of an element from an IMAGE data set if the alias name entered in

response to the ELEMENT ALIAS prompt of the ADD FILE command string is not the same as the actual name of the data element in the data set.

FILE — Enter the name of an existing file (20 characters maximum). Enter the following file names for the appropriate subcommand:

If the subcommand is:

FILE - the file type must be one of the following:

MAST	IMAGE manual master data set
AUTO	IMAGE automatic master data set
DETL	IMAGE detail data set
KSAM	KSAM file
MPEF	MPE sequential file
MPER	MPE relative file
FORM	VPLUS form

CLASS-FILE - the file type must be one of the following:

BASE	IMAGE database
MAST	IMAGE manual master data set
AUTO	IMAGE automatic master data set
DETL	IMAGE detail data set
KSAM	KSAM file
MPEF	MPE sequential file
MPER	MPE relative file
VPLS	VPLUS forms file

FILE-LOC - the file type must be one of the following:

BASE	IMAGE database
KSAM	KSAM file
MPEF	MPE sequential file
MPER	MPE relative file
VPLS	VPLUS forms file

GROUP- the file must have at least one parent (BASE) if it is type MAST, AUTO, or DETL and the element must have previously been associated with the file.

FIELD NUMBER — Enter the field number of this element on the VPLUS form.

FILE ALIAS — Enter the name by which the file is known within the location (8 characters maximum). DEFAULT: If [[RETURN]] is pressed in response to this prompt, the alias name becomes the first 8 characters of the file name.

FILE SIZE Enter an integer value for the number of records in the file in that location (0 to 999999999). DEFAULT is 0.

GROUP Enter the name of an existing HP Inform group (20 characters maximum).

KEY ELEMENT Enter the name of an existing data element (20 characters maximum) that is as appropriate:

For an AUTO file, the search item and only item for an IMAGE automatic master set.

For a MAST file, the first element added to an IMAGE manual master set.

For a KSAM file, an element to be used as a key for a KSAM file.

KEY ELEMENT (N/Y)? Enter Y or Yes if this element is to be a key element for the KSAM file. DEFAULT is No.

LOCATION Enter the name of an existing location (20 characters maximum).

PARENT FILE Enter the name of the existing parent file (20 character maximum) of the database set specified by the preceding FILE prompt. The file type must be type BASE (IMAGE database).

PATH MASTER FILE Enter the name of an existing file (20 characters maximum) of type MAST or AUTO which is the chain header for a detail set path. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates that the element is not to be a search item.

PATH SORT ELEMENT Enter the name of an existing data element (20 characters maximum) that is the sort item for a sorted chain. This prompt is issued if a master data set's name was given in response to the prompt for PATH MASTER FILE. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates that this is not a sorted chain path.

PRIMARY KEY (N/Y)? This prompt generates data that will be used by the DICTCDE utility. Enter Y or YES if this element is to be a primary key for the KSAM file. This prompt will appear for each element associated with the KSAM file until a primary key is designated. Once an element is entered as a primary key, the prompt will not appear again for that file. DEFAULT is No. Note that if you respond to this prompt with NO (or you press [[RETURN]] as the default) then this element is assumed to be an alternate key for the KSAM file.

PRIMARY PATH (N/Y)? If the element is a search item, a Y indicates that it is the primary path for the detail data set.

Note that if a primary path has already been specified during the current prompting cycle of the ADD FILE command string, this prompt is not issued. Next time the ADD FILE command string is issued and the file is an IMAGE detail data set, this prompt is issued again; however,

responding Y to this prompt now results in the prompt PRIMARY ALREADY EXISTS, CONTINUE(Y/N)?. The default response of Y makes this the primary path and cancels the primary path previously identified; responding N or No does not make this the primary path and the primary path previously identified remains active.

PRIMARY/SECONDARY (P/S)? Enter P if this element is to be added to the Primary format for this file. Enter S if this element is to be added as a Secondary format for this file. DEFAULT is P (Primary format).

(See the discussion of the ADD command for more information on Primary and Secondary formats.) Note that Secondary formats are used for COBOL II programs only.

PROCEDURE Enter the name of an existing procedure (20 characters maximum).

PROCEDURE ALIAS Enter the name by which the procedure is known within the location (8 characters maximum). If [[RETURN]] is pressed in response to this prompt, the alias becomes the first 8 characters of the procedure name.

VALUE AS A LINK Enter -1, 0, or a positive integer to specify the preferred elements to be used in linking files. The numbers indicate the following:

-1	The element can not be used for linking.
0	The element may or may not be used for linking (DEFAULT).
1 or more	The element should be used as a link when possible; elements assigned a positive value form a prioritized list to be used when it is necessary to link files (the lower the positive integer, the higher the priority).

Change

Changes the description of a relationship established with the RELATE command.

Syntax

```
< [option] CHANGE subcommand CHG
```

Use the CHANGE command to change the description of a relationship or to change the attributes of a child entity in the relationship established with the RELATE command. A complete list of attributes which can be changed is given below under Subcommands.

The subcommands are used to identify the type of parent entity that defines the relationship. For example, CHANGE ELEMENT can change the starting position of a child element within the parent element and/or edit the textual description of the relationship.

Subcommands

Any of the following subcommands can be used with the CHANGE command:

CATAGORY	changes a category to category description
CLASS	changes a class to class description
ELEMENT	changes an element to element entry and/or description
FILE	changes a file to file entry and/or description
GROUP	changes an HP Inform group to group description
PROCEDURE	changes a procedure to procedure description

Example

```
>CHANGE ELEMENT
         PARENT ELEMENT> Name              < Enter the parent's name.
           CHILD ELEMENT> First-name       < The child to be changed.
          EDIT DESCRIPTION(Y/N)? N           N to go directly to
                                             attributes.
   CHILD ALIAS:          POSITION:           Existing attribute information.
      STOCK1                  1
              CHILD ALIAS> Stock2          < Identifies new element alias.
            START POSITION> 4              < Identifies new position of
                                             child within parent's list.
           CHILD ELEMENT>                  < [[RETURN]] to terminate command.
     >                                       New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string. Two asterisks (**) next to a prompt indicate that this prompt and those which follow are repeated until [[RETURN]] is pressed in response to this prompt.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
CHANGE	CATAGORY	*	PARENT CATAGORY	
		**	CHILD CATAGORY	
		*	EDIT COMMAND>>	
CHANGE	CLASS	*	PARENT CLASS	
		**	CHILD CLASS	
		*	EDIT COMMAND>>	
CHANGE	ELEMENT	*	PARENT ELEMENT	
		**	CHILD ELEMENT	
			EDIT DESCRIPTION (Y/N)	
		*	EDIT COMMAND	Issued only if Ye to EDIT DESCRIPTION.
			EDIT ATTRIBUTES (Y/N)	Issued only if Yes to EDIT DESCRIPTION.
			CHILD ALIAS	
			START POSITION	
CHANGE	FILE	*	PARENT FILE	
		**	CHILD FILE	
			EDIT DESCRIPTION (Y/N)?	
		*	EDIT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES (Y/N)?	Issued only if Yes to EDIT DESCRIPTION.
			CHILD ALIAS	
			CAPACITY	Issued if PARENT file type is BASE.
			BLOCKMAX	Issued if PARENT file type is BASE.
CHANGE	GROUP	*	PARENT GROUP	
		**	CHILD GROUP	
		*	EDIT COMMAND>>	
CHANGE	PROCEDURE	*	PARENT PROCEDURE	
		**	CHILD PROCEDURE	
		*	EDIT COMMAND>>	

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

BLOCKMAX	Enter a new maximum physical block length value (in words) for the child file within the parent file (128 to 2048). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.
CAPACITY	Enter a new value for the maximum number of entries for the child file (1 to 999999999). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.
CHILD ALIAS	When the subcommand ELEMENT is used, enter a new element alias (16 characters maximum). This is the name by which this element is known within the child-to-parent element relationship. The child alias will be used by the DICTCDE and DICTPDE utilities. This prompt allows data items within records to have the same name but different types or definitions. When the subcommand FILE is used, enter a new file alias (16 characters maximum) for the child file. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.
CHILD CATEGORY	Enter the name of an existing category (20 characters maximum) that is the child category in the entry.
CHILD CLASS	Enter a numeric value which is the identifying number (0 to 9999) of an existing security class, which is not an INFO CLASS.
CHILD ELEMENT	Enter the name of an existing data element (20 characters maximum) that is the child in the entry.
CHILD FILE	Enter the name of an existing file (20 characters maximum) that is the child in the entry.
CHILD GROUP	Enter the name of an existing HP Inform group that is the child in the entry (20 characters maximum).
CHILD PROCEDURE	Enter the name of an existing procedure that is the child in the entry (20 characters maximum).
EDIT ATTRIBUTES (Y/N)	Enter an N if no prompts for attributes are to be generated. (see Subcommand Prompts for which attributes can be changed).
EDIT COMMAND>>	Enter one of the EDIT COMMANDS described in Section III.
EDIT DESCRIPTION (Y/N)	Enter an N if editing is not to be done for the textual description of the relationship.
PARENT CATEGORY	Enter the name of an existing parent category (20

	characters maximum).
PARENT CLASS	Enter a numeric value which is the identifying number of an existing INFO type security class (0 to 9999).
PARENT ELEMENT	Enter the name of an existing element that is the parent in the entry (20 characters maximum).
PARENT FILE	Enter the name of an existing file that is the parent in the entry (20 characters maximum).
PARENT GROUP	Enter the name of an existing HP Inform group that is the parent in the entry (20 characters maximum). Note that the HP Inform group $MENU can not be changed.
PARENT PROCEDURE	Enter the name of an existing procedure that is the parent in the entry (20 characters maximum).
START POSITION	Enter a positive integer which is the new byte position of the start of the child element's data field within the parent element (1 to 9999). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

Create

Creates a new entity.

Syntax

```
[option] CREATE subcommand C
```

Use the CREATE command to define a new entity in the Dictionary. A complete list of entities that can be created in the Dictionary is given below under Subcommands. The subcommand identifies the type of entity to be defined. For example, CREATE GROUP creates a new HP Inform group and CREATE ELEMENT creates a new element entry in the Dictionary.

An entity must be CREATED before it can be used with any other command.

Subcommands

Any of the following subcommands can be used with the CREATE command:

CATEGORY	creates a new category entry
CLASS	creates a new user security class entry
ELEMENT	creates a new data element entry
FILE	creates a new file entry
GROUP	creates a new HP Inform group entry
LOCATION	creates a new physical location entry
PROCEDURE	creates a new procedure entry

Example

```
>CREATE GROUP
            GROUP> Customer Information&lt;       HP Inform group.
       LONG NAME> Customer Information-Marketing              <
            TYPE> MRKT                           User defined type.
  RESPONSIBILITY> Order Processing     <
     DESCRIPTION> This group will contain detail          <
                > information about Marketing's customers. <
                >                                        <
                                      [[RETURN]] to end
                                      command.
                                      New command prompt.
  >
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
CREATE	CATAGORY	*	CATAGORY	
			LONG NAME	
			TYPE	
			RESPONSIBILITY	
			DESCRIPTION	
CREATE	CLASS	*	CLASS	
			NAME	
			TYPE	
			PASSWORD	
			RESPONSIBILITY	
			RESTRICT CLASS TO A FILE (N/Y)	Issued only f CLASS type is not INFO.
			FILE	* If Yes to RESTRICT CLASS; file type must be BASE.
			DESCRIPTION	
CREATE	ELEMENT	*	ELEMENT	
			LONG NAME	
			TYPE	
			SIGN POSITION	Issued only if TYPE Z.
		*	ELEMENT REFERENCE	Issued only if TYPE “*”
		*	SIZE	Issued only if TYPE specified and TYPE in not B or “*”.
			DECIMAL	Issued only if TYPE specified and TYPE is numeric.
			STORAGE LENGTH(n)	Issued only if TYPE specified and TLYPE in not “*”, B or S.
			COUNT(1)	Issued only if TYPE specified.
			HEADING TEXT	
			ENTRY TEXT	
			EDIT MASK	
			MEASUREMENT UNITS	
			BLANK WHEN ZERO (N/Y)?	Issued only if TYPE Z or 9.

COMMAND	SUB-COMMAND		PROMPT	
			RIGHT USTIFY (N/Y)?	Issued only if TYPEX or U.
			SYNCHRONIZED (N/Y)?	Issued only if TYPE I, J, or K.
			RESPONSIBIILTY	
			DESCRIPTION	
CREATE	FILE	*	FILE	
			LONG NAME	
		*	TYPE	
			RESPONSIBILITY	
			DESCRIPTION	
			ADDITIONAL FILE ATTRIBUTES (N/Y)?	Issued only if file type is KSAM, MPEF or MPER.
			RECORD FORMAT (F/V/U/S)?	Issued only if file type is KSAM, MPEF or MPER.
			MINIMUM RECORD SIZE	Issued only if file type is KSAM, MPEF or MPER.
			BLOCKING (N/Y)?	Issued only if file type is KSAM, MPEF or MPER.
			UNIT (R/C)?	Issued if YES to BLOCKING.
			MINIMUM BLOCKNG	Issued if YES to BLOCKING.
			MAXIMUM BLOCKING	Issued if YES to BLOCKING.
			RECORDING MODE (A/B)	Issued only if file type is MPEF or MPER.
			DATA STORAGE TYPE (A/E)?	Issued only if file type is MPEF.
			CCTL (N/Y)?	Issued only if file type is MPEF.
			DEVICE CLASS (DA/UR/UT)?	Issued only if file type is MPEF.
CREATE	GROUP	*	GROUP	
			LONG NAME	
			TYPE	
			RESPONSIBILITY	
			DESCRIPTION	
CREATE	LOCATION	*	LOCATION	
			LONG NAME	
			GROUP	
			ACCOUNT	
			CPU	
			DESCRIPTION	

COMMAND	SUB-COMMAND		PROMPT
CREATE	PROCEDURE	*	PROCEDURE
			LONG NAME
			TYPE
			LANGUAGE
			RESPONSIBILITY
			DESCRIPTION

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

ACCOUNT — Enter the name for the MPE account associated with the location (8 characters maximum).

ADDITIONAL FILE ATTRIBUTES (N/Y)? — Enter Y or Yes to generate additional prompts ex which allow you to specify additional attributes for this file. These prompts generate data that will be used by the DICTCDE utility. DEFAULT is No.

BLANK WHEN ZERO (N/Y)? — This prompt generates data that will be used by the DICTCDE utility. Enter Y if the element is to be displayed as a blank when its value is equal to 0. DEFAULT is No.

BLOCKING (N/Y)? — This prompt generates data that can be used by the DICTCDE utility. Enter Y to specify the blocking factor of the file. DEFAULT is No.

CATEGORY — Enter the name for a new category (20 characters maximum).

CCTL(N/Y)? — This prompt generates data that will be used by the DICTCDE utility. Enter Y if the carriage control option for an output file is to be specified. DEFAULT is No.

CLASS — Enter a numeric value for the identity of a new security class (0 to 9999).

COUNT(1) — Enter the sub-element count if the data element is a compound element (1 to 9999). DEFAULT is 1.

CPU — Enter the identity of the machine associated with the location (8 characters maximum).

DATA STORAGE TYPE (A/E)? — This prompt generates data that will be used by the DICTCDE utility. Enter the following to specify the character code that represents how data is stored in the file:

A	ASCII character codes
E	EBCDIC character codes

DEFAULT is A.

DECIMAL Enter the integer value for the number of digits to the right of the decimal point (0 to 9999). Do not include a position for the decimal point itself. The largest value for each element type is as follows:

For types Z, P, and 9, the largest value is 27 digits.

For types I, J, and K, the largest value is 28 decimal digits.

For types E and R, the largest value is 27 digits without a decimal and 28 digits with a decimal.

This prompt is only issued for numeric type data elements. Note if the maximum value for SIZE is entered, a decimal value is required. DEFAULT is 0.

DESCRIPTION Enter a textual description which describes the entity (50 characters per description line allowed, unlimited lines).

DEVICE This prompt generates data that will be used by the DICTCDE utility. Enter the type of user-specified device in which the file resides (8 characters maximum). If [[RETURN]] is pressed in response to this prompt, then the MPE default device will be assumed.

DEVICE CLASS (DA/UR/UT)? This prompt generates data that will be used for the DICTCDE utility. Enter the class of device on which the file resides as follows:

DA	mass storage device
UT	utility device (such as a tape drive)
UR	unit record device (such as a card reader)

DEFAULT is DA.

EDIT MASK Enter a textual edit mask to be used by HP Inform, Report, and Transact to edit values when displayed for the data element (30 characters maximum). (Refer to Using Edit Masks in Section III.)

ELEMENT Enter the name for a new data element (20 characters maximum).

ELEMENT REFERENCE This prompt generates data that will be used by the DICTCDE and DICTPDE utilities. Enter the name of an existing element which is to be used as a back-reference. The new element to be created will have the same type, size, storage length, and sub-element count (if it is a compound element) as this element. If this element is a numeric type, the new element will also have the same number of digits to the right of the decimal.

ENTRY TEXT Enter textual information to be used by HP Inform, Report, and Transact as a prompt for input. Otherwise, the primary name is used in the prompt (30 characters maximum).

Note that the primary name will be used in HP Inform's and Report's SELECTION CRITERIA prompt unless an ENTRY TEXT has been specified.

FILE — If the subcommand is:

- FILE — enter the name for a new file (20 characters maximum). CAUTION: A base name can be up to 20 characters in the Dictionary. Dictionary accepts the full name such as BASE.GROUP.ACCOUNT as a valid name. DICTDBC, however, always uses the first 6 characters of the name as the base name. When DICTDBC encounters a base name in the Dictionary such as BASE.GROUP.ACCOUNT, your base name is created as "BASE.G" which is an invalid base name. To specify the physical location of the file, use the CREATE LOCATION and the ADD FILE-LOCATION commands.
- CLASS — enter the name of an existing file of type BASE (20 characters maximum).

GROUP — If the subcommand is:

- GROUP — enter the name for a new HP Inform group (20 characters maximum).
- LOCATION — enter the name of the MPE group associated with the location (8 characters maximum).

HEADING TEXT — Enter a textual heading to be used by HP Inform, Report, and Transact for reports. Otherwise the primary name is used in the heading (30 characters maximum).

Note that the primary name will be used in the report heading by HP Inform and Report unless a HEADING TEXT has been specified.

LANGUAGE — Enter the name of the implementation language for the procedure (10 characters maximum).

LOCATION — Enter the name of a physical location that identifies the MPE file group and account, and the machine where files and procedures reside (20 characters maximum).

LONG NAME — Enter a full textual description for the entity (50 characters maximum).

MAXIMUM BLOCKING — This prompt generates data that will be used by the DICTCDE utility. Enter the maximum blocking factor for the file. DEFAULT: is 0.

MAXIMUM RECORD SIZE — This prompt generates data that will be used by the DICTCDE utility. Enter the maximum number of characters in the record. DEFAULT is 0.

MEASUREMENT UNITS — Enter the type of measurement units (such as feet, dollars) used for the data element's value (10 characters maximum).

MINIMUM BLOCKING — This prompt generates data that will be used by the DICTCDE utility. Enter the minimum blocking factor for the file. DEFAULT is 0.

MINIMUM RECORD SIZE — This prompt generates data that will be used by the DICTCDE utility. Enter the minimum number of characters in the record. DEFAULT is 0.

NAME — Enter a textual name for the new security class (50 characters maximum).

PASSWORD — Enter a password for the new security class (8 characters maximum).

PROCEDURE — Enter the name for a new procedure (20 characters maximum).

RECORD FORMAT (F/V/U/S) — This prompt generates data that will be used by the DICTCDE utility. RECORD FORMAT specifies how the logical records are contained in the file. Enter:

F	for fixed length logical records
V	for variable length logical records
U	for undefined length logical records
S	for the multirecord or “multiblock” option

RECORDING MODE(A/B)? — This prompt generates data that will be used by the DICTCDE utility. Enter A if the recording mode of the file is ASCII. Enter B if the recording mode of the file is BINARY. DEFAULT is A.

RESPONSIBILITY — Enter the name of the person, department, or area responsible for the integrity of the new entity (20 characters maximum).

RESTRICT CLASS TO A FILE (Y/N)? — Enter a Y if the scope of the class is to be restricted to one file. A security class can be restricted to files of type BASE.

RIGHT JUSTIFY (N/Y)? — This prompt generates data that can be used by the DICTCDE utility. Enter Y or Yes to right justify alphabetic or alphanumeric data elements. DEFAULT is No.

SIGN POSITION — This prompt generates data that will be used by the DICTCDE utility. Enter the position of the sign for a signed numeric data element as follows:

LO	leading overpunched
LS	leading separate
TO	trailing overpunched
TS	trailing separate

DEFAULT: Pressing [[RETURN]] in response to this prompt indicates that there is no sign.

SIZE — Enter an integer value required to display an element (1 to 9999). Include a position for a decimal point if there is one, but do not include one for a sign. The largest value for each element type is as follows:

For type Z, P, and 9, the largest value is 28 digits.

For type I, J, and k, the largest value is 29 digits.

For type E and R, the largest value is 27 digits without a decimal and 28 digits with a decimal.

For type X and U, the largest value is 9999 digits.

STORAGE LENGTH (n) The system calculates the storage length in bytes and displays it as “n”. An over-ride value may be entered (1 to 9999), otherwise “n” is used.

SYNCHRONIZED (N/Y)? This prompt generates data that will be used by the DICTCDE utility. Enter Y or Yes to align data elements in order to facilitate arithmetic operations. DEFAULT is No.

TYPE If the subcommand is:

ELEMENT - enter one of the following values:

U	upper case ASCII
X	upper or lower case ASCII
9	numeric ASCII
Z	zoned decimal
P	packed decimal
I	integer binary
J	integer binary (COBOL)
K	logical
R	floating point (commercial)
E	floating point (E format)
B	boolean (PASCAL)
S	string (PASCAL)
*	back reference to an existing element (PASCAL)

If the type is numeric, a plus character (+) may immediately follow it to indicate that the data element can have only positive values. If a plus character is entered with a type other than numeric, it is ignored. Pressing [[RETURN]] in response to this prompt indicates that the data element does not have a type assigned.

FILE - enter one of the following values:

BASE	IMAGE database
MAST	IMAGE manual master data set
AUTO	IMAGE automatic master data set
DETL	IMAGE detail data set

KSAM	KSAM file
MPEF	MPE sequential file
MPER	MPE relative file
VPLS	VPLUS forms file
FORM	VPLUS form

CATEGORY, CLASS, GROUP, or PROCEDURE - enter up to four characters meaningful to you. Or, if this is a CLASS to be used for INFORM security, enter INFO as the class type.

UNIT(R/C)? This prompt generates data that will be used by the DICTCDE utility. Specify the unit used to determine the blocking factor with: R for RECORDS or C for CHARACTERS. DEFAULT is R.

Delete

Deletes an association which was established with the ADD command.

Syntax

```
[option] DELETE subcommand DEL
```

Use the DELETE command to delete an association between unlike entities established with the ADD command. Only the association is deleted, not the entities. A complete list of associations that can be deleted is given below under Subcommands.

The subcommands are used to identify the entity from which an association is to be deleted. For example, DELETE CLASS deletes an association of an element to a security class.

Subcommands

Any of the following subcommands can be used with the DELETE command:

CATEGORY	deletes an element to category association
CLASS	deletes an element to class association
CLASS-FILE	deletes a file to class association
CLASS-GROUP	deletes a group to class association
FILE	deletes an element to file association
FILE-LOC	deletes a file to location association
GROUP	deletes an element to HP Inform group association
PROCEDURE	deletes an element to procedure association
PROCEDURE-LOC	deletes a procedure to location association

Example

```
>DELETE FILE
            FILE> Empnum          File used in the
                                  association.
PRIMARY/SECONDARY(P/S)?&gt;P      Indicates element is in
                                  the Primary format.
          ELEMENT> Name         < Element to be deleted
                                  from the association.
ENTRY DELETED
            ELEMENT>              [[RETURN]] to terminate the
                                  command.
>                                 New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string. Two asterisks (**) next to a prompt indicate that this prompt and those which follow are repeated until [[RETURN]] is pressed in response to this prompt.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
DELETE	CATAGORY	*	CATAGORY	
		**	ELEMENT	
DELETE	CLASS	*	CLASS	
		**	ELEMENT	
DELETE	CLASS-FILE	*	CLASS	
		**	FILE	
DELETE	FILE	*	FILE	
		**	PRIMARY/SECONDARY (P/S)?	Issued if file type is MPEF, MPER, or KSAM.
		**	ELEMENT	
DELETE	C;ASS-GROUP	*	CLASS	
		**	GROUP	
DELETE	FILE-LOC	*	LOCATION	
		**	FILE	
DELETE	GROUP	*	GROUP	
		**	ELEMENT	
DELETE	PROCEDURE	*	PROCEDURE	
		**	ELEMENT	
DELETE	PROCEDURE-LOC	*	LOCATION	
		**	PROCEDURE	

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

CATEGORY Enter the name of an existing category used in the association (20 characters maximum).

CLASS Enter the identifying number of an existing security class used in the association (1 to 9999).

If the subcommand is CLASS-GROUP, enter the identifying number of an existing INFO type security class used in the association (1 to 9999).

ELEMENT Enter the name of an existing data element which is to be deleted from the

association (20 characters maximum).

FILE — If the subcommand is:

FILE	enter the name of an existing file used in the association (20 characters maximum).
FILE-LOC or CLASS-FILE	enter the name of an existing file which is to be deleted from the association (20 characters maximum).

GROUP — Enter the name of an existing HP Inform group used in the association (20 characters maximum).

LOCATION — Enter the name of an existing physical location used in the association (20 characters maximum).

PRIMARY/SECONDARY (P/S)? — Enter P if the element is associated to the Primary format for this file. Enter S if the element is associated to the Secondary format for the file. DEFAULT is P. (See the discussion of the ADD command for more information on Primary and Secondary formats.)

PROCEDURE — If the subcommand is:

PROCEDURE	enter the name of an existing procedure used in the association (20 characters maximum).
PROCEDURE -LOC	enter the name of an existing procedure which is to be deleted from the association (20 characters maximum).

Display

Displays information about an entity established with the CREATE, ADD, and/or RELATE commands.

Syntax

```
[option] DISPLAY subcommand D
```

Use the DISPLAY command to display the attributes of an entity and, if applicable, the direct relationships and the direct associations of that entity.

A direct relationship is the relationship between the displayed entity and its parent entity. A direct association is the association between the displayed entity and the entity that owns it (the entity with which the displayed entity is associated).

The subcommand identifies the entity to be displayed. If ELEMENT, FILE, or PROCEDURE is used as the subcommand, then the attribute, relationship, and association information for that entity is displayed. If CATEGORY, CLASS, or GROUP is used as the subcommand, the attribute and relationship information for that entity is displayed. If LOCATION is used as the subcommand, only the attribute information for that entity is displayed. A complete list of entities that can be displayed is given below under Subcommands. Selection Criteria can also be used to select the entity or entities to be displayed. Refer to Using Selection Criteria in Section III for more information.

To display the attribute information and, if applicable, the direct relationships and direct associations for all the entities for a subcommand, enter an exclamation mark (!) or press [[RETURN]] in response to the prompt. (Only 1 prompt is issued for each subcommand used.) Note that the exclamation mark and [[RETURN]] are used differently for the DISPLAY command than for most commands. The PRINT option can be used with the command string to print more descriptive information on the line printer in addition to the information normally displayed on the terminal. This information includes the information that was entered when the relationship or association was made.

Subcommands

Any of the following subcommands can be used with the DISPLAY command:

CATEGORY	displays attribute information for a category plus all the directly related categories
CLASS	displays attribute information for a user security class plus the directly related classes
ELEMENT	displays attribute information for an element plus the directly associated and/or related entities
FILE	displays attribute information for a file plus the directly associated and/or directly related entities
GROUP	displays attribute information for an HP Inform group plus the directly related entities and any associated classes

LOCATION	displays attribute information for a physical location
PROCEDURE	displays attribute information for a procedure plus the directly associated and/or directly related entities

Example

```
> DISPLAY ELEMENT
                  ELEMENT>; ACCOUNT                                    >
     ELEMENT:             TYPE: SIZE: DEC: LENGTH: COUNT: RESPONSIBILITY:
      ACCOUNT              X     10    0    10      1
             LONG NAME: MARKETING CUSTOMER ACCOUNT NUMBERS
           HEADING TEXT: CUSTOMER ACCOUNT NUMBERS
             ENTRY TEXT: CUSTOMER ACCOUNT NUMBER
              EDIT MASK:
      MEASUREMENT UNITS:
           RIGHT JUSTIFY:
           DATE CREATED: 83/06/01 BY MGR
           DATE CHANGED: 83/06/06 BY MGR
    DESCRIPTION:
      THIS ACCOUNT NUMBER IDENTIFIES THE CUSTOMERS
      WHO BUY THE LARGE INTEGRATED SYSTEMS.
       ELEMENTS THAT BACK REFERENCE ACCOUNT:
                REPORT1
                ACCTSTAT
       PRIMARY FORMAT ASSOCIATIONS:
         FILE:               TYPE:   ELEMENT(ALIAS):      FIELD NUMBER:
         SALES               MAST    ACCOUNT
                                     *KEY ELEMENT
    1 RECORD FOUND
    >
```

Subcommand Prompts

Enter the name for a specific entity or use SELECTION CRITERIA in response to a prompt. (Refer to Using Selection Criteria in Section III for more information.) Responding with an exclamation mark (!) or with [[RETURN]] will display information about all entities for the specified subcommand.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT
DELETE	CATAGORY	*	CATAGORY
DISPLAY	CATAGORY		CATAGORY
DISPLAY	CLASS		CLASS
DISPLAY	ELEMENT		ELEMENT
DISPLAY	FILE		FILE
DISPLAY	GROUP		GROUP
DISPLAY	LOCATION		LOCATION

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

CATEGORY	Enter the name of an existing category or Selection Criteria for which information is to be displayed.
CLASS	Enter the identifying number of an existing security class or Selection Criteria for which information is to be displayed.
ELEMENT	Enter the name of an existing element or Selection Criteria for which information is to be displayed.
FILE	Enter the name of an existing file or Selection Criteria for which information is to be displayed.
GROUP	Enter the name of an existing HP Inform group or Selection Criteria for which information is to be displayed.
LOCATION	Enter the name of an existing physical location or Selection Criteria for which information is to be displayed.
PROCEDURE	Enter the name of an existing procedure or Selection Criteria for which information is to be displayed.

Help

Provides a brief description of each DICTDBM command and subcommand.

Syntax

```
[option] HELP [subcommand]              H
```

Use the HELP command to obtain information on:

- all DICTDBM commands
- any DICTDBM command and its subcommands.

To review descriptions of all DICTDBM commands, use the HELP command without a subcommand.

To obtain a description of a specific DICTDBM command and its subcommands, use the HELP command with one of the following subcommands listed below.

All options except REPEAT can be used with this command.

Subcommands

Any of the following subcommands can be used with the HELP command:

ADD	LIST	REORDER
CHANGE	MODIFY	REPORT
CREATE	PURGE	RESEQUENCE
DELETE	RELATE	SECURE
DISPLAY	REMOVE	SHOW
EXIT	RENAME	UPDATE

Example

```
> HELP
```

ADD	Add an association between unlike entities
CHANGE	Change entry/description of relationship established with RELATE
CREATE	Create a new entity
DELETE	Delete association which was established with the ADD command
DISPLAY	Display info about entity established with CREATE, ADD, RELATE
EXIT	Terminate DICTDBM
HELP	Display information about the commands. Enter HELP command-name

LIST	List information for entity/entities with subset of attributes
MODIFY	Modify attributes and/or description of an entity
PURGE	Purge entity and all references to it
RELATE	Establish hierarchical relationship between like entities
REMOVE	Remove relationships between entities established with RELATE
RENAME	Rename an existing entity
REORDER	Reposition entity within relationship established with RELATE
REPORT	Report a sorted listing of elements
RESEQUENCE	Reposition element within association established with ADD
SECURE	Assign all elements of an IMAGE type file to a security class
SHOW	Show entity's association and relationships with child entities
UPDATE	Update association between entities established with ADD

```
> HELP ADD
```

ADD		Add an association between unlike entities
	CATEGORY	Add an element to a category
	CLASS	Add an element to a user security class
	CLASS-FILE	Add a file to a user security class
	CLASS-GROUP	Add a group to a user security class
	FILE	Add an element to a file
	FILE-LOC	Add a file to a physical location
	GROUP	Add an element to an INFORM group
	PROCEDURE	Add an element to a procedure
	PROCEDURE-LOC	Add a procedure to a physical location

List

Lists information for an entity or entities with a subset of the attributes.

Syntax

```
[option] LIST subcommand L
```

Use the LIST command to see the attributes of:

- a particular entity
- all entities of a particular type
- entities selected by using Selection Criteria (see Section III)

A complete list of entities for which a listing can be generated is given under Subcommands. The subcommand identifies the type of entity for which a listing is generated. The listing is displayed in the order in which the entities were entered. After the listing, the number of records displayed is given.

To list one entity, specify the entity's name in response to the first prompt. To list all of the entities for a specific subcommand, press the [[RETURN]] key for all the prompts or enter an exclamation character (!) in response to the first prompt. (Note that this is a different use of the [[RETURN]] key and of the exclamation mark character than for most commands.) The Selection Criteria can also be used to generate a listing. (See Section III for more detailed information.)

The PRINT option can be used with the command string to print the listing and additional information on the line printer. Along with the information normally displayed, the listing will include the description entered when the entity was created.

Subcommands

Any of the following subcommands can be used with the LIST command:

CATEGORY	lists the categories
CLASS	lists the security classes
ELEMENT	lists the data elements
FILE	lists the files
GROUP	lists the HP Inform groups
LOCATION	lists the physical locations PROCEDURE lists the procedures

Example

```
> REPEAT LIST ELEMENT
                    ELEMENT> PU^                                  <
                       TYPE> !                                    <
    ELEMENT:              TYPE: SIZE: DEC: LENGTH: COUNT: RESPONSIBILITY:
   PUB-DATE               X     6     0    6       1     M. Sides
      PURCH-DATE          X     6     0    6       1     M. Abear
      PURCH-NO            U     6     0    6       1     R. Park
      PUBLISHER           X     40    0    40      1     M. Sides
    4 RECORDS FOUND
                    ELEMENT> ^T                                   <
                       TYPE> !                                    <
    ELEMENT:              TYPE: SIZE: DEC: LENGTH: COUNT: RESPONSIBILITY:
      CREDIT              U     2     0    2       1     J. Smith
      SUBJECT             X     40    0    40      1     M. Sides
      SQUARE-FEET         X     8     0    8       1     R. Park
      ACCOUNT             X     10    0    10      1     M. Abear
      UNIT-COST           P     11    0    6       1     M. Abear
      EXISTING-STRUCT     U     2     0    2       1     J. Smith
      LOAN-AMT            P     11    2    6       1     M. Abear
    7 RECORDS FOUND
                   ELEMENT> ]                                     <
    >
```

Subcommand Prompts

You can enter specific information for any of the prompts or use SELECTION CRITERIA in response to a prompt. (Refer to Using Selection Criteria in Section III.) Responding with [[RETURN]] to all the prompts or with an exclamation mark (!) to the first prompt lists information about all entities for the specified subcommand.

You enter: You are prompted for:

COMMAND	SUB-COMMAND	PROMPT	
LIST	CATEGORY	CATEGORY	
		TYPE	
		RESPONSIBILITY	
LIST	CLASS	CLASS	
		TYPE	
		RESPONSIBILITY	
LIST	ELEMENT	ELEMENT	
		TYPE	
		ELEMENT REFERENCE	Issued only if TYPE is “*”.
		SIZE	Issued if TYPE is not “*”.
		DECIMAL	Issued only if TYPE is numeric and TYPE is not “*”.
		STORAGE LENGTH	Issued if TYPE is not “*”.
		COUNT	Issued if TYPE is not “*”.
		RESPONSIBILITY	Issued if TYPE is not “*”.
LIST	FILE	FILE	
		TYPE	
		RESPONSIBILITY	
LIST	GROUP	GROUP	
		TYPE	
		RESPONSIBILITY	
LIST	LOCATION	LOCATION	
		GROUP	
		ACCOUNT	
		C[I	
LIST	PROCEDURE	PROCEDURE	
		TYPE	
		LANGUAGE	
		RESPONSIBILITY	

Prompts

Depending on the subcommand used, one or more of the following prompts is issued.

ACCOUNT — Enter the name of the existing MPE account or Selection Criteria for which information is to be listed.

CATEGORY — Enter the name of the existing category or Selection Criteria for which information is to be listed.

CLASS — Enter the number of the existing security class or Selection Criteria for which information is to be listed.

COUNT — Enter the sub-element count if the data element is a compound element or Selection Criteria for which information is to be listed.

CPU — Enter the name of the machine or Selection Criteria for which information is to be listed.

DECIMAL — Enter the integer value for the number of digits to the right of the decimal point (0 to 9999) or Selection Criteria for which information is to be displayed. Do not include a position for the decimal point itself. The largest value for each element type is as follows:

For types Z, P, and 9, the largest value is 27 digits.

For types I, J, and K, the largest value is 28 digits.

For types E and R, the largest value is 9999 decimal digits.

This prompt is only issued for numeric type data elements. Note that if the maximum value for SIZE is entered, a decimal value is required.

ELEMENT — Enter the name of the existing data element or Selection Criteria for which information is to be listed.

ELEMENT REFERENCE — Enter the name of an existing element (20 characters maximum) which is to be used as the back-referenced element. The element to be listed has the same type, size, storage length, and sub-element count, and the same decimal placement (if applicable) as the back-referenced element.

FILE — Enter the name of the existing file or Selection Criteria for which information is to be listed.

GROUP — If the subcommand is:

- GROUP — enter the name of an existing HP Inform group or Selection Criteria for which information is to be listed.
- LOCATION — enter the name of an existing MPE group or Selection Criteria for which information is to be listed.

LANGUAGE — Enter the name of the implementation language or Selection Criteria for which information is to be listed.

LOCATION — Enter the name of the physical location or Selection Criteria for which information is to be listed.

PROCEDURE Enter the name of the existing procedure or Selection Criteria for which information is to be listed.

RESPONSIBIILTY Enter the name of the person, department, or area, or Selection Criteria for which information is to be listed.

SIZE Enter an integer value required to display an element (1 to 9999) or Selection Criteria for which information is to be listed. Include a position for a decimal point if there is one, but do not include one for a sign. The largest value for each element type is as follows:

For types Z, P, and 9, the largest value is 28 digits.

For types I, J, and K, the largest value is 29 digits.

For types E, R, X, and U, the largest value is 9999 digits.

STORAGE LENGTH Enter the storage length in bytes or Selection Criteria for which information is to be displayed.

TYPE If the subcommand is:

ELEMENT enter the attribute type or Selection Criteria for which information is to be listed.

FILE enter the file type or Selection Criteria for which information is to be listed.

CATEGORY, CLASS, GROUP, PROCEDURE - enter the character value or Selection Criteria for which information is to be listed.

Modify

Modifies the information for an entity which was created with the CREATE command.

Syntax

```
[option] MODIFY subcommand M
```

Use the MODIFY command to change the attributes and/or the description of an entity created with the CREATE command. The edit description can also be modified with the prompt EDIT COMMAND. Refer to Using Edit Description Commands in Section III for a complete list of EDIT COMMANDs.

The subcommand identifies the type of entity to be modified. For example, MODIFY CATEGORY, allows you to enter a new type designator, change the name of the person responsible, or enter a new textual name for the category. It also allows you to edit the textual description for the category. A complete list of entities that can be modified is given below under Subcommands.

Subcommands

Any of the following subcommands can be used with the MODIFY command:

CATEGORY	modifies a category
CLASS	modifies a security class
ELEMENT	modifies a data element
FILE	modifies a file
GROUP	modifies an HP Inform group
LOCATION	modifies a physical location
PROCEDURE	modifies a procedure

Example

```
>MODIFY CATEGORY
             CATEGORY> Marketing  <          Category to be modified.
     EDIT DESCRIPTION(Y/N)? n                No, to edit description.
     CATEGORY:           TYPE: RESPONSIBILITY:
     Marketing           SPLS  John Williams
            LONG NAME:
                    TYPE> Attribute prompt.
       RESPONSIBILITY> James Max                Changed name.
            LONG NAME> Marketing Supplies Sales  Added information.
    >                                            New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
MODIFY	CATEGORY	*	CATEGORY	
			EDIT DESCRIPTION (Y/N)?	
			EDIT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			TYPE	
			RESPONSIBILITY	
			LONG NAME	
MODIFY	CLASS	*	CLASS	
			EDIT DESCRIPTION (Y/N)?	
		*	EDIT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES (Y/N)?	Issued only if Yes to EDIT DESCRIPTION.
			TYPE	Issued only if CLASS type is not INFO or if CLASS type is INFO but the file is not associated or related to any other entity.
			PASSWORD	
			RESPONSIBILITY	
			LONG NAME	
			RESTRICTED FILE	Issued only if CLASS type is not INFO.
MODIFY	ELEMENT	*	ELEMENT	
			EDIT DESCRIPTION (Y/N)?	
		*	EDIT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES (Y/N)?	Issued only if Yes to EDIT DESCRIPTION.
			TYPE	
			SIGN POSITION	Issued only if TYPE Z.
			ELEMENT REERENCE	Issued only if TYPE is “*”.
			SIZE	Issued if TYPE is not “*”, B or S.
			COUNT	
			RESPONSIBILITY	

COMMAND	SUB-COMMAND		PROMPT	
			LONG NAME	
			HEADING TEXT	
			ENTRY TEXT	
			EDIT MASK	
			MEASUREMENT UNITS	
			BLANK WHEN ZERO (N/Y	Issued if TYPE is Z or 9.
			RIGHT JUSTIFY (N/Y)?	Issued if TYPE is Z or U.
			SYNCHRONIZED (N/Y)?	Issued if TYPE is I, J, or E.
MODIFY	FILE	*	FILE	
			EDIT DESCRIPTION (Y/N)?	
		*	EDIT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES (N/Y)?	Issued only if Yes to EDIT DESCRIPTION.
			TYPE	Issued only if the file has not be associated or related to any other entity.
			RESPONSIBILITY	
			LONG NAME	
			RECORD FORMAT (F/V/U/S)?	Issued only if file type is KSAM, MPEF, or MPER.
			MINIMUM RECORD SIZE	Issued only if TYPE is KSAM, MPEF or MPER and RECORD FORMAT is V or U.
			MAXIMUM RECORD SIZE	Issued only if file type is KSAM, MPEF, or MPER.
			MODIFY BLOCK (N/Y)?	Issued only if file type is KSAM, MPEF, or MPER.
			UNIT (R/C)?	Issued if YES to BLOCKING.
			BLOCK MINIMUM	Issued if YES to BLOCKING.
			BLOCK MAXIMUM	Issued if YES to BLOCKING.
			RECORDING MODE (A/B)?	Issued only if file type is MPEF or MPER.
			DATA STORAGE TYPE (A/E)?	Issued only if file type is MPEF.
			CCTL (N/Y)?	Issued only if file type is MPEF.
			DEVICE	Issued only if file type is MPEF.
			DEVICE CLASS (DA/UB/UT)?	Issued only if file type is MPEF.
MODIFY	GROUP	*	GROUP	

COMMAND	SUB-COMMAND		PROMPT	
			EDIT DESCRIPTION (Y/N)?	
		*	EDIT COMMAND>>	Issued only if file type is EDIT DESCRIPTION.
			EDIT ATTRIBUTES (Y/N)?	Issued only if file type is DESCRIPTION.
			TYPE	
			RESPONSIBILITY	
			LONG NAME	
MODIFY	LOCATION	*	LOCATION	
			EDIT DESCRIPTION (Y/N)?	
			EDIT COMMAND>>	Issued only if file type is EDIT DESCRIPTION.
			EDIT ATTRIBUTES (N/Y)?	Issued only if file type is EDIT DESCRIPTION.
			GROUP	
			ACCOUNT	
			CPU	
			LONG NAME	
MODIFY	PROCEDURE	*	PROCEDURE	
			EDIT DESCRIPTION (Y/N)?	
			EDIT COMMAND	Issued only if file type is EDIT DESCRIPTION.
			EDIT ATTRIBUTES (Y/N)?	Issued only if file type is EDIT DESCRIPTION.
			TYPE	
			LANGUAGE	
			RESPONSIBILITY	
			LONG NAME	

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

ACCOUNT — Enter a new name for the MPE account that is associated with the location (8 characters maximum).

DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

BLANK WHEN ZERO(N/Y)? — This prompt generates data that will be used by the DICTCDE utility. Enter Y if the element is to be displayed as a blank when its value is equal

to 0. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

CATEGORY — Enter the name of an existing category (20 characters maximum).

CCTL(N/Y) — This prompt generates data that will be used by the DICTCDE utility. Enter Y if the carriage control option for an output file is to be specified. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates that no change is to be made.

CLASS — Enter the identifying number of an existing security class (0 to 9999).

COUNT — Enter a positive integer value for a new sub-element count if the data element is a compound type (1 to 9999). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

CPU — Enter the new name of the machine that is associated with the location (8 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

DATA STORAGE TYPE (A/E)? — This prompt generates data that will be used by the DICTCDE utility. Enter one of the following to change the character code that represents how data is stored in the file:

A	ASCII character codes
E	EBCDIC character codes

DEFAULT: pressing [[RETURN]] indicates no change.

DECIMAL — Enter a new integer value for the number of digits to the right of the decimal point (0 to 9999). Do not include a position for the decimal point itself. The largest value for each element type is as follows:

For types Z, P, and 9, the largest value is 27 decimal digits.

For types I, J, and K, the largest value is 28 decimal digits.

For types E and R, the largest value is 9999 decimal digits.

Note that if the maximum value for SIZE is entered, a decimal value is required. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

DEVICE — This prompt generates data that will be used by the DICTCDE utility. Enter the type of user specified device on which the file resides (8 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates that no change is to be made.

DEVICE CLASS (DA/UR/UT)? This prompt generates data that will be used by the DICTCDE utility. Enter the class of device on which the file resides as follows:

DA	mass storage device
UT	UT utility device (such as a tape drive)
UR	unit record device (such as a card reader)

DEFAULT: Pressing [[RETURN]] indicates no change is to be made.

EDIT ATTRIBUTES (Y/N)? Enter an N if attribute prompts are not to be issued.

EDIT COMMAND>> Enter one of the EDIT COMMANDS described under Using Edit Description Commands in Section III.

EDIT DESCRIPTION (Y/N)? Enter an N if description text editing is not required.

EDIT MASK Enter a new textual edit mask to be used to edit values when displayed for the data element (30 characters). This is used by HP Inform, Report, and Transact. (Refer to Using Edit Masks in Section III.) DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

ELEMENT Enter the name of an existing data element (20 characters maximum).

ELEMENT REFERENCE This prompt generates data that will be used by the DICTPDE utility. Enter the name of an existing element that is being back-referenced. The element being modified will have the same type, size, storage length, sub-element count and decimal position of the ELEMENT REFERENCE. DEFAULT: Pressing [[RETURN]] in response to this prompts indicates the ELEMENT REFERENCE does not change.

ENTRY TEXT Enter new textual information to be used by HP Inform, Report, and Transact as a prompt for input. This text replaces the primary name in the prompt (30 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

FILE Enter the name of an existing file (20 characters maximum).

GROUP If the subcommand is:

- GROUP — enter the name of an existing HP Inform group (20 characters maximum). NOTE: $MENU can not be modified.
- LOCATION — enter the name of a new MPE group to be associated with the location (8 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

HEADING TEXT Enter a new textual heading to be used by HP Inform, Report, and Transact for the heading in reports. This heading replaces the data element's primary name in reports (30 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

LANGUAGE Enter the name of a new implementation language for the procedure (10 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

LOCATION Enter the name of an existing physical location (20 characters maximum).

LONG NAME Enter a new, full textual name for the entity (50 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

MAXIMUM BLOCKING This prompt generates data that will be used by the DICTCDE utility. Enter the maximum blocking factor for the file. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

MAXIMUM RECORD SIZE This prompt generates data that will be used by the DICTCDE utility. Enter the maximum number of characters in the record. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

MEASUREMENT UNITS Enter a new type of measurement units (such as feet, dollars) for the data element's value (10 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

MINIMUM BLOCKING This prompt generates data that will be used by the DICTCDE utility. Enter the minimum blocking factor for the file. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

MINIMUMRECORD SIZE This prompt generates data that will be used by the DICTCDE utility. Enter the minimum number of characters in the record. DEFAULT: Pressing [[RETURN]] indicates no change.

MODIFY BLOCKING (N/Y)? This prompt generates additional prompts to modify the blocking factor of the file. Those prompts generate data that can be used by the DICTCDE utility. DEFAULT is No.

PASSWORD Enter a new password for the user security class (8 characters maximum). A blank value (pressing the space bar followed by [[RETURN]]) removes the password. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

PROCEDURE Enter the name of an existing procedure (20 characters maximum).

RECORD FORMAT (F/V/U/S) This prompt generates data that will be used by the DICTCDE utility for COBOL II programs. Enter one of the following to specify how the logical records are contained in the file:

F	for fixed length logical records
V	for variable length logical records
U	for undefined length logical records
S	for the multirecord or “multiblock” option

DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

RECORDING MODE(A/B)? This prompt generates data that will be used by the DICTCDE utility. Enter A to change the recording mode of the file to ASCII. Enter B to change the recording mode of the file to BINARY. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

RESPONSIBILITY Enter a new name for the person, department, or area responsible for the integrity of the entity (20 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

RESTRICTED FILE If the scope of the security class is to be limited to a file, enter the name of the file (20 characters maximum). The file type must be BASE. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

RIGHT JUSTIFY (N/Y)? This prompt generates data that will be used by the DICTCDE utility. Enter Y or Yes to right justify alphabetic or alphanumeric data elements. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

SIGN POSITION This prompt generates data that will be used by the DICTCDE utility. Enter the position of the sign for a signed numeric data element as follows:

LO	leading overpunched
LS	leading separate
TO	trailing overpunched
TS	trailing separate

DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

SIZE Enter a new integer value required to display an element (1 to 9999). Include a position for a decimal point if there is one, but do not include one for a sign. The largest value for each type is as follows:

For types Z, P, and 9, the largest value is 28 digits.

For types I, J, and K, the largest value is 29 digits.

For types E, R, X, and U, the largest value is 9999 digits.

DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

STORAGE LENGTH(n) The calculated storage length in bytes is displayed as “n”. An over-ride value may be entered (1 to 9999). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

SYNCHRONIZED (N/Y)? This prompt generates data that will be used by the DICTCDE utility. Enter Y or Yes to align the data elements in order to facilitate arithmetic operations. DEFAULT: Pressing [[RETURN]] in response to this prompt

indicates no change.

TYPE If the subcommand is:

CATEGORY, CLASS, GROUP, or PROCEDURE enter a new user defined type (4 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

ELEMENT enter a new type for the data element as follows:

U	upper case ASCII
X	upper or lower case ASCII
9	numeric ASCII
Z	zoned decimal
P	packed decimal
I	integer binary
J	integer binary (COBOL)
K	logical
R	floating point (commercial)
E	floating point (E format)
S	string (PASCAL)
B	boolean (PASCAL)
*	back reference to an existing element (PASCAL)

If the type used is numeric, a plus character (+) following it indicates that the data element may have only positive values. If a plus character is used with any other type, it is ignored. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

FILE - enter a new file type based on one of the following types:

BASE	IMAGE database name
MAST IMAGE	manual master data set
AUTO	IMAGE automatic master data set
DETL	IMAGE detail data set
FORM	VPLUS form
VLPS	VPLUS forms file name
MPEF	MPE sequential file
MPER	MPE relative file
KSAM	KSAM file

DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

UNIT(R/C)? This prompt generates data that will be used by the DICTCDE utility. Specify the unit used to determine the blocking factor with R for RECORDS or C for CHARACTERS. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

Purge

Purges an entity which was created with the CREATE command from the Dictionary.

Syntax

```
[option] PURGE subcommand P
```

Use the PURGE command to purge an entity and all references to that entity from the Dictionary. A subcommand is used to identify the kind of entity to be purged. For example, PURGE FILE purges the specified file and all references to the file from the Dictionary. That is, if a master file and an associated primary path element are purged, the associated sort element, primary path, and path file information are purged as well. A complete list of entities that can be purged is given below under Subcommands.

Before the PURGE command is implemented, the Dictionary allows you to cancel the command with the following prompt:

```
DELETE ALL ENTRIES(N/Y)?
```

A response to this prompt other than Y cancels the PURGE command.

Subcommands

Any of the following subcommands can be used with the PURGE command:

CATEGORY	purges the category and all references to it
CLASS	purges the security class and all references to it
ELEMENT	purges the data element and all references to it
FILE	purges the file and all references to it; optionally purges related files and associated elements
GROUP	purges an HP Inform group and all references to it
LOCATION	purges the location and all references to it
PROCEDURE	purges the procedure and all references to it

Example

```
>PURGE ELEMENT
                    ELEMENT> Element 1  < Identifies element
    DELETE ALL ENTRIES(N/Y)?> Y           to be purged.
    ALL ENTRIES DELETED                   Confirms request to
                                          purge element.
    >                                     New command prompt.
    > PURGE FILE
                      FILE> Base1     < Identifies BASE file
                                          to be purged.
     DELETE ALL ENTRIES(N/Y)?> Y           Confirms request to purge Base1.
    PURGE RELATED CHILD FILES(Y/N)?> Y    Requests that all related
                                          child files be purged.
     PURGE ASSOCIATED ELEMENTS(Y/N)?> Y   Requests that child file
                                          elements not associated
                                          with another file be purged.
LIST ALL PURGED ENTITIES(Y/N)?> Y         Requests that purged
                                          entities be displayed.
     FILE PURGED:  BASE1
    CHILD FILES:            FILES PURGED:
       ELEMENTS:                ELEMENTS PURGED:
    HP-DIV-MSTR             HP-DIV-MSTR
        HP-DIV-NO
        HP-DIV-NAME
        HP-GROUP
     COMMODITY-MSTR          COMMODITY-MSTR
        COMMODITY-TYPE
        COMMODITY-NAME
    VENDOR-DETAIL           VENDOR-DETAIL
        COMMODITY-TYPE
        C-VEND-NUM             C-VEND-NUM
        VEND-NAME
               Q1-RATING         Q1-RATING
               Q2-RATING         Q2-RATING
               Q3-RATING         Q3-RATING
     >                                    New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string.

You enter:			You are prompted for:	
COMMAND	**SUB-COMMAND**		**PROMPT**	
PURGE	CATEGORY	*	CATEGORY	
			DELETE ALL ENTRIES (N/Y)?	
PURGE	CLASS	*	CLASS	
			DELETE ALL ENTRIES (N/Y)?	
PURGE	ELEMENT	*	ELEMENT	
			DELETE ALL ENTRIES (N/Y)?	
PURGE	FILE	*	FILE	
			1. if FILE is BASE or VPLS file	
			DELETE ALL ENTRIES (N/Y)?	
			PURGE RELATED CHILD FILES(Y/N)?	You must respond YES, Y, NO, or N; no default response occurs.
			ANSWER YES OR NO.	Issued if response to above was not YES or NO.
			PURGE ASSOCIATED ELEMENTS(Y/N)?	Issued if response to PURGE RELATED CHILD FILES was YES. You must respond YES, Y, NO, or N; no default response occurs.
			ANSWER YES OR NO.	Issued if response to above was not YES or NO.
			LIST ALL PURGED ENTITIES(Y/N)?	Issued if response to PURGE RELATED CHILD FILES or PURGE ASSOCIATED ELEMENTS was YES.
			2. if FILE is MAST, DETL, AUTO, MPEF, KSAM, or FORM file	
			DELETE ALL ENTRIES(N/Y)?	
			PURGE ASSOCIATED ELEMENTS(Y/N)?	You must respond YES, Y, NO, or N. No default response occurs. ANSWER YES OR NO. Issued if response for above was not YES or NO.
			LIST ALL PURGED ENTITIES(Y/N)?	Issued if response to PURGE ASSOCIATED ELEMENTS was YES.
PURGE	GROUP	*	GROUP	
		*	DELETE ALL ENTRIES(N/Y)?	
PURGE	LOCATION	*	LOCATION	
			DELETE ALL ENTRIES(N/Y)?	
PURGE	PROCEDURE	*	PROCEDURE	
			DELETE ALL ENTRIES(N/Y)?	

Prompts

Depending on the subcommand used, one of the following prompts is issued:

ANSWER YES OR NO — You must respond to this prompt with Y, Yes, N or No.

CATEGORY — Enter the name of an existing category which is to be purged from the Dictionary (20 characters maximum).

CLASS — Enter the identifying number of an existing security class which is to be purged from the Dictionary (0 to 9999).

DELETE ALL ENTRIES (N/Y)? — Enter Y to purge the entity and all references to it from the Dictionary. Note that entering N or pressing [[RETURN]] in response to this prompt means that no action is taken.

ELEMENT — Enter the name of an existing data element which is to be purged from the Dictionary (20 characters maximum).

FILE — Enter the name of an existing file which is to be purged from the Dictionary (20 characters maximum).

GROUP — Enter the name of an existing HP Inform group which is to be purged from the Dictionary (20 characters maximum). NOTE: $MENU cannot be purged.

LIST ALL PURGED ENTITIES (Y/N)? — Enter Y or press [[RETURN]] to display all related files and elements as well as files and elements that were purged. Enter N if you do not want this listing.

LOCATION — Enter the name of an existing physical location which is to be purged from the Dictionary (20 characters maximum).

PROCEDURE — Enter the name of an existing procedure which is to be purged from the Dictionary (20 characters maximum).

PURGE RELATED CHILD — Enter Y to purge all related child files and FILES(Y/N)? all references to these files. Note that a child file is not purged if it is related to another parent file. Enter N to purge only the specified file and all references to that file. You must enter YES, Y, NO, or N; there is no default response.

PURGE ASSOCIATED ELEMENTS (Y/N)? — Enter Y to purge all associated (i.e., member) elements and all references to these elements. Note that an element is not purged if it is associated with other files or with other elements. Enter N if you do not want associated elements purged. You must enter YES, Y, NO, or N; there is no default response.

Relate

Establishes a hierarchical relationship between like entities.

Syntax

```
[option] RELATE subcommand REL
```

Use the RELATE command to establish a hierarchical relationship between entities. The relationship must be between entities of the same type. That is, only categories can be related to categories, files to files, and so forth. A complete list of entities that can be related is given below under Subcommands.

The subcommand identifies the type of entity for which a relationship is established. For example, RELATE FILE establishes a relationship between a file of type BASE and a file of type MAST, DETL, or AUTO.

Child entities are ordered within the parent's entry list in the order in which they were related to the parent. This order can be changed through the REORDER command.

Subcommands

Any of the following subcommands can be used with the RELATE command:

CATEGORY	relates a category to a category
CLASS	relates a class to a class
ELEMENT	elates a data element to a data element
FILE	relates a file to a file
GROUP	relates an HP Inform group to a group
PROCEDURE	relates a procedure to a procedure

Example

```
> RELATE ELEMENT
        PARENT ELEMENT> Emp-num          <     Name of parent element.
         CHILD ELEMENT> Emp-div          <     Name of child element.
           CHILD ALIAS> Emp-num-id       <     Name by which the child
                                               element is known in the
                                               relationship.
  GENERATE BYTE POSITION(N/Y)?> y        <     Calculates byte offset
                                               so that element will
                                               follow the last element
                                               related to parent element
    DESCRIPTION > The parent element describes the employee number <
                > and the child element describes the employee     <
                > division number.                                 <
                >                                                  <
                                         [[RETURN]] ends prompting
                                         for description; another
                                         ends command string.
>                                        New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string. Two asterisks (**) next to a prompt indicate that this prompt and those which follow are repeated until [[RETURN]] is pressed in response to this prompt.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
RELATE	CATEGORY	*	PARENT CATEGORY	
		**	CHILD CATEGORY	
			DESCRIPTION	
RELATE	CLASS	*	PARENT CLASS	
		**	CHILD CLASS	
			DESCRIPTION	
RELATE	ELEMENT	*	PARENT ELEMENT	
		**	CHILD ELEMENT	
			CHILD ALIZS	
			GENERATE BYTE POSITION (N/Y)?	
			BYTE POSITION	Issued if No or [[RETURN]] to GENERATE BYTE POSITION.
			DESCRIPTION	
RELATE	FILE	*	PARENT FILE	
		**	CHILD FILE	
			CHILD ALIAS	
			CAPACITY	Issued only if parent file is type BASE.
			BLOCKMAX	Issued only if parent file is type BASE.
			DESCRIPTION	
RELATE	GROUP	*	PARENT GROUP	
		**	CHILD GROUP	
			DESCRIPTION	
RELATE	PROCEDURE	*	PARENT PROCEDURE	
		**	CHILD PROCEDURE	
			DESCRIPTION	

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

BLOCKMAX — Enter the maximum physical block length (in words) for the file within the database file (128 to 2048). DEFAULT is 512.

BYTE POSITION — Enter a positive integer value for the starting position of the child data element field within the parent data element field. First byte position is 1. DEFAULT is 1.

CAPACITY — Enter an integer value for the maximum number of entries for the file within the database file (0 to 999999999). DEFAULT is 0.

CHILD ALIAS — When the subcommand ELEMENT is used, enter the name by which the child element is known within the child to parent element relationship (20 characters maximum). The child alias will be used by the DICTCDE and DICTPDE utilities.

When the subcommand FILE is used, enter the name by which the child file is known within the parent file (16 characters maximum). DEFAULT: If [[RETURN]] is pressed in response to this prompt, the alias becomes the child's file name, up to the first 16 characters.

CHILD CATEGORY — Enter the name of an existing category that is the child in the relationship (20 characters maximum).

CHILD CLASS — Enter a numeric value which is the identifying number of an existing security class not of type INFO (0 to 9999).

CHILD ELEMENT — Enter the name of an existing data element that is the child in the relationship (20 characters maximum).

CHILD FILE — Enter the name of an existing file that is the child in the relationship according to the following restrictions:

If the parent file is type:

BASE - the child file must be one of the following types:

MAST	(IMAGE manual master data set)
AUTO	(IMAGE automatic master data set)
DETL	(IMAGE detail data set)

VPLS - the child file must be the following type:

FORM (VPLUS form)

CHILD GROUP — Enter the name of an existing HP Inform group that is the child in the relationship (20 characters maximum).

CHILD PROCEDURE — Enter the name of an existing procedure that is the child in the relationship (20 characters maximum).

DESCRIPTION Enter a textual description of the relationship (50 characters per description line allowed, unlimited lines).

GENERATE BYTE This prompt generates data that will be used by the POSITION (N/Y)? DICTPDE and DICTCDE utilities. Enter Y or Yes to generate the byte offset for this child element so that it will immediately follow the last child element related to the parent element. If this element is the first child element to be related to the parent element, the byte position generated will be 1. DEFAULT is No.

PARENT CATEGORY Enter the name of an existing category that is the parent in the relationship (20 characters maximum).

PARENT CLASS Enter a numeric value which is the identifying number of an existing INFO type security class (0 to 9999).

PARENT ELEMENT Enter the name of the existing data element that is the parent in the relationship.

PARENT FILE Enter the name of an existing file that is the parent in the relationship. The file type must be one of the following:

BASE	(IMAGE database)
VPLS	(VPLUS forms file)

PARENT GROUP Enter the name of an existing HP Inform group that is the parent in the relationship (20 characters maximum). Note that if the child group is to be related to the highest group in the group tree structure, enter “$MENU” as the parent group's name.

PARENT Enter the name of an existing procedure that is the PROCEDURE parent in the relationship.

Remove

Removes a relationship between entities which was established with the RELATE command.

Syntax

```
[option] REMOVE subcommand REM
```

Use the REMOVE command to remove the relationship between entities which was established with the RELATE command. This command does not purge the entities themselves but the relationship between the entities. A complete list of the relationships that can be removed is given below under Subcommands. The subcommand identifies the type of relationship that is removed. For example, REMOVE FILE, removes the relationship between a parent file and a child file.

Subcommands

Any of the following subcommands can be used with the REMOVE command:

CATEGORY	removes the relationship between a child and a parent category
CLASS	removes the relationship between a child and a parent class
ELEMENT	removes the relationship between a child and a parent element
FILE	removes the relationship between a child and a parent file
GROUP	removes the relationship between a child and a parent HP Inform group
PROCEDURE	removes the relationship between a child and a parent procedure

Example

```
>REMOVE FILE
              PARENT FILE> Ordmgt        <   Name of a database.
              CHILD FILE> Customer       <   Name of a MAST data set
                                             related to the database.
 ENTRY DELETED                               DICTDBM's response.
              CHILD FILE>                <   [[RETURN]] to terminate the
                                             command.
 >                                           New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string. Two asterisks (**) next to a prompt indicate that this prompt and those which follow are repeated until [[RETURN]] is pressed in response to this prompt.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT
REMOVE	CATEGORY	*	PARENT CATEGORY
		**	CHILD CATEGORY
REMOVE	CLASS	*	PARENT CLASS
		**	CHILD CLASS
REMOVE	ELEMENT	*	PARENT ELEMENT
		**	CHILD ELEMENT
REMOVE	FILE	*	PARENT FILE
		**	CHILD FILE
REMOVE	GROUP	*	PARENT GROUP
		**	CHILD GROUP
REMOVE	PROCEDURE	*	PARENT PROCEDURE
		**	CHILD PROCEDURE

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

CHILD CATEGORY	Enter the name of a child category from which the relationship is to be removed (20 characters maximum).
CHILD CLASS	Enter a numeric value which is the identifying number of the child security class from which the relationship is to be removed.
CHILD ELEMENT	Enter the name of a child element from which the relationship is to be removed (20 characters maximum).
CHILD FILE	Enter the name of a child file from which the relationship is to be removed (20 characters maximum).
CHILD GROUP	Enter the name of a child HP Inform group from which the relationship is to be removed (20 characters maximum).
CHILD PROCEDURE	Enter the name of a child procedure from which the relationship is to be removed (20 characters maximum).
PARENT CATEGORY	Enter the name of a parent category from which a relationship is to be removed (20 characters maximum).

PARENT CLASS	Enter a numeric value which is the identifying number of the parent security class from which a relationship is to be removed.
PARENT ELEMENT	Enter the name of a parent element from which a relationship is to be removed (20 characters maximum).
PARENT FILE	Enter the name of a parent file from which a relationship is to be removed (20 characters maximum).
PARENT GROUP	Enter the name of a parent HP Inform group from which a relationship is to be removed (20 characters maximum). Enter “$MENU” if the child group is related to the highest group in the group tree structure.
PARENT PROCEDURE	Enter the name of a parent procedure from which a relationship is to be removed (20 characters maximum).

Rename

Renames an existing entity which was created with the CREATE command.

Syntax

```
[option] RENAME subcommand REN
```

Use the RENAME command to change the name of an entity. When an entity is renamed, every reference to the entity in the Dictionary is also changed. A complete list of the entities that can be renamed is given under Subcommands.

The subcommand is used to identify the type of entity for which a change is to occur. For example, RENAME CLASS changes the identifying number for the specified class.

Subcommands

Any of the following subcommands can be used with the RENAME command:

CATEGORY	renames an existing category
CLASS	renames an existing identifying number of a security class
ELEMENT	renames an existing data element
FILE	renames an existing file
GROUP	renames an existing HP Inform group
LOCATION	renames an existing physical location
PROCEDURE	renames an existing procedure

Example

```
>RENAME CLASS
                CLASS> 11  <            Changes the identifying number
            NEW NUMBER> 111 <           from 11 to 111.
>                                       New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string. Two asterisks (**) next to a prompt indicate that this prompt and those which follow are repeated until [[RETURN]] is pressed in response to this prompt.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT
RENAME	CATEGORY	*	CATEGORY
		*	NEW NAME
RENAME	CLASS	*	CLASS
		*	NEW NUMBER
RENAME	ELEMENT	*	ELEMENT
		*	NEW NAME
RENAME	FILE	*	FILE
		*	NEW NAME
RENAME	GROUP	*	GROUP
		*	NEW NAME
RENAME	LOCATION	*	LOCATION
			NEW NAME
RENAME	PROCEDURE	*	PROCEDURE
		*	NEW NAME

Prompts

depending on the subcommand used, one or more of the following prompts is issued:

CATEGORY Enter the name of an existing category (20 characters maximum).

CLASS Enter the numeric value which is the identifying number of an existing user security class (0 to 9999).

ELEMENT Enter the name of an existing data element (20 characters maximum).

FILE Enter the name of an existing file (20 characters maximum).

GROUP Enter the name of an existing HP Inform group (20 characters maximum). NOTE: $MENU can not be renamed.

LOCATION Enter the name of an existing location (20 characters maximum).

NEW NAME Enter a new unique name for the entity (20 characters maximum).

NEW NUMBER Enter a new unique numeric value for the user security class (0 to 9999).

Reorder

Reorders an entity's position in the parent's list of relationships which was established with the RELATE command.

Syntax

```
[option] REORDER subcommand REO
```

Use the REORDER command to change the position of a child within the parent's entry list of relationships. A complete list of entities that can be reordered is given below under Subcommands. The subcommand identifies the type of relationship for which an entity's position is to be reordered. For example, REORDER CATEGORY, changes the position of a child category within a parent category's list of relationships. To determine the current position of the child entity in the parent's entry list and the name of the child entity before which it is to be placed, use the SHOW command. (Refer to the SHOW COMMAND, in this section for more information.) If [[RETURN]] is pressed in response to the NEW POSITION prompt for this command, the child entity will be placed at the end of the parent's entry list of relationships.

Subcommands

Any of the following subcommands can be used with the REORDER command:

CATEGORY	reorders a child category within a parent category's entry list
ELEMENT	reorders a child data element within a parent data element's entry list
FILE	reorders a child file within a parent file's entry list
GROUP	reorders a child HP Inform group within a parent group's entry list
PROCEDURE	reorders a child procedure within a parent procedure's entry list

Example

```
>REOREDER CATEGORY
         PARENT CATEGORY> Director       <
          CHILD CATEGORY> Marketing      <    Category to be reordered.
            NEW POSITION> Manufacturing  <    New position is before
                                              this category.
          CHILD CATEGORY>                <    Terminate the command.
 >                                            New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT
REORDER	CATEGORY	*	PARENT CATEGORY
		**	CHILD CATEGORTY
			NEW POSITION
REORDER	ELEMENT	*	PARENT ELEMENT
		**	CHILD ELEMENT
			NEW POSITION
REORDER	FILE	*	PARENT FILE
		**	CHILD FILE
REORDER	GROUP	*	PARENT GROUP
		**	CHILD GROUP
			NEW POSITION
REORDER	PROCEDURE	*	PARENT PROCEDURE
		**	CHILD PROCEDURE
			NEW POSITION

Prompts

Depending on which subcommand you use, one or more of the following prompts is issued:

CHILD CATEGORY	Enter the name of a child category that is to be reordered (20 characters maximum).
CHILD ELEMENT	Enter the name of a child data element that is to be reordered (20 characters maximum).
CHILD FILE	Enter the name of a child file that is to be reordered (20 characters maximum).
CHILD GROUP	Enter the name of a child HP Inform group that is to be reordered (20 characters maximum).
CHILD PROCEDURE	Enter the name of a child procedure that is to be reordered (20 characters maximum).
NEW POSITION	Enter the name of an existing child entity before which the reordered child is to be placed. DEFAULT: Pressing [[RETURN]] in response to this prompt places the reordered child at the end of the parent's list.
PARENT CATEGORY	Enter the name of an existing category that is the parent in the relationship (20 characters maximum).

PARENT ELEMENT	Enter the name of an existing data element that is the parent in the relationship (20 characters maximum).
PARENT FILE	Enter the name of an existing file that is the parent in the relationship (20 characters maximum).
PARENT GROUP	Enter the name of an existing HP Inform group that is the parent in the relationship (20 characters maximum).
PARENT PROCEDURE	Enter the name of an existing procedure that is the parent in the relationship (20 characters maximum).

Report

Reports a sorted listing of data elements.

Syntax

```
[option] REPORT subcommand R
```

Use the REPORT command to display an alphabetized list of data elements owned by an entity or by that entity's children. Note that any element that is owned by more than one entity is listed only once by the REPORT command.

The subcommand identifies the entity to be listed. For example, REPORT CLASS lists the elements owned by the named user security class. REPORT ELEMENT lists all the elements defined in the Dictionary. Also, if the subcommand used is CATEGORY, FILE, GROUPS, or PROCEDURE and the entity name entered is for a child entity, the elements owned by that entity are reported. If the name entered in response to one of these subcommands is for a parent entity, all the elements for the related child entities are reported. If the name entered in response to the PROCEDURE subcommand prompt is for a parent procedure any elements associated with that procedure or any of its related child procedures will also be reported.

The PRINT option can be used with the command string to direct the report to the line printer. The report would then include descriptive information in addition to the information normally displayed on the terminal. The description information is the description that was entered when the entity was created or when an association was made.

Subcommands

Any of the following subcommands can be used with the REPORT command:

CATEGORY	reports all the data elements associated with a category or with related child categories
CLASS	reports all the data elements associated with a security class
ELEMENT	reports all the data elements defined in the Dictionary
FILE	reports all the data elements associated with a file or with related child files
GROUP	reports all the data elements associated with an HP Inform group or with related child groups
PROCEDURE	reports all the data elements associated with a procedure or with related child procedures, including any owned by the named procedure

Example

```
 > REPEAT REPORT FILE
                        FILE> product          <
      PRIMARY/SECONDARY(P/S)?                  <
LIST OF PRIMARY FORMAT DATA ELEMENTS UNDER FILE: PRODUCT
     ELEMENT(PRIMARY):    TYPE: SIZE: DEC: LENGTH: COUNT:
      DESCRIPTION          X     30    0    30      1
      PROD-NO              U     8     0    8       1
                       FILE&gt; sales <
LIST OF PRIMARY FORMAT DATA ELEMENTS UNDER FILE: SALES
     ELEMENT(PRIMARY):    TYPE: SIZE: DEC: LENGTH: COUNT:
       ACCOUNT              X     10    0    10      1
       DELIV-DATE           X     6     0    6       1
       DESCRIPTION          X     30    0    30      1
       PRICE                P     11    0    6       1
       PROD-NO              U     8     0    8       1
       PURCH-DATE           X     6     0    6       1
       PURCH-NO             U     6     0    6       1
       QUANTITY             P     7     0    4       1
       TOTAL                P     11    0    6       1
                        FILE> customer          <
      PRIMARY/SECONDARY(P/S)?                    <
      LIST OF PRIMARY FORMAT DATA ELEMENTS UNDER FILE: CUSTOMER
     ELEMENT(PRIMARY):    TYPE: SIZE: DEC: LENGTH: COUNT:
       ACCOUNT              X     10    0    10      1
       CITY-NAME            X     14    0    14      1
       CREDIT               U     2     0    2       1
       DELIV-DATE           X     6     0    6       1
       DESCRIPTION          X     30    0    30      1
       FIRST-NAME           U     18    0    18      1
       LAST-NAME            X     20    0    20      1
       PRICE                P     11    0    6       1
       PROD-NO              U     8     0    8       1
       PURCH-DATE           X     6     0    6       1
       PURCH-NO             U     6     0    6       1
       STATE                X     2     0    2       1
       STR-ADDRESS          X     22    0    22      1
```

```
      TOTAL                P     11    0    6       1
      ZIP                  X     10    0    10      1
                       FILE>                   <   [[RETURN]] key pressed.
>
```

Subcommand Prompts

For the prompts shown with an asterisk, you must enter a response other than [[RETURN]] in order to implement the command string.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
REPORT	CATEGORY	*	CATEGORY	
REPORT	CLASS	*	CLASS	
RE[PORT	ELEMENT		(No prompts)	
REPORT	FILE	*	FILE	
		*	PRIMARY/SECONDARY(P/S)?	Issued only if file type is KSAM, MPEF or MPER.
REPORT	GROUP	*	GROUP	
REPORT	PROCEDURE	*	PROCEDURE	

Prompts<

Depending on the subcommand used, one of the following prompts is issued:

CATEGORY	Enter the name of an existing category (20 characters maximum).
CLASS	Enter the numeric value which is the identifying number of an existing security class (0 to 9999).
FILE	Enter the name of an existing file (20 characters maximum).
GROUP	Enter the name of an existing HP Inform group (20 characters maximum).
PRIMARY/SECONDARY (P/S)?	Enter P (or press [[RETURN]]) if the elements to be reported are in the primary format for that file. Enter S if the elements are in the secondary format. (See the discussion of the ADD command for more information on primary and secondary formats for a file.) DEFAULT is P.
PROCEDURE	Enter the name of an existing procedure (20 characters maximum).

Resequence

Resequences a data element within an association established with the ADD command.

Syntax

```
[option] RESEQUENCE subcommand RES
```

Use the RESEQUENCE command to change a data element's position within an entity's list of associations which was established by the ADD command. A complete list of entities for which data elements can be resequenced is given below under Subcommands.

The subcommand identifies the type of entity for which a resequencing is to occur. For example, RESEQUENCE FILE changes the position of a data element in the list of elements associated with that file.

The name of the element before which the resequenced element is to be placed is entered in response to the NEW POSITION prompt. Pressing [[RETURN]] in response to this prompt places the resequenced element at the end of the association list for the entity. To see the current list of associations and current positions, use the SHOW command.

Subcommands

Any of the following subcommands can be used with the RESEQUENCE command:

CATEGORY	resequences a data element within a category
CLASS	resequences a data element within a security class
FILE	resequences a data element within a file
GROUP	resequences a data element within an HP Inform group
PROCEDURE	resequences a data element within a procedure

Example

```
>RESEQUENCE FILE
               FILE> Customer        <       The file's name.
  PRIMARY/SECONDARY(P/S)?            <       Indicates element is in the
                                             Primary format.
           ELEMENT> Last-name        <       Element to be resequenced.
       NEW POSITION> Street-addr     <       New position is before
                                             this element.
           ELEMENT>                  [[RETURN]] to terminate
                                             command.
  >                                          New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string. Two asterisks (**) next to a prompt indicate that this prompt and those which follow are repeated until [[RETURN]] is pressed in response to this prompt.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
RESEQUENCE	CATEGORY	*	CATEGORY	
		**	ELEMENT	
			NEW POSITION	
RESEQUENCE	CLASS	*	CLASS	
		**	ELEMENT	
			NEW POSITION	
RESEQUENCE	FILE	*	FILE	
		*	PRIMARY/SECONDARY(P/S)?	Issued only if file type is KSAM, MPEF or MPER.
RESEQUENCE	GROUP	*	GROUP	
		**	ELEMENT	
			NEW POSITION	
RESEQUENCE	PROCEDURE	*	PROCEDURE	
		**	ELEMENT	
			NEW POSITION	

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

CATEGORY — Enter the name of an existing category in which a data element is to be resequenced (20 characters maximum).

CLASS — Enter the numeric value which identifies an existing security class in which a data element is to be resequenced (0 to 9999).

ELEMENT — Enter the name of the existing data element which is to be repositioned within the association list (20 characters maximum).

FILE — Enter the name of an existing file in which a data element is to be resequenced (20 characters maximum).

GROUP — Enter the name of an existing HP Inform group in which a data element is to be resequenced (20 characters maximum).

NEW POSITION — Enter the name of the data element before which the resequenced data element is to be placed. DEFAULT: Pressing [[RETURN]] in response to

this prompt places the resequenced data element at the end of the association list.

PRIMARY/SECONDARY (P/S)? Enter P (or press [[RETURN]]) if the element to be resequenced is in the primary format for the file. Enter S if the element is in the secondary format. (See the discussion of the ADD command for more information on primary and secondary formats.) DEFAULT is P.

PROCEDURE Enter the name of an existing procedure in which a data element is to be resequenced (20 characters maximum).

Secure

Secures data elements, files and groups to a security class.

Syntax

```
[option] SECURE subcommand SEC
```

Use the SECURE command to either assign all of the data elements associated with a file to one security class or to assign a group and all its child groups to one security class. When using the SECURE command to assign elements to a security class, you may also assign the file and all its child files, if any, to that security class. Therefore, you can avoid adding each individual data element, one at a time, to a security class with the ADD CLASS command string. However, when using the SECURE command to assign a group to a security class, no elements belonging to the group or its child groups will be added to the security class.

After all data elements, files, or groups are secured to a designated class, the security class and capability association can be selectively changed for an element with the UPDATE command. Unwanted associations can be selectively deleted with the DELETE command. If a data element was already secured to a security class (with a previous SECURE command) another SECURE command can be used to change the access capability previously defined. The new SECURE command deletes the association made earlier and creates a new association which may specify a different type of access capability.

Subcommand

FILE	secures all data elements in a file to a security class and secures the file and its child files to the security class
GROUP	secures a group and its child groups to a security class

Example

```
> SECURE FILE
                FILE> Customer          <     File to be secured.
               CLASS> 1                  <     Security class identifier.
   ACCESS CAPABILITY> M  <                     Type of access.
       ELEMENTS WILL BE SECURED TO CLASS.
       SECURE FILE(S) TO CLASS(N/Y)? N
>                                              New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
SECURE	FILE	*	FILE	
		*	CLASS	
		*	ACCESS CAPABILITY	
			ELEMENTS WILL BE SECURED TO CLASS.	
			SECURE FILE(S) TO CLASS(N/Y)?	
			FILE ACCESS CAPABILITY	Issued only if Yes to SECURE FILE(S) TO CLASS and file type is BASE, MAST, DETL, or AUTO
SECURE	GROUP	*	GROUP	
		*	CLASS	

Prompts

The following prompts are issued for the SECURE command:

ACCESS CAPABILITY Enter the type of access capability given to the user security class for all the data elements in the specified file. Choose from one of the following:

R	read only
U	read and update only
M	read write and modify
X	null read/write list “(/)”

Refer to the IMAGE Reference Manual for a detailed explanation of the meanings of user class access capabilities.

CLASS Enter the numeric integer (0 to 9999) for an existing user security class (not of type INFO) when securing data elements or a file to a class.

FILE Enter the name of an existing file (20 characters maximum). The type of file must be one of the following:

BASE	IMAGE database
MAST	IMAGE manual master data set
AUTO	IMAGE automatic master data set
DETL	IMAGE detail data set
MPEF	MPE Sequential File

	MPER	MPE Relative File
	KSAM	KSAM file

FILE ACCESS CAPABILITY — Enter the type of capability to be given to the user security class for the IMAGE file as follows:

R	= read only
W	= read, write, modify
X	= null read/write list “(/)”

Refer to the IMAGE Reference Manual for a detailed explanation of the user class access capabilities.

GROUP — Enter the name of the INFORM group to be secured (20 characters maximum).

SECURE FILE(S) TO CLASS (N/Y)? — Enter Y or Yes to secure the file(s) to the user security class. If the FILE type is BASE, then that BASE file, all its data sets, and all associated elements will be added to the CLASS.

Show

Shows all the relationships for an entity and the associations for child entities.

Syntax

```
[option] SHOW subcommand S
```

Use the SHOW command to display the hierarchical relationship for an entity and the association information for child entities. Relationships are established using the RELATE command and associations are established using the ADD command.

The subcommand identifies the entity to be displayed. For example, the subcommands LOCATION and CLASS show the association information for a location and class. The subcommands CATEGORY, PROCEDURE, FILE, and GROUP show the relationship and the association information for their respective entities. The subcommand ELEMENT shows only the relationship information for the element.

The relationship information is displayed as a tree structure from the point of entry down. After the parent entity name is displayed, each child is displayed in the order in which the child was related to the parent. Data elements associated with each child can also be displayed depending on your response to the prompt to do so. If you choose to display the data elements, an asterisk (*) will appear next to an element to indicate that it is a key or search element within that file. The PRINT option can be used with the command string to redirect what is shown by this command to the line printer. Description information will be printed in addition to the information normally displayed on the terminal. The description information is the information that was entered when the relationship or association was made.

Subcommands

Any of the following subcommands can be used with the SHOW command:

CATEGORY	shows all the relationships for a category plus associated data elements where applicable
CLASS	shows all the associations for a user security class and all the related child classes plus all associated groups.
ELEMENT	shows all the relationships for a data element
FILE	shows all the relationships and links for a file plus associated data elements where applicable
GROUP	shows all the relationships for an HP Inform group plus associated data elements where applicable and access information for the group
LOCATION	shows all the associations for a location
PROCEDURE	shows all the relationships for a procedure plus associated data elements where applicable

Example

```
>SHOW FILE
                      FILE> HOUSES              <
    SHOW ALL FILE ELEMENTS(Y/N)?> y
    FILE:                TYPE: RESPONSIBILITY:
    HOUSES               BASE
    FILE(ALIAS):     TYPE:     FILE(PRIMARY):       CAPACITY:
    CITY-MASTER      MAST      CITY-MASTER          101
           ELEMENT(ALIAS):           PROPERTIES:       ELEMENT(PRIMARY):
           CITY-ABBR           *      U (4,0,4)          CITY-ABBR
           CITYNAME                   U (20,0,20)        CITYNAME
    FILE(ALIAS):     TYPE:     FILE(PRIMARY):       CAPACITY:
    BATH-MASTER      AUTO      BATH-MASTER          31
           ELEMENT(ALIAS):           PROPERTIES:       ELEMENT(PRIMARY):
           NUMBER-BATHS        *      X (4,0,4)          NUMBER-BATHS
    FILE(ALIAS):     TYPE:     FILE(PRIMARY):       CAPACITY:
    BEDS-MASTER      AUTO      BEDS-MASTER          11
           ELEMENT(ALIAS):           PROPERTIES:       ELEMENT(PRIMARY):
           NUMBER-BEDS         *      X (2,0,2)          NUMBER-BEDS
    FILE(ALIAS):     TYPE:     FILE(PRIMARY):       CAPACITY:
    RESIDENTIAL      DETL      RESIDENTIAL          200
           ELEMENT(ALIAS):           PROPERTIES:       ELEMENT(PRIMARY):
           CITY-ABBR           *      U (4,0,4)          CITY-ABBR
                                 CHAIN MASTER SET: CITY-MASTER
           NUMBER-BEDS         *      X (2,0,2)          NUMBER-BEDS
                                 CHAIN MASTER SET: BEDS-MASTER
           NUMBER-BATHS        *      X (4,0,4)          NUMBER-BATHS
                                 CHAIN MASTER SET: BATH-MASTER
           CURRENT-OWNER              U (20,0,20)        CURRENT-OWNER
           OWNERS-PHONENR             X (10,0,10)        OWNERS-PHONENR
                .
                .
                .
    >
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
SHOW	CATEGORY	*	CATEGORY	
SHOW	CLASS	*	CLASS	
SHOW	ELEMENT	*	ELEMENT	
SHIOW	FILE		FILE	
			PRIMARY/SECONDARY(P/S)?	Issued if file type is KSAM, MPEF or MPER.
SHOW	GROUP	*	GROUP	
SHOW	LOCATION	*	LOCATION	
SHOW	PROCEDURE	*	PROCEDURE	

Prompts

Depending on which subcommand you use, one of the following prompts is issued:

CATEGORY Enter the name of an existing category for which relationships are to be shown (20 characters maximum).

CLASS Enter the name of an existing user security class for which a list of associations is to be given (4 characters maximum).

ELEMENT Enter the name of an existing data element for which relationships are to be shown (20 characters maximum).

FILE Enter the name of an existing file for which relationships are to be shown (20 characters maximum).

GROUP Enter the name of an existing HP Inform group for which relationships and access information are to be shown (20 characters maximum). Note that a pound sign ( ) indicates the element will not be displayed on HP Inform's Data Names Menu.

LOCATION Enter the name of an existing location for which a list of associations is to be given (20 characters maximum).

PRIMARY/SECONDARY (P/S)? Enter P (or press [[RETURN]]) if the primary format for that file is to shown. Enter S if the secondary format for that file is to be shown. (See the discussion of the ADD command for more information on primary and secondary formats for a file.) DEFAULT is P.

PROCEDURE Enter the name of an existing procedure for which relationships are to be shown (20 characters maximum).

Update

Updates an association between entities which was established with the ADD command.

Syntax

```
[option] UPDATE subcommand U
```

Use the UPDATE command to change the association for an element, a file, or a procedure established with the ADD command. A complete list of entities for which an association can be changed is given below under Subcommands.

The subcommand identifies the type of entity association that is changed. For example, UPDATE CATEGORY, changes the description of the association or the element alias name.

Subcommands

Any of the following subcommands can be used with the UPDATE command:

CATEGORY	updates a data element to a category association
CLASS	updates a data element to a user security class association
CLASS-FILE	updates a file to a user security class association
CLASS-GROUP	updates a group to a security class association
FILE	updates a data element to a file association
FILE-LOC	updates a file to a location association
GROUP	updates a data element to an HP Inform group association
PROCEDURE	updates a data element to a procedure association
PROCEDURE-LOC	updates a procedure to a location association

Example

```
> UPDATE CATEGORY
           CATEGORY> Marketing          <       Category's name.!!
            ELEMENT> Account            <       Element to be updated.
  EDIT DESCRIPTION(Y/N)? N                      No, to edit description
                                                of the relationship.
  ELEMENT ALIAS:
    ACCOUNT                                     Current alias.
      ELEMENT ALIAS> Account-Marketing <        New element alias.
           ELEMENT>                             [[RETURN]] to end the
                                                command.
 >                                              New command prompt.
```

Subcommand Prompts

For the prompts shown with an asterisk (*), you must enter a response other than [[RETURN]] in order to implement the command string. Two asterisks (**) next to a prompt indicate that this prompt and those which follow are repeated until [[RETURN]] is pressed in response to this prompt.

You enter: You are prompted for:

COMMAND	SUB-COMMAND		PROMPT	
UPDATE	CATEGORY	*	CATEGORY	
		**	ELEMENT	
			ELEMENT DESCRIPTION (Y/N)?	
		*	EDIT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES(Y/N)?	Issued only if Yes to EDIT DESCRIPTION.
			ELEMENT ALIAS	
UPDATE	CLASS	*	CLASS	
		**	ELEMENT	
			EDIT DESCRIPTION(Y/N)?	
		*	EDDIT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES(Y/N)?	Issued only if Yes to EDIT DESCRIPTION.
			ACCESS CAPABILITY	
UPDATE	CLASS-FILE	*	CLASS	
		**	GROUP	
		*	EDIT COMMAND>>	
UPDATE	FILE	*	FILE	
			PRIMARY/SECONDARY(P/S)?	Issued if file type is KSAM, MPEF or MPER.
		**	ELEMENT	
			EDIT DESCRIPTION(Y/N)?	
			EDIT COMMANDS>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES(Y/N)	Issued only if Yes to EDIT DESCRIPTION.
			1. if file type is MAST:	
			ELEMENT ALIAS	
			NEW KEY ELEMENT(N/Y)?	Issued if element is not the key.

COMMAND	SUB-COMMAND		PROMPT	
			2. if file type is AUTO, MPEF, MPER or FORM:	
			ELEMENT ALIAS	
			FIELD NUMBER	Issued if file type is FORM.
			3. if file type is DETL:	
			ELEMENT ALIAS	
			PATH FILE	
			SORT ELEMENT	Issued if element is a search item.
			CANCEL PRIMARY PATH(N/Y)?	Issued if element is a search item and the primary path.;
			PRIMARY PATH(N/Y)?	Issued if element is a search item but is not the primary path.
			4. if file type is KSAM:	
			ELEMENT ALIAS	
			CANCEL KEY(N/Y)?	Issued if element is in the Primary format and is currently a KSAM key.
			KEY ELEMENT(N/Y)?	Issued if element is in the Primary format, is not currently a KSAM key and element is not TYPE B, S or *.
			NEW PRIMARY KEY(N/Y)?	Issued if element is in the Primary format, is a key element but not the primary key.
			CANCEL PRIMARY KEY(N/Y)?	Issued if element is in the Primary format and is currently a primary key.
			DUPLICATE KEYS(N/Y)?	Issued if element is in the Primary format, is a key element and duplicate keys are currently not allowed.
			CANCEL DUPLICATE KEYS(N/Y)?	Issued if element is in the Primary format, is a key element and duplicate keys are currently allowed.
UPDATE	FILE-LOC	*	LOCATION	
		**	FILE	
			EDIT DESCRIPTION(Y/N)?	
		*	EDIT COMMANDS>>	Issued if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES(Y/N)?	Issued if Yes to EDIT DESCRIPTION.
			FILE SIZE	Issued if file type is MPEF or KSAM.

COMMAND	SUB-COMMAND		PROMPT	
			FILE ALIAS	
UPDAT	GROUP	*	GROUP	
		**	ELEMENT	
			EDIT DESCRIPTION(Y/N)?	
		*	ELEMENT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES(Y/N)?	Issued only if Yes to EDIT DESCRIPTION.
			ELEMENT ALIAS	
			PARENT FILE	Issued if file belongs to more than one database.
			VALUE AS A LINK	
			CHANGE ELEMENT TO DISPLAY(N/Y)?	Issued if element is currently nondisplay.
			CHANGE ELEMENT TO NONDISPLAY(N/Y)?	Issued if element is currently display.
UPDATE	PROCEDURE	*	PROCEDURE	
		**	ELEMENT	
			EDIT DESCRIPTION(Y/N)?	
		*	EDIT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES(Y/N)?	Issued only if Yes to EDIT DESCRIPTION.
			ELEMENT ALIAS	
UPDATE	PROCEDURE-LOC	*	LOCATION	
		**	PROCEDURE	
			EDIT DESCRIPTION(Y/N)?	
		*	EDIT COMMAND>>	Issued only if Yes to EDIT DESCRIPTION.
			EDIT ATTRIBUTES(Y/N)?	Issued only if Yes to EDIT DESCRIPTION.
			PROCEDURE ALIAS	

Prompts

Depending on the subcommand used, one or more of the following prompts is issued:

ACCESS CAPABILITY — Enter a new type of capability to be given to the user security class for the file or element using one of the following:

If the subcommand is:

CLASS-FILE - enter one of the following:

R	= read only
W	= read, write, modify
X	= null read/write list “(/)”

CLASS - enter one of the following:

R	= read only
U	= read and update only
M	= read, write and modify
X	= null read/write list “(/)”

Refer to the IMAGE Reference Manual for a detailed explanation of the meanings of user class access capabilities. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

CANCEL DUPLICATES (N/Y)? — This prompt generates data that will be used by the DICTCDE utility. Enter Y or Yes if duplicate key values are no longer allowed for this KSAM file. DEFAULT is N.

CANCEL KEY (N/Y) — Enter a Y if the data element is no longer to be a key in the KSAM file.

CANCEL PRIMARY KEYS(N/Y)? — This prompt generates data that will be used by the DICTCDE utility. Enter Y or Yes if this element is no longer a primary key. DEFAULT is N.

CANCEL PRIMARY PATH(N/Y)? — Enter Y or Yes if the data element is no longer to be the primary path.

CATEGORY — Enter the name of an existing category used in the association (20 characters maximum).

CHANGE ELEMENT TO DISPLAY (N/Y)? — Enter Y to change the element to a display element. This means it will be displayed for reporting on HP Inform's Data Names Menu.

CHANGE ELEMENT TO NONDISPLAY (N/Y) — Enter Y to change the element to a nondisplay element. This means it will not be displayed for reporting on HP Inform's Data Names Menu.

CLASS — Enter a positive numeric value which is the identifying number (0 to 9999) of the existing user security class used in the association.

If the subcommand used is CLASS-GROUP, enter a numeric value (0 to 9999) which is the identifying number of the INFO type user security class.

DUPLICATES (N/Y)? This prompt generates data that will be used by the DICTCDE utility. Enter Y or Yes to allow this key to have duplicate key values for this KSAM file. DEFAULT is N.

EDIT ATTRIBUTES (Y/N)? Enter an N and the prompts to edit the attributes will not be generated (see Subcommand Prompts for which attributes can be changed).

EDIT COMMAND>> Enter one of the EDIT COMMANDS described in Using Edit Description Commands in Section III.

EDIT DESCRIPTION (Y/N)? Enter an N if editing is not to be done for the textual description of the association.

ELEMENT Enter the name of an existing data element used in the association (20 characters maximum).

ELEMENT ALIAS Enter the name by which the data element is known within the file. The maximum characters allowed are:

15	for FORM
16	for MAST, AUTO or DETL
20	for MPEF, MPER or KSAM
60	for CATEGORY
20	for an HP Inform GROUP
16	for PROCEDURE

DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

Note that HP Inform and Report will not be able to retrieve the values of an element from an IMAGE data set if the response to the ELEMENT ALIAS prompt of the ADD FILE command string is not the actual name of the data element in the data set.

FIELD NUMBER Enter the field number of this element on the VPLUS form. DEFAULT is 0.

FILE If the subcommand is:

CLASS-FILE or GROUP - enter the name of a file (20 characters maximum). The file type must be one of the following:

MAST	IMAGE manual master data set
AUTO	IMAGE automatic master data set
DETL	IMAGE detail data set

KSAM	KSAM file
MPEF	MPE sequential file
MPER	MPE relative file
VPLS	VPLUS forms file

FILE - enter the name of a file (20 characters maximum). The file type must be one of the following:

MAST	IMAGE manual master data set
AUTO	IMAGE automatic master data set
DETL	IMAGE detail data set
KSAM	KSAM file
MPEF	MPE sequential file
MPER	MPE relative file
VPLS	VPLUS form

FILE ALIAS Enter a new name by which the file is to be known within the location (8 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

FILE SIZE Enter a new value for the number of records of the file in the location (0 to 999999999). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

GROUP Enter the name of an existing HP Inform group used in the association (20 characters maximum).

KEY ELEMENT (N/Y)? Enter a Y if the data element is to become the key element in the KSAM file.

LOCATION Enter the name of an existing location used in the association (20 characters maximum).

NEW KEY ELEMENT (N/Y)? Enter a Y if the data element is to become the new key item for the MAST file.

NEW PRIMARY KEY (N/Y)? This prompt generates data that will be used by the DICTCDE utility. Enter a Y or Yes if this element is to be the new primary key for this file. Any previous primary key will be canceled when Y or Yes is entered as a response to this prompt. Enter N or No is this element is not to be the primary key. DEFAULT is N.

PARENT FILE Enter the name of an existing file (20 characters maximum) that is the parent of the database set specified in response to the FILE prompt. The file type must be type BASE (IMAGE database).

PATH FILE Enter the name of an existing file of type MAST or AUTO that is the new chain header for a detail set path (20 characters maximum). A blank value

indicates that the data element is no longer to be a search item in the DETL file. DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

PRIMARY PATH (N/Y)? Enter a Y if the data element is to become the primary path. If another data element already exists as the primary path, a Y response automatically cancels it.

PRIMARY/SECONDARY (P/S)? Enter P (or press [[RETURN]]) if the element to be updated is in the primary format for the file. Enter S if the element is in the secondary format for the file. (See the discussion of the ADD command for more information on primary and secondary formats.) DEFAULT is P.

PROCEDURE Enter the name of an existing procedure used in the association (20 characters maximum).

PROCEDURE ALIAS Enter a new name by which the procedure is to be known within the location (8 characters maximum). DEFAULT: Pressing [[RETURN]] in response to this prompt indicates no change.

SORT ELEMENT Enter the name of an existing data element that is the new sort item for a sorted chain (20 characters maximum). This prompt is only issued if a path file exists. A blank value (pressing the space bar followed by [[RETURN]]) indicates that the chain is no longer sorted.

DEFAULT: Pressing [[RETURN]] following this prompt indicates no change

VALUE AS A LINK Enter -1, 0, or a positive integer to specify which elements should preferably be used in linking files. The numbers indicate the following:

-1	The element can not be used for linking.
0	The element may or may not be used for linking (DEFAULT).
1 or greater	The element should be used as a link when possible; elements assigned a positive link value form a prioritized list to be used when it is necessary to link files (the lower the positive integer, the higher the priority).

5 Dictionary/3000 Utilities

Overview

Dictionary/3000 provides a set of interactive utilities that allow a Dictionary or database administrator to easily create and maintain entries in the Dictionary. The Dictionary utilities can be used to create, maintain or load IMAGE database entries and to create VPLUS forms file entries in the Dictionary. The utilities can also be used to clean the Dictionary of superfluous information that may accumulate after continued DICTDBM use.

Table 5-1 gives a summary of the utilities with a brief description of their function.

Table 5-1. The Dictionary Utilities

Utility	Function
DICTDBA (Database Audit)	reports on the usage statistics and checks the internal linkages of an IMAGE database.
DICTDBC (Database Creation)	uses the information in the Dictionary to create a schema and root file for an IMAGE database.
DICTDBD (Database Definition)	loads the existing definition of an IMAGE database into the Dictionary using the root file.
DICTDBU (Database Unload)	copies the data entries of an IMAGE database to disc or tape.
DICTDBL (Database Load)	loads data entries from tape or disc into the IMAGE database files.
DICTDBM, UTIL (Data Dictionary Cleanup)	cleans up data Dictionary after several months of DICTDBM use.
DICTVPD (VPLUS Forms File Definition)	loads the existing definition of a VPLUS forms file into the Dictionary.

Dictionary/3000 also provides an initialization utility and two extract utilities. The initialization utility, DICTINIT, is used to create and initialize the Dictionary. (For more information on DICTINIT, see Appendix C.) The extract utilities, DICTPDE and DICTCDE, can be used to generate PASCAL and COBOL definitions from the entities defined in the Dictionary. (For more information on DICTPDE, see Chapter 6. For more information on DICTCDE, see Chapter 7.)

Initiating the Utilities

After logging on, any of the Dictionary utilities can be run. The utilities in this chapter assume that the Dictionary exists in the PUB group of your log-on account. If the Dictionary you wish to use exists in a different MPE group and/or account, you must first identify it by issuing a file equation before running the utility. The file equation should be as follows:

```
FILE DICT.PUB=DICT.group.account
```

To initiate any of the utilities issue an MPE RUN command, as follows:

```
RUN utility.PUB.SYS
```

Changing Output Files

DICTDBM uses the formal file designator TRANOUT to direct output to your terminal and TRANLIST to direct output to the line printer. The other utilities use DICTOUT to direct output to your terminal and DICTLIST to direct output to the line printer. To redirect these output files, use MPE file equations to redefine the file designators prior to running the utility.

Using the Utilities to Restructure a Database

It is possible to make changes to the design of an existing IMAGE database by using the Dictionary utilities. The following steps should be performed when restructuring a database:

1. As a safety precaution, use the utility program, DBSTORE.PUB. SYS to store the old database, before performing any of the following steps.
2. Run the Database Unload utility (DICTDBU) program, copying the existing contents of a database to a disc file or a tape file.
3. Run the Database Maintenance program (DICTDBM) program and make the desired changes to the database by using the appropriate commands. For example, items may need to be moved from one data set to another or added to the middle of a data set, or data types or database/data set names may need to be changed, now is the time to make these changes.
4. Run the HP DBUTIL utility program, purging the old database. (See the IMAGE/3000 Reference Manual for details on DBUTIL.)
5. Run the Database Creation utility (DICTDBC) program, creating a new schema file and a new root file for the redesigned database.
6. Run the HP DBUTIL utility program, creating and initializing the new database. (See the IMAGE/3000 Reference Manual for details on DBUTIL.)
7. Run the Database Load utility (DICTDBL) program, loading the new database with the contents of the old database. The Database Load utility program compares the old schema file with the new schema file and, if necessary, transforms the data field formats of the old database to agree with the formats of the new database.

DICTDBA

Database Audit Utility

The Database Audit utility (DICTDBA) reports on the usage statistics and checks the linkages for an IMAGE database. The report can include information on synonyms and chains for master sets and chain statistics for detail sets. DICTDBA also checks for broken chains.

This utility provides two reports, one at the terminal and one is also printed on the line printer. The terminal report is a summary report, while the line printer report includes more detailed information.

With this utility, a database can be audited for synonym information, for chain information by search items, or for specific information on chain lengths.

The following description explains how to execute the DICTDBA utility program and gives an explanation of each prompt.

To execute DICTDBA enter the following MPE command:

```
RUN DICTDBA.PUB.SYS
```

After an acknowledgment message, a prompt is issued to determine the type of audit report to be generated as follows:

```
RUN MODE(SYNONYMS/CHAINS/LINKS)>
```

Only one of the following report types can be run each time DICTDBA is executed. The following is a description of each report:

SYNONYMS	the report displayed at the terminal shows the master set's name, the type of master, the number of entries in the master, the capacity of the master, and the usage percentage of the master. The line printer report additionally includes the number of entries with synonyms, and the shortest and longest synonym chain lengths.
CHAINS	the terminal report includes the same information as the SYNONYM report, plus information on each detail set. The line printer report includes statistics by search item on the number of chains, the shortest and longest chain length, and the average chain length. If a broken chain is detected, a warning is given.
LINKS	the terminal report includes the same information as the SYNONYM report, the CHAINS report, plus more specific information for broken chains. The line printer report additionally includes information on the forward and backward read counts for the search item. In this mode the chain entries are physically read to check the integrity of the chains. Therefore, this report takes appreciably longer to run in this mode.

Enter the type of report chosen. You may enter either the full name of the report as shown above or the first letter of the report name. If [[RETURN]] is pressed in response to this prompt, DICTDBA will terminate.

The following prompts are then issued to identify the database to be audited:

```
BASE>
BASE PASSWORD>
MODE>
```

Enter the name of the database in response to the BASE prompt, and enter a password that grants read access to the entire database in response to the PASSWORD prompt. If an incorrect password is entered, the prompt is reissued. Pressing [[RETURN]] in response to the prompt for BASE terminates the utility.

Enter the access mode to be used to open and read the database in response to the MODE prompt. If [[RETURN]] is pressed in response to this prompt, the database will be accessed in the default mode, which is 5. (See the IMAGE Reference Manual for details on “Access Modes”.) Note that when DICTDBA is run, the database will be locked on the data set level, unless the database is opened in modes 3, 7, 8.

DICTDBA will then begin to execute the report.

The following examples show how to execute DICTDBA and briefly explain the responses given to the prompts.

Example

The following is an example of a SYNONYM report:

```
:RUN DICTDBA.PUB.SYS                    Executes the audit utility.
<The Dictionary/3000 DB Auditor program banner appears here.>
RUN MODE(SYNONYMS/CHAINS/LINKS)> S      Select S for SYNONYM run mode.
BASE> SHPMGT                            Name of database to be audited.
BASE PASSWORD>                           Password is not displayed; MANAGER
                                        was used.
MODE>                                   [[RETURN]] entered; opens database
                                        in default access mode 5.
```

The Terminal SYNONYMS report is generated as follows:

```
PRODUCT-MASTER  M: 22/101  [21%]      Set name; set type; current number
                                       of entries/set capacity; capacity
                                      usage percentage.
PO-MASTER  A: 0/311  [0%]
ITEM-MASTER  M: 44/311  [14%]
END OF PROGRAM
```

The Line printer SYNONYMS report is generated as follows:

```
RUN MODE: SYNONYMS                      Identifies run mode selected.
  BASE: SHPMGT.PUB.HOWE                 Identifies the selected base.
PRODUCT-MASTER  M: 22/101  [21%]       Same information as terminal
                                       report.
```

```
  NUMBER OF ENTRIES WITH SYNONYMS: 0  [0%]
PO-MASTER  A: 0/311  [0%]
    SET IS EMPTY
ITEM-MASTER  M: 44/311   [14%]
NUMBER OF ENTRIES WITH SYNONYMS: 1  [2%]  Detail information on
  SHORTEST SYNONYM CHAIN LENGTH: 1        synonym and chain;
    LONGEST SYNONYM CHAIN LENGTH: 1       synonym % = (synonym count
                                          100)/entry count, in this
                                          case (1x100)/44 = 2%.
  END OF AUDIT RUN
```

The following is an example of a CHAINS report.

```
:RUN DICTDBA.PUB.SYS                  How to execute the utility.
<The Dictionary/3000 DB Auditor program banner appears here.>
RUN MODE(SYNONYMS/CHAINS/LINKS)> C    Select C for CHAINS run mode.
BASE> SHPMGT                          Name of base to be audited.
BASE PASSWORD>                        Password is not displayed;
                                      MANAGER was used.
MODE>                                 [[RETURN]] entered; opens
database
                                      in default access mode 5.
```

The Terminal CHAINS report is generated as follows:

```
PRODUCT-MASTER  M: 22/101  [21%]      Set name; set type; current number
                                      of entries/set capacity; capacity
                                    usage percentage.
PO-MASTER  A: 0/311  [0%]
ITEM-MASTER  M: 44/311  [14%]
EXPLODE  D: 397/1023  [38%]           DETAIL data set information
                                      included for CHAINS run mode.
ITEM-DETAIL  D: 0/507  [0%]
COSTING  D: 0/1008  [0%]
END OF PROGRAM
```

The Line printer CHAINS report is generated as follows:

```
RUN MODE:  CHAINS                     Identifies run mode selected.
  BASE: SHPMGT.PUB.HOWE               Identifies the selected base.
PRODUCT-MASTER  M: 22/101  [21%]      Same information as terminal
                                      report.
   NUMBER OF ENTRIES WITH SYNONYMS: 0  [0%]
```

```
PO-MASTER  A: 0/311  [0%]
     SET IS EMPTY
ITEM-MASTER  M: 44/311  [14%]
  NUMBER OF ENTRIES WITH SYNONYMS: 1  [2%]  Synonym chain information
    SHORTEST SYNONYM CHAIN LENGTH: 1        for master set.
      ONGEST SYNONYM CHAIN LENGTH: 1
EXPLODE  D: 397/1023  [38%]
   SEARCH ITEM: PRODUCT-NO             Information at the search item
                                       level for detail data set.
         NUMBER OF CHAINS: 22          Chain information begins.
      SHORTEST CHAIN LENGTH: 18
       LONGEST CHAIN LENGTH: 19
       AVERAGE CHAIN LENGTH: 18
   SEARCH ITEM: ITEM-NO
         NUMBER OF CHAINS: 44
      SHORTEST CHAIN LENGTH: 2
       LONGEST CHAIN LENGTH: 24
       AVERAGE CHAIN LENGTH: 9
ITEM-DETAIL  D: 0/507  [0%]
     SET IS EMPTY
COSTING  D: 0/1008  [0%]
     SET IS EMPTY
  END OF AUDIT RUN
```

The following is an example of a LINKS report.

```
:RUN DICTDBA.PUB.SYS                  How to execute the audit utility.
<The Dictionary/3000 DB Auditor program banner appears here.>
 RUN MODE(SYNONYMS/CHAINS/LINKS)> L   Select mode L for LINKS.
 BASE> SHPMGT                         Name of base to be audited.
 BASE PASSWORD>                       Password is not displayed;
                                      MANAGER was used.
 MODE>                                [[RETURN]] entered; opens database
                                      in default access mode 5.
```

The Terminal LINKS report is generated as follows:

```
PRODUCT-MASTER  M: 22/101  [21%]      Set name; set type; current number
                                       of entries/set capacity; capacity
                                       usage percentage.
 PO-MASTER  A: 0/311  [0%]
```

```
ITEM-MASTER  M: 44/311  [14%]
EXPLODE  D: 397/1023  [38%]
ITEM-DETAIL  D: 0/507  [0%]
COSTING  D: 0/1008  [0%]
END OF PROGRAM
```

The Line printer LINKS report is generated as follows:

```
RUN MODE: LINKS                          Identifies run mode selected.
  BASE: SHPMGT.PUB.HOWE                  Name of base to be audited.
PRODUCT-MASTER  M: 22/101  (21%)         Same information as terminal
                                         report.
   NUMBER OF ENTRIES WITH SYNONYMS: 0  (0%)
     FORWARD SERIAL READ COUNT: 22
     REVERSE SERIAL READ COUNT: 22
PO-MASTER  A: 0/311  (0%)
     SET IS EMPTY
ITEM-MASTER  M: 44/311  (14%)
   NUMBER OF ENTRIES WITH SYNONYMS: 1  (2%)  Synonym information for
     SHORTEST SYNONYM CHAIN LENGTH: 1        the master set.
      LONGEST SYNONYM CHAIN LENGTH: 1
     FORWARD SERIAL READ COUNT: 44
     REVERSE SERIAL READ COUNT: 44
EXPLODE  D:  397/1023  (38%)
     FORWARD SERIAL READ COUNT: 397
     REVERSE SERIAL READ COUNT: 397
   SEARCH ITEM: PRODUCT-NO               Detail information by search item.
          NUMBER OF CHAINS: 22
     SHORTEST CHAIN LENGTH: 18
      LONGEST CHAIN LENGTH: 19
      AVERAGE CHAIN LENGTH: 18
      FORWARD CHAIN READ COUNT: 397
      REVERSE CHAIN READ COUNT: 397
   SEARCH ITEM:  ITEM-NO                 Second search item in the set.
          NUMBER OF CHAINS: 44
     SHORTEST CHAIN LENGTH: 2
      LONGEST CHAIN LENGTH: 24
      AVERAGE CHAIN LENGTH: 9
      FORWARD CHAIN READ COUNT: 397
```

```
      REVERSE CHAIN READ COUNT: 397
ITEM-DETAIL  D: 0/507  (0%)
     SET IS EMPTY
COSTING  D: 0/1008  (0%)
     SET IS EMPTY
END OF AUDIT RUN
```

DICTDBC

Database Creation Utility

The Database Creation utility (DICTDBC) creates a schema and a root file for an IMAGE database. Once the schema and root file are created, the database files can be created by using the IMAGE/3000 utility DBUTIL. Refer to the IMAGE/3000 Reference Manual for detailed information on how to use DBUTIL.

DICTDBC prompts for information that identifies which database definition in the data Dictionary is to be used. It also prompts for information that is used by the schema processor when generating a schema and a root file. The IMAGE/3000 utility DBSCHEMA is automatically executed by DICTDBC. (For details on DBSCHEMA, see the IMAGE/3000 Reference Manual.)

DICTDBC will terminate when DBSCHEMA terminates. Once the schema and root file are created, the DBUTIL utility can be run to create the database files. (DICTDBC does not execute DBUTIL.) If errors prevent a root file from being created, the corrections must be made in the Dictionary before you can re-execute DICTDBC. The following description tells how to execute DICTDBC and gives an explanation of each prompt.

To execute DICTDBC enter the following MPE command:

```
RUN DICTDBC.PUB.SYS
```

After DICTDBC issues an acknowledgment message, you are prompted to enter a password as follows:

```
DICTIONARY PASSWORD>
```

The password entered must grant at least PROGRAMMER level access to the Dictionary. (See Appendix C for an explanation of the levels of access.) If [[RETURN]] is pressed in response to this prompt, DICTDBC will terminate.

Note that the password is not displayed on the terminal as it is entered. However, if the password is not entered correctly, an error message is displayed and the prompt reissued.

After the password is accepted, DICTDBC issues the following prompt:

```
BASE>
```

Enter the name of the database for which a schema and root file are to be created. A definition for this database must already exist in the Dictionary. If it does not, an error message is displayed and the prompt is reissued. If [[RETURN]] is pressed in response to this prompt DICTDBC will terminate.

Next, DICTDBC prompts for information to be used by the schema processor. The first prompt allows you to specify the options to be used when the schema is processed:

```
CONTROL LINE>
```

Any of the following options may be entered in response to the above prompt. These options are the same ones as described for the $CONTROL COMMAND in the IMAGE/3000 Reference Manual, except for BLOCKMAX. Note that the BLOCKMAX option is defined differently here. One or more of these options may be entered, separated

by commas.

LIST	causes each source record of the schema to be printed on the listfile.
NOLIST	specifies that only source records with errors be printed on the listfile.
ERRORS=nnn	sets the maximum number of errors to nnn. nnn may have a value between 0 and 999, inclusive. If more than three errors are detected, the Schema Processor terminates. The default value is 100.
LINES=nnnnn	sets the number of lines per page on the listfile nnnnn. nnnnn may have a value between 4 and 32767, inclusive. The default is 60 if listfile is a line printer and 32767 if it is not.
ROOT	causes the Schema Processor to create a root file if no errors are detected in the schema.
NOROOT	prevents the Schema Processor from creating a root file.
BLOCKMAX=nnnn	sets the maximum physical block length (in words) for a data set. If you do not include this parameter, the value declared in the data Dictionary is used. If you entered [[RETURN]] to the prompt for BLOCKMAX in the Dictionary, the default value of 512 is used. Refer to the RELATE FILE command string in Section IV for an explanation of BLOCKMAX.
TABLE	causes the Schema Processor to write a table of summary information about the data sets to the listfile device if no errors are detected.
NOTABLE	suppresses the TABLE option.

The defaults for the control line options are: LIST, ROOT, and TABLE.

DICTDBC continues prompting for schema processor information as follows:

```
SCHEMA FILE>
```

Enter the name of the schema file. It can be a temporary or a permanent file. To create a process temporary file, press [[RETURN]] in response to this prompt. If the schema file is to be a permanent file, enter the name of the file as your response. If it is a new file, DICTDBC will create the file. If the file already exists, DICTDBC will issue a warning before over-writing the file.

DBSCHEMA then generates an output listing of the schema and root file. The response made to the next prompt determines where this listing is sent:

```
LIST FILE>
```

Enter one of the following options in response to this prompt:

LP	sends the listing to the line printer.
NULL	suppresses the listing.

filename sends the listing to the existing or to a new disc file. Before over-writing an existing file, you are prompted to approve purging the existing file's contents.

**filename* sends the listing to the file identified in the back referenced file equation.

Pressing [[RETURN]] in response to this prompt sends the listing to the terminal. If security was only defined for the data items and now the security should only apply to the data set, enter YES to the following prompt:

```
APPLY SECURITY JUST TO SET LEVEL(N/Y)?>
```

If NO is entered or [[RETURN]] is pressed in response to this prompt, the security specified for the data items is written into the schema file. If YES is entered in response to this prompt, DICTDBC will write the set level security into the schema file. The set level security is derived from the class or classes assigned to the set in the data Dictionary or, if none are specified, it is extrapolated from data item security specifications.

DICTDBC then executes the schema processor. Regardless of whether a schema and a root file are created, DICTDBC terminates after the schema processor has executed. If errors prevent a root file from being created, the errors must be corrected before DICTDBC can be re-executed.

The following example shows how to execute DICTDBC and briefly explains the responses to the prompts.

Example

```
:RUN DICTDBC.PUB.SYS                    Executes DICTDBC.
    <The Dictionary/3000 DB Creator program banner appears here.>
DICTIONARY PASSWORD>                    Data Dictionary's password.
BASE> SHPMGT                            Name of base to be created.
CONTROL LINE>                           [[RETURN]] pressed; default used.
SCHEMA FILE>                             [[RETURN]] pressed; session temporary
                                        file created.
LIST FILE>                              [[RETURN]] pressed; listing sent to
                                        terminal.
APPLY SECURITY JUST TO SET LEVEL(N/Y)?>   [[RETURN]] pressed; use item level
                                        security.
SCHEMA GENERATION                       DICTDBC executes for you.
```

DBSCHEMA generates the following listing at your terminal:

```
DBSCHEMA PROCESSOR
PAGE 1   <The IMAGE/3000 DBSCHEMA program banner appears here.>
BEGIN DATABASE SHPMGT;
PASSWORDS:
              1 SALESPER;
              2 RECEIVING;
```

```
                3 BUYER;
                4 MANAGER;
ITEMS:
     BUYER-NO,           X2       (3/4);
     DESCRIPTION,        X30           ;
     ITEM-NO,            X8            ;
     LIST-PRICE,         P12           ;
     PO-NUMBER,          U4            ;
     PRODUCT-NO,         U8            ;
     PURCHASE-COST,      P8            ;
     PURCHASE-DATE,      X6            ;
     PURCHASE-QTY,       P8            ;
     QTY-ALLOCATED,      P8            ;
     QTY-ON-HAND,        P8            ;
     QTY-ON-ORDER,       P8        (/2);
     QTY-REC,            P8            ;
     REORDER-PT,         P8            ;
     UNIT-COST,          P12     (/3,4);
     VENDOR-NO,          X8      (/3,4);
SETS:
NAME:   PRODUCT-MASTER,  MANUAL    (1,2/3,4);
ENTRY:  PRODUCT-NO      (1),
        DESCRIPTION,
        QTY-ON-HAND,
        QTY-ALLOCATED,
        UNIT-COST,
        LIST-PRICE;
CAPACITY: 101;
NAME:   PO-MASTER,       MANUAL    ;
ENTRY:  PO-NUMBER       (1);
CAPACITY: 311;
NAME:   ITEM-MASTER,     MANUAL     (1,2/3,4);
ENTRY:  ITEM-NO         (3),
        DESCRIPTION,
        QTY-ON-HAND,
        QTY-ON-ORDER,
        QTY-ALLOCATED,
        PURCHASE-COST,
```

```
        REORDER-PT;
CAPACITY: 311;
NAME:   ITEM-DETAIL,     DETAIL      (/3,4);
ENTRY:  ITEM-NO         (!ITEM-MASTER     ),
        VENDOR-NO,
        BUYER-NO;
CAPACITY: 507;
NAME:   EXPLODE,         DETAIL   (1,2,3/4);
ENTRY:  ITEM-NO         ( ITEM-MASTER     ),
        PRODUCT-NO      (!PRODUCT-MASTER  );
CAPACITY: 1023;
NAME:   COSTING,         DETAIL    (/2,3,4);
ENTRY:  ITEM-NO         ( ITEM-MASTER     (PURCHASE-DATE   )),
        PURCHASE-DATE,
        PURCHASE-QTY,
        PO-NUMBER       (!PO-MASTER       ),
        QTY-REC;
CAPACITY: 1032;
END.
 DATA SET      TYPE  FLD  PT  ENTR  MED  CAPACITY  BLK  BLK   DISC
   NAME              CNT  CT  LGTH  REC            FAC  LGTH  SPACE
PRODUCT-MASTER    M  6    1   29    39    101      13   508   36
PO-MASTER         M  1    1   2     12    311      42   507   36
ITEM-MASTER       M  7    3   29    49    311      10   491   132
ITEM-DETAIL       D  3    1   9     13    507      39   510   56
EXPLODE           D  2    2   8     16    1023     31   498   136
COSTING           D  5    2   13    21    1032     24   506   176
                           TOTAL DISC SECTORS INCLUDING ROOT: 583
NUMBER OF ERROR MESSAGES: 0
ITEM NAME COUNT: 16      DATA SET COUNT: 6
ROOT LENGTH: 630     BUFFER LENGTH: 510     TRAILER LENGTH: 256
ROOT FILE SHPMGT CREATED.             Schema and root file created.
END OF PROGRAM
```

DICTDBD

Database Definition Utility

The Database Definition utility (DICTDBD) enters the definition for an existing IMAGE database into the Dictionary. DICTDBD creates the entries for the database, the data sets, and the data items by using the root file. The database is entered in the Dictionary as a BASE type file. The automatic, master and detail data sets are entered as AUTO, MAST, and DETL type files respectively. The data items are entered as elements. DICTDBD also establishes the associations of the data items to the data sets and the relationships of the data sets to the database.

However, the existing security defined for the database is not transferred to the Dictionary by DICTDBD. To create or redefine security for the database, use the Dictionary Database Maintenance program (DICTDBM). See Section III of this manual on how to define security for a database.

Before the definition of the database is entered into the Dictionary, DICTDBD checks the existing entries to ensure that the database does not already exist in the Dictionary. If the database name is the same as the name for an existing entry, a message is issued. You are given the option to change the name of the database or to terminate DICTDBD. DICTDBD also checks the existing entries to avoid entering duplicate or redundant entries for data sets and data items. If any of the names used for the data sets or data items is the same as the name for an existing entry, a message is issued. For duplicate data sets and data items, you are given the option to either enter a new name for that entity or to use the existing entity in the Dictionary.

Whenever a name is changed for an entity being loaded into the Dictionary, the new name entered becomes the “primary” name for that entity. The original name then becomes the “alias” name for that entity in the Dictionary. Because HP Inform will reference these entities by their aliases, the original names used by VPLUS do not have to be changed to be consistent with the Dictionary. However, since DICTDBM uses the primary names for all entities, it is important to remember what the new primary names are for the entities whose names were changed. The following description explains how to execute DICTDBD and gives an explanation of each prompt. To execute DICTDBD enter the following MPE command:

```
RUN DICTDBD.PUB.SYS
```

After an acknowledgment message, DICTDBD prompts for a password to open and read the Dictionary. The prompt is:

```
DICTIONARY PASSWORD>
```

The password that you enter must grant at least PROGRAMMER level access to the Dictionary. (See Appendix C for an explanation of the levels of access.) If you press [[RETURN]] in response to this prompt, the utility terminates. The password that you enter is not displayed on your terminal, but if it is not entered correctly, an error message is displayed and the prompt reissued.

After accepting the password you entered, the utility prompts you to enter specific database identification information as follows:

```
BASE>
BASE PASSWORD>
MODE>
```

You respond to the prompt for BASE by entering the name of the database to be defined in the Dictionary for you. The BASE PASSWORD prompt should grant read access to the utility for the named database.

The MODE prompt indicates the access mode to be used by the utility to open and read the database. The database can be opened in any mode. If [[RETURN]] is entered as the response, the base is opened in the default mode which is 5. (Refer to the IMAGE/3000 Reference Manual for details on “Access Modes”.)

The process of transferring the definition of the database begins after you respond to the prompt for MODE. A message “LOADING DATA DICTIONARY” is displayed. The utility begins by checking whether the entire definition of the database will fit in the Dictionary. If it will not, the following prompt is issued:

```
ENTIRE DEFINITION OF DATABASE WILL NOT FIT IN DICTIONARY.
PROCEED (N/Y)?
```

A response of “N” or [[RETURN]] terminates the utility. Responding “Y” allows the utility to continue; however, the utility will issue an error message and terminate when any one of the data sets is full. Note that if the entire definition of the database will not fit in your Dictionary, you can run DICTINIT to re-initialize Dictionary with larger capacities specified (see Appendix C).

Next, the utility checks the schema of the named database with the entries in the Dictionary. If there are no entries identical to those in your database, the transfer is completed and the DICTDBD utility program is terminated. You can immediately execute the DICTDBM program and see the transferred database definition.

If an entry already exists for the database name, the following message and prompt are issued:

```
*ERROR: BASE ALREADY DEFINED IN DATA DICTIONARY
     DO YOU WANT TO LOAD UNDER A DIFFERENT NAME(Y/N)?>
```

A response of an “N” terminates the execution of the program. If a “Y” or [[RETURN]] is entered, the following prompt to rename your database is issued:

```
NEW BASE NAME>
```

The name you enter must be unique - that is, it must not already exist in the Dictionary. If it does, an error message is displayed and the prompt reissued. The utility program terminates if [[RETURN]] is entered as your response to this prompt.

Next the ITEMS as listed in the schema are checked. If there are no existing Dictionary entries, the loading process continues. If an entry already exists and is compatible with the item in the database, the following message and prompt are displayed:

```
COMPATIBLE DEFINITION ALREADY IN DICTIONARY FOR ELEMENT element
USE EXISTING DEFINITION(N/Y)?>
```

If you enter a “Y”, the loading process continues. If you enter an “N” or [[RETURN]], you are prompted to rename the element as follows:

```
NEW PRIMARY ELEMENT NAME>
```

A unique name for the element must be entered. The name for the item in your database becomes the alias name for the item in the Dictionary. If you enter [[RETURN]], the program does not load the element into the Dictionary and issues the following warning:

```
*WARNING: ELEMENT HAS NOT BEEN LOADED INTO DICTIONARY
```

Then the loading process continues. Each time an existing compatible entry is encountered, the above sequence occurs. If an incompatible item is encountered, the following sequence occurs:

```
INCOMPATIBLE DEFINITION ALREADY IN DICTIONARY FOR ELEMENT element
NEW PRIMARY ELEMENT NAME>
```

You must enter a unique element name in response to the prompt. If you enter [[RETURN]], the program does not load the element into the Dictionary and issues a warning message.

After checking the data items, the utility compares the data set definitions with existing Dictionary entries. The definition of a data set includes the data items within the set. If there are no other identical entries, the transfer is completed and the program terminates.

If an identical entry for the data set exists, the following message and prompt are issued:

```
DEFINITION ALREADY IN DICTIONARY FOR SET set
USE EXISTING DEFINITION(N/Y)?>
```

If you respond with an “N”, you are prompted to rename the set as follows:

```
 NEW PRIMARY FILE NAME>
```

A new, unique name, must be entered. By entering “Y” or [[RETURN]], the data set is not loaded and a warning message as follows is issued:

```
*WARNING: DATA SET HAS NOT BEEN LOADED INTO DICTIONARY
```

Each time the program encounters an existing identical definition, the above process is repeated. If an existing definition is incompatible, the following message and prompt are issued:

```
DEFINITION ALREADY IN DICTIONARY FOR SET set
NEW PRIMARY FILE NAME>
```

A unique name must be entered for the data set. If [[RETURN]] is used, a warning is issued.

The above process continues until the entire schema has been compared with existing Dictionary entries. When the comparison is complete, the transfer, where indicated by your responses to the prompts, is completed. The DICTDBM program could be executed to view the transferred definition.

The following examples show how to execute the Database Definition utility with a brief explanation of the prompts and responses.

In the first example, the database is unique, and no renaming is required. The second example shows a database that is not totally unique and the prompts issued by the utility to resolve the definition transfer.

Example

The following example shows the use of DICTDBD, using a unique database name:

```
:RUN DICTDBD.PUB.SYS                How to execute this utility.
<The Dictionary/3000 DB Info Loader program banner appears here.>
DICTIONARY PASSWORD>                Password that grants modify
                                    access used.
BASE> SHPMGT                        Name of existing IMAGE database.
BASE PASSWORD>                      MANAGER is password used.
MODE>                                 [[RETURN]] pressed; default used.
LOADING DATA DICTIONARY             Utility loading definition of
                                    base into data Dictionary.
END OF PROGRAM
```

The following example shows how DICTDBD is used to rename the database name:

```
:RUN DICTDBD.PUB.SYS              How to execute this utility.
<The Dictionary/3000 DB Info Loader program banner appears here.>
DICTIONARY PASSWORD>                Password that grants modify access.
BASE> SHPMGT                      Base to be defined in Dictionary.
BASE PASSWORD>                    MANAGER was used.
MODE>                             [[RETURN]] pressed; default used.
LOADING DATA DICTIONARY
*ERROR: BASE ALREADY DEFINED IN DATA DICTIONARY
DO YOU WANT TO LOAD UNDER A DIFFERENT NAME(Y/N)?>Y
NEW BASE NAME> MYSHPM             Will be name of base in Dictionary.
 COMPATIBLE DEFINITION ALREADY IN DICTIONARY FOR ELEMENT DESCRIPTION
DO YOU WANT ELEMENT TO HAVE A DIFFERENT NAME(N/Y)?>   [[RETURN]] pressed.
 COMPATIBLE DEFINITION ALREADY IN DICTIONARY FOR ELEMENT ITEM-NO
DO YOU WANT ELEMENT TO HAVE A DIFFERENT NAME(N/Y)?> Y
NEW PRIMARY ELEMENT NAME> MY-ITEM-NO
  COMPATIBLE DEFINITION ALREADY IN DICTIONARY FOR ELEMENT QTY-REC
DO YOU WANT ELEMENT TO HAVE A DIFFERENT NAME(N/Y)?> Y
NEW PRIMARY ELEMENT NAME>             [[RETURN]] pressed.
*WARNING: DATA ELEMENT QTY-REC HAS NOT BEEN LOADED INTO DICTIONARY
  INCOMPATIBLE DEFINITION ALREADY IN DICTIONARY FOR ELEMENT PO-NUMBER
NEW PRIMARY ELEMENT NAME>             [[RETURN]] pressed.
```

```
*WARNING: DATA ELEMENT PO-NUMBER HAS NOT BEEN LOADED INTO DICTIONARY
 DEFINITION ALREADY IN DICTIONARY FOR SET PRODUCT-MASTER
USE EXISTING DEFINITION(N/Y)?> Y      "Y" to use existing definition.
 DEFINITION ALREADY IN DICTIONARY FOR SET ITEM-MASTER
USE EXISTING DEFINITION(N/Y)?>        [[RETURN]] pressed; existing
                                      definition not used.
NEW PRIMARY FILE NAME> PARTS-MASTER
 DEFINITION ALREADY IN DICTIONARY FOR SET EXPLODE
USE EXISTING DEFINITION(N/Y)?> N
NEW PRIMARY FILE NAME>                [[RETURN]] pressed.
*WARNING: DATA SET EXPLODE HAS NOT BEEN LOADED INTO DICTIONARY
 INCOMPATIBLE DEFINITION ALREADY IN DICTIONARY FOR SET COSTING
NEW PRIMARY FILE NAME> MY-COSTING
END OF PROGRAM
```

DICTDBU

Database Unload Utility

The Database Unload utility (DICTDBU) unloads the contents of an existing IMAGE database. The database can be unloaded and stored either to a disc file or to a tape file. Additionally, the database can be edited during the unload process. Editing the unload process allows you to selectively unload, that is, choose whether or not to unload a particular data set. When the unloading process is complete, an audit report is produced.

The following description tells how to execute the DICTDBU utility program and gives an explanation of each prompt.

To execute this utility enter the following MPE command:

```
RUN DICTDBU.PUB.SYS
```

After an acknowledgment message, the unload utility issues the following prompt to establish a file for the storage of the database content:

```
STORE FILE>
```

If the database is to be stored on disc, enter a file name in response to the prompt. The utility builds the file for you if it is a new file. For an existing file, the utility issues a warning message and waits for you to indicate that it can be overwritten before it continues.

If the database is to be stored on tape, enter [[RETURN]] in response to the STORE FILE prompt. The utility issues the following message:

```
TAPE FILE REQUESTED(Y/N)?>
```

If you enter a “Y” or [[RETURN]], a request to mount a tape is issued on the system console; otherwise, the prompt for STORE FILE is reissued. If necessary, the utility can handle multiple tape reel stores.

When the utility completes the unloading process, it generates an audit report about the unloading process. You specify where the report is to be sent by responding to the following prompt:

```
LIST FILE>
```

Enter one of the following options in response to this prompt:

LP	sends the report to the line printer. If there is no device LP configured on your system, use a file equation to redirect DICTLIST to another device.
NULL	suppresses the audit listing.
filename	sends the report to the existing or to a new disc file. Before over-writing an existing file, you are prompted to approve purging the existing file's contents.
**filename*	sends the report to the file identified in the back-referenced file equation. Pressing [[RETURN]] in response to this prompt sends the report to your terminal.

After accepting your response to the prompt for LIST FILE, the utility requests the name of the database to be unloaded with the following prompt:

```
BASE>
```

Enter the name of the base in response to this prompt. If [[RETURN]] is entered, an error message is displayed and the program is terminated.

Next, you are prompted to enter a password as follows:

```
BASE PASSWORD>
```

In response to this prompt, enter a password that grants read access to the named database. The password is not displayed as you enter it. If the password entered is incorrect or does not grant read access to the database, a database operation error occurs during the unload process. The utility issues an error message which tells you why the database error occurred and then terminates.

To complete the information required to unload the database, the following prompt is issued:

```
MODE>
```

Your response to the MODE prompt indicates the access mode to be used to open the database. Any valid access mode is acceptable as a response to this prompt. If [[RETURN]] is entered, the program uses the default mode, which is 3. (Refer to the IMAGE/3000 Reference Manual for details on “Access Modes”.)

If the database named in response to the BASE prompt cannot be found, the following message is issued after the prompt for access mode:

```
*ERROR: NO SUCH DATABASE
```

Prompts are then repeated starting with the BASE prompt.

Next, prompts pertaining to the unloading process method are issued. The first prompt is as follows:

```
UNLOAD AUTOMATIC MASTER SETS(N/Y)?>
```

If automatic master sets are to be converted into manual master sets, you should respond with a “Y”; otherwise, it is unnecessary to unload automatic master sets.

The next prompt identifies how the detail sets are to be unloaded. The prompt is as follows:

```
UNLOAD DETAIL SETS BY CHAIN(Y/N)?>
```

Although it takes longer to unload detail data sets with a chained read, improvements in access time of chained reads can be seen after the database is reloaded. Enter “N” as your response if you do not want to unload the sets with a chained read.

The following prompt allows you to choose which data sets are or are not to be unloaded:

```
UNLOAD EDIT(N/Y)?>
```

You must enter a “Y” to selectively unload the data sets. If you choose to selectively unload the data sets, you are prompted as follows:

```
set name    set type:    UNLOAD(Y/N/C/X)?>
```

You can choose only one of the unload types. A description of each is as follows:

Y	which is short for YES, specifies that the data set is to be unloaded. (This is the default.)
N	which is short for NO, specifies that this data set is not to be unloaded.
C	which is short for CONTINUE, specifies that this data set and all remaining data sets are to be unloaded.
X	which is short for EXIT, specifies that this and all remaining data sets are not to be unloaded and the program terminates.

Note that entering a “C” ends the process of unloading data sets with “editing”, but it does not terminate the program. To terminate the program, an “X” must be entered. If [[RETURN]] is pressed in response to this prompt, the data set is unloaded.

If you chose to unload with a chained read and also chose to unload with editing, you will be prompted during the editing process to enter the search item's name for the detail set as follows:

```
SEARCH ITEM FOR CHAINED UNLOAD>
```

You can enter the name or allow the program to use the default, which is the primary path for the detail data set.

As each data set is unloaded, the program identifies the data set and the number of entries unloaded on your terminal screen. When the entire unload process is completed, the message “UNLOAD COMPLETED” is displayed, and the utility program terminates. An audit report is generated and sent to the file you indicated with your response to the LIST FILE prompt.

The following examples show how to execute the Database Unload utility program with an explanation of the responses given to the prompts.

Example

```
:RUN DICTDBU.PUB.SYS                    How to execute this utility.
<The Dictionary/3000 DB Unloader program banner appears here.>
STORE FILE> TEMPSTOR                    Name of a disc file.
LIST FILE>                              [[RETURN]] sends listing to terminal.
BASE> HOUSES                            Name of base to be unloaded.
BASE PASSWORD>                          Password used: MANAGER.
MODE>                                   [[RETURN]] pressed; default used.
UNLOAD AUTOMATIC MASTER SETS(N/Y)?>     [[RETURN]] pressed.
UNLOAD DETAIL SETS BY CHAIN(Y/N)?>      [[RETURN]] pressed.
UNLOAD EDIT(N/Y)?> Y                    Unload with editing.
ZONING-MASTER    M: UNLOAD(Y/N/C/X)?> Y       Yes, unload this set.
CITY-MASTER      M: UNLOAD(Y/N/C/X)?> Y
COMMERCIAL       D: UNLOAD(Y/N/C/X)?> C       Unload all remaining sets
                                              without EDIT prompts.
```

```
SEARCH ITEM FOR CHAINED UNLOAD> ZONING-CODE
ZONING-MASTER     M:6/31
     6 ENTRIES UNLOADED IN <1 CPU-SEC
    CITY-MASTER       M:20/101
      20 ENTRIES UNLOADED IN <1 CPU-SEC
    LIST-PRICE-MSTR   A:42/307
      AUTO NOT UNLOADED
    BATH-MASTER       A:2/31
      AUTO NOT UNLOADED
    BEDS-MASTER       A:4/11
      AUTO NOT UNLOADED
    COMMERCIAL        D:10/209
      10 ENTRIES UNLOADED IN <1 CPU-SEC
    RESIDENTIAL       D:40/200
      40 ENTRIES UNLOADED IN <1 CPU-SEC
    UNLOAD COMPLETED
    END OF PROGRAM
```

Identifies the search item of the detail data set.

Identifies what is unloaded.

DICTDBL

Database Load Utility

The Database Load utility (DICTDBL) reloads the contents of an IMAGE database. A comparison between the old and the new schemes is made before the loading process begins. If changes have been made, you are prompted to enter the changed information. Additionally, you can selectively reload the database, that is, you are given a choice to load or not to load a particular data set.

When the utility program completes the loading process, you are provided an audit report containing information about the database load. The following description tells how to execute the DICTDBL utility program and gives an explanation of each prompt.

To execute this utility enter the following MPE command:

```
RUN DICTDBL.PUB.SYS
```

After an acknowledgment message, the utility program issues the following prompt:

```
STORE FILE>
```

If the store file is a disc file, enter the file's name as your response to the prompt. An error message is issued if the named file cannot be found and the prompt is reissued. If the store file is a tape file, press [[RETURN]] as your response to this prompt. Before generating a tape request at the system console, the utility prompts you as follows:

```
TAPE FILE REQUESTED(Y/N)?>
```

If you enter a “Y” or [[RETURN]], a request to mount a tape is issued on the system console; otherwise, the prompt for STORE FILE is reissued. When the utility completes the loading process, it generates an audit report of the loading process. You can specify where the report is to be sent with the following prompt:

```
LIST FILE>
```

Enter one of the following options in response to the above prompt:

LP	sends the audit listing to the line printer. If there is no device LP configured on your system, use a file equation to redirect DICTLIST to another device.
NULL	suppresses the audit listing.
filename	sends the audit listing to the existing or to a new disc file. Before over-writing an existing file, you are prompted to approve purging the existing file's contents.
**filename*	sends the listing to the file identified in the back referenced equation.

Pressing [[RETURN]] in response to this prompt sends the audit listing to your terminal.

After accepting your response to the prompt for LIST FILE, the utility displays the stored database's name, then prompts you to select a RUN MODE for the loading process. The RUN MODE is the operating mode to be used to load the database. The prompt is as follows:

```
RUN MODE(LOAD/EDIT/SHOW/EXIT)>
```

In response to the prompt, you can choose one of four options, which have the following meaning:

LOAD	indicates the named database is to be loaded.
EDIT	indicates the named database is to be loaded and editing prompts issued for each data set.
SHOW	indicates the named database is not to be loaded, but the schema and set entry counts are to be displayed.
EXIT	indicates the utility is to be terminated.

Pressing [[RETURN]] in response to this prompt indicates that the default for RUN MODE is used, which is LOAD. In an operating mode of LOAD, you are prompted for information only when schema changes are encountered. In the EDIT mode, you can select an operating mode for the data sets. An operating mode for a data set is similar to the operating mode for a database. Next, the utility prompts you to enter the new name for the data base with the following prompt:

```
NEW BASE NAME>
```

If the name of the stored database has not changed, including qualifier information such as MPE group and account, press [[RETURN]] as your response; otherwise, enter the new name for the database.

If the named database cannot be found, the following prompt is issued:

```
BASE NOT FOUND>
```

This prompt indicates that the identified database, either from the stored file or entered in response to NEW BASE NAME, cannot be found. You should re-enter the base name in response to this prompt. If the prompt is reissued, terminate the program by entering [[RETURN]] and make sure the database is where you think it is. When the named database is found, you are prompted for additional database information as follows:

```
BASE PASSWORD>
MODE>
```

The password you enter in response to the password prompt should be one that allows modify access to the database. This password is not displayed as you enter it. If the password grants only partial access to the database, the utility will load only those parts of the database to which it has access; no warning is given. If the password is entered incorrectly or does not grant modify access, a database operation error occurs during the loading process. The utility issues the following questions if a database operation error occurs:

```
DISPLAY INPUT RECORD(Y/N)?>
CONTINUE SET LOAD(Y/N)?>
CONTINUE LOAD PROCESS(Y/N)?>
```

You can write the contents of the record to the indicated LIST FILE by entering a “Y” or [[RETURN]].

If you choose to continue the set load, the data entry in error is not loaded. If you choose not to continue, the utility allows you to continue loading the remaining sets in the database with the last prompt. Your response to MODE indicates the access mode to be used to open the database. Valid responses include 1, 2, 3, 4, and [[RETURN]]. Pressing [[RETURN]] indicates acceptance of the default mode, which is 3. (See the IMAGE/3000 Reference Manual for “Access Modes”.)

With the next prompt, you are given the choice to load in a fast I/O mode or not. The prompt is:

```
FAST I/O(Y/N)?>
```

FAST I/O means that the database buffers are not written at each modification.

The program begins to load the database after the FAST I/O prompt. If you chose LOAD as the operating mode for the database, you are prompted only when a difference between the old and new schema is encountered. If you chose EDIT as the operating mode, the following prompt is issued before the program loads each data set:

```
SET MODE(LOAD/EDIT/SKIP/CONT/EXIT)>
```

You can choose only one of the modes. A description for each is as follows:

LOAD	indicates that if a difference between the old and new schema occurs, issue prompts for the changed information.
EDIT	allows you to change the name of the data set and/or data elements during the loading operation.
SKIP	allows you to indicate that a data set is not to be loaded.
CONT	cancels the EDIT operation mode for the database, in which case the default of LOAD is used.
EXIT	indicates the utility program is to be terminated. Pressing [[RETURN]] in response to this prompt indicates the default of LOAD is used.

When the message LOAD COMPLETED is displayed, the program execution is terminated and the audit report is sent to the file indicated with the LIST FILE prompt.

The following examples show how to execute the Database Load utility program with an explanation of the responses given to the prompts. The examples show choosing a database operating mode of LOAD and of EDIT. The example using an operating mode of EDIT also shows choosing a data set operating mode of EDIT and of LOAD. The last example shows the audit report generated by the utility program for the database load.

Example

The following is an example of LOAD Run Mode:

```
:RUN DICTDBL.PUB.SYS                    How to execute this utility.
<The Dictionary/3000 DB Loader program banner appears here.>
STORE FILE> TEMP                        Name of file containing the
                                        contents of the stored base.
LIST FILE>                             [[RETURN]] sends report to terminal.
      BASE: SHPMGT.PUB.HOWE             Utility identifies base to load.
```

```
     RUN MODE(LOAD/EDIT/SHOW/EXIT)> LOAD  LOAD selected.
     NEW BASE NAME> SHPMGT.DEMO            Changing the MPE group for base.
     BASE PASSWORD>                       MANAGER was the password used.
     MODE> 1                               Opened for shared modify access.
 PRODUCT-MASTER    M 22/101               First data set to be loaded.
 PRODUCT-NO     : ITEM NOT FOUND, NEW ITEM NAME> PROD-NO  LOAD issues
       22 ENTRIES LOADED IN <1 CPU-SEC                    prompt when old
                                                         and new schemes
                                                         differ.
ITEM-MASTER       M 44/311
       44 ENTRIES LOADED IN 1 CPU-SEC
     EXPLODE           D 397/1054
     PRODUCT-NO      : ITEM NOT FOUND, NEW ITEM NAME> PROD-NO
       397 ENTRIES LOADED IN 20 CPU-SECS
     LOAD COMPLETED
     END OF PROGRAM
```

The following is an example of EDIT Run Mode:

```
     :RUN DICTDBL.PUB.SYS                  How to execute this utility.
     <The Dictionary/3000 DB Loader program banner appears here.>
     STORE FILE> TEMP                      Name of store file.
     LIST FILE>                           [[RETURN]] sends report to terminal.
          BASE: SHPMGT.PUB.HOWE             Program identifies base to load.
     RUN MODE(LOAD/EDIT/SHOW/EXIT)> EDIT   Editing prompts at set level
                                           requested.
    NEW BASE NAME> SHPMGT.DEMO             Loading to a different MPE group.
     BASE PASSWORD>                        MANAGER is password used.
     MODE> 1                                Opened for shared modify access.
     PRODUCT-MASTER    M 22/101                First data set to be loaded.
     SET MODE(LOAD/EDIT/SKIP/CONT/EXIT)> EDIT   Set level editing capability.
     PRODUCT-MASTER  : NEW SET NAME>            If the data set and/or items
     PRODUCT-NO      : NEW ITEM NAME> PROD-NO   have new names, enter new
     DESCRIPTION     : NEW ITEM NAME>           name after prompt as for
     QTY-ON-HAND     : NEW ITEM NAME>           PROD-NO.
     QTY-ALLOCATED   : NEW ITEM NAME>
     UNIT-COST       : NEW ITEM NAME>
     LIST-PRICE      : NEW ITEM NAME>
       22 ENTRIES LOADED IN <1 CPU-SEC
```

```
ITEM-MASTER       M 44/311                Next data set to be loaded.
SET MODE(LOAD/EDIT/SKIP/CONT/EXIT)> LOAD  No editing prompts generated
  44 ENTRIES LOADED IN 1 CPU-SEC          by choosing LOAD.
EXPLODE           D 397/1023              Next data set to be loaded.
SET MODE(LOAD/EDIT/SKIP/CONT/EXIT)> LOAD
PRODUCT-NO   : ITEM NOT FOUND, NEW ITEM NAME> PROD-NO  LOAD causes
  397 ENTRIES LOADED IN 20 CPU-SECS                    prompts when
                                                       old and new
LOAD COMPLETED                                           schemes differ.
END OF PROGRAM
```

The following is an example of Audit Listing:

```
<The Dictionary/3000 DB Loader program banner appears here.>
            DATABASE LOAD FROM STORE FILE TEMP.PUB.MARTIN
     BASE: SHPMGT.PUB.HOWE
LOADED TO: SHPMGT.DEMO.MARTIN
PRODUCT-MASTER    M 22/101
 ITEM PRODUCT-NO        LOADED TO PROD-NO
  22 ENTRIES LOADED IN <1 CPU-SEC
ITEM-MASTER       M 44/311
 44 ENTRIES LOADED IN 1 CPU-SEC
EXPLODE           D 397/1023
 ITEM PRODUCT-NO        LOADED TO PROD-NO
  397 ENTRIES LOADED IN 19 CPU-SECS
LOAD COMPLETED
```

DICTDBM,UTIL

Dictionary Clean Utility

As a result of continued DICTDBM use, a data Dictionary slowly accumulates superfluous information. This information can be periodically purged by using the Dictionary Clean Utility (DICTDBM,UTIL).

This utility, which can only be executed by the Dictionary creator, should be run when:

- DICTDBM has been used heavily for about 6 months, or
- the following message appears after signing onto DICTDBM:

```
*INFO:  DICTIONARY INTERNAL CLEANUP NEEDED (DICT 75)
```

To preserve the integrity of your Dictionary, ensure that this utility is only used on the entire data Dictionary which includes the root file and all data sets.

Before executing this utility, back up the data Dictionary. Also ensure that the temporary file used by DICTDBM,UTIL (DICTTEMP) is large enough to accommodate your data. By default, DICTTEMP holds 10,000 records (divided into 30 extents). These records originate from three data sets: DESCRIPTION-TEXT, FILE-PATH, and FILE-SORT. Run DICTDBA to determine the total number of records in these data sets. If the total is >10,000, use the FILE command to allocate adequate space as shown below:

```
:FILE DICTTEMP; DISC = your value.
```

Execute DICTDBM,UTIL by entering the following MPE command:

```
:RUN DICTDBM.PUB.SYS,UTIL
```

The utility issues the following message and prompt:

```
DICTIONARY SHOULD BE STORED OFF BEFORE RUNNING THIS UTILITY.
CONTINUE (Y/N)?>
```

Enter Y (to initiate cleanup) or N (to terminate utility execution). When the utility has finished processing, the message END OF PROGRAM is displayed and the program terminates.

Example

```
:RUN DICTDBM.PUB.SYS, UTIL                    How to execute this utility.
    <The Dictionary/3000 Clean program banner appears here.>
    DICTIONARY SHOULD BE STORED OFF BEFORE RUNNING THIS UTILITY.
    CONTINUE (Y/N)?> Y                        Must respond "Y" or "N";
         :                                    no default response occurs.
    END OF PROGRAM
```

DICTVPD

VPLUS Forms File Definition Utility

The VPLUS Forms File Definition utility (DICTVPD) enters the definition for an existing VPLUS forms file into the Dictionary. This utility creates the entries for the forms file, the forms in the forms file, and the data fields for the forms in the Dictionary. The forms file is entered in the Dictionary as a VPLS type file. The forms are entered as FORM type files and the data fields are entered as data elements in the Dictionary.

DICTVPD also establishes the associations of the data fields with the forms and the relationships between the forms and the forms file. However, form family relationships are not established in the Dictionary by DICTVPD. In the Dictionary, the parent forms and the child forms in a form family will only be related to the forms file.

Before the definition of a forms file is loaded in the Dictionary, DICTVPD checks the existing entries to avoid entering duplicate or redundant entries. If the name used for a forms file, form or data field being loaded is the same as the name of an existing entry, a warning message is issued. For a duplicate forms file name or a duplicate data field name, you are given the option to either enter a new name or to use the existing entity in the Dictionary. For a duplicate form name, you are given the option to either skip that form (that form would not be loaded, but subsequent forms would be loaded), purge the existing form (the form name and all associations with the form would be purged but the elements associated with the file would not be purged), or enter a new name for the form to be loaded.

Whenever a name is changed for an entity being loaded into the Dictionary, the new name entered becomes the “primary” name for that entity. The original name then becomes the “alias” name for that entity in the Dictionary. Because HP Inform will reference these entities by their aliases, the original names used by VPLUS do not have to be changed to be consistent with the Dictionary. However, since DICTDBM uses the primary names for all entities, it is important to remember what the new primary names are for the entities whose names were changed.

The following description explains how to execute DICTVPD and gives an explanation of each prompt.

This utility assumes that the Dictionary to be used by DICTVPD is in DICT.PUB in the Logan account. If the forms file is to be loaded in a Dictionary which is not in that group and account, use an MPE file equation to redefine the Dictionary's location before running this utility.

To execute DICTVPD, enter the following MPE command:

```
RUN DICTVPD.PUB.SYS
```

After an acknowledgment message, DICTVPD prompts for the Dictionary password to open and read the Dictionary as follows:

```
DICTIONARY PASSWORD>
```

Enter the Dictionary password and press [[RETURN]]. The password must grant at least PROGRAMMER level access to the Dictionary. (See Appendix C for an explanation of the

levels of access.) This password will not be displayed on the terminal as it is entered. However, it must be entered correctly or an error will result and the prompt will be reissued. If [[RETURN]] is pressed as the only response to this prompt DICTVPD will terminate. After the password is accepted DICTVPD then prompts for the name of the forms file to be loaded into the Dictionary.

```
FORMS FILE NAME>
```

Enter the forms file name (20 characters maximum).

DICTVPD now checks the Dictionary for any existing entries which may have the same name as the name entered. If an entry already exists with this name, the following warning and prompt are issued:

```
*WARNING: FORMS FILE ALREADY NAMED IN DATA DICTIONARY
DO YOU WANT TO LOAD UNDER A DIFFERENT NAME(Y/N)>
```

If N is entered in response to this prompt, DICTVPD will load the forms file into existing file defined in the Dictionary. If Y (the default) is entered in response to this prompt, a new prompt is issued to rename the forms file to be loaded into the Dictionary.

```
NEW FORMSFILE NAME>
```

Enter the new forms file name. This name must be unique to the Dictionary; otherwise, the above warning and prompt will be reissued. This name becomes the primary name for this forms file in the Dictionary.

Once the forms file name is accepted, the following prompt is issued:

```
SELECT DATA CONVERSION(Default/Char)>
```

The response to this prompt determines how the data types defined for the data fields are converted to Dictionary compatible data types. For instance, if Char (or C) is entered as the response to this prompt, then any data types defined in the forms file will be converted to Dictionary type X when loaded into the Dictionary. The size and storage length will be the same as the field length. If Default (or D) is entered as the response to this prompt, then the data types defined in the forms file will be converted and entered into the Dictionary as shown on the next page

VPLUS		DICTIONARY/3000			
TYPE	**FIELD LENGTH**	**TYPE**	**SIZE**	**DECIMAL**	**STORAGE LENGTH**
CHAR	L	X	L	O	L
NUM	L (if L is 1)	R	1	0	4
	L (if L is 2, 3, 4, 5 or 6)	R	L-1	0	8
	L (if L is 7 or greater)	R	L-1	0	8
NUMn	L (if L is 1)	I	1	n	2
	L (if L is 2, 3, 4, or 5)	I	L-1	n	2
	L (if L is 6, 7, 8, 9 or 10)	I	L-1	n	4
	L (if L is 11 or greater)	P	L-1	n	* (L + 1)/2
DIG	L (if L is 1, 2, 3, or 4)	I+	L	0	2
	L (if L is 5, 6, 7, 8 or 9)	I+	L	0	4
	L (if L is 10 or greater)	P+	L	0	* (L +2)/2
IMPn	L (if L is 1	I	1	n	2
	L (if L is 2, 3, 4, or 5)	I	L-1	n	2
	L (if L is 6, 7, 8, 9 or 10)	I	L-1	n	2
	L (if L is 11 or greater)	P	L-1	n	* (L + 1)/2

L = VPLUS Field Length

* when computing these numbers always round down

The following examples use the chart above to determine how data types, size, decimal and storage lengths are converted into the dictionary if the elements have NOT been previously defined and you have selected DEFAULT conversion.

Example

```
      VPLUS                              DICTIONARY
Type     Field Length            Type(Size,Decimal,Storage Length)
CHAR          10                          X(10,0,10)
```

Dictionary converts the CHAR to type X and takes the VPLUS field length as its size and storage length. Decimal is 0.

```
NUM           10                          R(9,0,8)
```

Dictionary converts type to R. Size is 9 because the VPLUS field length is 10, and 10 - 1 is 9. Decimal is 0. Storage length is 8.

```
NUM            1                          R(1,0,4)
```

Dictionary converts type to R. Size is 1. Decimal is 0. Storage length is 4.

```
NUM5          10                          I(9,5,4)
(n=5)
```

Dictionary converts type to I. Size is 9 because the VPLUS field length is 10, and 10 - 1 is 9. Decimal is equal to n which is 5. Storage length is 4.

```
NUM5          20                          P(19,5,10)
(n=5)
```

Dictionary converts type to P. Size is 19 because the VPLUS field length is 20, and 20 - 1 is 19. Decimal is equal to n which is 5. Storage length is 10 because (20 +1)/2 = 10 (rounded down).

```
DIG           10                          P+(10,0,6)
```

Dictionary converts type to P+. Size is equal to the VPLUS field length which is 10. Decimal is 0. Storage length is 6 because (10 + 2)/2 = 6.

```
IMP5           10                         I(9,5,4)
(n=5)
```

Dictionary converts type to I. Size is 9 because the VPLUS field length is 10, and 10 - 1 is 9. Decimal is equal to n which is 5. Storage length is 4.

```
IMP5           20                         P(19,5,10)
(n=5)
```

Dictionary converts type to P. Size is 19 because the VPLUS field length is 20, and 20 - 1 is 19. Decimal is equal to n which is 5. Storage length is 10 because (20 + 1)/2 = 10 (rounded down).

If the data fields (elements) have not been previously defined in the Dictionary, enter N. This indicates that new element entries are to be created when the elements are loaded into the Dictionary.

```
DATA ELEMENTS ALREADY DEFINED(Y/N)>
```

If the data fields for the forms file have already been defined as elements in the Dictionary, enter Y in response to the prompt. This indicates that DICTVPD will use the existing element entries to define the data fields as long as the existing element data is compatible with the data for the data fields to be loaded. If DICTVPD finds that the two element definitions are not compatible, an error will be displayed as the utility tries to load the data field.

DICTVPD determines that an existing dictionary element and a VPLUS field are compatible if the display size of the dictionary element is 1 less than (for the signed numeric data types 9, Z, I, P, R, K, J, and E) or equal to (for the remaining data types such as 9+, Z+, I+, etc.) the VPLUS field length. The reason that the dictionary display size for signed numeric data types must be 1 less than the VPLUS field length is that the display size stored in the dictionary does not include a character position for the sign while the VPLUS field length does. If DICTVPD determines that the dictionary element and the VPLUS field are incompatible, the user receives an error message and is prompted for a new primary element name.

After DICTVPD loads the forms file in the Dictionary, a reference list is generated which shows the loaded forms file, forms, and data fields. Also included in the reference list are the data names, aliases, data dispositions and data types as they were loaded in the Dictionary. This list can be a useful tool to ensure that the loaded forms and data are the correct ones and to provide future reference documentation. (See the example following the discussion of DICTVPD for a sample reference list.)

The next prompt allows you to direct the reference list to the terminal, the line printer, on a specified disc file, or to suppress the listing entirely.

```
LIST FILE>
```

Enter one of the following options in response to the LIST FILE prompt:

LP	sends the reference list to the line printer. If there is no device LP configured on your system, use a file equation to redirect DICTLIST to another device.
NULL	suppresses the list.
filename	sends the list to a disc file. If the specified file already exists, an error message will be issued and you will be asked if the contents of the file should be purged. If not, you will be prompted for a new list file.
**filename*	sends the listing to the file identified in the back-referenced file equation.
[[RETURN]]	displays the list on $STDLIST after the forms file has been loaded.

DICTVPD then issues the prompt:

```
CHANGE UNDERSCORE TO HYPHEN(Y/N)>
```

Although the Dictionary and VPLUS recognize an underscore character ("_") as a valid character, IMAGE does not. (However, IMAGE does recognize a hyphen ("-")). Therefore, in order to ensure compatibility between IMAGE and the forms file, you may choose to allow DICTVPD to change any occurrence of an underscore character in the forms file to a hyphen.

Enter N to keep the underscore character as itself when the forms file is loaded.

After you respond to the above prompt, DICTVPD loads the forms file in the Dictionary and displays the following message:

```
LOADING DATA DICTIONARY
```

The utility then begins the loading process for the forms and data fields. Since DICTVPD allows you to selectively load forms into the Dictionary, you can specify which forms are to be loaded for this forms file with the following prompt:

```
FORM NAME TO BE LOADED(or "@"/"?")>
```

Enter the name of an individual form belonging to the forms file. If you are not sure of the names of the forms in the forms file, enter "?". DICTVPD will display the list of forms that may be loaded. DICTVPD will then reprompt you to enter the form name. If an individual form name is entered in response to this prompt, DICTVPD will continue to prompt for form names until no other form names are entered and [[RETURN]] is pressed.

Enter "@" to load all forms in the forms file. Note that if this forms file is a new entry in the Dictionary, the VPLUS reserved form $REFRESH will also be loaded in the Dictionary for

this forms file. This form will not have any data fields associated to it. Also, $REFRESH will not appear on the reference list for the forms file but will appear when the Dictionary is accessed by the program DICTDBM.

As each form is loaded in the Dictionary, the following message will appear:

```
LOADING FORM: formname
```

If DICTVPD finds an existing entry in the Dictionary with the same name as the form to be loaded, the following warning and prompt are issued:

```
*WARNING: DICTIONARY ALREADY CONTAINED FORM: formname
SKIP, PURGE OR ENTER NEW FORM(S/P/N)>
```

If S is entered in response to this prompt, DICTVPD will ignore the current form and proceed to load the next form in the forms file. If P is entered, DICTVPD will purge the existing form in the Dictionary along with all the associations it may have. Note that the elements themselves are not purged, only the associations to the form are purged. Once the existing form has been purged, DICTVPD will load the form which belongs to the forms file in the Dictionary. If N is entered in response to this prompt, a new prompt is issued to rename the form to be loaded as follows:

```
NEW FORM NAME>
```

Enter a new name for the form. This name must be unique to the Dictionary or DICTVPD will reissue the warning and prompts above. The original name used for the form in the forms file becomes the alias name in the Dictionary. The new name entered becomes the primary name in the Dictionary.

As each form is loaded in the Dictionary, DICTVPD checks that there are no data field name conflicts between the existing elements defined in the Dictionary and the data fields currently being loaded. For instance, if Y was entered in response to the DATA ELEMENTS ALREADY DEFINED prompt and DICTVPD can not find the existing entry for a data field being loaded, the following message will be issued:

```
ELEMENT NOT DEFINED, NEW ELEMENT LOADED: element
```

This indicates that DICTVPD has loaded the specified data field from the forms file in the Dictionary as a new element entry.

If N was entered in response to the DATA ELEMENTS ALREADY DEFINED prompt, DICTVPD will still check for existing data elements in the Dictionary with the same name and compatible definition. If DICTVPD finds a duplicate entry in the Dictionary and the data type and data size are compatible with the data field to be loaded, DICTVPD will issue the following message and prompt:

```
COMPATIBLE DEFINITION ALREADY IN DICTIONARY FOR ELEMENT element
DO YOU WANT ELEMENT TO HAVE A DIFFERENT NAME(N/Y)>
```

Enter N to use the existing entry in the Dictionary for this element. Enter Y to rename the element being loaded. A new prompt will then be issued:

```
NEW PRIMARY ELEMENT NAME>
```

Enter the new primary name for the data field being loaded in the Dictionary. Once again, the original name becomes the alias for the element. If [[RETURN]] is pressed in response to this prompt, the data field is not loaded in the Dictionary and the following warning is issued:

```
*WARNING: ELEMENT HAS NOT BEEN LOADED INTO DICTIONARY
```

If N was entered in response to the DATA ELEMENTS ALREADY DEFINED prompt and DICTVPD finds an existing element in the Dictionary with the same name but an incompatible definition, the following message and prompt are issued:

```
INCOMPATIBLE DEFINITION ALREADY IN DICTIONARY FOR ELEMENT: element
NEW PRIMARY ELEMENT NAME>
```

Enter the new primary element name for the element being loaded.

Note that a new element name must be entered and the existing element can not be used for this data field because the data type and data size for the existing element are not compatible with the data field's definition. If [[RETURN]] is pressed in response to this prompt the data field will not be loaded and a warning will be issued (as shown above).

After all forms and respective data fields have been loaded into the Dictionary, DICTVPD will generate the listing of the loaded forms file, the forms and the data fields. (See the LIST FILE prompt for information on this listing. See the example that follows for a sample reference list.)

After the listing is produced, DICTVPD terminates.

Example

The following is an example of DICTVPD:

```
RUN DICTVPD.PUB.SYS                How to execute this utility.
<The Dictionary/3000 VPLUS Loader program banner appears here.>
DICTIONARY PASSWORD> ;             Password for modify access used.
FORMS FILE NAME> Formf1
SELECT DATA CONVERSION (Default/Char)> D
DATA ELEMENTS ALREADY DEFINED (Y/N)> N
LIST FILE> LP                     The report is sent to the Line Printer.
CHANGE UNDERSCORE TO HYPHEN (Y/N)> Y
LOADING DATA DICTIONARY
FORM NAME TO BE LOADED (or "@"/"?")> @  All forms in the forms file to be
                                        loaded.
LOADING FORM: FORMA
LOADING FORM: FORMB
Name                    Alias              NEW/OLD Type
FORMF1                                     NEW     VPLS
FORMA                                      NEW     FORM
```

```
F5                                NEW     X ( 5, 0, 5)
F10                               NEW     X (10, 0,10)
F20                               NEW     X (20, 0,10)
F2                                NEW     X ( 2, 0, 2)
UNDER_SCORE_TO                    NEW     X (14, 0,14)
FORMB                             NEW     FORM
CHAR                              NEW     X ( 5, 0, 5)
NUMN                              NEW     I ( 4, 3, 2)
NUM                               NEW     R ( 4, 0, 4)
DIG                               NEW     I+( 5, 0, 4)
IMPN                              NEW     I ( 4, 2, 2)
IMP                               NEW     I ( 4, 0, 2)
MDY                               NEW     X ( 5, 0, 5)
DMY                               NEW     X ( 5, 0, 5)
YMD
```

6 The Dictionary DICTPDE Utility

Overview

DICTPDE, the PASCAL Definition Extract Utility, is an interactive, command-driven utility provided by the Dictionary. This utility extracts data and file definitions from the dictionary and generates the corresponding PASCAL declarations for the data definitions. The generated declarations are echoed to the terminal and written to an output file which can be used for PASCAL programs.

When a parent entity is extracted from the Dictionary, DICTPDE generates a PASCAL record with the related child entities as the “fields” within the record. The primary name of the parent entity is used as the record name. The aliases of the child entities are used as the field names. (The aliases are used because they define how the child entities are known by the parent entity.) Note that DICTPDE will not generate a data declaration for a file which does not have any child files or any elements associated with it.

When a data declaration is generated for an entity, DICTPDE checks that the entity name, byte position and byte offsets as defined in the Dictionary (for that entity) are compatible with PASCAL. If the byte position and byte offsets for the entity are not PASCAL compatible, an error message is issued and DICTPDE converts the byte position and byte offsets to compatible PASCAL code. (Refer to PASCAL Data Type Mappings later in this section, for more detailed information.) If the entity name defined in the Dictionary is an illegal PASCAL name (for instance, the name contains a hyphen) or the entity name is a PASCAL reserved word, an error message is issued and DICTPDE will convert the names to legal PASCAL names. (Refer to DICTPDE Naming Considerations, later in this section for more detailed information.)

Initiating DICTPDE

DICTPDE can be initiated from within a session or from within a job stream. When initiated from within a session, DICTPDE can be executed to accept commands from a command file or DICTPDE can be run interactively (commands are entered at the terminal). When initiated from within a job stream, DICTPDE can be executed to accept commands from a command file or from the job stream itself.

The following describes how to execute DICTPDE interactively (from within a session). For information on how to execute DICPTDE to accept commands from a command file (either in a session or a job stream) see Changing Input, List and Output Files later in this section.

DICTPDE assumes that the Dictionary resides in DICT.PUB of the logon account. To redefine the Dictionary, you can use an MPE file equation or the ALTER DICT command-subcommand. (ALTER DICT is discussed in detail later in this section.)

To execute DICTPDE, enter the following MPE command:

```
RUN DICTPDE.PUB.SYS
```

After the DICTPDE banner is displayed, the following prompt is issued:

```
DICTIONARY PASSWORD>
```

Enter the password for the Dictionary. If the Dictionary password was entered incorrectly or was an invalid password for that Dictionary, DICTPDE will issue the following error message:

```
*DICT ERROR: INVALID PASSWORD. CAN NOT OPEN DICTIONARY!
```

The DICTIONARY PASSWORD prompt will then be reissued.

If DICTPDE can not open the Dictionary, the following error message will be issued:

```
*DICT ERROR: CAN NOT OPEN DICTIONARY!
```

Check to see that the Dictionary is in the correct group and account. If a file equation was used to redirect the Dictionary, check that the Dictionary name was entered correctly. DICTPDE will then issue a prompt that identifies the current Dictionary and asks for a new Dictionary to be entered. Once the specified Dictionary and password have been accepted, DICTPDE will issued an “>” which indicates that the utility is in command mode. If you are not able to open a Dictionary to be used for code generation, you must enter a colon (:) or hit CONTROL Y to terminate DICTPDE. Otherwise, DICTPDE will continue to prompt for a Dictionary name and password until a specified Dictionary can be opened.

Once DICTPDE is in command mode, you may enter any of the DICTPDE commands.

Table 6-1 lists the DICTPDE commands and gives a brief description of their function.

Table 6-1. The DICTPDE commands

COMMAND	FUNCTION
ALTER	Changes an option for generating declarations.
EXIT	Terminates the DICTPDE utility.
GENERATE	Generates PASCAL declarations for element and file definitions in the Dictionary, and for the VPLUS COMAREA and IMAGE parameters.
HELP	Provides a description of the DICTPDE commands.
LIST	Displays the entities extracted from the Dictionary.

Table 6-2 shows the DICTPDE Commands and their respective subcommands.

ALTER	EXIT	GENERATE	HELP	LIST
|	|	|	|	|
|	|	|	|	|
All	(None)	Comarea	ALTER	All
Dict		Element	EXIT	Element
Kind		File	GENERATE	File
Name		Imageparms	HELP	
Output			LIST	
Shift				
Test				
Vplus				

Changing Input, List and Output Files

DICTPDE uses the formal file designators DICTIN as the input file for DICTPDE commands, DICTLOG as the list file for the commands entered interactively, and DICTOUT as the output file for the generated source code.

When DICTPDE is run interactively, DICTIN is the terminal (DICTPDE commands are accepted as they are entered at the terminal). DICTPDE can also be executed to accept commands from a command file (in either a session or a job stream) by redirecting DICTIN to a command file. To redirect DICTIN to command file, use the following MPE file equation:

```
FILE DICTIN = command file name
```

The next time DICTPDE is run (in either a session or job stream), the input commands will be read from this file. Remember, before you can initiate DICTPDE to accept commands from a command file, you must first create the command file using the EDITOR and enter the DICTPDE commands into the file.

DICTPDE can also be executed to accept commands from a command file that contains the DICTPDE commands previously entered at the terminal. To do so, redirect the list file DICTLOG to a disc file, and then run DICTPDE interactively. If the file specified by DICTLOG does not already exist, DICTPDE will build the file for you. All the commands entered interactively for that session will be saved in this disc file in the order you entered them. Before DICTPDE is run again, redirect DICTIN to the disc file used for DICTLOG and reset DICTLOG. The next time DICTPDE is run interactively, the commands will be read from the disc file as they were entered in the previous session. This process is shown below:

1. FILE DICTLOG = command file	Commands are written to this file.
2. RESET DICTIN	Do this if DICTIN was directed to a file.
3. RUN DICTPDE.PUB.SYS	Run DICTPDE and enter the commands.
4. Exit DICTPDE	Exit this session.
5. FILE DICTIN = command file	DICTIN will take commands from this file.
6. RESET DICTLOG	DICTLOG must be reset.
7. RUN DICTPDE.PUB.SYS	Commands are now taken from the command file.

The data declarations generated by DICTPDE are echoed to $STDLIST (which is the terminal when DICTPDE is run interactively) and written to an output file. DICTPDE uses the output file DICTOUT (a fixed ASCII file with 80 bytes) unless a different output file is specified. To specify a different output file, you can either use an MPE file equation to redirect DICTOUT or you can use the ALTER OUTPUT command-subcommand. (ALTER OUTPUT is discussed in detail later in this section.) The specified file can be a new file or an existing file. If the file is new, DICTPDE will build that file as a fixed ASCII file with 80 bytes. If the file already exists, DICTPDE will append the generated code to the existing contents of that file.

The following is an example of how to run DICTPDE within a job stream. Note that in this example, DICTPDE will accept commands from the command file COMMANDS and write the generated declarations to the output file OUTPUT.

```
!JOB CODEGEN, LYNN.ACCTS
!FILE DICTIN = COMMANDS
!FILE DICTOUT = OUTPUT
!RUN DICTPDE.PUB.SYS
 MGR
!EOJ
```

Note that the Dictionary password must be supplied in the job stream immediately after the MPE RUN command for DICTPDE. In the above case, the password used is MGR. Also, remember to create the command file COMMANDS with the EDITOR before streaming this job. (Don't forget to include an EXIT command at the end of the command file to terminate DICTPDE.)

ALTER

Changes an option used when generating data declarations from the Dictionary.

Syntax

```
ALTER subcommand A
```

Use the ALTER command to change the default options for generating data declarations for the entities extracted from the Dictionary. The subcommand identifies the option to be changed. Each ALTER command-subcommand generates a prompt that allows you to specify how an option is to be changed, except for the ALTER ALL command-subcommand. ALTER ALL generates all the prompts which are issued for the other subcommands.

A complete list of subcommands is given below.

Subcommands

The following subcommands can be used with the ALTER command:

ALL	changes all the options for code generation.
A	
DICT	changes the dictionary to be used for code generation.
D	
KIND	changes the kind of declaration statement generated (TYPE or VAR).
K	
NAME	changes the generated code to include string constants for the
N	extracted entities.
OUTPUT	changes the output file.
O	
SHIFT	changes the type case of the generated code in the output file.
S	
TEXT	changes the generated code to include comments.
T	
VPLUS	changes the generated declarations for a VPLUS form to include
V	PASCAL arrays for data fields in that form.

The following gives a discussion of each of the subcommands used with the ALTER command and the prompts that they generate.

>ALTER ALL

Changes all the options used for generating PASCAL data declarations for the entities extracted from the Dictionary.

Prompts

The following prompts are issued when the command-subcommand ALTER ALL is entered:

```
DICTIONARY NAME (current dictionary )>
DICTIONARY PASSWORD>
SOURCE OUTPUT FILE (<current file )>
EXTRACT AS TYPE OR VAR (T/V)>
GENERATE COMMENTS (N/Y)>
GENERATE NAMES AS STRING CONSTANTS (N/Y)>|
GENERATE TABLES FOR VPLUS EXTRACTS (N/Y)>
LOWER OR UPPER CASE (L/U)>
```

When ALTER ALL is used, the above prompts are issued one at a time. (After a response is made to one prompt, the next prompt is issued.) Note that the ALTER ALL command-subcommand generates all the prompts issued for the other ALTER subcommands. The following explains each of the above prompts in detail:

Discussion

DICTIONARY NAME (current dictionary)>
: Enter the Dictionary name, group and account to be opened and used for further code generation. DICTPDE displays the current Dictionary being used in (current dictionary). Data and file definitions are extracted from the new Dictionary specified until a new ALTER DICT command-subcommand is used.

  Pressing [[RETURN]] in response to this prompt indicates that the Dictionary as displayed in (current dictionary) does not change. (The password prompt will not be issued.)

  If an invalid Dictionary is entered in response to this prompt, DICTPDE will issue an error message and reprompt for the Dictionary name. DICTPDE will continue to prompt for the Dictionary name until a valid Dictionary name is entered or DICTPDE is terminated.

DICTIONARY PASSWORD>
: Enter the Dictionary password for the new Dictionary to be used. The password entered must grant at least REPORT level access to that Dictionary. If the new Dictionary specified is the same as the previous Dictionary, the password prompt will not be issued.

If an invalid password is entered in response to this prompt, DICTPDE will issue an error message and reprompt for the Dictionary password. DICTPDE will continue to prompt for the Dictionary password until a valid password is entered or DICTPDE is terminated.

When a new Dictionary name and valid password are entered, DICTPDE closes the previous Dictionary used and opens the new Dictionary to be used.

SOURCE OUTPUT FILE (current output file)>

Enter the name of the file to redirect output to. This output includes the data declarations generated by DICTPDE and, if specified, the string constants generated for the entity names. DICTPDE displays the current output file as (current output file). (current output file) will be DICTOUT unless otherwise specified by an earlier ALTER OUTPUT command-subcommand or an MPE file equation was used to redirect DICTOUT.

Output may be directed to a new or existing disc file (ASCII, fixed file of 80 bytes) or to the terminal. If the specified file is a new file, DICTPDE will build the file for you. If the specified file already exists, DICTPDE will issue a warning message and append the generated source code to the existing contents of the specified file. To direct this output to the terminal instead of to a disc file, enter $STDLIST in response to this prompt.

Pressing [[RETURN]] in response to this prompt indicates that the output file does not change. This means that if no previous output file had been specified by an ALTER OUTPUT command, then the output file would still be DICTOUT. If a file had been specified by a previous ALTER OUTPUT command, that file would remain the output file.

EXTRACT AS TYPE OR VAR (T/V)>

Enter T (or press [[RETURN]]) if the extracted data definitions are to be generated as TYPE data declarations. Enter V if the data definitions extracted are to be generated as VAR data declarations. Note that DICTPDE assumes that the definitions are to be generated as TYPE declarations unless V is entered in response to this prompt.

Pressing [[RETURN]] in response to this prompt indicates that the data definitions are to be generated as TYPE data declarations.

GENERATE COMMENTS (N/Y)>

Enter Y to generate comment lines which describe the entities extracted from the Dictionary. These comment lines are written to the output file along with the data declarations and, if specified, the string constants generated for the file names. DICTPDE assumes that comment lines are not generated unless Y is entered in response to this prompt.

Pressing [[RETURN]] in response to this prompt indicates that no comment lines are generated for the extracted entities from the Dictionary.

GENERATE NAMES AS STRING CONSTANTS

(N/Y)> Enter Y if PASCAL string constants are to be generated for the specified file names extracted from the Dictionary. DICTPDE assumes that string constants are not to be generated for the file names unless Y is entered in response to this prompt.

Pressing [[RETURN]] in response to this prompt indicates that string constants are not to be generated.

GENERATE TABLES FOR VPLUS EXTRACTS

(N/Y)> Enter Y to generate PASCAL arrays for the fields in a VPLUS form in addition to the data declarations generated for the form. DICTPDE assumes that no arrays are generated unless Y is entered in response to this prompt.

Pressing [[RETURN]] in response to this prompt indicates that no additional arrays are generated for a VPLUS form.

LOWER OR UPPER CASE

(L/U)> Enter U if the generated code is to be written in UPPER CASE characters to the output file. Enter L if the generated code is to be written in LOWER CASE characters. DICTPDE assumes the generated code will be written in LOWER CASE characters to the output file unless U is entered in response to this prompt.

Pressing [[RETURN]] in response to this prompt indicates that the generated code will be written in LOWER CASE characters to the output file.

>ALTER DICT

Changes the Dictionary that contains the data definitions to be extracted for code generation.

Prompts

The following prompts are issued when the command-subcommand ALTER DICT is entered:

```
DICTIONARY NAME (current dictionary )>
DICTIONARY PASSWORD>
```

DICTIONARY NAME (current dictionary)>
: Enter the new Dictionary name, group and account to be opened and used for further code generation. DICTPDE displays the current Dictionary being used in (current dictionary). Data and file definitions are extracted from the new Dictionary specified until a new ALTER DICT command-subcommand is used.

  Pressing [[RETURN]] in response to this prompt indicates that the Dictionary as displayed in (current dictionary) does not change. (The password prompt will not be issued.)

  If an invalid Dictionary is entered in response to this prompt, DICTPDE will issue an error message and reprompt for the Dictionary name. DICTPDE will continue to prompt for the Dictionary name until a valid Dictionary name is entered or DICTPDE is terminated.

DICTIONARY PASSWORD>
: Enter the Dictionary password for the new Dictionary to be used. The password entered must grant at least REPORT level access to that Dictionary.

  If the new Dictionary specified is the same as the previous Dictionary, the password prompt will not be issued.

  If an invalid password is entered in response to this prompt, DICTPDE will issue an error message and reprompt for the Dictionary password. DICTPDE will continue to prompt for the Dictionary password until a valid password is entered or DICTPDE is terminated.

  When a new Dictionary name and valid password are entered, DICTPDE closes the previous Dictionary used and opens the new Dictionary to be used.

Discussion

ALTER DICT allows you to use a Dictionary other than the Dictionary in DICT.PUB of your logon account without using an MPE file equation to redefine the Dictionary. Therefore, file and element definitions can be extracted from different Dictionaries without terminating DICTPDE.

Example

```
>alter dict
DICTIONARY NAME (DICT.PUB)>          Enter the new Dictionary.
dict.pub.howe
DICTIONARY PASSWORD> mgr             Enter the password for the
                                     specified Dictionary.
```

>ALTER KIND

Changes the kind of data declaration generated (either as VAR or TYPE) for the entities extracted from the Dictionary.

Prompts

The following prompt is issued when the command-subcommand ALTER KIND is entered:

```
EXTRACT AS TYPE OR VAR (T/V)>
```

EXTRACT AS TYPE OR VAR (T/V)>

Enter T (or press [[RETURN]]) if the extracted data definitions are to be generated as TYPE data declarations. Enter V if the data definitions extracted are to be generated as VAR data declarations. Note that DICTPDE assumes that the definitions are to be generated as TYPE declarations unless V is entered in response to this prompt.

When generating VAR declarations for an element that back-references another element in the Dictionary, the element used as the back-reference will be generated as a TYPE declaration.

Pressing [[RETURN]] in response to this prompt indicates that the data definitions are to be generated as TYPE data declarations.

Discussion

ALTER KIND allows you to specify whether the data definitions will be generated as TYPE or VAR declarations. ALTER KIND also allows you to alternately generate declarations as TYPE or VAR. Note that a data definition may NOT be generated as both TYPE and VAR declarations in the same output file. However, you may choose to generate a data definition as a VAR declaration for one output file, and to generate the same data definition as a TYPE declaration for a different output file. (To change the output file, see the ALTER OUTPUT command-subcommand in this section.)

Example

```
>alter kind
EXTRACT AS TYPE OR VAR (T/V)> v       The code generated after this
                                      command will be generated as VAR
                                      declarations.
```

>ALTER NAME

Changes the generated source code to include PASCAL string constants for the file names extracted from the Dictionary.

Prompts

The following prompt is issued when the command-subcommand ALTER NAME is entered:

```
GENERATE NAMES AS STRING CONSTANTS (N/Y)>
```

GENERATE NAMES AS STRING CONSTANTS (N/Y)>

Enter Y if PASCAL string constants are to be generated for the specified file names extracted from the Dictionary. DICTPDE assumes that string constants are not to be generated for the file names unless Y is entered in response to this prompt.

Pressing [[RETURN]] in response to this prompt indicates that string constants are not to be generated.

Discussion

ALTER NAME allows you to generate string constants for the specified file names extracted from the Dictionary. The string constants are generated for the specified file name only. If the specified file is a parent file, string constants are generated for the parent file name only. String constants are not generated for the child files which are related to the specified parent file.

Example

```
 >alter name
GENERATE NAMES AS STRING CONSTANTS    String constants will be
(N/Y)> y
                                      generated for the subsequent
                                      files generated.
```

>ALTER OUTPUT

Changes the output file that the PASCAL data declarations will be written to.

Prompts

The following prompt is issued when the command-subcommand ALTER OUTPUT is entered:

```
SOURCE OUTPUT FILE (current output file )>
```

SOURCE OUTPUT FILE (current output file)>
: Enter the name of the file to redirect output to. This output includes the data declarations generated by DICTPDE and, if specified, the string constants generated for the entity names. DICTPDE displays the current output file as (current output file). (current output file) will be DICTOUT unless specified otherwise by an earlier ALTER OUTPUT command-subcommand or an MPE file equation used to redirect DICTOUT.

  Output may be directed to a new or existing disc file (ASCII, fixed file of 80 bytes) or to the terminal. If the specified file is a new file, DICTPDE will build the file for you. If the specified file already exists, DICTPDE will issue a warning message and append the generated source code to the existing contents of the specified file. To direct this output to the terminal, enter $STDLIST in response to this prompt.

  Pressing [[RETURN]] in response to this prompt indicates that the output file does not change. This means that if no previous output file had been specified by an ALTER OUTPUT command, then the output file would still be DICTOUT. If a file had been specified by a previous ALTER OUTPUT command, that file would remain the output file.

Discussion

ALTER OUTPUT allows you to change the output file to a specified disc file or to the terminal without using an MPE file equation to redirect DICTOUT. (The file equation is used before running DICTPDE.) DICTPDE also allows you to alternate the output files while running DICTPDE. For instance, VAR declarations could be generated and written to one output file and TYPE declarations could be generated and written to a different file. Then all VAR declarations would be kept in one file and all TYPE declarations would be kept in a different file. (See the ALTER KIND command-subcommand for alternating the kinds of declarations generated.)

Example

```
>alter output
    SOURCE OUTPUT FILE (DICTOUT)> pasout.pub.howe
```

The output file is changed fromDICTOUT toPASOUT.PUB.HOWE. Any subsequent code generated will be written toPASOUT.PUB.HOWE until a newALTER OUTPUT command-subcommand is used.

>ALTER SHIFT

Changes the type case of the code generated by DICTPDE.

Prompts

The following prompt is issued when the command-subcommand ALTER SHIFT is entered:

```
LOWER OR UPPER CASE (L/U)>
```

LOWER OR UPPER CASE (L/U)>
: Enter U if the generated code is to be written in UPPER CASE characters to the output file. Enter L if the generated code is to be written in LOWER CASE characters. DICTPDE assumes the generated code will be written in LOWER CASE characters to the output file unless U is entered in response to this prompt.

  Pressing [[RETURN]] in response to this prompt indicates that the generated code will be written in LOWER CASE characters to the output file.

Discussion

ALTER SHIFT allows you to choose your personal preference for the type case used for the code generated by DICTPDE. You can also use the ALTER SHIFT command-subcommand to maintain type case consistency between the output files used for code generation and the PASCAL programs that these files may be used for.

Example

```
>alter shift
LOWER OR UPPER CASE (L/U)> u          Any subsequent code generated
                                      will be written in UPPER CASE
                                      characters to the output file.
```

>ALTER TEXT

Changes the generated code to include comments lines about the extracted entities.

Prompts

The following prompt is issued when the command-subcommand ALTER TEXT is entered:

```
GENERATE COMMENTS (N/Y)>
```

GENERATE COMMENTS (N/Y)>

Enter Y to generate comment lines which describe the entities extracted from the Dictionary. These comment lines are written to the output file along with the data declarations and, if specified, the string constants generated for the file names. DICTPDE assumes that comment lines are not generated unless Y is entered in response to this prompt.

Pressing [[RETURN]] in response to this prompt indicates that no comment lines are generated for the extracted entities from the Dictionary.

Discussion

ALTER TEXT allows you to generate additional code for the entities extracted from the Dictionary in the form of comment lines. These comment lines include the following information about the extracted entity:

entity	name of the entity extracted from the Dictionary.
entity-name	the entity long-name (as it was entered in DICTDBM).
entity-resp	the name of the person, department, or area responsible for the integrity of the entity.
date-change	the date of the latest change made to the entity in the Dictionary.
date-create	the date the entity was created in the Dictionary.
identity-change	the identity of the person, department, or area that made the last change to the entity in the Dictionary.
identity-create	the identity of the person, department or area that created the entity in the Dictionary.

Example

```
>alter text
     GENERATE COMMENTS (N/Y)> y
     >generate file
     FILE(S)> account
     var
        account_rec =
         record
          firstname : longreal;
          lastname  : packed array[1..20] of char;
          address   :
           record
             case integer of
                0 :
                 (
                   buffer      : packed array[1..20] of char
                 );
                1 :
                 (
                   streetname  : packed array[1..10] of ' '..'Z'
                 );
                2 :
                 (
                   city        : packed array[1..10] of char
                 );
           end;
          phone    : packed array[1..20] of char
        end;
     {file                  : account                  }
     {file_name             : customer accounts        }
     {file_resp             : manager                  }
     {date_change           : 83/08/04                 }
     {date_create           : 82/09/02                 }
     {identity_change       : b. lewis                 }
     {identity_create       : manager                  }
```

>ALTER VPLUS

Changes the generated code to include PASCAL arrays for the fields in a VPLUS form.

Prompts

The following prompt is issued when the command-subcommand ALTER VPLUS is entered:

```
GENERATE TABLES FOR VPLUS EXTRACTS (N/Y)>
```

GENERATE TABLES FOR VPLUS EXTRACTS (N/Y)>
: Enter Y to generate PASCAL arrays for the fields in a VPLUS form in addition to the data declarations generated for the form. DICTPDE assumes that no arrays are generated unless Y is entered in response to this prompt. Enter N if no arrays are to be generated for the fields in a VPLUS form. Pressing [[RETURN]] in response to this prompt indicates that no arrays are generated.

Discussion

ALTER VPLUS allows you to generate additional code for the data fields in a VPLUS form. The additional code includes PASCAL arrays for the fields in the form and a buffer for the FORM declaration.

The arrays generated for the data fields are as follows:

off
: the byte offsets for each field in the form.

len
: the byte lengths of each of the fields in the form.

num
: the numbers of the fields within the FORM

Example

```
>alter vplus
    GENERATE TABLES FOR VPLUS EXTRACTS (N/Y)> y
    >generate file
    FILE(S)> forma                              Specifies a VPLUS form.
    type
       forma =
        record
          case integer of
             0 :
              (
               f5                   : packed array[1..6] of char;
               f10                  : packed array[1..10] of char;
               f20                  : packed array[1..20] of char;
               f2                   : packed array[1..2] of char;
               under_score_to       : packed array[1..14] of char;
              );
             1 :
              (
               forma_buf            : packed array[1..53] of char
              );
           end;
       forma_array = array[1..5] of integer;
    const
        forma_off = forma_array[1, 7, 17, 37, 39];
        forma_len = forma_array[6, 10, 20, 2, 14];
        forma_num = forma_array[1,2,3,4,5];
```

EXIT

Terminates DICTPDE.

Syntax

```
EXIT
E
```

Note that no subcommands are used with the EXIT command.

Discussion

EXIT allows you to terminate DICTPDE.

GENERATE

Generates PASCAL data declarations for the entities defined in the Dictionary.

Syntax

```
GENERATE subcommand G
```

Use the GENERATE command to extract data definitions from the Dictionary and to generate the corresponding PASCAL data declarations. This command is also used to generate the data structures for the VPLUS defined COMAREA and the data structures for the IMAGE parameters. (See the GENERATE COMAREA and the GENERATE IMAGEPARMS command-subcommands discussed later in this section.)

The subcommand identifies the entity to be extracted from the Dictionary. Each GENERATE command-subcommand issues a prompt which allows you to specify the entity name, except for GENERATE COMAREA. GENERATE COMAREA does not issue any prompts. A complete list of subcommands is given below.

The data declarations and COMAREA generated by the GENERATE command are echoed to the terminal and written to an output file (DICTOUT by default). The data declarations for a specified entity and the COMAREA data structure can be generated only once to an output file.

Subcommands

The following subcommands can be used with the GENERATE command:

COMAREA	generates the data structures for the VPLUS COMAREA.
C	
ELEMENT	extracts element definitions from the dictionary and generates the
E	corresponding data declarations.
FILE	extracts file definitions from the dictionary and generates the
F	corresponding data declarations.
IMAGEPARMS	generates the data structures for the IMAGE parameters.
I	

The following gives a discussion of each of the subcommands used with the GENERATE command and the prompts that they generate.

>GENERATE COMAREA

Generates the PASCAL data structures for the VPLUS COMAREA.

Prompts

GENERATE COMAREA does not issue any additional prompts. DICTPDE will begin generating the VPLUS COMAREA as soon as the GENERATE COMAREA command-subcommand is entered.

Discussion

GENERATE COMAREA allows you to generate the TYPE, CONST and VAR declarations for the VPLUS COMAREA. (The COMAREA is the data area which must be allocated in a program to be able to call VPLUS procedures.) GENERATE COMAREA generates the basic structure of the COMAREA. This command-subcommand does not extract any data definitions from the Dictionary to generate this COMAREA data structure. The generated data structures are the same as the data structures shown in the PASCAL/3000 Reference Manual. (See the PASCAL/3000 Reference Manual for more information on PASCAL and the VPLUS COMAREA.)

Example

```
>generate comarea
    type
       word = -32768..32767;
       vplus_comarea =
        record
          cstatus       : word;
          language      : word;
          comarealen   : word;
          usrbuflen     : word;
          cmode         : word;
          lastkey       : word;
          numerrs       : word;
          windowenh     : word;
          multiusage    : word;
          labeloptions  : word;
          cfname        : packed array[1..16] of char;
          nfname        : packed array[1..16] of char;
          repeatapp     : word;
          freezapp      : word;
          cfnumlines    : word;
          dbuflen       : word;
          skip2         : word;
          lookahead     : word;
          deleteflag    : word;
          showcontrol   : word;
          skip4         : word;
          printfilnum   : word;
          filerrnum     : word;
          errfilenum    : word;
          formstrsize   : word;
          skip6         : word;
          skip7         : word;
          skip8         : word;
          numrecs       : integer;
          recnum        : integer;
          skip9         : packed array[1..4] of char;
```

```
        term_filen    : word;
        skip10        : packed array[1..10] of char;
        retries       : word;
        term_options  : word;
        environ       : word;
        usertime      : word;
        identifier    : word;
        labelinfo     : word;
    end;
    const
    com_area_init =
     vplus_comarea
       [
        cstatus       : 0,
        language      : 5,
        comarealen    : 60,
        usrbuflen     : 0,
        cmode         : 0,
        lastkey       : 0,
        numerrs       : 0,
        windowenh     : 0,
        multiusage    : 0,
        labeloptions  : 0,
        cfname        : '                ',
        nfname        : '                ',
        repeatapp     : 0,
        freezapp      : 0,
        cfnumlines    : 0,
        dbuflen       : 0,
        skip2         : 0,
        lookahead     : 0,
        deleteflag    : 0,
        showcontrol   : 0,
        skip4         : 0,
        printfilnum   : 0,
        filerrnum     : 0,
        errfilenum    : 0,
        formstrsize   : 0,
```

```
      skip6          : 0,
      skip7          : 0,
      skip8          : 0,
      numrecs        : 0,
      recnum         : 0,
      skip9          : #0#0#0#0,
      term_filen     : 0,
      skip10         : #0#0#0#0#0#0#0#0#0#0,
      retries        : 0,
      term_options   : 0,
      environ        : 0,
      usertime        : 0,
      identifier     : 0,
      labelinfo      : 0,
  ];
var
   com_area : vplus_comarea;
   termfilename  :  packed array[1 .. 6] of char;
   message_buff  :  packed array[1 .. 72] of char;
   message_buff_len  :  word;
   msglen  :  word;
   buflen  :  word;
   fieldnum  :  word;
```

>GENERATE ELEMENTS

Generates PASCAL data declarations for element definitions in the Dictionary.

Prompts

The following prompt is issued when the command-subcommand GENERATE ELEMENT is entered:

```
ELEMENT(S)>
```

ELEMENT(S)> Enter the name(s) of the elements to be extracted from the Dictionary and for which the data declaration(s) will be generated. More than one element name can be entered in one command line as long as each element is separated by a comma (,) or a space. Also, the element names can not exceed the first 72 characters of the command line. DICTPDE will continue to prompt for element names until [[RETURN]] is pressed in response to this prompt.

If a parent element is entered in response to this prompt, DICTPDE will generate data declarations for the parent element as well as for the related child elements. The parent element will be generated as a record. The primary name of the parent element will be the record name. The child elements will be generated as fields within the record. The aliases of the child elements will be the field names (unless a child element does not have an alias; then its primary name will be the field name).

A data declaration can be generated for a specified element only once to an output file. However, many data declarations can be generated for a specified element if each declaration generated for that element is written to a different output file.

Pressing [[RETURN]] in response to this prompt indicates that no more element names are to be entered. DICTPDE will then stop prompting for element(s) and begin the code generation process. The generated code is echoed to the terminal as it is written to the output file.

Discussion

GENERATE ELEMENT allows you to identify the element(s) to be used for code generation.

Example

The following examples show how the GENERATE ELEMENT command-subcommand is used and the code that it generates.

The first example shows the data declaration generated for a simple element. The element “account” does not have any child elements related to it.

The element “account” is defined in the Dictionary as follows:

```
ELEMENT         = account
ELEMENT-TYPE    = I
ELEMENT-LENGTH  = 4
```

The code is generated as follows:

```
> generate element                    Or G E could have been entered.
ELEMENT NAME(S)> account

                                      The generated code is echoed to
TYPE                                  the terminal.
ACCOUNT = INTEGER;
```

The next example shows the code generated for elements that have ELEMENT-COUNTS in the Dictionary that are greater than 1. The two elements have no child entities related to them. The elements are defined in the Dictionary as follows:

```
ELEMENT         = PART1     PART2
ELEMENT-TYPE    = I         P
ELEMENT-LENGTH  = 2         10
ELEMENT-COUNT   = 10        4
```

The code is generated as follows:

```
> generate element
ELEMENT NAME(S)> part1 part2
TYPE
   PART1 = ARRAY[1..10] OF -32768..32767;
   PART2 = ARRAY[1..4] OF ARRAY[1..10] OF 0..255;
```

Note that ELEMENT-COUNT is used to determine the index of the array. ELEMENT-TYPE and ELEMENT-LENGTH are used in determining the base type of the array.

The third example shows the code generated for an element whose type is defined as an asterisk “*” in the Dictionary. (This element back-references another element.) The element is defined in the Dictionary as follows:

```
ELEMENT         = CUSTNUM
ELEMENT-TYPE    = *
ELEMENT-REFERENCE = CUSTNAME
```

CUSTNAME is defined in the Dictionary as:

```
ELEMENT         = CUSTNAME
ELEMENT-TYPE    = X
ELEMENT-LENGTH  = 14
```

The TYPE declaration for the element CUSTNUM is generated as follows:

```
>generate element
ELEMENT(S)> custnum
```

```
   TYPE
      CUSTNAME = PACKED ARRAY[1..14] OF CHAR;
      CUSTNUM : CUSTNAME;
```

The VAR declaration for the element CUSTNUM is generated as follows:

```
TYPE
   CUSTNAME = PACKED ARRAY[1..14] OF CHAR;
VAR
   CUSTNUM = CUSTNAME
```

Note that the code for the ELEMENT-REFERENCE is generated first as a TYPE declaration.

The next example shows the code generated for a parent element. (This element has child elements related to it.) In this example, the element SALES has the elements PRODUCT, PRICE and PACKAGE related to it. These elements are defined in the Dictionary as:

```
PARENT-ELEMENT   = SALES
ELEMENT-TYPE     = X
ELEMENT-LENGTH   = 50
CHILD-ELEMENTS   = PRODUCT     PRICE     PACKAGE
ELEMENT-TYPE     = X           X         X
ELEMENT-LENGTH   = 12          28        10
```

The code for the parent element SALES is generated as follows:

```
>generate element
ELEMENT(S)> sales
TYPE
   SALES =
    RECORD
      PRODUCT  : PACKED ARRAY[1..12] OF CHAR;
      PRICE    : PACKED ARRAY[1..28] OF CHAR;
      PACKAGE  : PACKED ARRAY[1..10] OF CHAR
    END;
```

Note that in this case the sum of the storage lengths for the child elements equals the storage length for the parent element as defined in the Dictionary.

In the next example, code is again generated for a parent element. However, in this example, the sum of the storage lengths for the child elements does not equal the storage length for the parent element.

```
PARENT-ELEMENT   = ADDRESS
ELEMENT-TYPE     = X
ELEMENT-LENGTH   = 48
CHILD-ELEMENTS   = STREET      NUMB
```

```
ELEMENT-TYPE    = X           I
ELEMENT-LENGTH  = 12          4
```

The code for the parent element ADDRESS is generated as follows:

```
>generate element
ELEMENT(S)> address
TYPE
   ADDRESS =
    RECORD
       CASE INTEGER OF
          0 :
           (
             BUFFER : PACKED ARRAY[1..48] OF CHAR;
           );
          1 :
           (
             DUMMY0 : PACKED ARRAY[1..3] OF CHAR;
             STREET : PACKED ARRAY[1..12] OF CHAR;
             DUMMY1 : PACKED ARRAY[1..15] OF CHAR;
             NUMB   : INTEGER
           );
    END;
```

When the sum of the storage lengths for the child elements does not equal the storage length for the parent element, DICTPDE will generate “BUFFER” or “DUMMY” fields in order to insure the correct storage length for the parent element. The BUFFER and DUMMY fields begin at the correct byte offsets within the parent element.

>GENERATE FILE

Generates the PASCAL declarations for the file definitions in the Dictionary.

Prompts

The following prompt is issued when the command-subcommand GENERATE FILE is entered:

```
FILE(S)>
```

FILE(S)> Enter the name(s) of the files to be extracted from the Dictionary and from which data declarations will be generated. More than one file can be entered in a command line as long as each file name is separated by a comma (,) or a space. Also, the file names can not exceed the first 72 characters in the command line. DICTPDE will continue to prompt for file names until [[RETURN]] is entered in response to this prompt.

If a parent file is entered in response to this prompt, DICTPDE will generate data declarations for the parent file as well as for the related child files and associated elements. The parent file will be generated as a record. The primary name of the parent file will be the record name. The child elements will be generated as fields within the record. The aliases of the child files will be the field names (unless a child file does not have an alias, then the first 8 characters of the primary name will be used as the alias). Note that any element associated to the secondary format for a file will not be extracted by DICTPDE.

A data declaration can be generated for a specified file only once to an output file. However, many data declarations can be generated for a specified file if each declaration generated for that file is written to a different output file.

Pressing [[RETURN]] in response to this prompt indicates that no more file names are to be entered. DICTPDE will then stop prompting for file(s) names and begin the code generation process. The generated code will be echoed to the terminal as it is written to the output file.

Discussion

GENERATE FILE allows you to identify the files to be used for code generation.

Example

The following examples show how the GENERATE FILE command-subcommand is used and the code that it generates.

The first example shows the data declaration generated for a simple file; that is, a file without any child files. The file type for a simple file must be defined in the Dictionary as: FORM, AUTO, DETL, MAST, KSAM, MPEF or MPER.

The file CUSTFORM is defined in the Dictionary as follows:

```
FILE           : CUSTFORM
FILE-TYPE      : FORM
FILE-ALIAS     : CUSTALIAS
```

CUSTFORM has the following elements associated with it:

```
ELEMENT        : ACCOUNT        LAST-NAME      FIRST-NAME    CREDIT
ELEMENT-ALIAS  : ACCOUNTALIAS
ELEMENT-TYPE   : I              X              X             R
ELEMENT-LENGTH : 4              16             10            4
```

The code is generated as follows:

```
>generate file
FILES(S)> custform
TYPE
    CUSTFORM =
     RECORD
       ACCOUNTALIAS  : INTEGER;
       LAST_NAME     : PACKED ARRAY[1..16] OF CHAR;
       FIRST_NAME    : PACKED ARRAY[1..10] OF CHAR;
       CREDIT        : REAL
     END;
```

Note that the alias name for the element ACCOUNT was used when the code was generated. This is the name by which this element is known by the file CUSTFORM. Also note that the hyphens in the element names have been replaced by underscores when the code was generated. (See Naming Considerations later in this section for more information on converting hyphens to underscores.)

The next example shows the code generated for a parent file (this file must be defined in the Dictionary as type BASE or VPLUS). The file STOREFF is defined in the Dictionary as follows:

```
FILE           : STOREFF
FILE-TYPE      : VPLS
```

The following files are related to STOREFF:

```
FILE           : CUSTFORM     DATEFORM     INVENTFORM
FILE-TYPE      : FORM         FORM         FORM
FILE-ALIAS     : CUSTALIAS
```

The file CUSTFORM has the following elements associated with it:

```
FILE           : CUSTFORM
ELEMENT        : ACCOUNT       LAST-NAME     FIRST-NAME    CREDIT
ELEMENT-ALIAS  : ACCOUNTALIAS
```

```
ELEMENT-TYPE   : I            X            X            R
ELEMENT-LENGTH : 4            16           10           4
```

The file DATEFORM has the following elements associated with it:

```
FILE           : DATEFORM
ELEMENT        : DATE
ELEMENT-ALIAS  :
ELEMENT-TYPE   : X
ELEMENT-LENGTH : 6
```

The file INVENTFORM has the following elements associated with it:

```
FILE           : INVENTFORM
ELEMENT        : STOCKNUM     SUPPLIER     BINNUM       UNIT-COST
ELEMENT-ALIAS  :                           BINNUMALIAS
ELEMENT-TYPE   : X            X            R            R
ELEMENT-LENGTH : 8            16           4            4
```

The code is generated for STOREFF as follows:

```
>generate file
FILE(S)> storeff
TYPE
   CUSTALIAS =
    RECORD
      ACCOUNTALIAS  : INTEGER
      LAST_NAME     : PACKED ARRAY[1..16] OF CHAR;
      FIRST_NAME    : PACKED ARRAY[1..10] OF CHAR;
      CREDIT        : REAL
    END;
   DATEFORM =
    RECORD
      DATE : PACKED ARRAY[1..6] OF CHAR;
    END;
   INVENTFORM =
    RECORD
      STOCKNUM    : PACKED ARRAY[1..8] OF CHAR;
      SUPPLIER    : PACKED ARRAY[1..16] OF CHAR;
      BINNUMALIAS : REAL;
      UNIT_COST   : REAL
    END;
```

>GENERATE IMAGEPARMS

Generates the PASCAL data structures for the IMAGE standard parameters.

Prompts

GENERATE IMAGEPARMS does not issue any additional prompts. DICTPDE will begin generating the IMAGE parameters as soon as the GENERATE IMAGEPARMS command-subcommand is entered. These parameters include the DB-STATUS array declaration, the eight MODE declarations, and the utility LIST declarations.

Discussion

GENERATE IMAGEPARMS allows you to generate the TYPE, CONST and VAR declarations for the IMAGE standard parameters. This command-subcommand does not extract any data definitions from the Dictionary to generate these parameters.

Example

```
>generate imageparms
    type
     status_type =
      record
        c_word          : -32768..32767;
        stat2           : -32768..32767;
        stat3_4         : integer;
        stat5_6         : integer;
        stat7_8         : integer;
        stat9_10        : integer;
     end;
    var
     status            : status_type;
     password          : packed array [1..8] of char;
     dummy_list        : packed array [1..2] of char; (*init '; '*)
     all_items         : packed array [1..2] of char; (*init '@;'*)
     previous_list     : packed array [1..2] of char; (*init '*;'*)
     mode1             : -32768..32767;               (*init 1*)
     mode2             : -32768..32767;               (*init 2*)
     mode3             : -32768..32767;               (*init 3*)
     mode4             : -32768..32767;               (*init 4*)
     mode5             : -32768..32767;               (*init 5*)
     mode6             : -32768..32767;               (*init 6*)
     mode7             : -32768..32767;               (*init 7*)
     mode8             : -32768..32767;               (*init 8*)
```

HELP

Displays a description of the DICTPDE commands and subcommands.

Syntax

```
HELP subcommand
    H
    ?
```

Use the HELP command to display information about each of the DICTPDE commands and their respective subcommands. Note that you can also enter "?" to invoke this command. The HELP command uses all the DICTPDE commands as its subcommands. The DICTPDE command (used as a subcommand) identifies the command for which a description will be displayed, as shown below:

Subcommands

The following subcommands can be used with the HELP command:

ALTER	defines the ALTER command and its subcommands.
A	
EXIT	defines the EXIT command and alternate ways to terminate DICTPDE.
E	
GENERATE	defines the GENERATE command and its subcommand
G	
HELP	defines each of the DICTPDE commands.
H	
LIST	defines the LIST command and its subcommands.
L	

Note that the HELP command can not be used when DICTPDE is run from a job stream.

LIST

Lists the entities that have been extracted by DICTPDE from the Dictionary.

Syntax

```
LIST subcommand
    L
```

Use the LIST command to display the entities that have been extracted from the Dictionary. These are the entities for which data declarations have been generated. The LIST command also shows the types of the entities extracted (whether file or element), the kind of code generated for each entity (whether VAR or TYPE), and the output file the generated code was written to. The subcommand identifies the entity type (file or element) to be listed. All the entities of that type are listed.

A complete list of subcommands is given below.

Subcommands

The following subcommands can be used with the LIST command:

ALL	lists all the files and elements extracted from the Dictionary.
A	
ELEMENT	lists the elements extracted from the Dictionary.
E	
FILE	lists the files extracted from the Dictionary.
F	

A discussion of each of the subcommands used with the LIST command and the prompts that they generate follows.

>LIST ALL

Displays all the entities extracted from the Dictionary.

Prompts

LIST ALL does not generate any additional prompts. This command-subcommand displays the element and file names as they were extracted from the Dictionary. (The entities identified as files are listed first. The entities identified as elements are listed after the files.) LIST ALL also helps you to keep track of the entities that have been extracted from the Dictionary and to determine what output files contain the data declarations for what entities.

Discussion

LIST ALL allows you to list all the elements and files extracted from the Dictionary. LIST ALL also displays information about the extracted entities.

Example

```
>list all
    entity-name          entity     kind    output-file
    product              file       type    dictout.pub.howe
    sales                file       type    dictout.pub.howe
    price                file       var     accts.pub.howe
    street               element    var     accts.pub.howe
    city                 element    var     accts.pub.howe
    state                element    type    dictout.pub.howe
    zip                  element    type    dictout.pub.howe
```

>LIST ELEMENT

Displays the elements extracted from the Dictionary.

Prompts

LIST ELEMENT does not generate any additional prompts. This command-subcommand displays the element names as they were extracted from the Dictionary. LIST ELEMENT also shows the kind of data declaration generated for each of the extracted elements and the output file each element declaration was written to.

Discussion

LIST ELEMENT allows you to list all the elements extracted from the Dictionary in the order in which they were extracted. LIST ELEMENT also helps you to keep track of the elements that were extracted and to determine what output files contain the data declarations for what elements.

Example

```
>list element
   entity-name    entity        kind        output-file
   street         element       var         accts.pub.howe
   city           element       var         accts.pub.howe
   state          element       type        dictout.pub.howe
   zip            element       type        dictout.pub.howe
```

>LIST FILE

Displays the files extracted from the Dictionary.

Prompts

LIST FILE does not generate any additional prompts. This command-subcommand displays the file names as they were extracted from the Dictionary. LIST FILE also shows the kind of data declaration generated for each of the extracted files and the output file that each declaration was written to.

Discussion

LIST FILE allows you to list all the files extracted from the Dictionary in the order in which they were extracted. LIST FILE also helps you to keep track of the files that have been extracted and to determine what output files contain the data declarations for what file declarations.

Example

```
>list file
    entity-name          entity     kind     output-file
    product              file       var      accts.pub.howe
    sales                file       var      accts.pub.howe
    price                file       type     dictout.pub.howe
    credit               file       type     dictout.pub.howe
```

DICTPDE Naming Considerations

When an entity is extracted from the Dictionary, DICTPDE checks that the entity name is compatible with PASCAL. If the entity name defined in the Dictionary is an illegal PASCAL name (for example, the name contains a hyphen) or the entity name is a PASCAL reserved word, an error message will be issued and DICTPDE will convert the names to legal PASCAL names. Note that the names generated by DICTPDE will not be reflected in the Dictionary.

The following describes how DICTPDE converts entity names that are not compatible with PASCAL.

If the entity name as defined in the Dictionary begins with a character other than an alphabetic character, DICTPDE converts the first letter of the name to a d (for “dummy”) as shown below: If the entity name in the Dictionary is:

```
#stock
```

The following PASCAL name will be generated:

```
dstock
```

If an entity name as defined in the Dictionary consists of any characters other than alphabetic or numeric characters, or underscores (__), that character will be converted to an underscore. (Any hyphens or minus signs are converted to underscores.) If the entity name in the Dictionary is:

```
cust-stock#
```

The following PASCAL name will be generated:

```
cust_stock_
```

If the entity name extracted from the Dictionary is a PASCAL reserved word, DICTPDE will append “__reswd” to the end of the entity name. For example, if the entity name in the Dictionary is:

```
array
```

The following PASCAL name will be generated:

```
array_reswd
```

PASCAL Data Type Mappings

To be compatible with PASCAL, the data types for the extracted entities as defined in the Dictionary must be mapped to PASCAL data types. The following shows how the Dictionary data types are mapped to compatible PASCAL data types. Note that the PASCAL type occupies the same storage length as the data types defined in the Dictionary. (The storage length is computed from the ELEMENT-LENGTH and ELEMENT-COUNT entries for an entity in the Dictionary.)

Any ASCII character:

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE    = X
ELEMENT-LENGTH  = N
```

The corresponding PASCAL data type will be generated:

```
CHAR (if N=1) or
PACKED ARRAY[1..N] OF CHAR (if N > 1)
```

For example, if an entity is defined in the Dictionary as:

```
ELEMENT          = ACCOUNT_NAME
ELEMENT-TYPE     = X
ELEMENT-LENGTH   = 20
```

The PASCAL code will be generated as:

```
ACCOUNT_NAME = PACKED ARRAY[1..20] OF CHAR
```

Uppercase alphanumeric string:

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = U
ELEMENT-LENGTH   = N
```

The corresponding PASCAL data type will be generated:

```
' '..'Z' (if N = 1) or
PACKED ARRAY[1..N] OF ' '..'Z' (if N > 1)
```

Note that the PASCAL subrange ' '..'Z' restricts characters from being lowercase.

Numeric ASCII string:

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = 9 or 9+
ELEMENT-LENGTH   = N
```

The corresponding PASCAL data type will be generated:

```
'0'..'9' (if N = 1) or
```

```
PACKED ARRAY[1..N] OF '0'..'9' (if N > 1)
```

However, you must check that the minus sign doesn't appear in the data.

Zoned decimal:

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = Z
ELEMENT-LENGTH   = N
```

The corresponding PASCAL data type will be generated:

```
'+'..'}' (if N = 1) or
PACKED ARRAY[1..N] OF '0'..'}' (if N > 1)
```

The above mapping indicates that the zoned decimal consists of (N - 1) digits with a trailing overpunch.

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = Z+
ELEMENT-LENGTH   = N
```

The corresponding PASCAL data type will be generated:

```
'+'..'{' (if N = 1) or
PACKED ARRAY[1..N] OF '0'..'{' (if N > 1)
```

Logical value (absolute binary):

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = K (or K+)
ELEMENT-LENGTH   = 2
```

The corresponding PASCAL data type will be generated:

```
SET of 0..15
```

For any other ELEMENT-LENGTHs, an “Undefined PASCAL type” will be generated.

Packed decimal:

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = P (or P+)
ELEMENT-LENGTH   = N
```

The corresponding PASCAL data type will be generated:

```
#0..#255 (if N = 1) or
PACKED ARRAY[1..N] OF #0..#255 (if N > 1)
```

Boolean:

If the entity is defined in the Dictionary as

```
ELEMENT-TYPE = B
```

The corresponding PASCAL data type will be generated:

```
BOOLEAN
```

Note that ELEMENT-LENGTH is not needed in this case.

String:

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = S
ELEMENT-LENGTH   = N
```

The corresponding PASCAL data type will be generated:

```
STRING[N]
```

Note that the storage length is not the ELEMENT-LENGTH for string data types.

Integer number:

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = I (or J)
ELEMENT-LENGTH   = 2
```

The corresponding PASCAL data type will be generated:

```
-32768..32767
```

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = I (or J)
ELEMENT-LENGTH   = 4
```

The corresponding PASCAL data type will be generated:

```
INTEGER
```

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = I+ (or J+)
ELEMENT-LENGTH   = 2
```

The corresponding PASCAL data type will be generated:

```
0..32767
```

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = I+ (or J+)
ELEMENT-LENGTH   = 4
```

The corresponding PASCAL data type will be generated:

```
0..MAXINT
```

Real number:

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE     = E (or R)
```

```
ELEMENT-LENGTH    = 4
```

The corresponding PASCAL data type will be generated:

```
REAL
```

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE      = E (or R)
ELEMENT-LENGTH    = 8
```

The corresponding PASCAL data type will be generated:

```
LONGREAL
```

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE      = E+ (or R+)
ELEMENT-LENGTH    = 4
```

The corresponding PASCAL data type will be generated:

```
REAL
```

If the entity is defined in the Dictionary as:

```
ELEMENT-TYPE      = E+ (or R+)
ELEMENT-LENGTH    = 8
```

The corresponding PASCAL data type will be generated:

```
LONGREAL
```

For all other cases:

For an element that back-references another element data type, PASCAL will map the data type of the back referenced element to a compatible PASCAL data type.

If the element-length or the byte position is not valid for a REAL, LOGICAL, or INTEGER data type, DICTPDE will generate the following:

```
PACKED ARRAY[1..ELEMENT_LENGTH] OF CHAR
```

A warning is also issued indicating that even though the element length or byte position is not valid for that data type, a data declaration was generated for that entity. Please note that these entities should not be accessed for use in PASCAL programs unless the correct PASCAL can be made for the data type. (In order to do so, the invalid element length or byte position should be corrected in the Dictionary.)

7 The Dictionary DICTCDE Utility

Overview

DICTCDE, the COBOL Definition Extract Utility, is an interactive, prompt-driven utility provided by Dictionary/3000. This utility extracts data and file definitions from the Dictionary and generates the corresponding COBOL source code for the data definitions. When a parent entity is extracted from the Dictionary, DICTCDE also generates the code for any related child entities. The entities extracted from the Dictionary can be generated by their primary names or by their aliases. (Refer to Changing the DICTCDE Options discussed later in this section for more information on using the entities' aliases.) As the source code is generated, it is echoed to the terminal and written to the COBOL copylib that you specify.

When source code is generated for an entity, DICTCDE checks that the entity name and data type as defined in the Dictionary for that entity are compatible with COBOL. If the data type for the entity is not COBOL compatible, DICTCDE will issue a warning message and convert the data type to compatible COBOL code. (See COBOL Data Type Mappings later in this section for more detailed information.) If the entity name as defined in the Dictionary is an illegal COBOL name (for instance, if the name begins with a hyphen) or the entity name is a COBOL reserved word, DICTCDE will issue a warning message and convert the name to a legal COBOL name. (See DICTCDE Naming Considerations later in this section for more detailed information.)

DICTCDE also provides an interactive help facility which gives a brief description of the DICTCDE prompts and a set of options which allows you to change the defaults used by DICTCDE.

This section describes how to run DICTCDE and use its help facility, defines the options that can be changed, and gives examples of the source code generated by DICTCDE.

Special Character Responses

The following characters have special meaning when used in response to a DICTCDE prompt:

]	the previous prompt issued is redisplayed. It does not affect any source code that has already been generated.
]]	the main prompt is reissued. This prompt allows you to generate code, change the program options, or exit DICTCDE. (This prompt is discussed later in this section.)
!	DICTCDE responds as if you had pressed [[RETURN]] in response to all prompts that follow for that prompting cycle. A prompting cycle is complete when the main prompt is reissued.
[[CONTROL]] Y	the main prompt is reissued (as with]]).
?	the Help facility is invoked for a description of a prompt.
:*mpecommand*	the MPE command is executed without leaving DICTCDE.

Initiating DICTCDE

DICTCDE can be initiated from within a session or from within a job stream. The following describes how to execute DICTCDE interactively (from within a session). For information on how to execute DICTCDE from within a job stream, see Changing Input, List and Output Files discussed later in this section.

DICTCDE assumes that the Dictionary resides in DICT.PUB of the logon account. To redefine the Dictionary, you can use an MPE file equation or you can use the option provided by DICTCDE to change the Dictionary. (The Dictionary option is discussed in detail later in this section.)

To execute DICTCDE, enter the following MPE command:

```
RUN DICTCDE.PUB.SYS
```

After the DICTCDE banner is displayed, the following message and prompt are issued:

```
Type ? at any prompt for help
Dictionary password>
```

The above message indicates that the Help facility can be invoked by entering a question mark (?) in response to any prompt. The Help facility gives a brief description of the prompt in question.

Enter the password for the Dictionary. If the Dictionary password is entered incorrectly or is an invalid password for the Dictionary, DICTCDE will issue the following error message:

```
Password is invalid
```

DICTCDE will then continue to prompt for the Dictionary password until a valid password is entered. If [[RETURN]] is pressed in response to this prompt, DICTCDE will terminate.

If DICTCDE cannot open the Dictionary, the following error message are issued:

```
ERROR: Cannot open dictionary dictionary name
```

Check to see that the Dictionary is in the correct group and account. If a file equation was used to redirect the Dictionary, check that the Dictionary name was entered correctly. DICTCDE will then ask for a new Dictionary to be entered and will also prompt for the password for that Dictionary.

Once the specified Dictionary and password have been accepted by DICTCDE, the following prompt is issued:

```
Copylib file name>
```

Enter the COBOL copylib file name to which you are directing the generated source code. The copylib file name may be up to 35 characters maximum and may be fully qualified and have a lockword. The name must be a valid MPE file name. If the specified copylib file already exists, DICTCDE will open that copylib file and append any source code generated to the contents of that file. If the specified file name exists, but it is not in KSAM copylib format, DICTCDE will issue an error message and then reprompt for the copylib file name. If the specified file name does not exist, DICTCDE will issue the following prompt:

```
File filename  does not exist, create it (N/Y)>
```

Enter Y to create the specified copylib file. If N,],]], [[RETURN]], or [[CONTROL]] Y is entered in response to this prompt, DICTCDE will again prompt for the Copylib file name.

Since a COBOL copylib is in KSAM file format, a key file must be associated with the KSAM copylib file. Before DICTCDE creates a new KSAM copylib file, you are prompted for the key file name for the new copylib as follows:

```
Key file name>
```

Enter the copylib key file name (8 characters maximum) for the new copylib file. This file will be created in the group and account you logged on with. Since this file can not be qualified, you should log on to the group and account you would like this file to be created in.

If],]], [[RETURN]] or [[CONTROL]] Y is entered in response to this prompt, DICTCDE will reprompt for the copylib file name.

If [[RETURN]] is pressed in response to the Copylib file name prompt, DICTCDE will terminate.

The DICTCDE Main Prompt

When the copylib file is either accepted or created by DICTCDE, the main prompt used for generating source code and for changing the DICTCDE options is issued:

File, Element, Parameters, Options, or Exit (F/E/P/O/EX)>

The responses allowed for this prompt indicate the following:

F	begins the prompting cycle for generating source code for IMAGE, VPLUS, KSAM, MPEF and MPER file definitions.
E	begins the prompting cycle for generating source code for element definitions.
P	begins the prompting cycle for generating source code for IMAGE, VPLUS, and KSAM standard parameters. (These parameters are discussed later in this section.)
O	begins the prompting cycle for changing the options provided by DICTCDE. (These options are discussed in detail later in this section.)
EX	exits DICTCDE.

Each of the above responses (except the EX response) generates additional prompts. The additional prompts are shown as each response is discussed in more detail.

Note that if],]], [[CONTROL]] Y, or [[RETURN]] is entered in response to the main prompt, DICTCDE will reissue the prompt.

Generating Code for Files

To generate source code for file definitions, enter F in response to the main prompt.

DICTCDE will then prompt for the file name as follows:

```
File name>
```

Enter the name of the file to be extracted from the Dictionary. Only one file may be extracted at a time. (You may not enter multiple file names in response to this prompt.) Note that if the file to be extracted is an IMAGE data set or a VPLUS form, you may qualify the data set or form by its parent file. To do so, enter the name of the database or forms file in parenthesis next to the file name as shown below:

```
File name> filename (parent filename)
```

When a file is identified by its parent file, then DICTCDE will generate the code for the file using the file's alias (if any) within the parent file if the alias option is enabled. (The alias option is discussed in more detail later in this section.)

DICTCDE then displays the file type of the specified file and continues to prompt for information. The prompts that are issued by DICTCDE depend on the type of the file specified.

IMAGE Database Files. If the specified file is an IMAGE database, DICTCDE will issue the following message and prompts:

```
File filename  is an IMAGE database
Define all data sets in one module (N/Y)>
```

Enter Y if the generated source code for all the data sets is to be written to one copylib module. This module will also contain the source code for the IMAGE special parameters. These are the IMAGE special name-constant parameters that identify the database and password for use by the IMAGE intrinsics. To generate the IMAGE standard parameters, see the discussion for the P (Parameters) response to the main prompt. (The IMAGE standard parameters include the DB-STATUS array definition, the eight MODES as constants, and the various LIST parameters.)

When all data sets are to be defined in one module, DICTCDE will prompt for the copylib module for the database definition:

```
Copylib module for database name >
```

Enter the name of the copylib module (8 characters maximum). DICTCDE then begins the generation process. The generated source code is echoed to the terminal as it is written to the copylib module. If [[RETURN]] is pressed in response to this prompt, DICTCDE will reissue the main prompt and no source code would be generated for the database.

If the source code generated for the data sets is to be written to different copylib modules, enter N or press [[RETURN]] in response to Define all data sets in one module (N/Y)>. DICTCDE will then prompt for the copylib module for each of the data sets in turn and for the IMAGE special parameters as shown:

```
Copylib module for database name  special parameters>
```

Enter the copylib module to which you will direct the source code for the IMAGE special parameters. If [[RETURN]] is pressed in response to this prompt, the code for the IMAGE special parameters will not be generated and DICTCDE will issue the next prompt.

```
Copylib module for data set name >
```

Enter the copylib module for the data set specified by DICTCDE in data set name. If [[RETURN]] is pressed in response to this prompt, the source code for this data set will not be generated and DICTCDE will prompt for the copylib module for the next data set that belongs to the specified database. Once [[RETURN]] has been pressed in response to a prompt for a data set, that data set is skipped (code can not be generated for that data set). If] is entered in response to this prompt, DICTCDE will again prompt for a file name.

After a copylib module has been defined for a data set, DICTCDE issues the following prompt:

```
Prefix for data items in data set name >
```

This prompt allows you to enter a prefix which is used in the COBOL identifier for each data item in the specified data set. The prefix you choose can be 8 characters maximum and must begin with a letter to be a valid COBOL identifier. If you would like a hyphen between the prefix and the data item name, be sure to include it as part of the prefix. (See Examples of Generated Code later in this section for an example of data items generated with prefixes.) When a prefix is assigned to a data item name, those data items can be referred to uniquely in a COBOL program without using the COBOL OF clause.

If [[RETURN]] is pressed in response to this prompt, the source code for that data set is generated without a prefix assigned to the data items. The code is echoed to the terminal as it is written to the copylib module. After the code for this data set is generated, DICTCDE will continue to prompt for copylib modules and prefixes for the remaining data sets in the database.

After prompting for each data set in the specified database, DICTCDE will return to the main prompt.

IMAGE Data Set Files. If the specified file is an IMAGE data set, DICTCDE will issue the following message and prompts:

```
File filename  is an IMAGE MASTER data set.
```

(DICTCDE also tells you what type of data set the file is.)

```
Copylib module for data set name >
```

Enter the name of the copylib module (8 characters maximum). If [[RETURN]] is pressed in response to this prompt, DICTCDE will return to the main prompt and no code will be generated.

Once the copylib module for the specified data set has been entered, DICTCDE will prompt for the following:

```
Prefix for data items in data set name >
```

Enter the prefix to be used in the COBOL identifier for the data items in the data set. The prefix you choose can be up to 8 characters and must begin with a letter to be a valid COBOL identifier. If you would like a hyphen between the prefix and the data item name, be sure to include the hyphen as part of the prefix. (See Examples of Generated Code later in this section for an example of data items generated with prefixes.) When a prefix is assigned to a data item, that data item can be referred to uniquely in a COBOL program without qualification.

If [[RETURN]] is pressed in response to this prompt, the source code for the data set is generated without a prefix assigned to the data items. The code is echoed to the terminal as it is written to the copylib module.

After the source code has been generated for the data set, DICTCDE will return to the main prompt.

VPLUS Forms Files. If the specified file is a VPLUS forms file, DICTCDE will issue the following message and prompts:

```
File filename  is a VPLUS forms file.
Define all forms in one module (N/Y)>
```

Enter Y if the generated source code for all the forms in the forms file is to be written to one copylib module. This module will also contain the source code for the forms file name. DICTCDE will then prompt for the copylib module for the source code for the entire forms file as follows:

```
Copylib module for forms file name >
```

Enter the name of the copylib module. DICTCDE will then issue the next prompt:

```
Define field number tables (N/Y)>
```

Enter Y to generate a field number table for each form in the forms file. The field number tables are useful in calls to some of the VPLUS intrinsics, such as VSETERROR. The tables provide the field numbers for each field in the form. Each field in the form is represented by a data item in the table. The data items are COMP variables initialized to the field number. (See the Examples of Generated Code later in this section for examples of the field number tables.) Enter N or press [[RETURN]] in response to this prompt if you do not want field number tables generated for the forms.

The data items defined as numeric VPLUS fields in the Dictionary only appear as ASCII fields in the VPLUS buffer. Because of this, DICTCDE asks whether these fields are to be defined as PIC 9 or PIC X when the code is generated:

```
Define numeric fields as PIC 9 or PIC X (9/X)>
```

Enter X to define every numeric field in the forms file as ASCII alphanumeric (PICTURE CLAUSE PIC X). Enter 9 or press [[RETURN]] to define every numeric field as ASCII numeric only (PICTURE CLAUSE PIC 9). If field edits have been defined for the numeric fields in the forms file, it may be more appropriate to define the numeric fields as PIC 9. However, if you choose to define the numeric fields as PIC 9, the forms must only allow numeric data to be entered in those fields.

DICTCDE will then begin the generation process. The generated code is echoed to the terminal as it is written to the copylib module.

If the source code generated for the forms is to be written to different copylib modules, enter N or press [[RETURN]] in response to Define all forms in one module (N/Y)>. DICTCDE will then issue the following prompts:

```
Copylib module for forms file name constant>
```

Enter the copylib module to which you are directing the source code for the forms file name. DICTCDE will echo the code generated for the forms file name to the terminal as it is written to the copylib module. If [[RETURN]] is pressed in response to this prompt, this code will not be generated and the next prompts will be issued.

The next two prompts apply to all the forms in the forms file to be generated. These are the same prompts issued when the entire forms file is to be defined in one copylib module.

```
Define field number tables (N/Y)>
```

Enter Y to generate field number tables for each form to be generated. Enter N or press [[RETURN]] if you do not want to generate field number tables.

```
Define numeric fields as PIC 9 or PIC X (9/X)>
```

Enter X to define every numeric field in every form generated as PIC X. Enter 9 or press [[RETURN]] to define every field as PIC 9. (Remember, if 9 is entered in response to this prompt, the forms must only allow numeric data to be entered in those fields.) DICTCDE then begins prompting for copylib modules for each of the forms in the forms file.

```
Copylib module for formname >
```

Enter the copylib module for the form specified by DICTCDE in formname. If [[RETURN]] is pressed in response to this prompt, the source code for this form will not be generated. (The field number tables will not be generated for that form either.) Once [[RETURN]] has been pressed in response to the copylib module for a form, that form is skipped (code can not be generated for that form) and DICTCDE will then prompt for the copylib module for

the next form in the forms file.

After the copylib module has been defined for a form, DICTCDE issues the following prompt:

```
Prefix for data items in formname >
```

Enter a prefix to be used in the COBOL identifier for the data item in the form. The prefix you choose can be up to 8 characters long and must begin with a letter to be a valid COBOL identifier. If you would like a hyphen between the prefix and the data item name, be sure to include the hyphen as part of the prefix you enter. (See Examples of Generated Code later in this section for an example of data items generated with prefixes.) When a prefix is assigned to a data item, that data item can be referred to uniquely in a COBOL program without using the COBOL OF clause.

If [[RETURN]] is pressed in response to this prompt, the source code for the data set is generated without a prefix assigned to the data items. The code is echoed to the terminal as it is written to the copylib module. DICTCDE then prompts for the copylib module for the next form in the forms file.

After the source code has been generated for the forms file, DICTCDE will return to the main prompt.

VPLUS Forms. If the specified file is a VPLUS form, DICTCDE will issue the following message and prompts:

```
File formname  is a VPLUS form.
```

Note that the next prompts are also issued when the file is a forms file.

```
Define field number tables (N/Y)>
```

Enter Y to generate field number tables for the specified form. Enter N or press [[RETURN]] if you do not want field number tables generated.

```
Define numeric fields as PIC 9 or PIC X (9/X)>
```

Enter X to define every numeric field in the specified form as PIC X. Enter 9 or press [[RETURN]] to define every numeric field as PIC 9. (Remember, if 9 is entered in response to this prompt, the form must only allow numeric data to be entered in those fields.)

DICTCDE next prompts for the copylib module.

```
Copylib module for formname >
```

Enter the copylib module for the specified form. If [[RETURN]] is pressed in response to this prompt, the source code for this form will not be generated and DICTCDE will reprompt for a file name. (The field number tables will not be generated for the form either.)

After the copylib module has been defined for a form, DICTCDE issues the following prompt:

```
Prefix for data items in formname >
```

Enter a prefix to be used in the COBOL identifier for the data items in the form. The prefix you choose can be up to 8 characters long and must begin with a letter to be a valid COBOL identifier. If you would like a hyphen between the prefix and the data item name, be sure to include the hyphen as part of the prefix you enter. (See Examples of Generated Code later in this section for an example of data items generated with prefixes.) When a

prefix is assigned to a data item, that data item can be referred to uniquely in a COBOL program without using the COBOL OF clause.

If [[RETURN]] is pressed in response to this prompt, the source code for the form is generated without a prefix assigned to the data items. The code is echoed to the terminal as it is written to the copylib module.

After the source code has been generated for the form, DICTCDE will return to the main prompt.

MPEF and MPER Files. If the specified file is defined in the Dictionary as MPEF, DICTCDE will issue the following message:

```
File filename  is an MPE file.
```

If the specified file is defined in the Dictionary as MPER, DICTCDE will issue the following message:

```
File filename  is an MPE relative file.
```

The next prompts issued by DICTCDE allow you to choose what COBOL program sections are to be defined for the file. DICTCDE allows you to generate source code for the ENVIRONMENT DIVISION and the DATA DIVISION of a COBOL program.

```
Copylib module for SELECT statement>
```

Enter the copylib module for the code generated for the SELECT statement. The SELECT statement identifies the file to be used in a program and is part of the ENVIRONMENT DIVISION of a COBOL program. When a copylib module is entered in response to this prompt, DICTCDE will then prompt for the copylib module for the next program section. If [[RETURN]] is pressed in response to this prompt, no SELECT statement will be generated, and DICTCDE will continue to prompt for the copylib module for the next program section.

```
Copylib module for FILE SECTION entry>
```

Enter the copylib module for the code generated for the FILE SECTION entry. The FILE SECTION defines the file to be used in the program and is part of the DATA DIVISION of the program. The record layout for the file (the source code for the data items in the file) is defined in this section. However, you may choose to generate the record layout for the file in WORKING-STORAGE, instead of in the FILE SECTION. If a copylib module is entered for the FILE SECTION, but you choose to generate the record layout for the file in WORKING-STORAGE, the FD statement would contain a single, 01-level definition. This definition is generated to serve as a buffer for READs and WRITEs to the file. If a copylib module is not entered for the FILE SECTION, the record definition for the file may still be generated in WORKING-STORAGE.

If [[RETURN]] is pressed in response to this prompt, no FILE SECTION entry will be generated and DICTCDE will continue to prompt for the copylib module for the next program section. When a copylib module is entered for the FILE SECTION, DICTCDE prompts for the following:

```
Define filename  in FILE SECTION as an FD or SD file (F/S)>
```

Enter F to define filename as a data file. Enter S to define filename as a sort file. If [[RETURN]] is pressed in response to this prompt, filename will be defined as a data file.

Note that the FILE SECTION for an MPER file can only be generated as an FD file.

(Therefore, this prompt is not issued for MPER files.)

DICTCDE then issues the next prompt:

```
Copylib module for WORKING-STORAGE record>
```

Enter the copylib module for the WORKING-STORAGE record. The WORKING STORAGE record will contain the record layout for the file. If a copylib module was entered in response to the prompt for the FILE SECTION entry, and [[RETURN]] is pressed in response to this prompt, the record layout of the file will be defined in the FILE SECTION. If no copylib module was entered in response to the prompt for the FILE SECTION entry, and [[RETURN]] is also pressed in response to this prompt, no record layout for the file will be generated.

If the copylib modules have been defined for either the FILE SECTION or for WORKING-STORAGE, DICTCDE prompts for the following:

```
Prefix for data items in filename >
```

Enter the prefix to be used in the COBOL identifier for each data item in the file. The prefix can be 8 characters maximum and must begin with a letter to be a valid COBOL identifier. This prefix can be used to uniquely identify the data items to the file. (See Examples of Generated Code which is discussed later in this section for an example of data items generated with prefixes.)

If [[RETURN]] is pressed in response to this prompt, the source code for the file is generated without a prefix assigned to the data items. The code is echoed to the terminal as it is written to the copylib module. DICTCDE then returns to the main prompt.

KSAM Files. If the specified file is a KSAM file, DICTCDE will issue the following message and prompts:

```
File filename  is a KSAM file.
```

DICTCDE allows you to use KSAM files in two ways. First, you may use the KSAM support provided by COBOL II. To do so, you would generate the SELECT statement and FILE SECTION, but you would not generate the record layout in WORKING STORAGE. Second, you may use the intrinsics supplied by KSAM for use by COBOL. To do this, you would generate the record layout in WORKING STORAGE to serve as a data buffer parameter to the KSAM intrinsics, but you would not generate the SELECT statement or the FILE SECTION entry.

The next prompts issued by DICTCDE allow you to choose what COBOL program sections are to be defined for the file. DICTCDE allows you to generate source code for the ENVIRONMENT DIVISION and the DATA DIVISION of a COBOL program as it did for an MPE file.

```
Copylib module for SELECT statement>
```

Enter the copylib module for the code generated for the SELECT statement. The SELECT statement identifies the file to be used in a COBOL program and is part of the ENVIRONMENT DIVISION for the program. When a copylib module is entered in response to this prompt, DICTCDE will then prompt for the copylib module for the next program section. If [[RETURN]] is pressed in response to this prompt, no SELECT statement will be generated and DICTCDE will continue to the next prompt.

```
Copylib module for FILE SECTION entry>
```

Enter the copylib module for the code generated for the FILE SECTION entry. The FILE SECTION defines the file to be used in the program and is part of the DATA DIVISION for a program. The record layout for the file (the source code for the data items in the file) is also defined in this section. However, you may choose to generate the record layout for the file in WORKING-STORAGE, instead of in the FILE SECTION. If a copylib module is entered for the FILE SECTION, but you choose to generate the record layout for the file in WORKING-STORAGE, the FD statement would be generated as a single, 01-level definition. This definition is generated to serve as a buffer for READs and WRITEs to the file. If a copylib module is not entered for the FILE SECTION, the record definition for the file may still be generated in WORKING-STORAGE.

If [[RETURN]] is pressed in response to this prompt, no FILE SECTION entry will be generated and DICTCDE will continue to prompt for the copylib module for the next program section.

```
Copylib module for WORKING-STORAGE record>
```

Enter the copylib module for the WORKING-STORAGE record. The WORKING-STORAGE record will contain the record layout for the file. If a copylib module was entered in response to the prompt for the FILE SECTION entry and [[RETURN]] is pressed in response to this prompt, the record layout of the file will be defined in the FILE SECTION. If no copylib module was entered in response to the prompt for the FILE SECTION entry, and [[RETURN]] is pressed in response to this prompt, no record layout for the file will be generated.

When a copylib module is defined for the WORKING-STORAGE record for a KSAM file, the following prompt is also issued by DICTCDE:

```
Include KSAM FILETABLE parameter (N/Y)>
```

Enter Y to generate the KSAM FILETABLE parameter for this file. The KSAM FILETABLE parameter is useful if you use the intrinsics provided by KSAM for COBOL, rather than the KSAM support provided by COBOL II.

When the copylib modules have been defined for either the FILE SECTION or for WORKING-STORAGE, DICTCDE prompts for the following:

```
Prefix for data items in filename >
```

Enter the prefix to be used in the COBOL identifier for each data item in the file. The prefix can be 8 characters maximum and must begin with a letter to be a valid COBOL identifier. This prefix can be used to uniquely identify the data items to the file. (See Examples of Generated Code later in this section for an example of data items generated with prefixes.)

If [[RETURN]] is pressed in response to this prompt, the source code for the KSAM file is generated without a prefix assigned to the data items in the file. The code is echoed to the terminal as it is written to the copylib module.

After the source code has been generated for the KSAM file, DICTCDE will return to the main prompt.

Generating Code for Elements

To generate source code for element definitions, enter E in response to the main prompt.

DICTCDE will then prompt for the element name:

```
Element name>
```

Enter the name of the element to be extracted from the Dictionary. Only one element may be extracted at a time. (You may not enter more than 1 element name in response to this prompt.) DICTCDE then prompts for the copylib module for the source code for the specified element:

```
Copylib module for element name >
```

Enter the name of the copylib module for this element. If [[RETURN]] is pressed in response to this prompt, no code will be generated for this element and DICTCDE will return to the main prompt.

After the copylib module has been defined for the element, DICTCDE will prompt for the following:

```
Prefix for data items in element name >
```

Enter the prefix to be used as the COBOL identifier for any child elements this element may have. The prefix can be up to 8 characters long and must begin with a letter to be a valid COBOL identifier. When a prefix is assigned to a child element, that element can be uniquely identified with its parent element. For example, for the parent element RECORD1, the following code is generated if the prefix REC1- is entered in response to this prompt:

```
000100
000200 01  RECORD1.
000300     05  REC1-FIELD1   PIC X(2).
000400     05  REC1-FIELD2   PIC X(8).
000500     05  REC1-FIELD3   PIC X(6).
```

If the specified element does not have any related child elements, the prefix will be ignored. If [[RETURN]] is pressed in response to this prompt, the code is generated without any prefixes assigned to the child elements (if there are any). The code is echoed to the terminal as it is written in the WORKING-STORAGE record.

Generating Code for Standard Parameters

DICTCDE allows you to generate source code for the IMAGE, VPLUS, and KSAM standard parameters. To generate this source code, enter P in response to the main prompt. The code for the standard parameters is generated in WORKING-STORAGE.

DICTCDE will issue the following prompt:

```
Copylib module for IMAGE standard parameters>
```

Enter the copylib module for the code generated for the IMAGE standard parameters. The IMAGE standard parameters include the DB-STATUS array declaration, the eight MODE declarations, and the utility LIST declarations. The code for the IMAGE standard parameters is echoed to the terminal as it is written to the copylib module. (See Examples of Generated Code later in this section for an example of the code generated for the IMAGE standard parameters.) If [[RETURN]] is pressed in response to this prompt, the IMAGE standard parameters will not be generated.

DICTCDE next prompts for the copylib module for the VPLUS parameters:

```
Copylib module for VPLUS standard parameters>
```

Enter the copylib module for the code generated for the VPLUS standard parameters. The VPLUS standard parameters include the VPLUS COMAREA definition and a character string which identifies the terminal. (The terminal identification string is passed to the VPLUS intrinsic VOPENTERM.) The code for the VPLUS standard parameters is echoed to the terminal as it is written to the copylib module. (See Examples of Generated Code later in this section for an example of the code generated for the VPLUS standard parameters.) If [[RETURN]] is pressed in response to this prompt, the VPLUS standard parameters will not be generated.

DICTCDE will next prompt for the KSAM standard parameters:

```
Copylib module for KSAM standard parameters>
```

Enter the copylib module for the code generated for the KSAM standard parameters. The KSAM standard parameters are useful if you use the intrinsics provided by KSAM for COBOL, rather than the COBOL II KSAM support. These parameters include a general KSAM FILETABLE parameter and a STAT parameter. The general FILETABLE parameter does not reference any file name and can be used for any file with the appropriate initialization. The STAT parameter contains STATUS-KEY-1 and STATUS-KEY-2 for use with the KSAM COBOL intrinsics. The code for the KSAM standard parameters is echoed to the terminal as it is written to the copylib module. (See Examples of Generated Code later in this section for an example of the code generated for the KSAM standard parameters.) If [[RETURN]] is pressed in response to this prompt, the KSAM standard parameters will not be generated.

After the code has been generated for the standard parameters, DICTCDE returns to the main prompt.

Changing the DICTCDE Options

DICTCDE provides a set of options which allows you to change the Dictionary used, the copylib used, and the DICTCDE defaults; and it allows you to run the COBOL copylib editor, COBEDIT.

To change any of the options offered by DICTCDE, enter an O in response to the main prompt. DICTCDE will then issue a series of prompts which allow you to change any of the options. Each of the prompts displays the value currently being used by DICTCDE in either of two ways. For some of the prompts, the current value is shown within parenthesis as a single value. For example, the current value for the following prompt is DICT.PUB:

```
Dictionary name (DICT.PUB)>
```

For prompts which ask a choice to be made, the current value is also within parenthesis. It is the value to the left of the slash. In this example, the current value is Y (Yes):

```
List definitions on the terminal (Y/N)>
```

When O is entered in response to the main prompt, DICTCDE issues the following message:

```
Press RETURN to retain current value and move to the next option.
```

The following is a list of all the prompts which are used to change the values for the options and a brief explanation of each of them. (A more detailed explanation for each of these prompts follows the list.)

Dictionary name (current Dictionary)>	allows you to change the Dictionary from which you extract definitions.
Copylib file name (current copylib)>	allows you to change the copylib for the generated code.
Run COBEDIT (N/Y)>	allows you to run the copylib editor, COBEDIT, from within DICTCDE.
List definitions on the terminal (Y/N)>	allows you to suppress the generated code from being displayed on the terminal.
Comment the definitions (N/Y)>	allows you to generate comment lines for the extracted entities.
Use Primary name or Alias COBOL identifier (P/A)>	allows you to use the aliases for the extracted entities.
Qualify data sets and forms with their parent files (N/Y)>	allows you to use the name of the database or forms file as the prefix for the data set or form.
Use edit mask for PICTURE clause (Y/N)>	allows you to create display fields from items in the Dictionary that have edit masks.

Prompt for record data item prefixes (Y/N)>	allows you to suppress the prompts for prefixes for data items.
Prompt for all sections of MPE/KSAM definitions (Y/N)>	allows you to suppress the prompts for the ENVIRONMENT DIVISION and FILE SECTION.
Prompt for VPLUS field types and number tables (Y/N)>	allows you to suppress the prompts for selecting the VPLUS field types and the VPLUS field number tables.

DICTCDE will then issue the first option:

```
Dictionary name (current dictionary )>
```

Enter the Dictionary name, group and account to be opened and used for further code generation. DICTCDE displays the current Dictionary being used in (current dictionary). Data and file definitions are extracted from the new Dictionary specified until this option is changed again. When a new Dictionary name is specified, DICTCDE will also issue the following prompt:

```
Dictionary password>
```

Enter the Dictionary password for the new Dictionary to be used. If the new Dictionary specified is the same as the previous Dictionary, the password prompt will not be issued. If [[RETURN]] is pressed in response to this option, the Dictionary used will not change and DICTCDE will continue to the next option as follows:

```
Copylib file name (current copylib )>
```

Enter the copylib file name (35 characters maximum) to be opened and used for further code generation. DICTCDE displays the current copylib file being used in current copylib. If the new copylib file already exists, DICTCDE will open that copylib file and append any further source code generated to the contents of that file. If the specified copylib file name exists, but it is not in KSAM copylib format, DICTCDE will issue an error message and reprompt for the copylib file name. If the specified file name does not exist, DICTCDE will issue the following prompt:

```
File filename  does not exist, create it (N/Y)>
```

If N is entered in response to this prompt, DICTCDE will reprompt for the copylib file name. If Y is entered in response to this prompt, DICTCDE will prompt for the key file name for the new KSAM copylib to be created:

```
Key file name>
```

Enter the key file name for the new copylib file (8 characters maximum). If [[RETURN]] is pressed in response to this prompt, DICTCDE will reprompt for the copylib file name.

Once a new copylib is accepted, or if [[RETURN]] is pressed in response to the copylib file option prompt, DICTCDE would continue to the next option as follows:

```
Run COBEDIT (N/Y)>
```

Enter Y to run the copylib editor COBEDIT from within DICTCDE. DICTCDE will then close the current copylib file being used and invoke COBEDIT for you. (For information on running COBEDIT, see the COBOL II Reference Manual.) When you exit COBEDIT,

DICTCDE will reopen the copylib file and continue prompting for the next option.

If N is entered or [[RETURN]] is pressed in response to the COBEDIT option prompt, DICTCDE will also prompt for the next option as follows:

```
List definitions on the terminal (Y/N)>
```

Enter N to suppress the generated code from being displayed at the terminal (or printed in the job listing if DICTCDE is run from within a job stream). Enter Y to echo the source code as it is generated to the terminal (or printed in the job listing). If [[RETURN]] is pressed in response to this option prompt, DICTCDE will continue to the next option.

```
Comment the definitions (N/Y)>
```

Enter Y to generate comment lines which describe the entities extracted from the Dictionary. These comment lines are written to WORKING-STORAGE, at the end of any other code generated for an entity. These comment lines include the following information about the extracted entity:

entity	the name of the entity extracted from the Dictionary.
entity -name	the entity long-name (as it was specified in DICTDBM).
entity -resp	the name of the person, department, or area responsible for the integrity of the entity.
date-change	the date of the latest change made to the entity in the Dictionary.
date-create	the date the entity was created in the Dictionary.
identity-change	the identity of the person, department, or area who made the last change to the entity in the Dictionary.
identity-create	the identity of the person, department, or area who created the entity in the Dictionary.

For example, the following code is generated as comments for the MPE file ACCOUNT:

```
000100
000200* file                : ACCOUNT
000300* file-name           : CUSTOMER ACCOUNTS
000400* file-resp           : MANAGER
000500* date-change         : 83/11/01
000600* date-create         : 83/11/15
000700* identity-change     : MGR
000800* identity-create     : MGR
```

Enter N in response to this prompt to suppress the comment lines from being generated and to continue to the next prompt. If [[RETURN]] is pressed in response to this prompt, DICTCDE will also continue to the next option:

```
Use Primary name or Alias for COBOL identifier (P/A)>
```

Enter P to extract the Dictionary primary name to be used as the COBOL identifier for an entity when code is generated. Enter A to extract the Dictionary alias (whenever an alias is defined in the Dictionary) to be used as the COBOL identifier for an entity when code is

generated. Using aliases as COBOL identifiers identifies the child entities by the names that are known by the parent entities. Regardless of whether the primary names or aliases are used as the COBOL identifiers, the constants used for passing data set and search item names to IMAGE will be generated using the aliases (if the aliases have been defined in the Dictionary).

If [[RETURN]] is pressed in response to this prompt, DICTCDE will continue to the next option:

```
Qualify data sets and forms with their parent files (N/Y)>
```

Enter Y to use the parent file as a prefix in the COBOL identifier generated for the data sets or forms. When generating code for a database or forms file, DICTCDE allows you to qualify the related IMAGE data sets or VPLUS forms by the name of the parent file. DICTCDE will use the parent file name as a prefix in the COBOL identifier generated for the related data sets or forms. This will help you to differentiate between data sets and forms with common names. DICTCDE will also qualify single data sets or forms by their parent file name, if the parent file was specified, when the data set or form was extracted from the Dictionary. (The parent file was entered in parenthesis next to the file name in response to the File name prompt.)

Note that if the prefixes, file name and suffixes in the COBOL identifier total more than 30 characters, DICTCDE will truncate the COBOL identifier to the allowable 30 characters. DICTCDE will truncate the suffix first, then the prefix, so that the identifier does not exceed 30 characters.

Enter N if you do not wish to add the parent file name as a prefix in the COBOL identifiers generated for data sets and VPLUS forms. If [[RETURN]] is pressed in response to this prompt, DICTCDE will continue to the next option:

```
Use edit mask for PICTURE clause (Y/N)>
```

Enter Y to generate display fields for the elements defined with edit masks in the Dictionary. DICTCDE will convert the edit mask in the Dictionary to a COBOL PICTURE clause. When code is generated for elements with edit masks, the storage length is computed from the size of the edit mask, not from the storage length information in the Dictionary. However, the storage length is changed when the code is generated. For entities which contain child elements with edit masks, the record layout for the parent entity is also changed. Therefore, the byte offsets for the elements with edit masks must be defined in the Dictionary according to the edit mask size, not by the data storage requirements, if you wish to use this option. (For more information on the code generated for edit masks, see Edit Masks discussed under Examples of Generated Code later in this section.)

Enter N to ignore any edit masks defined for the elements. The PICTURE clause for an element will then be generated from the data type and storage length defined in the Dictionary.

If [[RETURN]] is pressed in response to this prompt, DICTCDE will continue to the next option:

```
Prompt for record data item prefixes (Y/N)>
```

Enter N to suppress the prompts asking for prefixes for the record data items. Enter Y to have the prompts issued for data item prefixes wherever DICTCDE normally issues them.

If [[RETURN]] is pressed in response to this prompt, DICTCDE will continue to the next option:

```
Prompt for all sections of MPE/KSAM definitions (Y/N)>
```

Enter N to suppress the prompts for the copylib modules for the SELECT statement and the FILE SECTION entry for an MPEF, MPER, or KSAM file. (Only the copylib for the WORKING-STORAGE record will be prompted for.) Enter Y to prompt for the copylib modules for the SELECT statement, FILE SECTION entry, and WORKING-STORAGE record for an MPEF, MPER, or KSAM file as DICTCDE normally issues them.

If [[RETURN]] is pressed in response to this prompt, DICTCDE will continue to the last option as follows:

```
Prompt for VPLUS field types and number tables (Y/N)>
```

Enter N to suppress the prompts for the VPLUS field types (PIC 9 or PIC X) and for the VPLUS field number tables for the forms files and forms extracted from the Dictionary. If these prompts are not issued, all the forms extracted from the Dictionary will be generated without field number tables, and the numeric fields in the forms will be generated as PIC 9 numeric fields.

When a response is made, or [[RETURN]] is pressed in response to this prompt, DICTCDE will reissue the main prompt.

Changing Input, List and Output Files

DICTCDE uses the formal file designators DICTIN for the input file, DICTOUT for the output file and DICTLOG for the response log file. The responses made to DICTCDE interactively are logged to the editor file DICTLOG. This file is session-temporary. An MPE SAVE command must be used to save this file as a permanent file (see example below). All responses are logged except for those made to COBEDIT or any MPE commands used while running DICTCDE.

DICTCDE can be run interactively or from within a job stream. DICTCDE can also be executed to accept previously entered responses from the terminal as follows:

1. RUN DICTCDE.PUB.SYS	Responses are made at the terminal and written to DICTLOG.
2. SAVE DICTLOG	Saves DICTLOG as a permanent file.
3. RENAME DICTLOG, LOGFILE	Renames DICTLOG to the file LOGFILE.
4. FILE DICTIN=LOGFILE	The input will be the responses previously entered.
5. RUN DICTCDE.PUB.SYS	DICTCDE now accepts the responses in LOGFILE.
6. RESET DICTIN	

Please note that when DICTLOG is used as the input file, DICTCDE expects that all the same prompts will be issued as when DICTCDE was previously run. For example, if you enter a new copylib name to be used the first time DICTCDE is run, you must purge the copylib name before re-running DICTCDE using DICTLOG as the list file. (If the copylib is not purged, the sequence of responses in DICTLOG will not follow the order of the prompts that will be issued by DICTCDE.) If the specified copylib already existed the first time DICTCDE was run, it must also exist when DICTCDE is run again using DICTLOG as the list file.

All terminal output, including the echoed source code, is written to DICTOUT (which is the terminal when DICTCDE is run interactively). The terminal output can be written to a disc file by redirecting DICTOUT to a disc file as shown below:

```
:FILE DICTOUT; SAVE
```

The terminal output will be echoed to the terminal and written to the disc file. Note that the MPE SAVE option must be part of the file equation or the file will not be saved as a permanent file. This file must not already exist (it will be created for you). If this file already exists, DICTCDE will open $STDLIST instead and issue a warning message.

Examples of Generated Code

The following provides a description and example of the COBOL source code generated for the entities extracted from the Dictionary.

IMAGE Database Definitions

DICTCDE allows you to generate the definition for an entire database or for selected data sets. The code generated for the database includes the database name, the record layouts for all its related data sets, and the constants containing the names of the search items. The data set records generated include the data set name and the associated data items. The data set name is generated as an 01-level identifier and the data items are generated as 05-level identifiers. Any child elements which are related to those data items are generated as 10-level identifiers, 15-level identifiers, and so forth.

The IMAGE special parameters can also be generated for an IMAGE database. These are the IMAGE special name-constant parameters that identify the database, password, and search items which are used by the IMAGE intrinsics. The code generated for the IMAGE special parameters is shown below:

Base parameter:

```
000300 01  DB-SHPMGT                PIC X(9) VALUE "  SHPMGT"
```

Password parameter:

```
000400 01  SHPMGT-PWD                PIC X(10) VALUE "         ".
```

The IMAGE standard parameters can also be generated for the database. (See the discussion of Parameters under Examples of Generated Code for more information.) Note that all definitions for an IMAGE database are generated in WORKING-STORAGE in the DATA DIVISION for a COBOL program.

Example

The following is an example of the code generated for an IMAGE Database:

```
000100
000200 01  CUST-M-DATA.
000300     05  DELETE-WORD         PIC X(2).
000400     05  CUST-NUM            PIC S9(8).
000500     05  CUST-NAME           PIC X(30).
000600     05  CUST-STREET         PIC X(30).
000700     05  CUST-CITY           PIC X(20).
000800     05  CUST-STATE          PIC X(2).
000900     05  CUST-ZIP            PIC S9(10).
001000     05  CUST-PHONE          PIC X(12).
001100     05  CUST-CREDIT-LIM     PIC S9(8).
001200     05  CUST-BALANCE        PIC S9(10).
```

```
001300
001400 01  DS-CUST-M         PIC X(7) VALUE "CUST-M ".
001500
001600 01  DATE-A-DATA.
001700     05  X-DATE         PIC X(8).
001800
001900 01  DS-DATE-A         PIC X(7) VALUE "DATE-A ".
002000
002100 01  INVOICE-M-DATA.
002200     05  DELETE-WORD        PIC X(2).
002300     05  INVOICE-NUM        PIC S9(8).
002400
002500 01  DS-INVOICE-M      PIC X(10) VALUE "INVOICE-M ".
002600
002700 01  ITEM-M-DATA
002800     05  DELETE-WORD        PIC X(2).
002900     05  ITEM-NUM           PIC S9(6).
003000     05  ITEM-DESC          PIC X(30).
003100     05  ITEM-UNIT-PRICE    PIC S9(8).
003200     05  ITEM-QOH           PIC S9(6).
003300     05  ITEM-ROP           PIC S9(6).
003400
003500 01  DS-ITEM-M         PIC X(7) VALUE "ITEM-M ".
003600
003700 01  INVOICE-D-DATA.
003800     05  DELETE-WORD        PIC X(2).
003900     05  INVOICE-NUM        PIC S9(8).
004000     05  CUST-NUM           PIC S9(8).
004100     05  INVOICE-TOTAL      PIC S9(10).
004200     05  INVOICE-DATE       PIC X(8).
004300
004400 01  DS-INVOICE-D      PIC X(10) VALUE "INVOICE-D ".
004500
004600 01  LINE-ITEMS-D-DATA.
004700     05  DELETE-WORD        PIC X(2).
004800     05  INVOICE-NUM        PIC S9(8).
004900     05  ITEM-NUM           PIC S9(6).
005000     05  ITEM-QTY           PIC S9(6).
```

```
005100     05  ITEM-SELL-PRICE    PIC S9(8).
005200
005300 01  DS-LINE-ITEMS-D    PIC X(13) VALUE "LINE-ITEMS-D  ".
005400
005500
005600 01  DB-ORDERS          PIC X(9) VALUE  "  ORDERS  ".
005700 01  ORDERS-PWD         PIC X(10) VALUE  ";        ".
005800
005900 01  DI-CUST-NUM        PIC X(9) VALUE  "CUST-NUM ".
006000 01  DI-X-DATE          PIC X(5) VALUE  "DATE  ".
006100 01  DI-INVOICE-NUM     PIC X(12) VALUE  "INVOICE-NUM ".
006200 01  DI-ITEM-NUM        PIC X(9) VALUE  "ITEM-NUM ".
006300 01  DI-INVOICE-DATE    PIC X(13) VALUE  "INVOICE-DATE  ".
```

IMAGE Data Set Definitions

The code generated for an IMAGE data set includes the record layout of the data set, a name which can be passed as the *dset* parameter to IMAGE, and the names for the search items which can be passed as the item parameter to DBFIND. If the alias option is enabled when the code is generated, then any element that has an alias within the data set will be generated with its alias. Regardless of whether the alias option is enabled or not, the respective VALUE clauses for the data set and search items will use the aliases if they exist for those entities.

Example

The following is an example of the code generated for a data set without prefixes assigned:

```
000100
000200 01  CUST-M-DATA.
000300     05  DELETE-WORD        PIC X(2).
000400     05  CUST-NUM           PIC S9(8).
000500     05  CUST-NAME          PIC X(30).
000600     05  CUST-STREET        PIC X(30).
000700     05  CUST-CITY          PIC X(20).
000800     05  CUST-STATE         PIC X(2).
000900     05  CUST-ZIP           PIC S9(10).
001000     05  CUST-PHONE         PIC X(12).
001100     05  CUST-CREDIT-LIM    PIC S9(8).
001200     05  CUST-BALANCE       PIC S9(10).
001300
001400 01  DS-CUST-M         PIC X(7) VALUE "CUST-M ".
001500 01  DI-CUST-NUM       PIC X(9) VALUE "CUST-NUM ".
```

The following is an example of the code generated for a data set with the prefix “cm-” assigned:

```
000100
000200 01  CUST-M-DATA
000300     05  CM-DELETE-WORD      PIC X(2).
000400     05  CM-CUST-NUM         PIC S9(8).
000500     05  CM-CUST-NAME        PIC X(30).
000600     05  CM-CUST-STREET      PIC X(30).
000700     05  CM-CUST-CITY        PIC X(20).
000800     05  CM-CUST-STATE       PIC X(2).
000900     05  CM-CUST-ZIP         PIC S9(10).
001000     05  CM-CUST-PHONE       PIC X(12).
001100     05  CM-CUST-CREDIT-LIM  PIC S9(8).
001200     05  CM-CUST-BALANCE     PIC S9(10).
001300
001400 01  DS-CUST-M          PIC X(7) VALUE "CUST-M ".
001500 01  DI-CM-CUST-NUM        PIC X(9) VALUE "CUST-NUM ".
```

VPLUS Forms File Definitions

DICTCDE allows you to generate the definition for an entire forms file or for selected forms. The code generated for the forms file includes the forms file name and the record layouts for all the related forms. The form records generated include the form name and their associated elements. The form name is generated as an 01-level identifier, and the fields are generated as 05-level identifiers. DICTCDE can also generate field number tables for each form in the forms file. (See the discussion of VPLUS Forms Definition for an example of the field number tables generated for a VPLUS form.)

The VPLUS standard parameters can also be generated for a forms file. (See the discussion of Parameters under Examples of Generated Code for more information.)

Example

The following is an example of the code generated for a VPLUS forms file:

```
000100
000200 01  ADDCUST-DATA.
000300     05  ACCOUNT                PIC X(10).
000400     05  L-NAME                 PIC X(20).
000500     05  F-NAME                 PIC X(18).
000600     05  INITIAL                PIC X(1).
000700     05  STREET-ADDR            PIC X(22).
000800     05  CITY                   PIC X(14).
000900     05  STATE                  PIC X(2).
001000     05  ZIP-CODE               PIC X(5).
001100     05  CREDIT                 PIC X(2).
001200 01  ADDCUST-FIELDS.
001300     05  ACCOUNT-FIELDNO        PIC S9(4) COMP VALUE 17.
001400     05  L-NAME-FIELDNO         PIC S9(4) COMP VALUE 18.
001500     05  F-NAME-FIELDNO         PIC S9(4) COMP VALUE 19.
001600     05  INITIAL-FIELDNO        PIC S9(4) COMP VALUE 20.
001700     05  STREET-ADDR-FIELDNO    PIC S9(4) COMP VALUE 21.
001800     05  CITY-FIELDNO           PIC S9(4) COMP VALUE 22.
001900     05  STATE-FIELDNO          PIC S9(4) COMP VALUE 5.
002000     05  ZIP-CODE-FIELDNO       PIC S9(4) COMP VALUE 23.
002100     05  CREDIT-FIELDNO         PIC S9(4) COMP VALUE 24.
002200
002300 01  ADDPROD-DATA.
002400     05  PROD-NO                PIC X(8).
002500     05  DESCRIPTION            PIC X(30).
002600 01  ADDPROD-FIELDS
```

```
002700     05  PROD-NO-FIELDNO        PIC S9(4) COMP VALUE 10.
002800     05  DESCRIPTION-FIELDNO    PIC S9(4) COMP VALUE 11.
002900
003000 01  ADDSALE-DATA.
003100     05  ACCOUNT                PIC X(10).
003200     05  PROD-NO                PIC X(8).
003300     05  QUANTITY               PIC 9(8).
003400     05  PRICE                  PIC X(12).
003500     05  TOTAL                  PIC X(12).
003600     05  PURCH-DATE             PIC X(6).
003700     05  DELIV-DATE             PIC X(6).
003800     05  PURCH-NO               PIC X(6).
003900 01 ADDSALE-FIELDS
004000     05  ACCOUNT-FIELDNO        PIC S9(4) COMP VALUE 11.
004100     05  PROD-NO-FIELDNO        PIC S9(4) COMP VALUE 12.
004200     05  QUANTITY-FIELDNO       PIC S9(4) COMP VALUE 13.
004300     05  PRICE-FIELDNO          PIC S9(4) COMP VALUE 14.
004400     05  TOTAL-FIELDNO          PIC S9(4) COMP VALUE 15.
004500     05  PURCH-DATE-FIELDNO     PIC S9(4) COMP VALUE 16.
004600     05  DELIV-DATE-FIELDNO     PIC S9(4) COMP VALUE 17.
004700     05  PURCH-NO-FIELDNO       PIC S9(4) COMP VALUE 18.
004800
004900 01  CHGCUST-DATA.
005000     05  ACCOUNT                PIC X(10).
005100     05  L-NAME                 PIC X(20).
005200     05  F-NAME                 PIC X(18).
005300     05  INITIAL                PIC X(1).
005400     05  STREET-ADDR            PIC X(22).
005500
005600
005700 01  ORDERSFF-CONST     PIC X(9) VALUE "ORDERSFF ".
```

VPLUS Form Definitions

The code generated for a VPLUS form includes the record layout of the form and the field number table for the fields in the form. These tables provide the field number for each field in a form. (The field number is useful in some VPLUS intrinsics such as VSETERROR.) Each field in the form is represented by a data item in the table. The data items are COMP variables initialized to the field number value defined in the Dictionary. The following is an example of the field tables generated for a VPLUS form:

```
001100  01  DELSALE-FIELDS
001200      05  ACCOUNT-FIELDNO        PIC S9(4) COMP VALUE 11.
001300      05  PROD-NO-FIELDNO        PIC S9(4) COMP VALUE 12.
001400      05  QUANTITY-FIELDNO       PIC S9(4) COMP VALUE 13.
001500      05  PRICE-FIELDNO          PIC S9(4) COMP VALUE 14.
001600      05  TOTAL-FIELDNO          PIC S9(4) COMP VALUE 15.
001700      05  PURCH-DATE-FIELDNO     PIC S9(4) COMP VALUE 16.
001800      05  DELIV-DATE-FIELDNO     PIC S9(4) COMP VALUE 17.
001900      05  PURCH-NO-FIELDNO       PIC S9(4) COMP VALUE 18.
002000
```

If the alias option is enabled when the code is generated for a form, then any element that has an alias within that form will be generated with its alias as the COBOL identifier for that element.

Example

The following is an example of the code generated for a form:

```
000100
000200  01  DELSALE-DATA.
000300      05  ACCOUNT                PIC X(10).
000400      05  PROD-NO                PIC X(8).
000500      05  QUANTITY               PIC 9(8).
000600      05  PRICE                  PIC X(12).
000700      05  TOTAL                  PIC X(12).
000800      05  PURCH-DATE             PIC X(6).
000900      05  DELIV-DATE             PIC X(6).
001000      05  PURCH-NO               PIC X(6).
001100  01  DELSALE-FIELDS
001200      05  ACCOUNT-FIELDNO        PIC S9(4) COMP VALUE 11.
001300      05  PROD-NO-FIELDNO        PIC S9(4) COMP VALUE 12.
001400      05  QUANTITY-FIELDNO       PIC S9(4) COMP VALUE 13.
001500      05  PRICE-FIELDNO          PIC S9(4) COMP VALUE 14.
001600      05  TOTAL-FIELDNO          PIC S9(4) COMP VALUE 15.
001700      05  PURCH-DATE-FIELDNO     PIC S9(4) COMP VALUE 16.
001800      05  DELIV-DATE-FIELDNO     PIC S9(4) COMP VALUE 17.
001900      05  PURCH-NO-FIELDNO       PIC S9(4) COMP VALUE 18.
002000
```

MPE File Definitions

DICTCDE allows you to generate source code for the ENVIRONMENT DIVISION and DATA DIVISION for an MPE file. For the ENVIRONMENT DIVISION portion, DICTCDE generates the SELECT statement for an MPE file as follows:

```
000100
000200     SELECT CUSTMPE
000300         ASSIGN "file-info "
000400         ORGANIZATION IS SEQUENTIAL.
```

DICTCDE generates the SELECT statement for an MPER (MPE Relative) file as follows:

```
000100
000200     SELECT CUSTMPER
000300         ASSIGN "file-info "
000400         ORGANIZATION IS RELATIVE
```

The file-info string contains the filename, CLASS, RECORDING MODE, DEVICE, CCTL, and the FILE SIZE fields as they are defined in the Dictionary. Note that all these fields, except for DEVICE, are generated only if they are defined in the Dictionary for that file. If CCTL was defined but DEVICE was not, then the DEVICE field would be generated as LP. Otherwise, DICTCDE will generate DEVICE as it was defined in the Dictionary. If a file is related to a LOCATION in the Dictionary, then the alias (if it is defined), and the group and account defined by the LOCATION will be used for the file name field.

For the DATA DIVISION portion, DICTCDE generates an entry for the FILE SECTION and the WORKING-STORAGE record. The FILE SECTION for an MPE sequential file can be generated as an FD file or as an SD file. The FILE SECTION for an MPER file can only be generated as an FD file.

The FILE SECTION is generated as follows:

```
000100
000200 FD  CUSTMPE
000300     BLOCK CONTAINS block-size  RECORDS
000400     RECORD CONTAINS recsize  CHARACTERS
000500     CODE-SET IS alphabet-name
000600     RECORDING MODE IS recording-mode.
```

The above parameters are generated from the following entries defined in the Dictionary for an MPE file:

block-size	BLOCKING(N/Y)?
	UNIT(R/C)?
	MINIMUM
	MAXIMUM
recsize	MINIMUM RECORD SIZE
	MAXIMUM RECORD SIZE
alphabet-name	DATA STORAGE TYPE(A/E)?
recording-mode	RECORD FORMAT(F/V/U/S)?

Since the recsize parameter is generated from the RECSIZE entry in the Dictionary, not from a total of the storage sizes of the elements in the file, care should be taken when defining the file's record size in the Dictionary. Otherwise, the record size defined for the file may conflict with the record layout of the file. (Refer to Section 4, The DICTDBM Commands, for more information on the above DICTDBM prompts.)

If any of the entries were not defined in the Dictionary for the above parameters, the corresponding clause for those parameters will not be generated.

The record layout of the file can be generated in the FD statement or in WORKING-STORAGE. The record layout identifies the data items associated with the file. Each data item in the file is generated as an 05-level identifier. The child elements of those data items are generated as 10-level identifiers, 15-level identifiers and so forth. If the alias option is enabled when the code is generated, then any data item that has an alias within the file will be generated with its alias as the COBOL identifier.

When the record layout of the file is generated in the FILE SECTION, it will immediately follow the FD Statement as shown below:

```
000100
000200 FD  CUSTMPE
000300     RECORD CONTAINS 1023 CHARACTERS
000400     RECORDING MODE IS S.
000500
000600 01  CUSTMPE-DATA.
000700     05  CUST-NUM               PIC  S9(8).
000800     05  CUST-CREDIT-LIM        PIC  S9(8).
000900     05  CUST-BALANCE           PIC  S9(10).
001000     05  INVOICE-NUM            PIC  S9(8).
001100     05  INVOICE-TOTAL          PIC  S9(10).
001200     05  INVOICE-DATE           PIC  X(8).
001300     05  FILLER                 PIC  X(971).
```

When the record layout is generated in WORKING-STORAGE and no FD Statement is generated, the code will be generated as shown below:

```
000100
000200 01  CUSTMPE-DATA.
000300     05  CUST-NUM               PIC  S9(8).
000400     05  CUST-CREDIT-LIM        PIC  S9(8).
000500     05  CUST-BALANCE           PIC  S9(10).
000600     05  INVOICE-NUM            PIC  S9(8).
000700     05  INVOICE-TOTAL          PIC  S9(10).
000800     05  INVOICE-DATE           PIC  X(8).
000900     05  FILLER                 PIC  X(971).
```

When the record layout is generated in WORKING-STORAGE and the FD Statement is also generated, the 01-level definition will appear in the FD Statement to serve as a buffer for READs and WRITEs to the file. The generated code is shown below:

```
000100
000200 FD  CUSTMPE
000300     RECORD CONTAINS 1023 CHARACTERS
000400     RECORDING MODE IS S.
000500 01  CUSTMPE-REC                PIC  X(1023).
000100
000200 01  CUSTMPE-DATA.
000300     05  CUST-NUM               PIC  S9(8).
000400     05  CUST-CREDIT-LIM        PIC  S9(8).
000500     05  CUST-BALANCE           PIC  S9(10).
000600     05  INVOICE-NUM            PIC  S9(8).
000700     05  INVOICE-TOTAL          PIC  S9(10).
000800     05  INVOICE-DATE           PIC  X(8).
000900     05  FILLER                 PIC  X(971).
```

Note that the size generated for CUSTMPE-REC is derived from the Dictionary definition for the record size of the file. However, if the record size of the file was not defined in the Dictionary, DICTCDE will compute its size from the record layout size. For a file with multiple record formats, only the size of the primary record format will be used in computing the file's record size. Files with multiple record layouts can also be generated by DICTCDE. If the record layout is to be generated in the FD statement and the file has more than one record layout, DICTCDE will use an implicit REDEFINES to define the multiple record layouts. (In other words, DICTCDE lays out the multiple record layouts, one after the other, which implicitly shows their relationship.) Each record layout will follow the other in the FD Statement as follows:

```
000100
000200 FD  CUSTMPE
```

```
000300 01  CUSTMPE-DATA.
000400     05  RECORD-LAYOUT-1   PIC X(80).
000500 01  CUSTMPE-DATA2.
000600     05  RECORD-LAYOUT-2-1 PIC X(40).
000700     05  RECORD-LAYOUT-2-2 PIC X(40).
```

If the record layout is to be generated in WORKING-STORAGE and the file has more than one record format, DICTCDE will use an explicit REDEFINES to define the multiple record layouts. (In other words, DICTCDE will use the REDEFINES clause to show the relationship between the different record layouts.) The generated code for the FD Statement and for WORKING-STORAGE is shown below:

```
000100
000200 FD  CUSTMPE
000300 01  CUSTMPE-REC               PIC X(80).
000100
000200 01  CUSTMPE-DATA.
000300     05  RECORD-LAYOUT-1        PIC X(80).
000400 01  CUSTMPE-DATA2 REDEFINES CUSTMPE-DATA.
000500     05  RECORD-LAYOUT-2-1      PIC X(40).
000600     05  RECORD-LAYOUT-2-2      PIC X(40).
```

Example

The following is an example of the code generated for an MPE file. (Note that the SELECT STATEMENT and FD statement have both been generated. The record layout has been generated in WORKING-STORAGE.)

```
000100
000200     SELECT CUSTMPE
000300         ASSIGN "CUSTMPE,,,DISC"
000400         ORGANIZATION IS SEQUENTIAL.
000500
000600 FD  CUSTMPE
000700     RECORD CONTAINS 1023 CHARACTERS
000800     RECORDING MODE IS S.
000900 01  CUSTMPE-REC               PIC X(1023)
001000
001100 01  CUSTMPE-DATA.
001200     05  CUST-NUM          PIC S9(8).
001300     05  CUST-CREDIT-LIM   PIC S9(8).
001400     05  CUST-BALANCE      PIC S9(10).
001500     05  INVOICE-NUM       PIC S9(8).
```

```
001600     05  INVOICE-TOTAL     PIC S9(10).
001700     05  INVOICE-DATE      PIC X(8).
001800     05  FILLER            PIC X(971).
```

KSAM File Definitions

DICTCDE allows you to generate source code for the ENVIRONMENT DIVISION and DATA DIVISION for a KSAM file.

For the ENVIRONMENT DIVISION portion, DICTCDE generates the SELECT statement for a KSAM file as follows:

```
000100
000200     SELECT KORDER
000300         ASSIGN "file-info "
000400         ORGANIZATION IS INDEXED
000500         RECORD KEY IS CUST-NUM
000600         ALTERNATE RECORD KEY INVOICE-NUM
000700         ALTERNATE RECORD KEY INVOICE-DATE WITH DUPLICATES.
```

The file-info string contains the filename and the FILE SIZE fields as they are defined in the Dictionary. Note that these fields are generated only if they are defined in the Dictionary for that file. If a file is related to a LOCATION in the Dictionary, then the alias (if any), group, and account defined by the LOCATION will be used for the file name field. DICTCDE also generates the key clauses according to the definitions of the items in the KSAM file. For the DATA DIVISION portion, DICTCDE generates an entry for the FILE SECTION and the WORKING-STORAGE record.

The FILE SECTION is generated as follows:

```
000100
000200 FD  KORDER
000300     BLOCK CONTAINS block-size  RECORDS
000400     RECORD CONTAINS recsize  CHARACTERS
000500     RECORDING MODE IS recording-mode.
```

The above parameters are generated from the following entries defined in the Dictionary for a KSAM file:

block-size	BLOCKING (N/Y)?
	UNIT (R/C)?
	MINIMUM
	MAXIMUM
recsize	MAXIMUM RECORD SIZE
recording-mode	RECORD FORMAT (F/V/U/S)?

Since the recsize parameter is generated from the entries in the Dictionary, not from a total of the storage sizes of the elements in the file, care should be taken when defining the file's record size in the Dictionary. Otherwise, the record size defined for the file may conflict with the record layout of the file. (Refer to Chapter 4, DICTDBM Commands, for more information on the above DICTDBM prompts.)

If any of the entries were not defined in the Dictionary for the above parameters, the corresponding clause for those parameters would not be generated.

The record layout of the file can be generated in the FD statement or in WORKING-STORAGE. The record layout identifies the data items associated with the file. Each data item in the file is generated as an 05-level identifier. The child elements of those data items are generated as 10-level identifiers, 15-level identifiers and so forth. If the alias option is enabled when the code is generated, then any data item that has an alias within the file will be generated with its alias as the COBOL identifier.

When the record layout of the file is generated in the FILE SECTION, it will immediately follow the FD Statement as shown below:

```
000100
000200 FD  KORDER
000300     RECORDING MODE IS F.
000400
000500 01  KORDER-DATA.
000600     05  CUST-NUM               PIC  S9(8).
000700     05  CUST-CREDIT-LIM        PIC  S9(8).
000800     05  CUST-BALANCE           PIC  S9(10).
000900     05  INVOICE-NUM            PIC  S9(8).
001000     05  INVOICE-TOTAL          PIC  S9(10).
001100     05  INVOICE-DATE           PIC  X(8).
```

When the record layout is generated in WORKING-STORAGE, and no copylib module has been specified for the FD Statement, the code will be generated as shown below:

```
000100
000200 01  KORDER-DATA.
000300     05  CUST-NUM               PIC  S9(8).
000400     05  CUST-CREDIT-LIM        PIC  S9(8).
000500     05  CUST-BALANCE           PIC  S9(10).
000600     05  INVOICE-NUM            PIC  S9(8).
000700     05  INVOICE-TOTAL          PIC  S9(10).
000800     05  INVOICE-DATE           PIC  X(8).
```

When the record layout is generated in WORKING-STORAGE, and the FD Statement is also generated, the 01-level definition will appear in the FD Statement to serve as a buffer for READs and WRITEs to the file. The generated code is shown below:

```
000100
000200 FD  KORDER
000300     RECORDING MODE IS F.
000400 01  KORDER-REC                PIC  X(52).
000500
000600 01  KORDER-DATA.
000700     05  CUST-NUM               PIC  S9(8).
000800     05  CUST-CREDIT-LIM        PIC  S9(8).
000900     05  CUST-BALANCE           PIC  S9(10).
001000     05  INVOICE-NUM            PIC  S9(8).
001100     05  INVOICE-TOTAL          PIC  S9(10).
001200     05  INVOICE-DATE           PIC  X(8).
```

This will only be of use when the file is being accessed by the KSAM COBOL intrinsics. If the file is defined in the SELECT Statement as ORGANIZATION INDEXED, and the keys have been defined, then COBOL requires that the record layout be generated in the FILE SECTION.

Note that the size generated for KORDER-REC is derived from the Dictionary definition for the record size of the file. However, if the record size of the file was not defined in the Dictionary, DICTCDE will compute its size from the record layout size. For a file with multiple record formats, only the size of the primary record format will be used in computing the file's record size.

KSAM files with multiple record layouts can also be generated by DICTCDE. If the record layout is to be generated in the FD statement, and the file has more than one record layout, DICTCDE will use an implicit REDEFINES to define the multiple record layouts. Each record layout will follow the other in the FD Statement as follows:

```
000100
000200 FD  KORDER
000300 01  KORDER-DATA.
000400     05  RECORD-LAYOUT-1   PIC X(80).
000500 01  KORDER-DATA2.
000600     05  RECORD-LAYOUT-2-1 PIC X(40).
000700     05  RECORD-LAYOUT-2-2 PIC X(40).
```

Note that KORDER-DATA2 is defined in the Dictionary as a parent element with 2 child elements.

If the record layout is to be generated in WORKING-STORAGE and the file has more than one record format, DICTCDE will use an explicit REDEFINES to define the multiple record layouts. The generated code for the FD Statement and for WORKING-STORAGE is

shown below.

```
000100
000200 01  KORDER-DATA.
000300     05  RECORD-LAYOUT-1        PIC X(80).
000400 01  KORDER-DATA2 REDEFINES KORDER-DATA.
000500     05  RECORD-LAYOUT-2-1      PIC X(40).
000600     05  RECORD-LAYOUT-2-2      PIC X(40).
```

You can also generate the KSAM FILETABLE parameter along with the WORKING-STORAGE record layout for the KSAM file. The FILETABLE parameter is the special parameter required by KSAM to be passed to each KSAM COBOL intrinsic. The following is an example of the FILETABLE parameter generated for the file KORDER:

```
000100
000200 01  KORDER-FILETAB.
000300     05  FILENUMBER             PIC S9(4) COMP VALUE 0.
000400     05  FILENAME               PIC X(8) VALUE "KORDER".
000500     05  I-O-TYPE               PIC S9(4) COMP VALUE 0.
000600     05  A-MODE                 PIC S9(4) COMP VALUE 0.
000700     05  PREV-OP                PIC S9(4) COMP VALUE 0.
```

DICTCDE will also generate the KSAM standard parameters. The KSAM standard parameters include a general FILETABLE parameter that does not reference a file name and a STAT parameter that contains STATUS-KEY1 and STATUS-KEY2 for use with the KSAM COBOL intrinsics. The standard FILETABLE can be used for any KSAM file as long as it is initialized correctly. The following is an example of the KSAM standard parameters:

```
000100
000200 01  FILETABLE.
000300     05  FILENUMBER             PIC S9(4) COMP VALUE 0.
000400     05  FILENAME               PIC X(8)       VALUE SPACES.
000500     05  I-O-TYPE               PIC S9(4) COMP VALUE 0.
000600     05  A-MODE                 PIC S9(4) COMP VALUE 0.
000700     05  PREV-OP                PIC S9(4) COMP VALUE 0.
000800 01  STAT.
000900     05  STATUS-KEY-1           PIC X.
001000     05  STATUS-KEY-2           PIC X.
```

Example

The following is an example of the code generated for a KSAM file. (Note that the SELECT statement and FD statement have both been generated.)

```
000100
000200     SELECT KORDER
000300         ASSIGN "KORDER"
000400         ORGANIZATION IS INDEXED
000500         RECORD KEY IS CUST-NUM
000600         ALTERNATE RECORD KEY INVOICE-NUM
000700         ALTERNATE RECORD KEY INVOICE-DATE WITH DUPLICATES.
000800
000900 FD  KORDER
001000     RECORDING MODE IS F.
001100 01  KORDER-DATA.
001200     05  CUST-NUM          PIC S9(8).
001300     05  CUST-CREDIT-LIM   PIC S9(8).
001400     05  CUST-BALANCE      PIC S9(10).
001500     05  INVOICE-NUM       PIC S9(8).
001600     05  INVOICE-TOTAL     PIC S9(10).
001700     05  INVOICE-DATE      PIC X(8).
```

Element Definitions

DICTCDE allows you to generate source code for any data element defined in the Dictionary. The code is generated in WORKING-STORAGE. Each element is generated as an 01-level data definition. Any child elements related to the element are generated as an 05-level identifier, 10-level identifier and so forth. If the alias option is enabled when the code is generated, then any child element will be generated with its alias as its COBOL identifier.

Example

The following is an example of the code generated for an element with no child elements:

```
000200 01  CUST-NUM          PIC S9(8).
```

The next example shows the code generated for an element that has child elements:

```
000100
000200 01  RECORD1.
000300     05  FIELD1      PIC X(2).
000400     05  FIELD2      PIC X(8).
000500     05  FIELD3      PIC X(6).
```

Compound Element Definitions

DICTCDE generates an array declaration for an element if its sub-element count is defined in the Dictionary as greater than one. Since arrays cannot appear at the 01-level in COBOL, DICTCDE will generate an array as an 05-level identifier, with a FILLER as an 01-level identifier.

For example, if an element is defined in the Dictionary as:

```
ELEMENT        = PRICE
ELEMENT-TYPE   = I
ELEMENT-SIZE   = 2
ELEMENT-LENGTH = 2
ELEMENT-COUNT  = 10
```

the following code will be generated:

```
000100
000200 01  FILLER.
000300     05  PRICE PIC S9(2) COMP OCCURS 10.
```

If the element is defined in the Dictionary as:

```
ELEMENT         = ACCOUNT
ELEMENT-TYPE    = P
ELEMENT-SIZE    = 10
ELEMENT-DECIMAL = 1
ELEMENT-LENGTH  = 6
ELEMENT-COUNT   = 4
```

the following code will be generated:

```
000100
000200 01  FILLER.
000300     05  ACCOUNT PIC S9(8)V9(1) COMP-3 OCCURS 4.
```

Back-Referenced Elements

DICTCDE generates a COBOL table for an element if its element type is defined in the Dictionary as “*”. (This element back references another element.) The element reference becomes the name of the sublevel of the table. For example, if the entity is defined in the Dictionary as:

```
ELEMENT          = ACCOUNT
ELEMENT-TYPE     = *
ELEMENT-REFERENCE = ACCOUNT-MASTER
ELEMENT-COUNT    = 10
```

and the back-referenced element ACCOUNT-MASTER is defined in the Dictionary as:

```
ELEMENT        = ACCOUNT-MASTER
```

```
ELEMENT-TYPE   = X
ELEMENT-SIZE   = 14
ELEMENT-LENGTH = 14
ELEMENT-COUNT  = 1
```

the following code is generated:

```
000100
000200 01  FILLER.
000300     05  ACCOUNT OCCURS 10.
000400         10  ACCOUNT-MASTER PIC X(14).
```

If both the element and the element reference have COUNTS greater than one, a multi-dimensional table is generated. For example, an element is defined in the Dictionary as:

```
ELEMENT           = ACCOUNT
ELEMENT-TYPE      = *
ELEMENT-REFERENCE = ACCOUNT-MASTER
ELEMENT-COUNT     = 10
```

The back-referenced element ACCOUNT-MASTER is defined in the Dictionary as:

```
ELEMENT        = ACCOUNT-MASTER
ELEMENT-TYPE   = X
ELEMENT-SIZE   = 14
ELEMENT-LENGTH = 14
ELEMENT-COUNT  = 3
```

The code for ACCOUNT would be generated as follows:

```
000100
000200 01  FILLER.
000300     05  ACCOUNT OCCURS 10.
000400         10  ACCOUNT-MASTER PIC X(14) OCCURS 3.
```

The next example shows how DICTCDE generates code for an element that back-references an element that has child elements related to it. If an element is defined in the Dictionary as:

```
ELEMENT           = ACCOUNT
ELEMENT-TYPE      = *
ELEMENT-REFERENCE = ACCOUNT-MASTER
ELEMENT-COUNT     = 10
```

and the back-referenced element ACCOUNT-MASTER has the elements NAME, ADDRESS and COST related to it, then these elements are defined in the Dictionary as:

```
PARENT-ELEMENT = ACCOUNT-MASTER
ELEMENT-TYPE   = X
```

```
ELEMENT-SIZE   = 36
ELEMENT-LENGTH = 36
ELEMENT-COUNT  = 1
CHILD-ELEMENTS        = NAME     ADDRESS     COST
ELEMENT-TYPE          =  X          X         X
ELEMENT-SIZE          =  9          2         5
ELEMENT-LENGTH        =  9          2         5
ELEMENT-COUNT         =  1          1         5
ELEMENT-BYTE POSITION =  1         10        12
```

The code for the element ACCOUNT is generated as follows:

```
000100
000200 01  FILLER.
000300     05  ACCOUNT OCCURS 10.
000400         10  ACCOUNT-MASTER.
000500             15  NAME    PIC X(9).
000600             15  ADDRESS PIC X(2).
000700             15  COST    PIC X(5) OCCURS 5.
```

In the above case, ACCOUNT-MASTER is generated only to include NAME, ADDRESS, and COST as elements of table ACCOUNT.

Element to Element Relationships

DICTCDE handles element-to-element relationships (child elements to parent elements) differently depending on how the storage lengths and byte positions are defined in the Dictionary for the child elements. In the first example, the storage length of the parent element (as defined in the Dictionary) is equal to the sum of the storage lengths for the child elements. The byte positions for the child elements have been defined in the Dictionary to allow the child elements to consecutively follow each other within the parent element. For this example, the element SALES has the elements PRODUCT, PRICE and AMOUNT as child elements. The element SALES is defined in the Dictionary as:

```
PARENT-ELEMENT = SALES
ELEMENT-TYPE   = X
ELEMENT-SIZE   = 50
ELEMENT-LENGTH = 50
```

The child elements are defined in the Dictionary as:

```
CHILD-ELEMENTS        = PRODUCT     PRICE     AMOUNT
ELEMENT-TYPE          =    X          X          X
ELEMENT-SIZE          =   12         28         10
ELEMENT-LENGTH        =   12         28         10
ELEMENT-BYTE POSITION =    1         13         41
```

The code for element SALES is generated as follows:

```
000100
000200 01  SALES.
000300     05  PRODUCT PIC X(12).
000400     05  PRICE   PIC X(28).
000500     05  AMOUNT  PIC X(10).
```

Note that the COBOL record would be represented in storage as a memory array of a length of 50 bytes, with PRODUCT being from byte 1 to 12, PRICE from byte 13 to 40 and AMOUNT from byte 41 to 50.

In the next example of element-to-element relationships, the storage length for the parent element is defined in the Dictionary. The child elements are defined at specified offsets defined in the Dictionary, within the parent element's storage. If there is a gap between the memory areas of two consecutive child elements, DICTCDE generates FILLER items in order to place the child elements at the proper offset within the parent element and to assign the correct size to the parent element. Therefore, when the parent element is accessed, the correct amount of memory will be available for it. Also, when the child element is accessed, it will be at the correct offset within the parent. For example, the element ADDRESS has the elements STREET and YEARS related to it. The element ADDRESS is defined in the Dictionary as:

```
PARENT-ELEMENT = ADDRESS
ELEMENT-TYPE   = X
ELEMENT-SIZE   = 48
ELEMENT-LENGTH = 48
```

The child elements are defined in the Dictionary as:

```
CHILD-ELEMENTS         = STREET     YEARS
ELEMENT-TYPE           =   X          I
ELEMENT-SIZE           =   12         5
ELEMENT-LENGTH         =   12         4
ELEMENT-BYTE POSITION =    4        30
```

Note that ADDRESS requires a storage length of 48 bytes. Therefore, when DICTCDE generates the code for ADDRESS, a FILLER is used at the end to fill up the 48 bytes.

```
000100
000200 01  ADDRESS.
000300     05  FILLER PIC X(3).
000400     05  STREET PIC X(12).
000500     05  FILLER PIC X(14).
000600     05  YEARS  PIC S9(5) COMP.
000700     05  FILLER PIC X(15).
```

Forced REDEFINES

When the storage lengths and byte offsets for two child elements overlap each other within the parent element, the two child elements cannot be placed adjacently within the parent element record. DICTCDE generates these elements as a forced REDEFINES to insure they overlap exactly as defined in the Dictionary. For example, the element NAME has the child elements F-NAME and M-NAME. The parent element NAME is defined in the Dictionary as:

```
ELEMENT        = NAME
ELEMENT-TYPE   = X
ELEMENT-SIZE   = 44
ELEMENT-LENGTH = 44
```

The child elements are defined in the Dictionary as:

```
CHILD-ELEMENTS        = F-NAME     M-NAME
ELEMENT-TYPE          =    X          X
ELEMENT-SIZE          =   28         12
ELEMENT-LENGTH        =   28         12
ELEMENT-BYTE POSITION =    1         12
```

The code for the element NAME would be generated as follows:

```
000100
000200 01  NAME.
000300     05  F-NAME PIC X(28).
000400     05  FILLER PIC X(16).
000500 01  FILLER REDEFINES NAME.
000600     05  FILLER PIC X(11).
000700     05  M-NAME PIC X(12).
000800     05  FILLER PIC X(21).
```

Explicit REDEFINES

Overlapping child elements can also be generated by DICTCDE as explicit REDEFINES as long as the record layout for the parent element was defined in the Dictionary using the dummy element $REDEFINES. All child elements added to the parent element's list of relationships before $REDEFINES make up the first record layout. When $REDEFINES is related to the parent element, it implies that those child elements added to the parent's list of relationships after the $REDEFINES dummy, constitute an alternate record layout.

Note that when $REDEFINES is related to the parent element in DICTDBM, an alias must be defined for $REDEFINES so that DICTCDE can use the alias as the COBOL identifier for $REDEFINES. $REDEFINES can be related to the file only once. DICTCDE keeps track of the storage space of the first record layout and correctly lays out the subsequent elements after the $REDEFINES into alternate records, thus each alternate record format does not exceed the storage space of the first record. For example, the element ELEM10 has the child elements ELEM2, ELEM3, ELEM7, $REDEFINES and ELEM8 (in that order). The parent element ELEM10 is defined in the Dictionary as:

```
ELEMENT              = ELEM10
ELEMENT-TYPE         = X
ELEMENT-SIZE         = 28
ELEMENT-LENGTH       = 28
```

Suppose the child elements are defined in the Dictionary as:

```
CHILD-ELEMENTS         = ELEM2     ELEM3    ELEM7    $REDEFINES     ELEM8
ELEMENT-TYPE           =  X          I        X                       X
ELEMENT-SIZE           = 12          5       28                      10
ELEMENT-LENGTH         = 12          4       28                      10
ELEMENT-BYTE POSITION  =  1         13        1                       1
ELEMENT-ALIAS          =                              REDEFINE-REC
```

Note that if any of the above elements were not related to the parent element in the correct order, you could change the order with the REORDER command provided by DICTDBM.

The code for the element ELEM10 would be generated as follows:

```
000100
000200 01  ELEM10.
000300     05  ELEM2      PIC X(12).
000400     05  ELEM3      PIC S9(5) COMP.
000500     05  FILLER     PIC X(12).
000600 01  FILLER        REDEFINES ELEM10.
000700     05  ELEM7      PIC X(28).
000800 01  REDEFINE-REC  REDEFINES ELEM10.
000900     05  ELEM8      PIC X(10).
001000     05  FILLER     PIC X(18).
```

Edit Masks

Display fields can be generated for elements defined with edit masks in the Dictionary. If the edit mask is enabled (see the Options response to the main prompt), DICTCDE will generate a PICTURE clause for the edit mask. Table 7-1 shows how the edit masks are generated as COBOL PICTURE characters.

Table 7-1. Edit Mask to COBOL PICTURE Clause Characters

Dictionary Edit Mask	COBOL PICTURE Clause Characters	Comments
^	X or 9	X if data type is X or U, otherwise 9
, / $ * Z	, / $ * Z	These are valid COBOL characters
.		Note that only one period may appear
(blank)	B	
CR	CR	
DR	DR	COBOL only recognizes DB.
!	ERROR	COBOL does not have an equivalent
Any other	ERROR	No other characters are allowed

When code is generated for elements with edit masks, the display size for the elements is generated according to the display size defined in the Dictionary. However, the storage length is changed when code is generated for edit masks. For entities which contain child elements with edit masks, the record layout for the parent entity is also changed. Therefore, the byte offsets for the elements with edit masks must be defined in the Dictionary according to the edit mask size, not by the data storage requirements. (This may require changes to the existing definitions in the Dictionary.)

When defining a parent element or file with edit masks in the Dictionary, you should manually define the byte offsets for that entity in the Dictionary according to the COBOL storage lengths for their edit masks. Otherwise, the child elements will probably overlap each other. To compute the required storage length, add one byte for each COBOL PICTURE clause character. The storage length of the parent element should also be defined as the total storage lengths of the child elements with their edit masks, to ensure that the child elements all fit within the record layout.

In the next table, the parent element PART-MASTER has the child elements PART1, PART2, PART3, PART4, PART5, and PART6 related to it. Table 7-2 shows the edit masks and the correct byte offsets as defined in the Dictionary for the child elements and the COBOL PICTURE clause generated for those elements.

Table 7-2. Generating COBOL PICTURE Clauses

ELEMENT	EDIT MASK	BYTE/ POSITION	COBOL PICTURE Clause
Part1	^^^^^^/^^^^^^/^^	1	05 PART1 PIC XXXXXX/XXXXXX/XX.
Part2	^^^ ^^^ ^^^^	17	05 PART2 PIC XXXBXXXBXXXX.
Part3	$$$,$$$,$$$.^^	29	05 PART3 PIC $$$,$$$,$$$.99.
Part4	ZZZ,ZZZ.^^	43	05 PART4 PIC ZZZ,ZZZ.99.
Part5	^^,^^^.^^CR	53	05 PART5 PIC 99,999.99CR.
Part6	^^,^^^.^^DR	64	05 PART6 PIC 99,999.99DB

Code Generated for Standard Parameters

DICTCDE also generates source code for the IMAGE, VPLUS and KSAM standard parameters. (To generate these standard parameters, see the Parameter response to the main prompt issued by DICTCDE.)

The IMAGE standard parameters include the DB STATUS array declaration, the eight MODE declarations, and the utility LIST declarations. The IMAGE parameters are generated in the WORKING-STORAGE record as follows:

```
000100
000200 01  STATUS.
000300     05  C-WORD              PIC S9(4) COMP.
000400     05  STAT2               PIC S9(4) COMP.
000500     05  STAT3-4             PIC S9(9) COMP.
000600     05  STAT5-6             PIC S9(9) COMP.
000700     05  STAT7-8             PIC S9(9) COMP.
000800     05  STAT9-10            PIC S9(9) COMP.
000900 01  DUMMY-LIST              PIC X(2) VALUE "; ".
001000 01  ALL-ITEMS               PIC X(2) VALUE "@;".
001100 01  PREVIOUS-LIST           PIC X(2) VALUE "*;".
001200 01  MODE1                   PIC 9999 COMP VALUE 1.
001300 01  MODE2                   PIC 9999 COMP VALUE 2.
001400 01  MODE3                   PIC 9999 COMP VALUE 3.
001500 01  MODE4                   PIC 9999 COMP VALUE 4.
001600 01  MODE5                   PIC 9999 COMP VALUE 5.
```

```
001700 01  MODE6                  PIC 9999 COMP VALUE 6.
001800 01  MODE7                  PIC 9999 COMP VALUE 7.
001900 01  MODE8                  PIC 9999 COMP VALUE 8.
```

The VPLUS standard parameters include the VPLUS COMAREA definition and a character string which identifies the terminal. (The terminal ID is used for passing to VOPENTERM.) The VPLUS parameters are generated to the WORKING-STORAGE record as follows:

```
001800
001900 01  COMAREA.
002000     05  COM-STATUS         PIC S9(4)  COMP VALUE 0.
002100     05  COM-LANGUAGE       PIC S9(4)  COMP VALUE 0.
002200     05  COM-COMAREALEN     PIC S9(4)  COMP VALUE 60.
002300     05  FILLER             PIC S9(4)  COMP VALUE 0.
002400     05  COM-MODE           PIC S9(4)  COMP VALUE 0.
002500     05  COM-LASTKEY        PIC S9(4)  COMP VALUE 0.
002600     05  COM-NUMERRS        PIC S9(4)  COMP VALUE 0.
002700     05  FILLER             PIC S9(4)  COMP VALUE 0.
002800     05  FILLER             PIC S9(4)  COMP VALUE 0.
002900     05  FILLER             PIC S9(4)  COMP VALUE 0.
003000     05  COM-CFNAME         PIC X(15)       VALUE SPACES.
003100     05  FILLER             PIC X(1)        VALUE SPACES.
003200     05  COM-NFNAME         PIC X(15)       VALUE SPACES.
003300     05  FILLER             PIC X(1)        VALUE SPACES.
003400     05  COM-REPEATOPT      PIC S9(4)  COMP VALUE 0.
003500     05  COM-NFOPT          PIC S9(4)  COMP VALUE 0.
003600     05  FILLER             PIC S9(4)  COMP VALUE 0.
003700     05  COM-DBUFLEN        PIC S9(4)  COMP VALUE 0.
003800     05  FILLER             PIC S9(4)  COMP VALUE 0.
003900     05  FILLER             PIC S9(4)  COMP VALUE 0.
004000     05  COM-DELETEFLAG     PIC S9(4)  COMP VALUE 0.
004100     05  COM-SHOWCONTROL    PIC S9(4)  COMP VALUE 0.
004200     05  FILLER             PIC S9(4)  COMP VALUE 0.
004300     05  FILLER             PIC S9(4)  COMP VALUE 0.
004400     05  FILLER             PIC S9(4)  COMP VALUE 0.
004500     05  FILLER             PIC S9(4)  COMP VALUE 0.
004600     05  FILLER             PIC S9(4)  COMP VALUE 0.
004700     05  FILLER             PIC S9(4)  COMP VALUE 0.
004800     05  FILLER             PIC S9(4)  COMP VALUE 0.
```

```
004900     05  FILLER              PIC S9(4)  COMP VALUE 0.
005000     05  COM-NUMRECS         PIC S9(9)  COMP VALUE 0.
005100     05  COM-RECNUM          PIC S9(9)  COMP VALUE 0.
005200     05  FILLER              PIC S9(4)  COMP VALUE 0
005300     05  FILLER              PIC S9(4)  COMP VALUE 0.
005400     05  COM-TERMFILENUM     PIC S9(4)  COMP VALUE 0.
005500     05  FILLER              PIC S9(4)  COMP VALUE 0.
005600     05  FILLER              PIC S9(4)  COMP VALUE 0.
005700     05  FILLER              PIC S9(4)  COMP VALUE 0.
005800     05  FILLER              PIC S9(4)  COMP VALUE 0.
005900     05  FILLER              PIC S9(4)  COMP VALUE 0.
006000     05  FILLER              PIC S9(4)  COMP VALUE 0.
006100     05  COM-TERMOPTIONS     PIC S9(4)  COMP VALUE 0
006200     05  FILLER              PIC S9(4)  COMP VALUE 0.
006300     05  FILLER              PIC S9(4)  COMP VALUE 0.
006400     05  FILLER              PIC S9(4)  COMP VALUE 0.
006500     05  FILLER              PIC S9(4)  COMP VALUE 0.
006600 01  TERMFILENAME            PIC X(6)        VALUE "TERM  ".
006700 01  MESSAGE-BUF             PIC X(72)       VALUE SPACES.
006800 01  MESSAGE-BUF-LEN         PIC S9(4)  COMP VALUE 72.
006900 01  MSGLEN                  PIC S9(4)  COMP VALUE 0.
007000 01  FIELDNUM                PIC S9(4)  COMP VALUE 0.
007100 01  BUFLEN                  PIC S9(4)  COMP VALUE 0.
```

The KSAM standard parameters include a general KSAM FILETABLE which does not reference any particular file and the STAT parameters. The general KSAM FILETABLE can be used for any file as long as it is initialized correctly. The STAT parameters contain STATUS-KEY1 and STATUS-KEY2 which can be used by the KSAM COBOL intrinsics. The KSAM parameters are generated to WORKING-STORAGE as follows:

```
007300 01  FILETABLE.
007400     05  FILENUMBER          PIC S9(4)  COMP VALUE 0.
007500     05  FILENAME            PIC X(8)        VALUE SPACES.
007600     05  I-O-TYPE            PIC S9(4)  COMP VALUE 0.
007700     05  A-MODE              PIC S9(4)  COMP VALUE 0.
007800     05  PREV-OP             PIC S9(4)  COMP VALUE 0.
007900 01  STAT.
008000     05  STATUS-KEY-1        PIC X.
008100     05  STATUS-KEY-2        PIC X.
```

DICTCDE Naming Considerations

When an entity is extracted from the Dictionary, DICTCDE checks that the entity name is compatible with COBOL. If the entity name defined in the Dictionary is an illegal COBOL name or the entity name is a COBOL reserved word, an error message will be issued, and DICTCDE will convert the names to legal COBOL names. Note that the names generated by DICTCDE will not be reflected in the Dictionary.

The following describes how DICTCDE converts entity names that are not compatible with COBOL.

If the entity name, as defined in the Dictionary, begins with a hyphen, DICTCDE will add an X- to the beginning of the entity name.

Suppose the entity name in the Dictionary is:

```
-STOCK
```

the following COBOL name will be generated:

```
X--STOCK
```

If the entity name, as defined in the Dictionary, ends with a hyphen, DICTCDE will add an -X to the end of the entity name. Suppose the entity name in the Dictionary is: CUSTSTOCK- the following COBOL name will be generated:

```
CUSTSTOCK--X
```

If the entity name extracted from the Dictionary is a COBOL reserved word, DICTCDE will add an X- to the beginning of the word. Also, DICTCDE prints a comment next to the generated entity to inform you that this entity name is a COBOL reserved word. For example, if the entity is defined in the Dictionary as FILE, the following code will be generated:

```
X-FILE
```

Also, if the entity name contains any invalid COBOL characters, those characters will be replaced with a hyphen.

COBOL Data Type Mappings

To be compatible with COBOL, the data types defined in the Dictionary must be mapped to COBOL data types. The following shows how the Dictionary data types are mapped to compatible COBOL data types. Note that the COBOL data types occupy the same storage length as the data types defined in the Dictionary. However, some Dictionary definitions allow numbers larger than what COBOL will allow (for example, the integer data type). Also, for some data types, the Dictionary allows 28 and 29 numeric digit elements, whereas COBOL only allows numeric elements to have a maximum of 18 digits. Elements defined in the Dictionary that are larger than the maximum digits allowed by COBOL will be flagged as an error and converted to data type X when generated by DICTCDE.

Any ASCII Character

Suppose the entity is defined in the Dictionary as:

```
ELEMENT-TYPE   = X
ELEMENT-SIZE   = 20
ELEMENT-LENGTH = 20
```

the corresponding COBOL clause will be generated:

```
PIC X(20)
```

Uppercase Alphanumeric String

Suppose the entity is defined in the Dictionary as:

```
ELEMENT-TYPE   = U
ELEMENT-SIZE   = 20
ELEMENT-LENGTH = 20
```

the corresponding COBOL clause will be generated:

```
PIC X(20)
```

Numeric ASCII String

Suppose the entity is defined in the Dictionary as:

```
ELEMENT-TYPE    = 9 or 9+
ELEMENT-SIZE    = 8
ELEMENT-DECIMAL = 2
ELEMENT-LENGTH  = 7
```

the corresponding COBOL clause will be generated:

```
PIC 9(5)V9(2)
```

Zoned Decimal

Suppose the entity is defined in the Dictionary as:

```
ELEMENT-TYPE    = Z+
ELEMENT-SIZE    = 8
ELEMENT-DECIMAL = 2
ELEMENT-LENGTH  = 7
```

the corresponding COBOL clause will be generated:

```
PIC 9(5)V9(2)
```

Suppose the entity is defined in the Dictionary as:

```
ELEMENT-TYPE    = Z
ELEMENT-SIZE    = 8
ELEMENT-DECIMAL = 2
ELEMENT-LENGTH  = 7
```

the corresponding COBOL clause will be generated:

```
PIC S9(5)V9(2)
```

If the above entity is defined in the Dictionary as SIGN TRAILING SEPARATE, DICTCDE will generate the corresponding COBOL clause:

```
PIC S9(4)V9(2) SIGN TRAILING SEPARATE
```

Note that 1 less byte is generated because of the sign.

Boolean

Suppose the entity is defined in the Dictionary as:

```
ELEMENT-TYPE   = B
ELEMENT-SIZE   = 1
ELEMENT-LENGTH = 1
```

the corresponding COBOL clause will be generated:

```
PIC X(1)
```

Since this is an undefined COBOL type, DICTCDE will flag this element definition with a warning message.

String

Suppose the entity is defined in the Dictionary as:

```
ELEMENT-TYPE   = S
ELEMENT-SIZE   = 5
ELEMENT-LENGTH = 8
```

the corresponding COBOL clause will be generated:

```
PIC X(8)
```

Since this is an undefined COBOL type, DICTCDE will flag this element definition with a warning message.

Real Number

Suppose the entity is defined in the Dictionary as:

```
ELEMENT-TYPE    = E (or R)
ELEMENT-SIZE    = 7
ELEMENT-DECIMAL = 2
ELEMENT-LENGTH  = 4
```

the corresponding COBOL clause will be generated:

```
PIC X(4)
```

Since this is an undefined COBOL type, DICTCDE will flag this element definition with a warning message.

Integer Number and Logical Value

The data types, defined in the Dictionary as I, J, or K are mapped to COBOL COMPUTATIONAL data types. These data types are defined to take up the required amount of storage (either 2, 4 or 8 bytes). If the number of digits specified in the Dictionary for an entity are sufficient to fill the storage length (as defined in the Dictionary) then DICTCDE will use the number of digits as specified in the Dictionary. However, if the number of digits are insufficient, COBOL will use the minimum number of digits to fill the required storage length. Also, if the number of digits is too large for the specified storage size, DICTCDE will use the maximum number of digits that storage size allows.

For example, the following entities in the Dictionary are generated as follows:

```
ELEMENT-TYPE    = I
ELEMENT-SIZE    = 4
ELEMENT-DECIMAL = 0            {   PIC S9(4) COMP.
ELEMENT-LENGTH  = 2
ELEMENT-TYPE    = I
ELEMENT-SIZE    = 5
ELEMENT-DECIMAL = 2            {  PIC S9(2)V9(2) COMP.
ELEMENT-LENGTH  = 2
```

The decimal point for the above entity requires a digit place.

```
ELEMENT-TYPE    = I
ELEMENT-SIZE    = 4
ELEMENT-DECIMAL = 0            {  PIC S9(5) COMP.
ELEMENT-LENGTH  = 4
```

(Five digits used to force a 2-word integer.)

```
ELEMENT-TYPE    = I
```

```
ELEMENT-SIZE    = 5
ELEMENT-DECIMAL = 0            {  PIC S9(10) COMP.
ELEMENT-LENGTH  = 8
```

(Ten digits used to force a 4-word integer.)

Packed Decimal

The data types defined in the Dictionary as P, are mapped to COBOL COMPUTATIONAL-3 data types. However, this data type is defined in COBOL to occupy the same storage length as specified in the Dictionary, regardless of digit length. If the digit length specified in the Dictionary is sufficient to fill the storage length specified in the Dictionary, then DICTCDE will use that digit length. If the number of digits is insufficient to fill the specified storage length, then DICTCDE will use the minimum number of digits to fill the storage length for the entity. If the number of digits is too large, DICTCDE will use the maximum number of digits allowed for the storage length.

For example, the following entities in the Dictionary are generated as follows:

```
ELEMENT-TYPE    = P
ELEMENT-SIZE    = 8
ELEMENT-DECIMAL = 0            {  PIC S9(8) COMP-3.
ELEMENT-LENGTH  = 5
ELEMENT-TYPE    = P
ELEMENT-SIZE    = 9
ELEMENT-DECIMAL = 0            {  PIC S9(9) COMP-3.
ELEMENT-LENGTH  = 5
ELEMENT-TYPE    = P
ELEMENT-SIZE    = 8
ELEMENT-DECIMAL = 0            {  PIC S9(10) COMP-3.
ELEMENT-LENGTH  = 6
```

(Ten digits are used to force a 6 byte storage length.)

```
ELEMENT-TYPE    = P
ELEMENT-SIZE    = 8
ELEMENT-DECIMAL = 2            {  PIC S9(5)V9(2) COMP-3.
ELEMENT-LENGTH  = 4
```

The decimal point for the above entity requires a digit place.

VPLUS Data Items

VPLUS forms only contain ASCII characters as storage format. Therefore, regardless of the data types defined in the Dictionary, all VPLUS fields are DISPLAY fields. You may choose to convert all VPLUS fields to PIC X (regardless of the data type defined in the Dictionary). You may also choose to define numeric fields as PIC 9. But you must ensure that non-numeric data (except for a sign) will not appear in those fields. The sign for signed numeric elements must appear in the first character position unless the element was defined in the Dictionary as having a TRAILING-SEPARATE sign. The trailing separate sign must appear in the last character position.

The following shows the COBOL clauses generated for some VPLUS fields.

```
ELEMENT-TYPE    = X
ELEMENT-SIZE    = 8            {  PIC X(8).
ELEMENT-LENGTH  = 9
```

Note that the VPLUS field is generated by SIZE not by LENGTH. Also, the next examples assume that the PIC 9 numeric option was enabled.

```
ELEMENT-TYPE    = I+
ELEMENT-SIZE    = 8
ELEMENT-DECIMAL = 0            {  PIC 9(8).
ELEMENT-LENGTH  = 4
ELEMENT-TYPE    = I
ELEMENT-SIZE    = 8
ELEMENT-DECIMAL = 0            {  PIC S9(8) SIGN LEADING SEPARATE.
ELEMENT-LENGTH  = 4
```

Note that the sign field generated the SIGN LEADING SEPARATE clause.

```
ELEMENT-TYPE    = I
ELEMENT-SIZE    = 8
ELEMENT-DECIMAL = 2          {  PIC S9(5)V9(2) SIGN LEADING SEPARATE.
ELEMENT-LENGTH  = 4
```

A DICTDBM Error Messages

DICTDBM generates two types of messages on the user terminal - Error Messages and Information Messages. Information Messages are conditions that DICTDBM informs the user about, but are not errors. Both kinds of messages are found in this appendix or in the appropriate reference manual for the indicated subsystem.

Error Messages have the following format:

```
*ERROR:  error-message  (error-type error-number )
```

Information Messages have the following format:

```
*INFO:  error-message  (error-type error-number )
```

Error-type is one the following:

USER	The error is caused by a user of DICTDBM and may usually be corrected by entering a different response.
PROG	The error is due to an error in DICTDBM'S environment and may usually be corrected by the Systems Engineer.
SYSTEM	The error is due to constraints of the system DICTDBM is running on and may be corrected by the operator.
TRAP	The error is due to an internal error in DICTDBM and its occurrence should be called to the attention of the Systems Engineer.
DICT	The error is caused by the user and may usually be corrected by entering a different response. These are errors that help maintain the integrity of the dictionary.
IMAGE	The error is an IMAGE database error and is not documented in this appendix. Consult the *IMAGE/3000 Reference Manual*.
MPEF	The error is an MPE file system error and is not documented in this appendix. Consult the *MPE 3000 System Error Messages and Recovery Reference Manual*.

Error-number is the number listed in this manual for types USER, PROG, SYSTEM, TRAP or DICT. It is also the number of the error meaningful to either the IMAGE database system or the MPE file system.

USER Messages

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
USER 1	ENTRY NOT NUMERIC	Data item type is integer, floating point, or numeric ASCII and a non-numeric character has been detected in the data entry field.
USER 2	INPUT FIELD LONGER THAN *n*	Length of data entry exceeds the size (“n”) defined for the associated data item.
USER 3	ORIGINAL RECORD HAS BEEN RESTORED	An error has occurred on updating an entry in a data set and the original entry has been restored.
USER 4	NUMERIC INTEGER PART LONGER THAN *n*	Integer part of a decimal number exceeds the length (“n”) defined for the associated data item.
USER 5	NUMERIC DECIMAL PART LONGER THAN *n*	Decimal part of a decimal number exceeds the length (“n”) defined for the associated data item.
USER 6	MISSING COMMAND	A command option was entered without a command.
USER 7	INVALID COMMAND/OPTION: command/option	The command or command option entered is not valid.
USER 8	INVALID/MISSING SUB-COMMAND: *sub-command*	The sub-command entered is not valid for the command specified.
USER 18	ENTRY CANNOT BE NEGATIVE	The item has been declared to always be positive.
USER 19	CONNECTOR MUST BE ONE OF 'AND', 'OR' OR 'TO'	Connector must be 'and, 'or' or 'to'.
USER 20	OPERATOR MUST BE ONE OF 'EQ', 'NE', 'LE', 'GE', 'LT' OR 'GT'	Operator must be one of 'eq', 'ne', 'le', 'ge', 'lt' or 'gt'.
USER 21	UNDELIMITED TEXT STRING	A string value must terminate with a quote.

PROG Messages

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
USER 17	EXCEEDED MAXIMUM TREE DEPTH	Recursive code has gone through too many performs to trace tree structure. Reduce the depth of the parent/child tree.
USER 41	INCOMPATIBLE CODE FILE	The IPDIC.PUB code file is not compatible with the version of DICTDBM currently executing.
USER 42	INSUFFICIENT STACK FOR WORK SPACE RELOAD	DICTDBM needs more data stack for a work | space reorganization. Run DICTDBM with a MAXDATA greater than 25,000.
USER 43	INSUFFICIENT STACK FOR SYSTEM LOAD	The data stack required for the program is greater than the maximum specified for DICTDBM. Increase the stack size by specifying a MAXDATA on the run command that is greater than the default of 25,000.

SYSTEM Master

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
SYSTEM 1	SORT INITIALIZATION	Error in call to System SORT Utility. The probable cause is insufficient disc space for SORT scratch file.
SYSTEM 2	SORT FILE WRITE	Error on releasing record to the System SORT Utility. If cause is not apparent to the resident System Programmer, contact the Systems Engineer.
SYSTEM 3	SORT OUTPUT	Error on requesting record from System SORT Utility. If cause is not apparent to the resident System Programmer, contact the Systems Engineer.
SYSTEM 4	SORT END	Error in System SORT Utility during exit procedures. If cause is not apparent to the resident System Programmer, contact the Systems Engineer.
SYSTEM 5	CANNOT OPEN PRINT FILE	DICTDBM is unable to open the print file.
SYSTEM 7	CODE FILE READ	DICTDBM detected a read error while reading the code file.
SYSTEM 8	DISC SPACE NOT AVAILABLE FOR SORT FILE	Could not create scratch space for SORT operation.
SYSTEM 9	PRINT FILE ACCESS	The print device is unavailable. Check file equate or call the system operator.

TRAP Messages

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
TRAP 3	WORKSPACE EMPTY	Internal DICTDBM work space is empty. Please notify Systems Engineer.
TRAP 5	EMPTY CODE FILE	DICTDBM detected an empty code file.
TRAP 6	UNEXPECTED EOF IN CODE FILE	DICTDBM encountered an unexpected end-of-file while reading the code file. Please notify the Systems Engineer.
TRAP 7	ARITHMETIC CONVERSION FOR TABLE LITERAL	A program constant cannot be converted to the binary equivalent as the current statement requires.
TRAP 8	BROKEN WORK SPACE CHAIN	DICTDBM has detected a break in a work space link list. Please report the error condition to the Systems Engineer.
TRAP 9	ARITHMETIC TRAP	DICTDBM has detected an arithmetic trap condition in its internal processing. Please report the error condition and the internal address (given as %9.%99999) to the Systems Engineer.
TRAP 10	OUT OF RANGE PCODE ADDRESS	DICTDBM has detected a PCODE address out of the loaded range of transaction codes. Please report the error condition and the PCODE address (given as %999.%99999) to the Systems Engineer.
TRAP 11	DISPLAY FORMAT LEVEL OVERFLOW	DICTDBM has detected an overflow in managing format levels. Please report the error condition to the Systems Engineer.
TRAP 12	DISPLAY FORMAT LEVEL UNDERFLOW	DICTDBM has detected an underflow in managing format levels. Please report the error condition to the Systems Engineer.

DICT Messages

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICT 1	CONFLICTING LINE NUMBER SEQUENCE	Error messages of type DICT are usually due to an incorrect response to a prompt. After the message is issued, the prompt is reissued, allowing the user to respond to the prompt differently.
DICT 2	INVALID EDIT COMMAND, TRY 'HELP'	
DICT 3	NO DESCRIPTION LINES EXIST	
DICT 4	DUPLICATE ENTRY	
DICT 5	AUTO SET ALREADY HAS A KEY ELEMENT	
DICT 6	FILE TYPE MUST BE 'MAST' OR 'AUTO'	
DICT 7	SORT ELEMENT TYPE MUST BE 'X', 'U' OR 'K'	
DICT 8	CANNOT ADD AN ELEMENT TO A FILE OF TYPE type	
DICT 9	CANNOT ADD ELEMENTS TO A PARENT	
DICT 10	CANNOT ADD TO '$MENU'	
DICT 11	CAPABILITY MUST BE 'R', 'U', 'M', OR 'X'	
DICT 12	CANNOT ADD SECURITY TO FILE TYPE type	
DICT 13	CAPABILITY MUST BE 'R', 'W' OR 'X'	
DICT 14	CANNOT USE FILE TYPE type	
DICT 15	STORAGE LENGTH CANNOT BE ZERO	
DICT 16	TYPE MUST BE ONE OF: 'X', 'U', '9', 'Z', 'P', 'I', 'J', 'K', 'R', OR 'E'	

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICT 17	SIZE CANNOT BE ZERO	Error messages of type DICT are usually due to an incorrect response to a prompt. After the message is issued, the prompt is reissued, allowing the user to respond to the prompt differently.
DICT 18	DECIMAL LENGTH MUST BE LESS THAN SIZE	
DICT 19	SIZE TOO LARGE FOR ELEMENT TYPE	
DICT 20	CAN ONLY RESTRICT TO A TYPE 'BASE' FILE	
DICT 21	CANNOT PURGE '$MENU'	
DICT 22	CHILD LARGER THAN PARENT	
DICT 23	CHILD FIELD NOT FULLY WITHIN PARENT	
DICT 24	RELATIONSHIP NOT GENERATED - LOOP FOUND IN STRUCTURE	
DICT 25	CHILD FILE MUST BE TYPE 'MAST', 'AUTO' OR 'DETL'	
DICT 26	CHILD FILE MUST BE TYPE 'FORM'	
DICT 27	PARENT FILE MUST BE TYPE 'BASE' OR 'VPLS'	
DICT 28	BLOCKMAX MUST BE BETWEEN 128 AND 2048	
DICT 29	CANNOT BE A PARENT IF OWN ELEMENTS	
DICT 30	CANNOT RELATE '$MENU' AS A CHILD	
DICT 31	CANNOT SECURE A FILE OF TYPE type	
DICT 32	CHILD ELEMENT NOT FOUND	
DICT 33	CHILD FILE NOT FOUND	
DICT 34	CHILD PROCEDURE NOT FOUND	

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICT 35	CHILD CATEGORY NOT FOUND	Error messages of type DICT are usually due to an incorrect response to a prompt. After the message is issued, the prompt is reissued, allowing the user to respond to the prompt differently.
DICT 36	CHILD GROUP NOT FOUND	
DICT 37	TYPE MUST BE ONE F 'BASE', 'MAST', 'AUTO', 'DETL', KSAM', 'MPEF', 'VPLS', OR 'FORM'	
DICT 38	CANNOT MODIFY '$MENU'	
DICT 39	CATEGORY NOT A PARENT	
DICT 40	ELEMENT NOT A PARENT	
DICT 41	FILE NOT A PARENT	
DICT 42	GROUP NOT A PARENT	
DICT 43	PROCEDURE NOT A PARENT	
DICT 44	CATEGORY CONTAINS NO ELEMENTS	
DICT 45	CLASS CONTAINS NO ELEMENTS	
DICT 46	FILE CONTAINS NO ELEMENTS	
DICT 47	GROUP CONTAINS NO ELEMENTS	
DICT 48	PROCEDURE CONTAINS NO ELEMENTS	
DICT 49	ELEMENT NOT IN FILE	
DICT 50	ELEMENT NOT IN FILE	
DICT 51	ELEMENT NOT IN PROCEDURE	
DICT 52	ELEMENT NOT IN CATEGORY	
DICT 53	ELEMENT NOT IN GROUP	
DICT 54	ELEMENT NOT IN CLASS	
DICT 55	FILE NOT IN CLASS	

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICT 56	FILE NOT IN LOCATION	Error messages of type DICT are usually due to an incorrect response to a prompt. After the message is issued, the prompt is reissued, allowing the user to respond to the prompt differently.
DICT 57	PROCEDURE NOT IN LOCATION	
DICT 58	LINE NUMBER DOES NOT EXIST	
DICT 59	CANNOT RENAME '$MENU'	
DICT 60	NO PARENT FILE EXISTS FOR *file*	
DICT 61	RESPONSE REQUIRED TO PROMPT	
DICT 62	FILE IS NOT A PARENT OF *file*	
DICT 63	STORAGE LENGTH MUST BE EVEN FOR IMAGE	
DICT 64	INSUFFICIENT CAPABILITY	
DICT 65	MUST BE ONE OF 'Y', 'YES', 'N', 'NO', OR <CR>	
DICT 66	SEARCH ITEM CANNOT BE ITS OWN SORT ITEM	
DICT 67	LENGTH MUST BE ONE OF 2, 4, 8, OR 12	
DICT 68	LENGTH MUST BE LESS THAN 15	
DICT 69	LENGTH MUST BE LESS THAN 28	
DICT 70	LENGTH MUST BE ONE OF 4 OR 8	
DICT 71	NEW LENGTH TOO SMALL FOR CHILD: *element*	
DICT 72	NEW LENGTH TOO LARGE FOR PARENT: *element*	
DICT 73	LINK VALUE MUST BE GREATER THAN -2	

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICT 74	REPEAT OPTION INVALID WITH THIS COMMAND STRING	
DICT 75	DICTIONARY INTERNAL CLEANUP NEEDED	Creator of data Dictionary should run DICTDBM.PUB.SYS,UTIL.
DICT 76	INVALID RESPONSE	
DICT 77	RESPONSE MUST BE ONE OF ' ', 'LO', 'LS', 'TO', 'TS'	
DICT 78	CANNOT RELATE TO PARENT ELEMENT OF TYPE *type*	
DICT 79	LINE NUMBER MUST BE GREATER THAN ZERO	
DICT 80	MAXIMUM RECORD SIZE CANNOT BE LESS THAN MINIMUM RECORD SIZE	
DICT 81	MAXIMUM BLOCKING CANNOT BE LESS THAN MINIMUM BLOCKING	
DICT 82	CANNOT USE ELEMENT '$REDEFINES' AS AN ELEMENT REFERENCE	
DICT 83	CANNOT ADD ELEMENT OF TYPE *type* TO AN IMAGE FILE	
DICT 84	ELEMENT IS ASSOCIATED TO AN IMAGE FILE, CANNOT MODIFY TYPE TO *type*	
DICT 85	THIS ELEMENT HAS CHILDREN, CANNOT CHANGE TYPE TO *type*	
DICT 86	CANNOT USE ELEMENT '$REDEFINES' AS A SORT ELEMENT	
DICT 87	SPECIFIED CHILD ELEMENT IS CURRENTLY A BACK REFERENCE ELEMENT	An element that is coherently a back reference element cannot be used as a child element.

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICT 88	THIS ELEMENT IS CURRENTLY A CHILD ELEMENT	Error messages of type DICT are usually due to an incorrect response to a prompt. After the message is issued, the prompt is reissued, allowing the user to respond to the prompt differently.
DICT 89	LOOP FOUND - ELEMENT BACK REFERENCES ITSELF	
DICT 90	ELEMENT IS ASSOCIATED TO A GROUP, CANNOT MODIFY TYPE TO *type*	
DICT 91	CANNOT ADD ELEMENT '$REDEFINES' TO A FILE	
DICT 92	CANNOT ADD ELEMENT '$REDEFINES' TO A GROUP	
DICT 93	CANNOT ADD ELEMENT OF TYPE *type* TO A GROUP	
DICT 94	PARENT ELEMENT IS THE SAME AS THIS CHILD ELEMENT'S BACK REFERENCE	
DICT 95	CANNOT ADD ELEMENT '$REDEFINES' TO A CLASS	
DICT 96	CANNOT ADD ELEMENT '$REDEFINES' TO A PROCEDURE	
DICT 97	CANNOT ADD ELEMENT '$REDEFINES' TO A CATEGORY	
DICT 98	PARENT ELEMENT CANNOT BE '$REDEFINES'	
DICT 99	CANNOT EXCEED 10 LEVELS OF BACK REFERENCING	
DICT 100	THE NEW LENGTH OF REFERENCING ELEMENT *element* IS TOO LARGE FOR PARENT *parent element*	
DICT 101	CHILD '$REDEFINES' MUST HAVE AN ALIAS	

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICT 102	ELEMENT IS A KSAM FILE KEY, CANNOT MODIFY TYPE TO *type*	
DICT 103	CLASS IS ASSOCIATED TO AN ELEMENT, CANNOT MODIFY TYPE TO 'INFO'	
DICT 104	CLASS IS ASSOCIATED TO A FILE, | CANNOT MODIFY TYPE TO 'INFO'	
DICT 105	CLASS MUST BE TYPE 'INFO'	
DICT 106	CHILD CLASS NOT FOUND	
DICT 107	PARENT CLASS MUST BE TYPE 'INFO'	
DICT 108	CHILD CLASS CANNOT BE TYPE ‘INFO'	
DICT 109	CLASS NOT A PARENT	
DICT 110	GROUP NOT IN CLASS	
DICT 111	CANNOT ADD AN ELEMENT TO A CLASS OF TYPE 'INFO'	
DICT 112	CANNOT ADD A FILE TO A CLASS OF TYPE 'INFO'	
DICT 113	CANNOT SECURE A FILE TO A CLASS OF TYPE 'INFO'	
DICT 114	CLASS IS A CHILD CLASS, CANNOT MODIFY TYPE TO 'INFO'	

B UTILITY Error Messages

Error messages are issued for all the Dictionary/3000 utilities. The utilities generate several types of messages on the terminal. Each type of message is found in this appendix or in the appropriate reference manual for the indicated subsystem. The types of messages are:

1. Errors

   ```
   *ERROR:  error-message
   ```

   Errors are conditions that arise out of a user response or some operation during the utility process. A correct response or remedy for the error condition is required.

2. IMAGE errors

   ```
   *IMAGE ERROR:  error-message
   ```

   IMAGE errors occur when a call to the IMAGE data base management system cannot be performed. The error message returned by IMAGE is displayed. Refer to the IMAGE/3000 Reference Manual for more information.

3. File errors

   ```
   *FILE ERROR:  error-message
   ```

   File errors occur when a call to the MPE file system cannot be performed. The error message returned by the file system is displayed. Refer to the MPE 3000 Error Messages and Recovery Reference Manual for more information.

4. Warnings

   ```
   *WARNING:  message
   ```

   Warning messages are conditions that the utility program informs the user about. They are not fatal to the utility process.

5. Condition prompts

   ```
   condition message>
   ```

   Condition prompts inform the user of a certain situation that requires a decision by the user. The usual response is “Y” for yes or “N” for no.

   Note that DICTDBM and DICTINIT issue different messages than listed above. The DICTDBM messages are listed and explained in Appendix A. The DICTINIT messages are listed and explained at the end of Appendix C.

DICTCDE Messages

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICTCDE 1	*ERROR: PASSWORD IS INVALID	The password entered does not give access to the Dictionary.
DICTCDE 2	*ERROR: FILE! IS NOT IN COPYLIB FORMAT	The specified file is not in KSAM file format.
DICTCDE 3	*ERROR: RESPONSE MUST BE ! CHARACTERS OR LESS	
DICTCDE	*ERROR: RESPONSE MUST BE F, E, P, O, OR EX (? FOR HELP)	The user must input one of the responses given in the prompt.
DICTCDE 5	*ERROR: RESPONSE MUST BE Y OR N (? FOR HELP)	The user must input one of the responses given in the prompt.
DICTCDE 6	*ERROR: RESPONSE MUST BE 9 OR X (? FOR HELP)	The user must input one of the responses given in the prompt.
DICTCDE 7	*ERROR: RESPONSE MUST BE A OR P (? FOR HELP)	The user must input one of the responses given in the prompt.
DICTCDE 8	*ERROR: RESPONSE MUST BE F OR S (? FOR HELP)	The user must input one of the responses given in the prompt.
DICTCDE 9	*ERROR: ELEMENT ! IS NOT IN THE DICTIONARY	The element is not defined in the Dictionary.
DICTCDE 10	*ERROR: FILE ! IS NOT IN THE DICTIONARY	The file entity is not defined in the Dictionary.
DICTCDE 11	*ERROR: FILE ! HAS UNKNOWN TYPE !, CANNOT DEFINE IT	The file entity type must be one of MPEF, MPER, KSAM, BASE, MAST, DETL, FORM, VPLUS.
DICTCDE 14	*ERROR: FILE NAME IS MISSING OR BLANK	This error message occurs when the user specifies a file name as 'child(parent)', and the child name is blank, i.e., '(parent)'.
DICTCDE 33	*ERROR: COMMAND NOT PROGRAMMATICALLY EXECUTABLE	The MPE command cannot be executed.
DICTCDE 34	*ERROR: UNKNOWN COMMAND NAME (CIERR !)	The MPE command cannot be recognized.

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICTCDE 35	*ERROR: MPE ERROR IN COMMAND (CIERR !)	MPE could not execute the command.
DICTCDE 36	*ERROR: MPE CREATE PROCESS ERROR ! INVOKING COBEDIT	DICTCDE is unable to run COBEDIT. Check the user/group/account for PH capability, and check to see that the program file COBEDIT.PUB.SYS exists.
DICTCDE 37	*ERROR: END OF FILE FOUND READING FROM DICTIN	The command file DICTIN terminated while DICTCDE was expecting more responses to further prompts.
DICTCDE 38	*ERROR: MPE FILE SYSTEM ERROR ON TERMINAL	An MPE file system error on the terminal has occurred. Check the MPE file system error message for further details.
DICTCDE 39	*ERROR: MPE FILE SYSTEM ERROR ON !	An MPE file system error has occurred. Check the MPE file system error message for further details.
DICTCDE 40	*ERROR: RECORD ALREADY EXISTS IN COPYLIB WITH MODULE-SEQUENCE !&	This is reported only when DICTCDE has an internal error. DICTCDE, in sequencing the source in the module has found a duplicate sequence number already existing in the module.
DICTCDE 41	*ERROR: COPYLIB FILE ! IS FULL	The copylib module is full. Use COBEDIT to expand the copylib file.
DICTCDE 42	*ERROR: COPYLIB MODULE ! HAS NO MORE SEQUENCE NUMBERS	The copylib module has no more sequence numbers. Use COBEDIT to renumber the module.
DICTCDE 43	*ERROR: MPE FILE SYSTEM ERROR ON COPYLIB FILE !	An error has occurred in accessing the copylib file.
DICTCDE 44	*ERROR: ERROR WITH BATCH MODE OR EQUATED INPUT CAUSES TERMINATION &	When DICTCDE is run in batch, or with the commands driven off an disc file, i.e., DICTIN is equated, any error in the responses to prompts will cause DICTCDE to terminate.
DICTCDE 48	*ERROR: ! CAN ONLY BE DEFINED IN THE DATA DIVISION	This message is only returned when Toolset/3000 interface is used, and the user wants to generate a file of type FORM, MAST, DETL, AUTO.
DICTCDE 49	*ERROR: UNKNOWN COMMAND IN HP TOOLSET INTERFACE	Toolset/3000 has passed a command which cannot be recognized by DICTCDE.
DICTCDE 50	*ERROR: INFO PARAMETER IS INVALID	The info parameter passed through the RUN DICTCDE command is invalid.

Error No.	MESSAGE	EXPLANATION AND/OR ACTION
DICTCDE 51	*ERROR: CANNOT OPEN MESSAGE FILES FOR HP TOOLSET	DICTCDE and Toolset/3000 communicate to each other through message files. These message files could not be opened.
DICTCDE 52	*ERROR: INTERNAL ERROR IN HP TOOLSET INTERFACE	DICTCDE could not read the message files used to pass information to and from Toolset/3000.
DICTCDE 53	*ERROR: INTERNAL ERROR -	An error internal to DICTCDE has occurred.
DICTCDE 54	*ERROR: INTERNAL ERROR - DISCOVERED INVALID OPTION	An error internal to DICTCDE has occurred.
DICTCDE 55	*ERROR: INTERNAL ERROR - DISCOVERED INVALID FILE TYPE	An error internal to DICTCDE has occurred.
DICTCDE 56	*ERROR: INTERNAL ERROR - MODULE NAME IS NULL	An error internal to DICTCDE has occurred.
DICTCDE 57	*ERROR: INTERNAL ERROR - IN REDEFINES PROCESSING	An error internal to DICTCDE has occurred.
DICTCDE 58	*ERROR: INTERNAL ERROR !, PLEASE REPORT	An error internal to DICTCDE has occurred.
DICTCDE 59	*ERROR: CANNOT OPEN DICTIONARY !	The Dictionary DICT.PUB does not exist or is equated to a Dictionary which does not exist.
DICTCDE 60	*ERROR: CANNOT READ A CHILD FILE FOR !	DICTCDE has an internal error which prevents it from reading the attributes for the child file.
DICTCDE 61	*ERROR: CANNOT FIND PARENT ENTITY !	DICTCDE has an internal error which prevents it from reading the attributes for the parent entity.
DICTCDE 62	*ERROR: CANNOT READ PARENT/CHILD RELATIONSHIP FOR !	DICTCDE has an internal error which prevents it from reading the attributes for the parent/child relationship.
DICTCDE 63	*ERROR: CANNOT CLOSE DICTIONARY !	DICTCDE has encountered an internal error which prevented it from properly closing the Dictionary.
DICTCDE 64	*ERROR: DICTIONARY ERROR	DICTCDE has an internal error.

Warnings

MESSAGE	EXPLANATION AND/OR ACTION
*WARNING: FILE ! DOES NOT HAVE PARENT !	The user has specified a file name in the format 'child(parent)', and the child does not have the 'parent' specified.
*WARNING: ! CANNOT HAVE A PARENT; IGNORING PARENT	The user has specified a file name in the format 'child(parent)' and the file cannot have a parent, i.e., file types BASE, VPLUS, MPEF, MPER or KSAM.
*WARNING: PARENT FILE IS EMPTY; ASSUMING NO PARENT	The user specified a file name in the format 'child(parent)', and has left the parent name blank, i.e., 'child()'.
*WARNING: FILE ! DOES NOT HAVE A PRIMARY KEY	The file the user has specified is of type KSAM and does not have an element associated with it as its primary key.
*WARNING: DATA BASE HAS MORE THAN ! UNIQUE SEARCH ITEMS; & REMAINDER IGNORED	This warning message appears when an internal table of data base search items overflows.
*WARNING: RECORD LENGTH UPDATED TO FILE RECORD SIZE	When a file has multiple file formats defined, this warning message is issued if they are not of all the same size.
*WARNING: ! IS AN INVALID COBOL IDENTIFIER	The entity name has characters which are not legal in a COBOL name, or has a leading character which is not valid for a COBOL name.
*WARNING: ! IS A COBOL RESERVED WORD	The entity name is a reserved COBOL name. DICTCDE will adjust the name to a non-reserved word.
*WARNING: ! HAS BEEN LIMITED TO 30 CHARACTERS	The prefix with the entity name is longer than 30 characters. COBOL identifiers are limited to 30 characters. DICTCDE will adjust the name to 30 characters.
*WARNING: BACK REFERENCE NAME ! IS AN INVALID COBOL IDENTIFIER &	The entity name back referenced has characters which are not legal in a COBOL name.
*WARNING: CHILD NAME ! IS AN INVALID COBOL IDENTIFIER	The entity name has characters which are not legal in a COBOL name.
*WARNING: ! HAS AN UNDEFINED COBOL TYPE !	The type does not map to any COBOL data type. DICTCDE will map the type into a PIC X type with equivalent storage size.
*WARNING: ELEMENT ! HAS ZERO LENGTH	If an element is of size one, and has an explicit decimal point designated, then there is no room for the digit.

MESSAGE	EXPLANATION AND/OR ACTION
*WARNING: ELEMENT ! HAS NO SPACE FOR A SEPARATE SIGN	If an elements size is completely taken up by the digits, then this warning message is issued to warn the user that there is no room for the separate sign.
*WARNING: ELEMENT ! HAS MORE THAN 18 DIGITS	An element of type P can have a maximum size of 18.
*WARNING: ELEMENT ! HAS AN INVALID COMP STORAGE SIZE	An element of type I can have a maximum size of 8 bytes.
*WARNING: ELEMENT ! HAS AN INVALID COBOL EDIT MASK '!'	A character appears in the edit mask of an element which cannot be translated into a valid character which can appear in a COBOL picture clause.
*WARNING: LEVEL NUMBER FOR ! EXCEEDS 49	COBOL allows a maximum level number of 49. Every nested relationship increments the nesting level by 5. This error message indicates that the nesting of element to element relationships is too deep.
*WARNING: CONFIGURATION MESSAGE ! PREFIX > 8 CHARACTERS	
*WARNING: CONFIGURATION MESSAGE ! SUFFIX > 8 CHARACTERS	
*WARNING: CONFIGURATION MESSAGE 1 OPTION ! INVALID	

Info

MESSAGE	EXPLANATION AND/OR ACTION
*INFO: FILE ! DOES NOT HAVE ANY DATA SETS	The data base file entity does not have any data set file entities related to it. Only the data base special parameters will be generated.
*INFO: FILE ! DOES NOT HAVE ANY FORMS	The VPLUS file entity does not have any form file entities related to it. Only the VPLUS forms file special parameters will be generated.

DICTDBA Messages

Errors

MESSAGE	EXPLANATION AND/OR ACTION
*ERROR: BASE NAME TOO LONG	Data base name, including qualifiers, is greater than 24 characters.
*ERROR: DATA BASE OPERATION:	A data base error occurred when reading information. An IMAGE “DBEXPLAIN message block” is issued to explain the error condition.
*ERROR: INVALID MODE	Response to MODE prompt is a number other than 1 through 8.
*ERROR: INVALID RUN MODE	An incorrect response to RUN MODE prompt entered.
*ERROR: LIST FILE OPERATION	An error occurred during writing the list file (DICTLIST).
*ERROR: PASSWORD LONGER THAN 8 CHARACTERS	Response to BASE PASSWORD prompt is greater than 8 characters.
*ERROR: LIST FILE OPEN	Device LP is not available. Use a file equation to redirect DICTLIST.

Warnings

MESSAGE	EXPLANATION AND/OR ACTION
*WARNING: BROKEN CHAIN(S):	A broken chain has been encountered in a detail set. The data base should be unloaded serially with DICTDBU and then reloaded with DICTDBL to recover the data base.
*WARNING: CONTROL(Y) NOT ENABLED	Attempt to enable Control Y failed.
*WARNING: HIGH MASTER SET OCCUPANCY	The occupancy of a master set is higher than 80%. Consider increasing the data set capacity to reduce the occurrence of synonym chains (unless the capacity is only one entry).
*WARNING: SERIAL COUNT ERROR	Physical serial read count of a master set does not equal the IMAGE table count. The data base should be unloaded serially with DICTDBU and then reloaded with DICTDBL to recover the data base.

DICTDBC Messages

Errors

MESSAGE	EXPLANATION AND/OR ACTION
*ERROR: BAD PASSWORD	The Dictionary data base cannot be opened because the password entered is incorrect.
*ERROR: CANNOT OPEN DATA BASE DICT.PUB IMAGE error message	The Dictionary data base cannot be opened. The error message returned by IMAGE is shown.
*ERROR: DBSCHEMA PROCESSOR CANNOT BE ACTIVATED	Utility has been unable to activate the DBSCHEMA program. Contact your system manager.
*ERROR: DBSCHEMA PROCESSOR CANNOT BE INVOKED	Utility has been unable to invoke the DBSCHEMA program in the PUB group of the SYS account. Contact your system manager.
*ERROR: DICTIONARY ACCESS, PROBABLY AN INCOMPATIBLE DICTIONARY DATA BASE	The Dictionary schema is incompatible with current release of the utility.
*ERROR: DICTIONARY DATA BASE OPERATION	A data base error occurred. An IMAGE “DBEXPLAIN message block” is issued to explain the error condition.
*ERROR: EXCEEDED 255 TIMES IN BASE	Utility has found more than 255 data items for the base in the Dictionary which exceeds the limit for an IMAGE data base. The schema is generated without the excess data items.
*ERROR: EXCEEDED 63 CLASSES IN BASE	Utility has found more than 63 security classes for the base in the Dictionary which exceeds the limit for an IMAGE data base. The schema is generated without the excess security classes.
*ERROR: EXCEEDED 99 SETS IN BASE	Utility has found more than 99 data sets for the base in the Dictionary which exceeds the limit for an IMAGE data base. The schema is generated without the excess data sets.
*ERROR: FILE NAME TOO LONG	Response to SCHEMA FILE or LIST FILE is greater than 26 characters.
*ERROR: INVALID LIST FILE	Response to the LIST FILE prompt is not a valid file name.
*ERROR: INVALID STORAGE LENGTH FOR ITEM:	Utility has detected a Dictionary item item-name whose storage length is invalid for the given type. Correct the storage length in the Dictionary using the DICTDBM utility.

MESSAGE	EXPLANATION AND/OR ACTION
*ERROR: LINE LONGER THAN 70 CHARACTERS	The response to CONTROL LINE is longer than 70 characters.
*ERROR: PASSWORD DOES NOT ALLOW A ACCESS TO 'CLASS-PASSWORD'	Password does not grant sufficient capability to create a data base.
*ERROR: PASSWORD LONGER THAN 8 CHARACTERS	Response to DICTIONARY PASSWORD prompt is greater than 8 characters.
*ERROR: PROCESS TERMINATED ON FATAL ERROR CONDITION	A fatal error occurred during the utility process and the program has been terminated.
*ERROR: SCHEMA FILE WRITE	Error detected in writing schema file. A “file error tombstone” is issued to explain the error condition.
*ERROR: STORAGE LENGTH TOO SMALL FOR ITEM: item-name	Utility has detected a Dictionary item whose storage length definition is too small to contain values. Correct the item type, size, or storage length in the Dictionary using the DICTDBM utility.
*ERROR: TEMPORARY FILE ACCESS	Error occurred using the temporary file in preparing the schema file. A “file error tombstone” is issued to explain the error condition.
*ERROR: TEMPORARY FILE OPEN	A temporary file used in preparing the schema file cannot be opened. A “file error tombstone” is issued to explain the error condition
*ERROR: LIST FILE OPEN	Device LP is not available. Use a file equation to redirect DICTLIST.

MESSAGE	EXPLANATION AND/OR ACTION
*WARNING: BASE NAME TRUNCATED TO 6 CHARACTERS	The name of a base can be up to 20 characters in the Dictionary, but only the first 6 are used for generating the physical data base file name.
*WARNING: CONTROL(Y) NOT ENABLED	Attempt to enable Control Y failed.

MESSGE	EXPLANATION AND/OR ACTION
[TEMPORARY] FILE ALREADY EXISTS, PURGE OLD (N/Y)?>	The file name entered in response to the SCHEMA FILE prompt already exists as a temporary or a permanent file. Respond “Y” to purge the existing file and keep the new file.

DICTDBD Messages

Errors

MESSAGE	EXPLANATION AND/OR ACTION
*ERROR: BASE ALREADY DEFINED IN DATA DICTIONARY	The data base name is already in the Dictionary. The user is prompted as to whether the data base should be given a different name when loaded into the Dictionary.
*ERROR: BASE NAME TOO LONG	Data base name, including qualifiers, exceeds 24 characters.
*ERROR: DATA BASE OPERATION	A data base error occurred when reading information. An IMAGE “DBEXPLAIN message block” is issued to explain the error condition.
*ERROR: DATA DICTIONARY OPERATION	A data Dictionary error occurred. An IMAGE | “DBEXPLAIN message block” is issued to explain the error condition.
*ERROR: INVALID MODE	Response to MODE prompt is a number other than 1 through 8.
*ERROR: ITEM SEARCH IN DBINFO LIST	A system error occurred in processing “DBINFO” information from the data base. If the cause is not apparent, contact your system manager.
*ERROR: PASSWORD LONGER THAN 8 CHARACTERS	Response to DICTIONARY PASSWORD or BASE PASSWORD prompt is greater than 8 characters.
*ERROR: INSUFFICIENT ACCESS CAPABILITY TO *ERROR: INSUFFICIENT ACCESS CAPABILITY TO DATA BASE	Reenter response to BASE.

Warnings

MESSAGE	EXPLANATION AND/OR ACTION
*WARNING: CONTROL(Y) NOT ENABLED	“Arming” Control Y failed.
*WARNING: DATA SET HAS NOT BEEN LOADED INTO DICTIONARY	Response to the NEW PRIMARY FILE NAME prompt is RETURN, or response to USE EXISTING DEFINITION (N/Y)? prompt is “N” or RETURN.
*WARNING: ELEMENT HAS NOT BEEN LOADED INTO DICTIONARY	Response to the NEW PRIMARY ELEMENT NAME prompt is RETURN.

Condition Prompts

MESSAGE	EXPLANATION AND/OR ACTION
ENTIRE DEFINITION OF DATA BASE WILL NOT FIT IN DICTIONARY, PROCEED (N/Y)?	Definition of data base will not fit in dictionary. Respond “N” to terminate utility; respond “Y” to continue until one of the data sets is full, then terminate.

DICTDBL Messages

Errors

MESSAGE	EXPLANATION AND/OR ACTION
*ERROR: BAD DATA BASE NAME: *base-name*	The utility has encountered an invalid base name format in the store file. Probably the store file has bad data in it.
*ERROR: BASE LONGER THAN 6 CHARACTERS	Base name exceeds 6 characters.
*ERROR: BASE NAME TOO LONG	Data base name, including qualifiers, exceeds 24 characters.
*ERROR: DATA BASE OPERATION	A data base error occurred during the load. A “DBEXPLAIN message block” is issued to explain the error condition.
*ERROR: DEFERRED MODE REJECTED, CODE=*n*	The request for fast I/O has been rejected with an error code of “n”. Refer to the DBCONTROL intrinsic in the *IMAGE Reference Manual* for an explanation.
*ERROR: FILE EQUATION NOT ALLOWED	A back-referenced file name (*filename) entered as a response to the STORE FILE prompt is not allowed.
*ERROR: FILE NAME TOO LONG	Response to the LIST FILE prompt is greater than 26 characters.
*ERROR: INCOMPATIBLE STORE FILE	The store file is incompatible with the current release of the utility.
*ERROR: INCORRECT STORE TAPE MOUNTED	The tape mounted is not the correct store tape. Mount the correct tape.
*ERROR: INCORRECT TAPE NUMBER: *n*	The given tape number “n” is out of sequence. Mount the correct tape.
*ERROR: INVALID MODE	Response to MODE prompt is a number other than 1 through 8.
*ERROR: INVALID RUN MODE	Response to the RUN MODE prompt is invalid. Enter one of the modes indicated in the prompt.
*ERROR: ITEM CONVERSION	A conversion error occurred on an item with properties that have changed in the new data base schema. The item data field is set to a null value of all binary zeros.

MESSAGE	EXPLANATION AND/OR ACTION
*ERROR: ITEM NAME TOO LONG	Data item name entered in response to the NEW ITEM NAME prompt is longer than 16 characters.
*ERROR: PASSWORD LONGER THAN 8 CHARACTERS	Response to the BASE PASSWORD prompt is greater than 8 characters.
*ERROR: PRINT FILE CANNOT BE OPENED	The print file for the listing cannot be opened. Enter a correct file name to the LIST FILE prompt.
*ERROR: SET NAME TOO LONG	Data set name entered in response to the NEW SET NAME prompt is longer than 16 characters.
*ERROR: STORE FILE OPEN	The store file cannot be opened. A “file error tombstone” is issued to explain the error condition.

Warnings

MESSAGE	EXPLANATION AND/OR ACTION
*WARNING: CONTROL(Y) NOT ENABLED	“Arming” Control Y failed.

Condition Prompts

MESSAGE	EXPLANATION AND/OR ACTION
CONTINUE SET LOAD(Y/N)?>	After an error or a Control Y interrupt, this prompt allows the user to continue the operation or to terminate the utility program.
CONTROL(Y) BREAK, CONTINUE(Y/N)?>	User has entered Control Y during a data set load. A response of “N” will terminate the utility program.
DISPLAY INPUT RECORD(Y/N)?>	After an IMAGE error in trying to store a new record in the data base, this prompt allows the user to dump the record to the list file in ASCII and octal format.
LIST FILE ALREADY EXISTS, PURGE OLD(N/Y)?>	The list file is being written to disc and a file of the same name already exists. Respond “Y” to purge the existing file and keep the new file.
TAPE FILE REQUESTED(Y/N)	This prompt is issued if RETURN is entered in response to the STORE FILE prompt. If an “N” response is given, the utility program terminates.

DICTDBU Messages

Errors

MESSAGE	EXPLANATION AND/OR ACTION
*ERROR: BASE LONGER THAN 6 CHARACTERS	Base name exceeds 6 characters.
*ERROR: DATA BASE OPERATION	A data base error occurred. A “DBEXPLAIN message block” is issued to explain the error condition.
*ERROR: DBINFO OPERATION	The utility encountered an error when using a “DBINFO” call to IMAGE. A “DBEXPLAIN message block” is issued to explain the error condition.
*ERROR: DBINFO - SERIAL UNLOAD INVOKED	The utility encountered an error in the DBINFO information from IMAGE for the data base structure. A “DBEXPLAIN message block” is issued to explain the error condition. The utility has rejected the request for the chained unload of the current data set and is performing a serial unload.
*ERROR: FILE EQUATION NOT ALLOWED	A back-referenced file name (*filename) entered as a response to the STORE FILE prompt is not allowed.
*ERROR: FILE NAME TOO LONG	Response to the LIST FILE prompt is greater than 26 characters.
*ERROR: INCORRECT SYNTAX IN USE OF PERIOD(S)	An incorrect qualified name entered in response to the BASE prompt.
*ERROR: INVALID MODE	Response to MODE prompt is a number other than 1 through 8.
*ERROR: ITEM NOT FOUND	Named item is not found in the definition of the set when EDIT MODE used.
*ERROR: LIST FILE	The utility encountered an error when accessing the list file. A “file error tombstone” is issued to explain the error condition.
*ERROR: PASSWORD LONGER THAN 8 CHARACTERS	Response to BASE PASSWORD prompt is greater than 8 characters.
*ERROR: PRINT FILE CANNOT BE OPENED	The print file for the listing cannot be opened. Enter a correct file name to the LIST FILE prompt.

MESSAGE	EXPLANATION AND/OR ACTION
*ERROR: SEARCH ITEM NOT FOUND	Named item is not a search item within the detail set when EDIT MODE used.
*ERROR: STORE FILE CANNOT BE CLOSED	The utility was unable to close the store file in either the permanent or temporary file domain. Contact your system manager.
*ERROR: STORE FILE EXISTS AND CANNOT BE ACCESSED	Accessing the named store file is a security violation.
*ERROR: STORE FILE OPEN	The store file cannot be opened. A “file error tombstone” is issued to explain the error condition.
*ERROR: STORE FILE WRITE	The utility encountered an error when writing the store file. A “file error tombstone” is issued to explain the error condition.

Warnings

MESSAGE	EXPLANATION AND/OR ACTION
*WARNING: CONTROL(Y) NOT ENABLED	“Arming” Control Y failed.
*WARNING: INSUFFICIENT GROUP DISC SPACE — STORE FILE IS SESSION TEMPORARY	The utility was unable to close the store file in the permanent file domain because of insufficient disc space allocation in the file group. The store file is session temporary and will be lost when the user logs off. If you would like to keep the store file as a permanent file, you should contact your system manager for a higher disc space allocation. Save the temporary file to the permanent domain by using the MPE SAVE command.

Condition Prompts

MESSAGE	EXPLANATION AND/OR ACTION
CONTINUE SET UNLOAD(Y/N)?>	After an error or a Control Y interrupt, this prompt allows the user to continue the operation or to terminate the utility program.
CONTROL(Y) BREAK, CONTINUE(Y/N)?>	User has entered Control Y during a data set unload. A response of “N” will terminate the utility program.
LIST FILE ALREADY EXISTS, PURGE OLD(N/Y)?>	The list file is being written to disc and a file of the same name already exists. Respond “Y” to purge the existing file and keep the new file.
STORE FILE ALREADY EXISTS, PURGE OLD(N/Y)>	The store file is being written to disc and a file of the same name already exists. Respond “Y” to purge the existing file and keep the new file.
TAPE FILE REQUESTED (Y/N)	This prompt is issued if RETURN is entered in response to the STORE FILE prompt. If an “N” response is given, the utility program terminates.

DICTPDE Messages

Errors

MESSAGE	EXPLANATION AND/OR ACTION
*DICT ERROR: CANNOT CLOSE DICTIONARY!	The Dictionary entered was not entered correctly or does not exist. Reenter with the correct Dictionary.
*DICT ERROR: INVALID PASSWORD. CANNOT OPEN DICTIONARY!	The Dictionary cannot be opened because the password entered is incorrect. Reenter with the correct password.
*ERROR: COMMAND AND SUBCOMMAND ONLY	Something other than the command and subcommand was entered as a response.
*ERROR: ILLEGAL COMMAND. PLEASE REENTER	Command entered was not a legal command. Reenter with GENERATE, ALTER, EXIT, HELP or LIST.
*ERROR: ILLEGAL COMMAND/SUBCOMMAND	
*ERROR: ILLEGAL EXTRACT (TYPE OR VAR)	Toolset/3000 has requested DICTPDE to extract entities as neither TYPE or VAR. Entities can only be extracted as TYPE or VAR.
*ERROR: ILLEGAL INFO STRING LENGTH	
*ERROR: ILLEGAL SUBCOMMAND. PLEASE REENTER	The subcommand entered is not a legal subcommand for that command.
*ERROR: INFO STRING MUST BE FROM HP TOOLSET	
*ERROR: INPUT TOO LONG	Too many characters were entered on one response line.
*ERROR: INVALID RESPONSE. PLEASE REENTER	The response entered does not match one of the responses given in the prompt.
*ERROR: NON-EXISTENT ELEMENT!	The element specified is not defined in the Dictionary.
*ERROR: NON-EXISTENT FILE!	The file specified is not defined in the Dictionary.
*ERROR: NON-EXISTENT FILE TYPE!	
*LANGUAGE ERROR: PREVIOUSLY EXTRACTED! AS A TYPE	The entity specified has already been extracted as a TYPE declaration.
*LANGUAGE ERROR: PREVIOUSLY EXTRACTED! AS A VAR	The entity specified has already been extracted as a VAR declaration.

MESSAGE	EXPLANATION AND/OR ACTION
*LANGUAGE ERROR: PREVIOUSLY GENERATED COMAREA	The VPLUS COMAREA was already generated in the output file.
*LANGUAGE ERROR: PREVIOUSLY GENERATED IMAGE PARAMETERS	The IMAGE parameters were already generated in the output file.
*LANGUAGE ERROR: ! IS ILLEGAL PASCAL NAME	The entity name consists of characters which are not legal PASCAL identifiers. DICTPDE will convert the name to a legal PASCAL name.
*LANGUAGE ERROR: ! IS PASCAL RESERVED WORD	The entity name is a PASCAL reserved word. DICTPDE will convert the name to a legal TYPE or VAR identifier.
*MPE ERROR: CANNOT FREE EXTRA DATA SEGMENT	
*MPE ERROR: CANNOT GET EXTRA DATA SEGMENT	
*MPE ERROR: ERROR IN WRITING TO EXTRA DATA SEGMENT	
*MPE ERROR: ERROR ON READING FROM EXTRA DATA SEGMENT	

Warnings

MESSAGE	EXPLANATION AND/OR ACTION
*PASCAL FILE ERROR: PASCERR!	
*PASGEN: ERROR DETECTED DURING EXECUTION	
*FILE WARNING: OUTPUT FILE ALREADY EXISTS. WILL APPEND	All declarations generated will be appended to the specified output file.
*FILE WARNING: TOO MANY OUTPUT FILES	
*WARNING: NO ELEMENTS OR FILES ARE RELATED TO THIS ENTITY.	

DICTVPD Messages

Errors

MESSAGE	EXPLANATION AND/OR ACTION
FAILURE TO OPEN DICT.PUB	The Dictionary DICT.PUB does not exist, or has been directed to a Dictionary which does not exist.
FAILURE TO OPEN FORMS FILE	The forms file could not be opened. The forms file does not exist, or is opened for exclusive access by some other program.
INCOMPATIBLE DICTIONARY/3000 VERSION	This version of the Dictionary does not match the version the utility expects. The Dictionary should be re-initialized using the DICTINIT utility and the re-initialize option.
INVALID FORM NAME	The form name entered is not in the forms file or is not in the correct syntax for a form name.
INVALID RESPONSE	The user input does not match one of the possible responses given in the prompt. Reenter response.
UNEXPECTED ERROR	DICTVPD has encountered an error internal to the utility.
*ERROR WHILE ACCESSING FILE: #	DICTVPD encountered an MPE file system error while reading the file.
*ERROR: ELEMENT NOT DEFINED	This error message will only be displayed if DICTVPD encounters an internal error. While relating the elements to the forms file, DICTVPD could not find this element in the Dictionary.
*ERROR: INTERNAL TABLE OVERFLOW (TOO MANY FORMS)	The internal table of DICTVPD have overflowed. There are too many forms defined in the form file to be loaded in one execution of DICTVPD.
*ERROR: LIST FILE ALREADY EXISTS.	The list file specified already exists.
*ERROR: NAME CANNOT BE QUALIFIED	The name entered should not be qualified with the group or account.

Warnings

MESSAGE	EXPLANATION AND/OR ACTION
*WARNING: DICTIONARY ALREADY CONTAINS FORM: #	The form to be loaded already exists in the Dictionary
*WARNING: ELEMENT HAS NOT BEEN LOADED INTO DICTIONARY	DICTVPD skipped the specified element
*WARNING: FORMS FILE ALREADY NAMED IN DATA DICTIONARY	Forms file to be loaded already exists in the Dictionary.

C DICTIONARY/3000 Initialization Procedure

Overview

Once Dictionary/3000 software has been installed on your system, you must execute the program DICTINIT to create and initialize a data dictionary. This program allows you to customize the dictionary for your application and to define the security for the dictionary. It also allows you to re-initialize an existing dictionary when you want to change the passwords for access or to change the capacities of the data sets in the dictionary.

The dictionary is created in the MPE group and account from which the DICTINIT program is run. The dictionary maintenance program (DICTDBM) and the dictionary utility programs all assume the dictionary exists in the PUB group of the log-on account. If you create a dictionary in a different group and/or account, you will first have to identify these before running DICTDBM or the utilities. You do this by issuing a file equation as follows:

```
FILE DICT.PUB=DICT.group.account
```

When running DICTINIT, you can define five levels of access to the dictionary. You do this by assigning a password to each level. The access levels are hierarchical — that is, each level is given the capability for its level and all levels below. The topmost level of MANAGER has complete and total access to the dictionary. The mid-level of INFORM has access for INFORM as well as for DOCUMENTATION and REPORT, and so forth.

Entries that pertain to security, such as security class numbers and passwords, can only be created and modified by someone using the password assigned to MANAGER. A description of each possible level and the access capability given to each follows:

LEVEL	ACCESS CAPABILITY	WHO SHOULD USE
MANAGER	Total access.	The person responsible for the data dictionary.
PROGRAMMER	Can modify entries except security entries; can read security entries.	Anyone who needs to create and maintain files and elements in the dictionary for application development.
INFORM	Can modify HP Inform/3000 group entries and all entries below; cannot read/modify security entries.	Anyone responsible for the maintenance of the HP Inform/3000 program.

LEVEL	ACCESS CAPABILITY	WHO SHOULD USE
DOCUMENTATION	Can modify documentation entries and all entries below; cannot read/modify security entries.	Anyone responsible for maintaining documentation entries in the dictionary (PROCEDURES, LOCATIONS, or CATEGORIES).
REPORT	Can modify NO entries; can read all entries except security entries.	Anyone who needs to write can read all entries except reports about the contents of the dictionary.

Once passwords for the various levels are assigned, there is no way for you to see them within Dictionary/3000. If you are the database creator, you can run DBUTIL and see the passwords. (Refer to the *IMAGE/3000 Reference Manual* for a description of the DBUTIL utility.) If not, you can run the DICTINIT program again and use the RE-INITIALIZATION option to assign new passwords. If you do not specify a password for a particular level, that security level is deleted - there is no access at that level. The MANAGER level cannot be deleted; you must specify a password for the MANAGER level.

There are two phases to the DICTINIT program. During the first phase (initialization), information provided by the Dictionary/3000 software is used to create a temporary dictionary which is automatically purged at the end of phase two. To create this temporary dictionary, a special stream job is automatically invoked by DICTINIT. If you use passwords at either the user, account, or group level when you log on, the stream job prompts you to enter them before it begins. You should monitor the status of the stream job to be sure no errors are encountered. If the job was able to complete, a message informing you to run DICTINIT.PUB.SYS,UPDATE for phase two is displayed. If the job aborts, the line printer listing will tell you why. After correcting the problem, you can begin again.

Phase two (customization) allows you to customize your dictionary or use the defaults, and to assign passwords for the levels of access. At the end of phase two another stream job is invoked. This job builds your permanent data dictionary. If the job in phase two completes successfully, you are informed that DICTINIT IS COMPLETE. If you do not receive this message, but the job has finished, check the line printer listing for the errors. After correcting them, rerun DICTINIT.PUB.SYS,UPDATE.

You can terminate DICTINIT by entering a CNTL-Y at any time. If you enter a CNTL-Y while running phase two, any changes that have been verified are accepted and become the defaults when phase two is run again. A complete list of all error messages, including what to do to recover, is given following the examples.

Example

To initialize a new data dictionary using the default capacities provided by HP, simply follow the process below - substituting your responses for the ones shown where appropriate. The first step is to log on to the group and account in which you want the dictionary to appear. Then follow the following example:

```
:RUN DICTINIT.PUB.SYS      How to start initialization process.
```

<The Dictionary/3000 initialization program banner appears here.>

Initialization/Re-initialization (I/R) >i Select “I” to build a new dictionary.

USER PASSWORD >	If logon passwords were used, enter them in response to these prompts.
ACCOUNT PASSWORD >LOCKIT	They are used by the batch job that installs a temporary dictionary.
GROUP PASSWORD > #J33	Number of the STREAMed job which performs step one (initialization).
END OF PROGRAM :	End of step one.
FROM/<your account>/PLEASE RUN DICTINIT.PUB.SYS,UPDATE	
	This message informs you that the job is complete, and you are to perform step two (customization).
:RUN DICTINIT.PUB.SYS,UPDATE	Performing step two.

<The Dictionary/3000 initialization program banner goes here.>

Initialization/Re-initialization (I/R) >i Again - new dictionary.

Dictionary capacities: Default or Provided (D/P) >d Use the default capacities.

Listing of the capacities for your examination.

DATA-ELEMENT	will have capacity 1001
DATA-FILE	will have capacity 503
DATA-PROCEDURE	will have capacity 203
DATA-CATEGORY	will have capacity 203
DATA-GROUP	will have capacity 503
DATA-CLASS	will have capacity 203
DATA-LOCATION	will have capacity 203
LINK-FILE	will have capacity 401
LINK-ELEMENT	will have capacity 401
LINK-DESCRIPTION	will have capacity 2003
DATA-REPORTLOC	will have capacity 503
ELEMENT-REFTYPE	will have capacity 500
ELEMENT-ELEMENT	will have capacity 500
FILE-FILE	will have capacity 250
PROCEDURE-PROCED	will have capacity 100
CATEGORY-CATEGOR	will have capacity 100
GROUP-GROUP	will have capacity 250
FILE-ELEMENT	will have capacity 2000
FILE-EL-SECOND	will have capacity 100

FILE-PATH	will have capacity 400
FILE-SORT	will have capacity 400
PROCEDURE-ELEMENT	will have capacity 500
CATEGORY-ELEMENT	will have capacity 500
Press RETURN to continue >	Press RETURN to see the rest of the display; it fills more than one screen in this case.
GROUP-ELEMENT	will have capacity 1000
CLASS-CLASS	will have capacity 500
CLASS-ELEMENT	will have capacity 3000
CLASS-FILE	will have capacity 500
CLASS-GROUP	will have capacity 500
FILE-LOCATION	will have capacity 500
PROCEDURE-LOCATI	will have capacity 200
DESCRIPTION-TEXT	will have capacity 5000
REPORT-LIST	will have capacity 500
Are the capacities correct? (Y/N) >Y	This prompt gives you a chance to answer “N” and change them; here we say yes.
Password for MANAGER access >DICTMGR	These prompts allow you to specify the security levels and their passwords; you can respond with a password or simply press RETURN for no access at that level.

Password for PROGRAMMER access >DICPRG

Password for INFORM access >DICINF

Password for DOCUMENTATION access >

Password for REPORT access >DICRPT

Program lists the passwords assigned.

Password for MANAGER	access will be DICTMGR
Password for PROGRAMMER	access will be DICPRG
Password for INFORM	access will be DICINF
There will be no DOCUMENTATION access allowed	RETURN was pressed, nullifying that level.
Password for REPORT	access will be DICRPT
Are the passwords correct? (Y/N) >y	You can change the passwords by responding “N”.
USER PASSWORD >	Again — any log-on passwords are needed to start the

	final initialization job which actually creates the new dictionary.

ACCOUNT PASSWORD >LOCKIT

GROUP PASSWORD >

#J34	Number of the final job.
END OF PROGRAM	End of customization step. The program is complete; your dictionary is initialized.

FROM/<your account>/DICTINIT IS COMPLETE

This example demonstrates how to enter your own capacities and change them after they are displayed. As before, log on to the group and account in which you want the dictionary before starting.

`:RUN DICTINIT.PUB.SYS`	How to execute the program.

<The Dictionary/3000 initialization program banner appears here>

Initialization/Re-initialization (I/R) >i	Select “I” to build a new dictionary.

USER PASSWORD >

ACCOUNT PASSWORD >LOCKIT	If a user, account, or group password is required, enter it in response to these prompts.

GROUP PASSWORD >
#J35

END OF PROGRAM
:

FROM/<your account>/PLEASE RUN `DICTINIT.PUB.SYS,UPDATE`	Informs you to run step two.

```
:RUN DICTINIT.PUB.SYS,UPDATE
```

<The Dictionary/3000 initialization program banner appears here.>

Initialization/Re-initialization (I/R) >i	Select “I” to build a new dictionary.
Dictionary capacities: Default or Provided (D/P) >p	Select “P” to enter new capacities; old ones are displayed.

DATA-ELEMENT	has capacity 1001
New capacity for	DATA-ELEMENT > 1002
DATA-FILE	has capacity 503
New capacity for	DATA-FILE > 504
DATA-PROCEDURE	has capacity 203
New capacity for	DATA-PROCEDURE > 204
DATA-CATEGORY	has capacity 203
New capacity for	DATA-CATEGORY > 204

DATA-GROUP New capacity for	has capacity 503 DATA-GROUP > 504	
DATA-CLASS New capacity for	has capacity 203 DATA-CLASS > 204	
DATA-LOCATION New capacity for	has capacity 203 DATA-LOCATION >	Pressing RETURN keeps the old capacity.
LINK-FILE New capacity for	has capacity 401 LINK-FILE > 402	
LINK-ELEMENT New capacity for	has capacity 401 LINK-ELEMENT > 402	
LINK-DESCRIPTION New capacity for	has capacity 2003 LINK-DESCRIPTION > 2004	
DATA-REPORTLOC New capacity for	has capacity 503 DATA-REPORTLOC > 504	
ELEMENT-REFTYPE New capacity for	has capacity 500 ELEMENT-REFTYPE > 501	
ELEMENT-ELEMENT New capacity for	has capacity 500 ELEMENT-ELEMENT > 501	
FILE-FILE New capacity for	has capacity 250 FILE-FILE > 251	
PROCEDURE-PROCED New capacity for	has capacity 100 PROCEDURE-PROCED > 101	
CATEGORY-CATEGOR New capacity for	has capacity 100 CATEGORY-CATEGOR > 101	
GROUP-GROUP New capacity for	has capacity 250 GROUP-GROUP > 251	
FILE-ELEMENT New capacity for	has capacity 2000 FILE-ELEMENT >	
FILE-EL-SECOND New capacity for	has capacity 100 FILE-EL-SECOND > 101	
FILE-PATH New capacity for	has capacity 400 FILE-PATH > 401	
FILE-SORT New capacity for	has capacity 400 FILE-SORT > 401	
PROCEDURE-ELEMEN New capacity for	has capacity 500 PROCEDURE-ELEMEN > 501	
CATEGORY-ELEMENT New capacity for	has capacity 500 CATEGORY-ELEMENT > 501	
GROUP-ELEMENT New capacity for	has capacity 1000 GROUP-ELEMENT > 1001	

CLASS-CLASS New capacity for	has capacity 500 CLASS-CLASS > 501	
CLASS-ELEMENT New capacity for	has capacity 3000 CLASS-ELEMENT > 3001	
CLASS-FILE New capacity for	has capacity 500 CLASS-FILE > 501	
CLASS-GROUP New capacity for	has capacity 500 CLASS-GROUP > 501	
FILE-LOCATION New capacity for	has capacity 500 FILE-LOCATION > 501	
PROCEDURE-LOCATI New capacity for	has capacity 200 PROCEDURE-LOCATI > 201	
DESCRIPTION-TEXT New capacity for	has capacity 5000 DESCRIPTION-TEXT > 5001	
REPORT-LIST New capacity for	has capacity 500 REPORT-LIST > 501	
DATA-ELEMENT	will have capacity 1002	Listing of capacities.
DATA-FILE	will have capacity 504	
DATA-PROCEDURE	will have capacity 204	
DATA-CATEGORY	will have capacity 204	
DATA-GROUP	will have capacity 504	
DATA-CLASS	will have capacity 204	
DATA-LOCATION	will have capacity 203	
LINK-FILE	will have capacity 402	
LINK-ELEMENT	will have capacity 402	
LINK-DESCRIPTION	will have capacity 2004	
DATA-REPORTLOC	will have capacity 504	
ELEMENT-REFTYPE	will have capacity 501	
ELEMENT-ELEMENT	will have capacity 501	
FILE-FILE	will have capacity 251	
PROCEDURE-PROCED	will have capacity 101	
CATEGORY-CATEGOR	will have capacity 101	
GROUP-GROUP	will have capacity 251	
FILE-ELEMENT	will have capacity 2000	
FILE-EL-SECOND	will have capacity 101	

```
FILE-PATH                will have capacity 401
FILE-SORT                will have capacity 401
Press RETURN to continue >

PROCEDURE-ELEMEN         will have capacity 501
CATEGORY-ELEMENT         will have capacity 501
GROUP-ELEMENT            will have capacity 1001
CLASS-CLASS              will have capacity 501
CLASS-ELEMENT            will have capacity 3001
CLASS-FILE               will have capacity 501
CLASS-GROUP              will have capacity 501
FILE-LOCATION            will have capacity 501
PROCEDURE-LOCATI         will have capacity 201
DESCRIPTION-TEXT         will have capacity 5001
REPORT-LIST              will have capacity 501

Are the capacities correct?
(Y/N) >n                 Enter "N" to change values.
Data-set name (DONE/ALL/name)
>report-list             Change capacity of data set REPORT-LIST.
                REPORT-LIST  has capacity 501
                New capacity for REPORT-LIST        > 200
                Data-set name (DONE/ALL/name) >description-text
                DESCRIPTION-TEXT has capacity 5001
                New capacity for DESCRIPTION-TEXT  > 5000
Data-set name (DONE/ALL/name) >done   Enter DONE to stop changing capacities.
DATA-ELEMENT will have capacity 1002 Listing of capacities.
    .
    .
    .
Press RETURN to continue >
    .
    .
    .
```

DESCRIPTION-TEXT will have capacity 5000

REPORT-LIST will have capacity 200

Are the capacities correct? (Y/N) >y

Password for MANAGER access >DICTMGR

Password for PROGRAMMER access > Entering RETURN indicates that there will be no access to the dictionary at these levels.

Password for INFORM access >

Password for DOCUMENTATION access >

Password for REPORT access >

Password for MANAGER access will be DICTMGR Listing of passwords

There will be no PROGRAMMER access allowed assigned.

There will be no INFORM access allowed

There will be no DOCUMENTATION access allowed

There will be no REPORT access allowed

Are the passwords correct? (Y/N) >y

```
USER PASSWORD >
ACCOUNT PASSWORD >LOCKIT
GROUP PASSWORD >
 #J36
END OF PROGRAM
:
```

FROM/<your account>/DICTINIT IS COMPLETE Program is complete; your dictionary is initialization.

This example shows a dictionary re-initialization. That is, there is already a data-dictionary in your group, and you want to resize it and/or change the passwords to access it.

:RUN DICTINIT.PUB.SYS How to execute the program.

<The Dictionary/3000 initialization program banner appears here>

Initialization/Re-initialization (I/R) Select “R” to re-initialize an existing dictionary.

>r

USER PASSWORD > If logon passwords were used, enter them in response to these prompts.

ACCOUNT PASSWORD >

GROUP PASSWORD >

Dictionary store file on Tape or Disk(T/D) >d

The present contents of the dictionary will be stored by DICTINIT, to be recovered into the new dictionary when it is created. This store file may be quite large, and you may not have the disk space to hold it. In this case, you may store it to tape. Otherwise, request DICTINIT to use a disk file, as in this example.

#J46

END OF PROGRAM

:

FROM/<your account>/PLEASE RUN DICTINIT.PUB.SYS,UPDATE End of step one.

:RUN DICTINIT.PUB.SYS,UPDATE Execute step two.

<The Dictionary/3000 initialization program banner appears here>

Initialization/Re-initialization (I/R) >r Select R to re-initialize.

Dictionary capacities: Default or Provided (D/P) >d Use defaults.

DATA-ELEMENT will have capacity 1001 Listing of capacities.

.

.

.

CATEGORY-ELEMENT will have capacity 500

Press RETURN to continue >

GROUP-ELEMENT will have capacity 1000

.

.

.

REPORT-LIST will have capacity 500

Are the capacities correct? (Y/N) >y

Password for MANAGER access >DICTMGR Enter passwords for the various security levels.

Password for PROGRAMMER access >
Password for INFORM access >DICINF

Password for DOCUMENTATION access >
Password for REPORT access >DICRPT

Password for MANAGER access will be DICTMGR Listing of passwords.
There will be no PROGRAMMER access allowed
Password for INFORM access will be DICINF
There will be no DOCUMENTATION access allowed
Password for REPORT access will be DICRPT

Are the passwords correct? (Y/N) >y

USER PASSWORD >	If log-on passwords were used, enter them in response to these prompts.
ACCOUNT PASSWORD >	
GROUP PASSWORD >	
Dictionary store file on Tape or Disk(T/D) >d	Indicates a disk file; must be answered as in step one.
#J50	
END OF PROGRAM	
:	
FROM/<your account>/DICTINIT IS COMPLETE	The program is complete; your dictionary is now re-initialized.

DICTINIT Error Messages

The DICTINIT error messages are divided into three groups. The first group corresponds to step 1 (initialization). The second group corresponds to step 2 (customization). The third group corresponds to the final initialization job stream at the end of step 2, which builds your permanent Dictionary.

Group 1 — Initialization Startup Errors

The error messages you might receive during the initialization startup phase (while running `DICTINIT.PUB.SYS`) are as follows:

MESSAGE	EXPLANATION AND/OR ACTION
YOU DO NOT HAVE BATCH ACCESS CAPABILITY	In order to run the job streams that initialize the Dictionary, the user, group, and account must all have BA capability. Consult your system or account manager.
PLEASE PURGE *file* AND RERUN DICTINIT	*file* is one of the files MDIC, DICT, MDSCH. These files are used in the initialization process, and so cannot appear in your log-on group. You should either rename them or purge them.
DICTIONARY DATA-BASE DICT MUST EXIST TO RE-INITIALIZE	When re-initializing, the database file named DICT must exist in your logon group and account. Check to make sure you are in the correct group and account, then retry.
ERROR OCCURRED WRITING STREAM FILE	This message is preceded by an MPE file system error message. Refer to the *MPE 3000 Error Messages and Recovery Reference Manual*. The error prevented opening, writing, or closing the stream file which will perform the batch processing. Consult your system manager.
JOB COULD NOT START - INVALID PASSWORD	One of the passwords required for logon (account, user, or group) was either missing or incorrect. Check your passwords and respond again to the password prompts. Note that leading blanks are not allowed.
COMMAND ERROR = *nnnn* ERROR OCCURRED STREAMING JOB	*nnnn* is the number of a CIERR or CIWARN message. Consult the *MPE 3000 Error Messages and Recovery Reference Manual* for the full text. The error occurred while executing a STREAM command to start the phase 1 job which creates the temporary Dictionary. Consult your system manager.

MESSAGE	EXPLANATION AND/OR ACTION
FROM */your account file /file system error message* FROM */your account* / ERROR OCCURRED OPENING FILE MDSCH FROM */your account* / ERROR OCCURRED WRITING FILE MDSCH FROM */your account* / ERROR OCCURRED CLOSING FILE MDSCH	You will receive the first line, which is an MPE file system error message detailing the error, and one of the next three lines, which will tell you what DICTINIT was trying to do at that time. MDSCH is a schema file describing the temporary Dictionary. Consult the *MPE 3000 Error Messages and Recovery Manual* to resolve the MPE error, then rerun DICTINIT. These messages comes from the job that was invoked for you.

Group 2 — Customization Errors

The error messages you might receive during the customization phase (while running `DICTINIT.PUB.SYS,UPDATE`) are as follows:

MESSAGE	EXPLANATION AND/OR ACTION
YOU DO NOT HAVE BATCH ACCESS CAPABILITY	In order to run the job streams that initialize the Dictionary, your group must have BA capability. Consult your system or account manager.
*DB ERROR: *IMAGE database error message*	This error occurs while manipulating the temporary Dictionary. Consult your system manager if you receive one of these messages. The database error message corresponds to an IMAGE error message from the DBERROR facility. Consult the *IMAGE/3000 Reference Manual*.
ERROR OCCURRED OPENING MDIC	This is preceded by a *DB ERROR: message. This error may occur if phase two is run before the stream job from phase one completes. Wait to receive the message to run the UPDATE portion of DICTINIT.
ERROR IN INITIALIZING CAPACITIES FROM MDIC	This is preceded by a *DB ERROR: message. This occurs while reading default capacities in MDIC.
ERROR IN UPDATING CAPACITIES TO MDIC	This is preceded by a *DB ERROR: message. The error occurs after you respond that the capacities are correct, and the information is being transferred.
ERROR OCCURRED CLOSING MDIC	Consult your system manager.

MESSAGE	EXPLANATION AND/OR ACTION
CAPACITY MUST BE A NUMBER CAPACITY MUST BE LESS THAN 2,147,483,647 RESPONSE MUST BE 10 CHARACTERS OR LESS CAPACITY MAY NOT BE ZERO	All these messages come from invalid responses to the NEW CAPACITY prompt. (Note that the capacity should probably be MUCH less than 2,147,483,647.)
THE DATA SET NAME DOES NOT EXIST	This comes from an invalid (or misspelled) data set name in response to the DATA-SET NAME (DONE/ALL/name) prompt.
YOU MUST ENTER A PASSWORD FOR MANAGER ACCESS	All the other security levels may be deleted by pressing RETURN in response to the PASSWORD prompt EXCEPT manager, which must have a password.
PASSWORD MUST BE 8 CHARACTERS OR LESS PASSWORD MAY NOT CONTAIN A SEMICOLON (;)	These messages are the result of invalid responses to a PASSWORD prompt. Reenter the PASSWORD.
UNEXPECTED ERROR: MANAGER PASSWORD IS NULL, PLEASE REENTER THE PASSWORDS	Reenter all the passwords.
ERROR OCCURRED UPDATING MDIC WITH NEW SECURITY	This is preceded by a *DB ERROR: message. Please consult your system manager before proceeding.
DISK STORE FILE MDSCH NOT FOUND	You responded D for the re-initialization prompt for tape or disk. In that case, the file MDSCH must be present, and must contain the contents of the old data-Dictionary. Either respond T, if on tape, or locate MDSCH.
JOB COULD NOT START - INVALID PASSWORD	One of the passwords required for logon (user, account, or group) was missing or incorrect. Check your passwords and respond again to the password prompts. Note that leading blanks are not allowed.
FILE ERROR: *MPE file system error message* ERROR OCCURRED WRITING STREAM FILE	This message is preceded by an MPE file system error message. See the *MPE 3000 Error Messages and Recovery Reference Manual*. The error prevented opening, writing, or closing the stream file which will perform the batch processing. Consult your system manager.
COMMAND ERROR = *nnnn* ERROR OCCURRED STREAMING JOB	*nnnn* is the number of a CIERR or CIWARN message. Consult the *MPE 3000 Error Messages and Recovery Reference Manual* for the full text. The error occurred while executing the STREAM command to start the phase 1 job, which creates the temporary Dictionary. Consult your system manager.

Group 3 — Final Initialization Errors

The error messages you might receive during the final initialization job at the end of phase 2 are as follows:

MESSAGE	EXPLANATION AND/OR ACTION
FROM/*your account* / #DB ERROR: *IMAGE database error message* FROM/*your account* / ERROR OCCURRED OPENING MDIC FROM/*your account* / ERROR OCCURRED READING CLASS *nn* FROM/*your account* / ERROR OCCURRED REINITIALIZING CLASS *nn* FROM/*your account* / ERROR OCCURRED CLOSING MDIC	You will receive the first line, followed by one of the next four lines. *nn* is the number of a security class entry in MDIC. Please consult your system manager before proceeding. These messages come from the job that was invoked for you.
FROM/your account / *DB ERROR: *IMAGE database error message* FROM/*your account* / ERROR OCCURRED OPENING DICT FROM/*your account* / ERROR OCCURRED INITIALIZING DIC-CONTROL FROM/*your account* / ERROR OCCURRED	The first message will appear, followed by one of the next six (*nn* is the number of a security class). Please consult your system manager before proceeding. These messages come from the job that was invoked for you.
INITIALIZING DATA-GROUP FROM/*your account* / ERROR OCCURRED DELETING A CLASS FROM/*your account* / ERROR OCCURRED INITIALIZING CLASS *nn* FROM/*your account* / ERROR OCCURRED CLOSING DICT	

D How HP INFORM Links Files to Generate Reports

When generating reports using HP Inform/3000, the user selects the desired elements from HP Inform/3000's Data Names Menu. If the user has chosen to report from elements organized into an HP Inform/3000 group, these elements may exist in various files - MPE files, KSAM files, or IMAGE data sets. HP Inform/3000 needs to be able to link these files so that the desired lines of the report can be generated.

It is important for the database administrator to understand how HP Inform/3000 links the various files. The linking process determines whether it is possible to generate the requested report and, if so, what specific data will be printed in the report.

Common elements (elements that exist in more than one file) are used to build links between the files which contain elements needed for the report. An element that is used in this way is called a “link element”.

Before reading this appendix, you should read Using Commands to Define HP Inform/3000 Groups in Section III.

NOTE An element which has been defined as a CHILD element through the RELATE ELEMENT command string can not be used as a link.

Direct Links

A link between two files, each of which contains the values of one or more elements needed for a report, is called a “direct link”.

When two files are linked, the value of the link element is retrieved from the first file and is used to determine the correct record to retrieve from the second file. If the link is from an IMAGE master set, IMAGE detail set, KSAM file, or MPE file to an MPE file, the link element must simply exist in both files; if the file you are linking to, however, is an IMAGE master, IMAGE detail, or KSAM file, there are additional requirements on the link element. These are:

1. If the file you are linking to is a KSAM file, the link element must be a common element (must exist in both files) and must be a key item in the KSAM file. Note that when the link is from a KSAM file to another KSAM file, the link element need only be a key item in the second file accessed.
2. If the file you are linking to is an IMAGE detail set, the link element must be a common element and must be a search item in the detail set. These are the only requirements even when the link is from one database to another. Note that when the link is from a detail set to another detail set, the link element need only be a search item in the second file accessed.
3. If the file you are linking to is an IMAGE master set, the link element must be a common element and must be a search item in the master set. These are the only requirements even when the link is from one database to another. Note that when the link is from a master set to another master set, the link element need only be a search item in the second file accessed. (There are two additional ways that a data set can be linked to a master set as long as the link is within one database; these are explained below under Indirect Links.)

The following table summarizes the above requirements for direct links according to what type of file is being linked to. Note that the file you are linking from can be any type - MPE, KSAM, IMAGE detail, or IMAGE master.

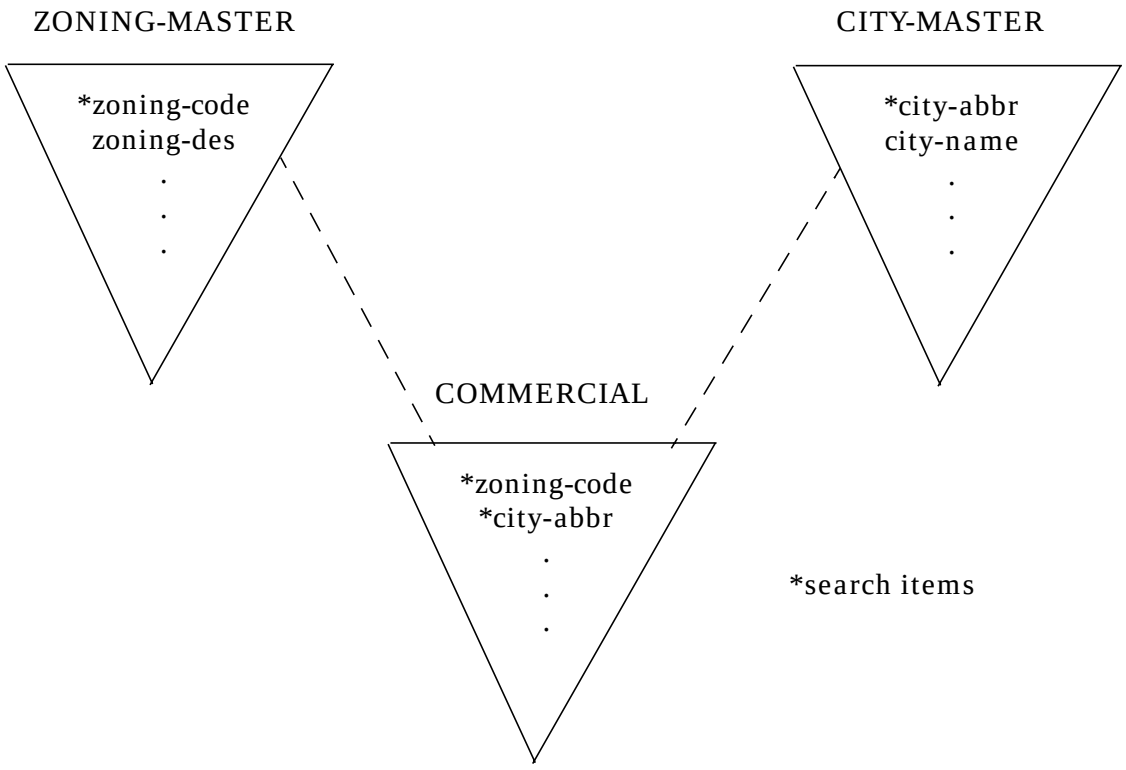
FILE BEING LINKED TO	REQUIREMENT ON LINK ELEMENT
MPE	Common element.
KSAM	Common element which is a key item in the KSAM file being linked to.
DETAIL	Common element which is a search item in the DETAIL data set being linked to.
MASTER	Common element which is a search item in the MASTER data set being linked to.

EXAMPLE 1 at the end of this appendix demonstrates direct links from an MPE file to a KSAM file and from an MPE file to an IMAGE detail set.

Indirect Links

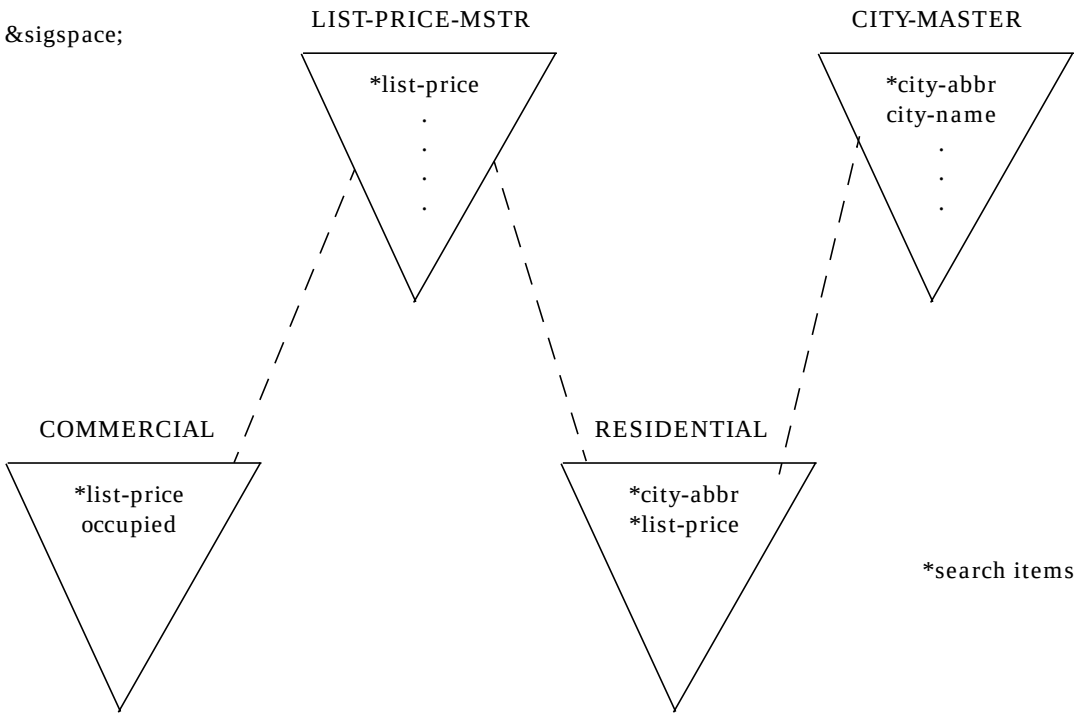
If two files containing elements needed for a report are linked through other files which contain no elements needed for the report, the link is called an “indirect link”. There are only two indirect links which HP Inform/3000 allows and both apply only to data sets within the same database. These links are the following:

1. If a master set needs to be linked to another master, a link can be made by going through a detail which has a search item in common with each of the masters. For example, assume the following is a partial representation of a data base:



Suppose the report requests the data element “zoning-des” from the master set ZONING-MASTER and the data element “city-name” from the master set CITY-MASTER. An indirect link can be made by going through the detail set COMMERCIAL. The master set ZONING-MASTER has the search item “zoning-code” which forms an IMAGE path to the detail set COMMERCIAL, and the master set CITY-MASTER has the search item “city-abbr” which also forms an IMAGE path to COMMERCIAL.

2. If a detail set needs to be linked to a master set, a link can be made by going through a second detail which has a search item in common with the first detail (forming IMAGE paths to a common master) and a second search item forming an IMAGE path to the desired master. For example, assume the following is a partial representation of a database:



Suppose the report requests the data element “occupied” from the detail set COMMERCIAL and the data element “city-name” from the master set CITY-MASTER. An indirect link can be formed by going through the detail set RESIDENTIAL. The detail set COMMERCIAL and the detail set RESIDENTIAL both have the search item “list-price” (which forms IMAGE paths to the common master set LIST-PRICE-MSTR); the detail set RESIDENTIAL has a second search item “city-abbr” which forms an IMAGE path to the master set CITY-MASTER.

As stated earlier, these are the only two indirect links which HP Inform/3000 allows and both apply only to data sets within one database.

The Driving File

The driving file is the file that is accessed first. Since at least one line of the report will be generated for each record retrieved from the driving file, the contents of a report could be quite different if the driving file is different. Files which meet the following criteria are considered:

1. the file must contain an element having a positive link value (greater than zero) which may or may not be included in the report, and
2. the file must contain an element to be included in the report.

   One, none, or more than one file might meet the above criteria:

   - If only one file meets these two criteria, then it will be chosen as the driving file.

- If no files meet these two criteria, then files which contain an element to be included in the report are considered in the following order:

  MPE files

  KSAM files

  IMAGE detail data sets

  IMAGE master data sets

If more than one file exists in the highest possible category, the file with the most elements to be included in the report is chosen as the driving file. If there are two such files or more, one of them is arbitrarily chosen.

- If more than one file meets these two criteria, the file which has the highest priority link value specified (lowest positive integer) is chosen as the driving file. If more than one of these files has the highest priority link value, then the driving file is chosen from among them as in the above second case.

NOTE The driving file will always be read serially. It is desirable to use an MPE file as the driving file whenever there is one which contains an element to be included in the report; this is because HP Inform/3000 must perform a serial read every time it accesses an MPE file to retrieve a value for a line of the report. If the MPE file is the driving file, this serial read is done only once; if it is not, then a serial read is done for each value of the element used to link to the MPE file. This is not true for KSAM files or IMAGE data sets.

Hence, to ensure that a particular file is the driving file whenever an element which exists in that file is requested for a report, add an element from that file to the group, specify the name of that file in response to the FILE prompt, and specify a positive link value in response to the VALUE AS A LINK prompt. The element used should be one which allows HP Inform/3000 to link directly with other files to obtain as many elements from the group as possible. The file you choose to specify in this manner should:

1. be an MPE file (see the NOTE earlier) if the group contains elements which only exist in that file or if the MPE file is needed by HP Inform/3000 to link directly to other files;
2. contain the appropriate subset of data desired for reports which will be generated from that group.

If more than one file satisfies (1) or (2), specify a link value of 1 for the file that contains the greatest number of elements from the group, 2 for the file that contains the next greatest number of elements from the group, and so on. Note that the more specific or limited a group is (that is, the less varied the kinds of reports that will be generated from it are), the more likely it is that a file which meets (2), above, can be specified.

EXAMPLE 1 at the end of this appendix demonstrates choosing a file which meets (1), above. EXAMPLE 2 demonstrates choosing a file which meets (2), above.

Linking Files

All elements in a group that have been assigned a positive link value form a prioritized list for use by HP Inform/3000's access algorithm when linking files. Elements which have been added to a group which will not be displayed on HP Inform/3000's Data Names Menu are included in this list if they have been assigned a positive link value. The following steps outline how HP Inform/3000 links files. If, at any particular step, all files containing elements needed for the report are linked, the steps which follow that one do not occur.

1. HP Inform/3000 first tries to use the element in the group with the highest priority link value (lowest positive integer) to directly link the driving file with any other files containing elements needed for the report.
2. HP Inform/3000 next tries to use the element in the group with the second highest priority link value to directly link those files linked in step (1) - that is, the driving file and any files which were successfully linked to it - with any other files containing elements needed for the report.
3. Next, the element in the group with the next highest priority link value is used to try to directly link those files linked in steps (1) and (2) with any other files containing elements needed for the report; and so on, until the prioritized list of elements with a positive link value is exhausted.
4. HP Inform/3000 next tries to use elements in the group with a link value of zero (the default) to directly link those files linked in steps (1), (2), and (3) with any other files containing elements needed for the report. The elements with a link value of zero are tried in an arbitrary order.
5. If all the files have still not been linked successfully, HP Inform/3000 tries to use elements which are not in the group to directly link any of the files linked in steps (1), (2), (3), and (4) with any of the remaining files. As always, a direct link is successful if the element exists in both files and if any additional requirements on the file being linked to are met (see “Direct Links” earlier in this appendix).
6. If still no link can be made and if the files HP Inform/3000 is attempting to link are IMAGE data sets, HP Inform/3000 will try the two allowable indirect links. If this fails to link all the needed files, the report can not be generated and the HP Inform/3000 user will receive an error message.

Default File Access

HP Inform/3000 provides defaults designed to maximize performance of the file accesses necessary during report production. One such default, which was explained in the discussion on the driving file, is that an MPE file will be used as the driving file whenever possible. In addition, if file names (in response to the FILE prompt) and link values that override the defaults have not been specified, HP Inform/3000 will try to follow these guidelines:

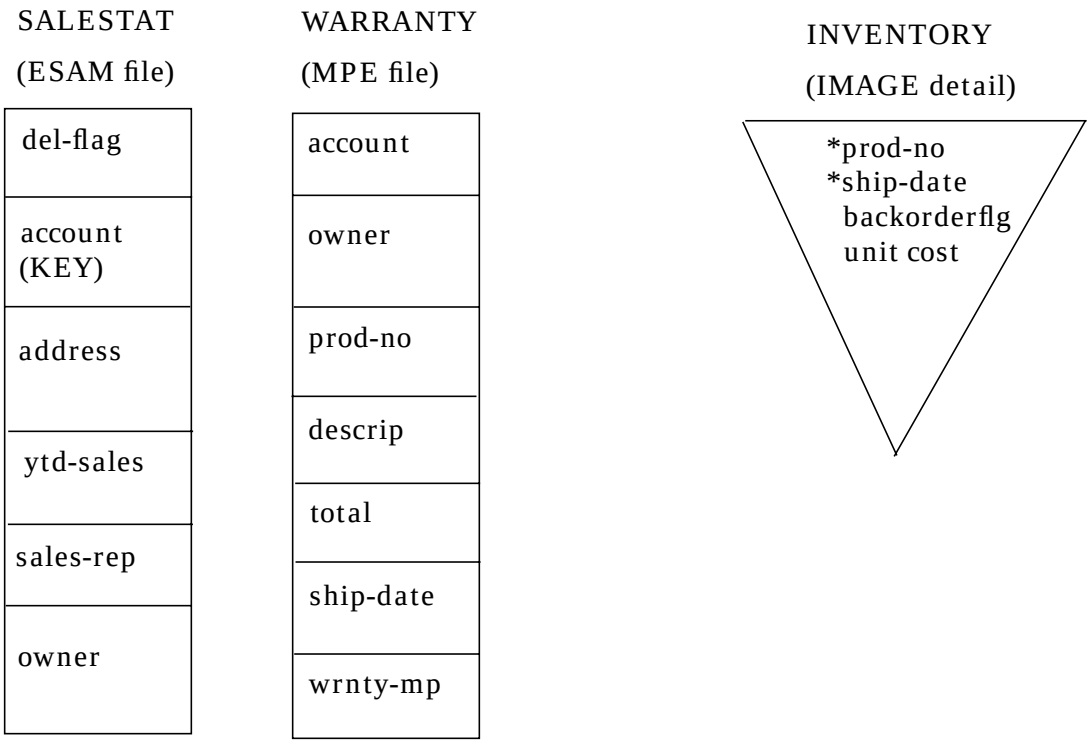
- Involve as few files as possible in the access.
- Access a KSAM file in preference to a database (since there is less overhead involved).
- Open as few databases as possible.
- Use master data sets whenever possible (since they will need to be accessed only once per key value, while detail data sets and KSAM files may require several accesses per key value).

Note that by specifying a file name in response to the FILE prompt, HP Inform/3000's access algorithm may not work efficiently and possibly not at all. It is best to not specify a file name in response to the FILE prompt unless necessary to ensure that the correct values of an element are retrieved. If a file name is specified when it is not necessary to do so, it is possible that more files will be accessed than HP Inform/3000 would otherwise access. In the worst case, it may result in HP Inform/3000 being unable to create a link - when there is no link element to the designated file.

HP Inform/3000 uses link elements to accomplish the above goals; the link elements are used in the order which produces maximum performance. The use of link values assigned to elements allows you to override this - to control which elements are tried to link files and in what order. However, the ability of HP Inform/3000 to maximize performance may be hampered by the improper use of link values.

Example

Assume the following files:



Suppose we wish to create a group called ACCOUNT ORDERS. The purpose of the ACCOUNT ORDERS group is to generate reports about the orders of each account. The group consists of the data elements:

account
address
owner
prod-no
unit-cost

By accessing the files SALESTAT and INVENTORY, we could obtain all the elements in the group; however, it is not possible to link these two files since there is no element which exists in both of them. The MPE file, however, also contains elements from the group and, in particular, contains the element “account” which exists in the KSAM file and which is a key item in the KSAM file; hence a direct link can be made from WARRANTY to SALESTAT. In addition, the MPE file contains two elements (“prod-no” and “ship-date”) which also exist in the IMAGE detail set and which are search items in the IMAGE detail; hence a direct link can be made from WARRANTY to INVENTORY.

Since the MPE file allows us to directly link to the two other files and since it is preferable to access an MPE file first, we want to make the MPE file the driving file. To accomplish this, we need to specify the file name WARRANTY in response to the FILE prompt and a link value of 1 in response to the VALUE AS A LINK prompt when adding an element from the file WARRANTY to the group. We would like to pick an element which will allow

HP Inform/3000 to link WARRANTY directly with other files to retrieve as many elements from the group as possible.

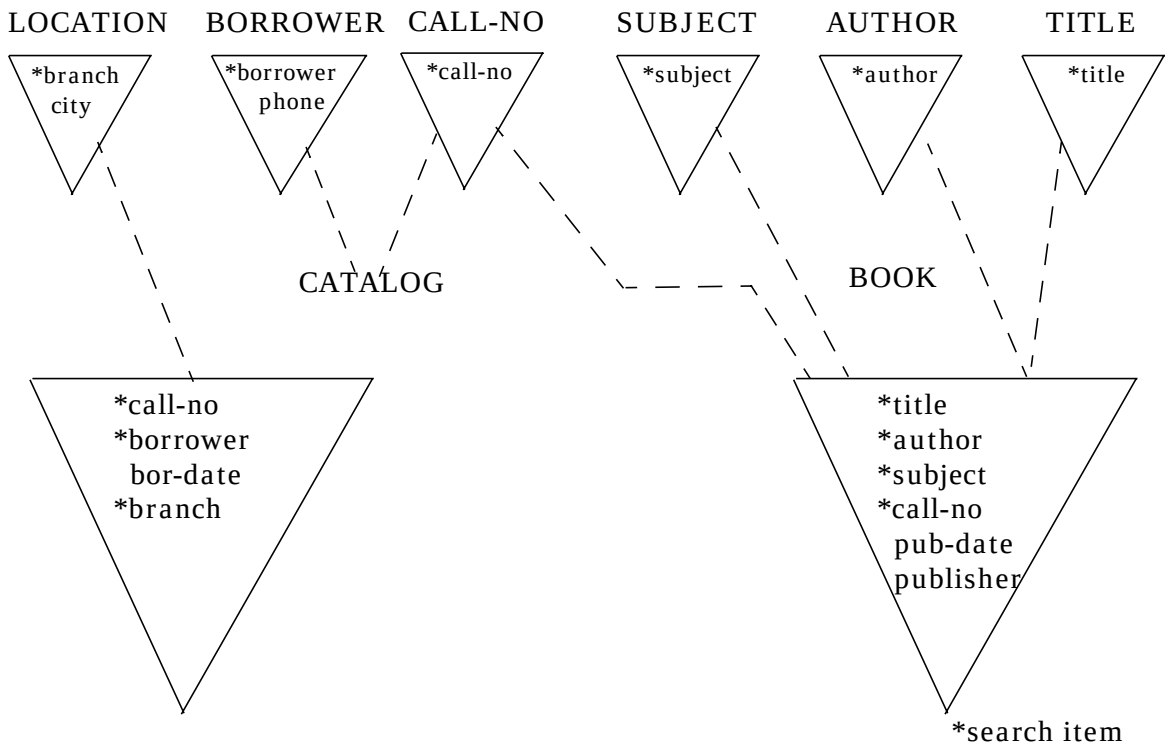
The elements “account”, “prod-no”, and “ship-date” are our three possibilities. Since “ship-date” is not in the group, we will use “account” or “prod-no”. The data element “account” links WARRANTY and SALESTAT, where “address” can be retrieved; the data element “prod-no” links WARRANTY and INVENTORY, where “unit-cost” can be retrieved. (Note that the values for “account”, “owner”, and “prod-no” will all be retrieved from the MPE file since it is accessed first.) Since both data elements allow us to link to one other file where one required element can be retrieved, it does not matter which one we use.

After linking WARRANTY with SALESTAT through the element “account”, HP Inform/3000 will try to use the element in the group with the next highest priority link value to directly link WARRANTY or SALESTAT with INVENTORY - assuming “prod-no” or “unit-cost” was requested for the report. To make this most efficient, we can give “prod-no” a link value of 2.

Thus, we could add elements to the ACCOUNT ORDERS group as follows:

```
>ADD GROUP
                  GROUP> account orders      <
                ELEMENT> unit-cost           <
          ELEMENT ALIAS> cost of unit        <
                   FILE>!                    <
                ELEMENT> address             <
          ELEMENT ALIAS> !                   <
                ELEMENT> owner               <
          ELEMENT ALIAS> !                   <
                ELEMENT> account             <
          ELEMENT ALIAS>                     <
                   FILE> warranty            <
       VALUE AS A LINK> 1    <
DISPLAY ELEMENT ON MENU (Y/N)?>!
                ELEMENT> prod-no             <
          ELEMENT ALIAS> product number      <
                   FILE>                     <
       VALUE AS A LINK> 2    <
DISPLAY ELEMENT ON MENU (Y/N)?>!
                ELEMENT>                     <
```

Assume the following database:



Suppose we wish to create a group called BOOKS OUT. The purpose of the BOOKS OUT group is to generate reports about books that are checked out of the library. The group consists of the data elements:

title
author
call-no
borrower
phone

Since the purpose of the BOOKS OUT group is to report on books that are checked out, it makes sense to access the detail set CATALOG as the first file; since CATALOG has only one entry for every book checked out, there will be at least one line of the report for each book that is checked out of the library. That is, CATALOG contains the appropriate subset of data which is desired for reports which will be generated from the BOOKS OUT group. BOOK, TITLE, and CALL-NO, on the other hand, have one entry for each book in the library; if one of these files is the first file accessed, there will be at least one line of the report for every book in the library. Accessing BORROWER or AUTHOR as the first file would also not be satisfactory, since BORROWER has one entry for each person registered to check out books from the library and AUTHOR has one entry for each author. (SUBJECT and LOCATION are not considered since they do not contain any elements from the group.)

To make CATALOG the driving file, we need to specify the file name CATALOG in response to the FILE prompt and a link value of 1 in response to the VALUE AS A LINK

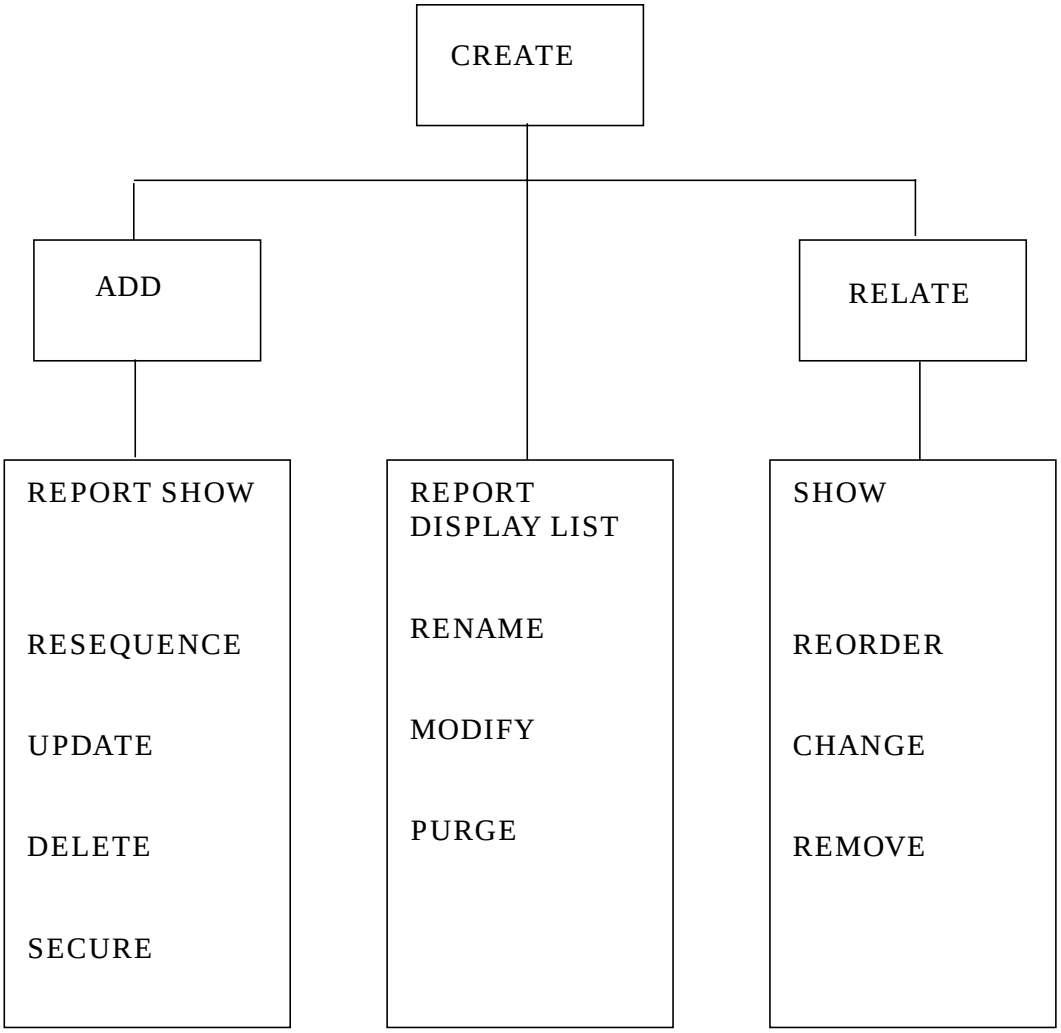
prompt when adding an element from the file CATALOG to the group. We would like to pick an element which will allow HP Inform/3000 to link CATALOG directly with other files to retrieve as many elements from the group as possible. The elements “call-no” and “borrower” are our two possibilities since they are search items. The data element “call-no” links CATALOG and BOOK, where “title” and “author” can be retrieved; the data element “borrower” links CATALOG and BORROWER, where only “phone” can be retrieved. Hence, we pick the element “call-no”.

After linking CATALOG and BOOK through the element “call-no”, HP Inform/3000 will try to use the element in the group with the next highest priority link value to directly link CATALOG or BOOK with BORROWER - assuming “phone” was requested for the report. To make this most efficient, we can give “borrower” a link value of 2. Thus, we could add elements to the BOOKS OUT group as follows:

```
>ADD GROUP
                  GROUP> books out           <
                ELEMENT> title               <
          ELEMENT ALIAS> !                   <
                ELEMENT> author              <
          ELEMENT ALIAS> !                   <
                ELEMENT> phone               <
          ELEMENT ALIAS> phone number        <
                   FILE> !                   <
                ELEMENT> call-no             <
          ELEMENT ALIAS> call number         <
                   FILE> catalog             <
       VALUE AS A LINK> 1    <
DISPLAY ELEMENT ON MENU (Y/N)?>!
                ELEMENT> borrower            <
          ELEMENT ALIAS>                     <
                   FILE>                     <
       VALUE AS A LINK> 2    <
DISPLAY ELEMENT ON MENU (Y/N)?>!
                ELEMENT>                     <
```

E DICTDBM Quick Reference

The following chart indicates the order in which the data dictionary commands can be used.



Command/Subcommand Guide

This is a quick reference guide for each command which shows what subcommands can be used and what the command/subcommand does.

ADD

CATEGORY	adds a data element to a category
CLASS	adds a data element to a user security class
CLASS-FILE	adds a file to a user security class
CLASS-GROUP	adds a group to a user security class
FILE	adds a data element to a file
FILE-LOC	adds a file to a physical location
GROUP	adds a data element to an HP Inform/3000 group
PROCEDURE	adds a data element to a procedure
PROCEDURE-LOC	adds a procedure to a physical location

CHANGE

CATEGORY	changes a category to category description
CLASS	changes a class to class description
ELEMENT	changes an element to element entry and/or description
FILE	changes a file to file entry and/or description
GROUP	changes an HP Inform/3000 group to group description
PROCEDURE	changes a procedure to procedure description

CREATE

CATEGORY	creates a new category
CLASS	creates a new user security class
ELEMENT	creates a new data element
FILE	creates a new file
GROUP	creates a new HP Inform/3000 group
LOCATION	creates a new physical location
PROCEDURE	creates a new procedure

DELETE

CATEGORY	deletes an element to category association
CLASS	deletes an element to class association
CLASS-FILE	deletes a file to class association
CLASS-GROUP	deletes a group to class association
FILE	deletes an element to file association
FILE-LOC	deletes a file to location association
GROUP	deletes an element to HP Inform/3000 group association
PROCEDURE	deletes an element to procedure association
PROCEDURE-LOC	deletes a procedure to location association

DISPLAY

CATEGORY	displays attribute information for a category plus all the directly related categories
CLASS	displays attribute information for a user security class
ELEMENT	displays attribute information for an element plus the directly associated and/or directly related entities
FILE	displays attribute information for a file plus the directly associated and/or directly related entities
GROUP	displays attribute information for an HP Inform/3000 group plus the directly related entities
LOCATION	displays attribute information for a physical location
PROCEDURE	displays attribute information for a procedure plus the directly associated and/or directly related entities

HELP

Provides brief description of any command and its sub-categories. HELP sub-categories are DICTDBM command names.

LIST

CATEGORY	lists the categories
CLASS	lists the security classes
ELEMENT	lists the data elements
FILE	lists the files
GROUP	lists the HP Inform/3000 groups
LOCATION	lists the physical locations
PROCEDURE	lists the procedures

MODIFY

CATEGORY	modifies a category
CLASS	modifies a security class
ELEMENT	modifies a data element
FILE	modifies a file
GROUP	modifies an HP Inform/3000 group
LOCATION	modifies a physical location
PROCEDURE	modifies a procedure

PURGE

CATEGORY	purges the category and all references to it
CLASS	purges the security class and all references to it
ELEMENT	purges the data element and all references to it
FILE	purges the file and all references to it, with the option to purge related files and associated elements.
GROUP	purges the HP Inform/3000 group and all references to it
LOCATION	purges the location and all references to it
PROCEDURE	purges the procedure and all references to it

RELATE

CATEGORY	relates a category to a category
CLASS	relates a class to a class
ELEMENT	relates a data element to a data element
FILE	relates a file to a file
GROUP	relates an HP Inform/3000 group to a group
PROCEDURE	relates a procedure to a procedure

REMOVE

CATEGORY	removes the relationship between a child and a parent category
CLASS	removes the relationship between a child and a parent class
ELEMENT	removes the relationship between a child and a parent element
FILE	removes the relationship between a child and a parent file
GROUP	removes the relationship between a child and a parent HP Inform/3000 group

PROCEDURE	removes the relationship between a child and a parent procedure

RENAME

CATEGORY	renames an existing category
CLASS	changes the existing identifying number of a security class
ELEMENT	renames an existing data element
FILE	renames an existing file
GROUP	renames an existing HP Inform/3000 group
LOCATION	renames an existing physical location
PROCEDURE	renames an existing procedure

REORDER

CATEGORY	reorders a child category within a parent category's entry list
ELEMENT	reorders a child data element within a parent data element's entry list
FILE	reorders a child file within a parent file's entry list
GROUP	reorders a child HP Inform/3000 group within a parent group's entry list
PROCEDURE	reorders a child procedure within a parent procedure's entry list

REPORT

CATEGORY	reports all the data elements associated with a category or with related child categories
CLASS	reports all the data elements associated with a security class
ELEMENT	reports all the data elements defined in the dictionary
FILE	reports all the data elements associated with a file or with related child files
GROUP	reports all the data elements associated with an HP Inform/3000 group or with related child groups
PROCEDURE	reports all the data elements associated with a procedure or with related child procedures, including any owned by the named procedure

RESEQUENCE

CATEGORY	resequences a data element within a category
CLASS	resequences a data element within a security class
FILE	resequences a data element within a file
GROUP	resequences a data element within an HP Inform/3000 group
PROCEDURE	resequences a data element within a procedure

SECURE

FILE	secures all data elements in a file and the file itself to a user security class
GROUP	secures a group to a user security class

SHOW

CATEGORY	shows all the relationships for a category plus associated data elements where applicable
CLASS	shows all the associations for a user security class
ELEMENT	shows all the relationships for a data element
FILE	shows all the relationships and links for a file plus associated data elements where applicable
GROUP	shows all the relationships for an HP Inform/3000 group plus associated data elements where applicable and access information for the group
LOCATION	shows all the associations for a location
PROCEDURE	shows all the relationships for a procedure plus associated data elements where applicable

UPDATE

CATEGORY	updates a data element to a category association
CLASS	updates a data element to a user security class association
CLASS-FILE	updates a file to a user security class association
CLASS-GROUP	updates a group to a user security class association
FILE	updates a data element to a file association&"
FILE-LOC	updates a file to a location association
GROUP	updates a data element to an HP Inform/3000 group association
PROCEDURE	updates a data element to a procedure association
PROCEDURE-LOC	updates a procedure to a location association

Glossary

A

access capability The type of access to be given to a user security class for a file or element, as follows:
R — allows the user identified by the security class to locate and read a file or an element;
U — allows the user identified by the security class to read and update values in a file or an element, except search or sort elements;
M — allows the user identified by the security class to read, update, add to, and delete from a file or an element;
X — allows only the creator to access the data set or data element;

account The name of an MPE account. (See the MPE Commands Reference manual)

alias An optional name you assign to an already defined entity for local use in programs, groups, and files.

association Ties together unlike entities such as an element and a file.

AUTO An IMAGE automatic master data set which contains only one data item - the search item - and is related to one or more IMAGE detail data sets.

B

BASE detail data sets.

blocking factor The number of records per block; a block is the smallest unit of data transferred by the file system

byte position The starting position for a data element within a parent data element.

C

category The name used to identify the natural relationships within an organization or a business.

child An entity which is subordinately related to another entity of the same type, such as:

CHILD CATEGORY a category that is subordinately related to another category;

CHILD ELEMENT a data element that is subordinately related to another data element;

CHILD FILE a file that is subordinately related to another file;

CHILD GROUP a group that is subordinately related to another group;

class A numeric value which can be assigned to data elements for the purpose of defining who may or may not access the file.

class-file A data dictionary entity to which data files can be assigned in order to define who may or may not have access to those files.

count The numeric value used to identify the number of sub-elements in a compound element.

D

decimal The number of decimal digits to allow for a numeric type data element.

DETL An IMAGE detail data set.

direct link A link between two files, where each file contains the values of one or more elements needed by HP Inform/3000 for a report.

driving file The file that HP Inform/3000 accesses first when linking files in order to generate a report; one line of the report is generated for each record retrieved from the driving file.

E

edit mask Text used to edit the value of data elements before displaying them.

element The smallest accessible unit of data in a database or file; the same as an 'item' in IMAGE/3000.

entry text The textual information used by HP Inform/3000, Report/3000, and Transact/3000 when prompting for input; if no entry text has been specified, the primary name is used in the prompt.

F

file A collection of logically related data elements; may be an IMAGE database or set, or a KSAM, MPE, or VPLUS file.

file-loc A data dictionary entity used to document the physical location of a file.

file size The maximum number of records a file is allows to contain.

form The name used to identify a form for a VPLUS forms file.

G

GROUP A collection of data elements in the data dictionary, established for the users of HP Inform/3000.

H

heading text The textual heading used by HP Inform/3000, Report/3000, and

Transact/3000 for labeling data elements in reports; if no heading text has been specified, the primary name is used

I

indirect link A link between two files, each containing elements needed by HP

Inform/3000 for a report, which is formed by linking through one or more other files which do not contain elements needed for the report.

K

key element Used to identify a data element which is the key in a KSAM file

KSAM Refers to a KSAM file. (See the KSAM/3000 Reference Manual.)

L

language The name of the implementation language used to write a program, procedure, or subroutine.

level The identifying number used to identify the position of an entity

within the hierarchical structure established with the RELATE command; 1 is the top of the structure.

link value A -1, 0, or positive integer assigned to a data element which is used to specify which elements should preferably be selected by HP Inform/3000 to link the desired elements into a logical record for reporting.

location A data dictionary entity which is used to document the physical location of data.

lockword Used to provide access to an MPE or KSAM file. (See the MPE Commands Reference Manual.)

long name A fuller textual name used to identify an entity.

M

MAST An IMAGE manual master data set.

measurement units Identifies the unit of measure of a data element, such as days, lbs., etc.

MPEF An MPE file.

P

parent The name of an entity in a hierarchical structure which owns child entities.

password Used to provide access to a file through a security system; also, dictionary password - allows a certain level access to a data dictionary.

primary The name given an entity when it is first created in the data dictionary; as opposed to an alias.

procedure The name used to identify a system, program, or routine which is recorded in the data dictionary.

procedure-loc A data dictionary entity which is used to document the physical location of a procedure.

R

relationship Ties together like entities such as a child file and a parent file.

responsibility The name of the person, area, or department that is responsible for the integrity of an entity.

RETURN A carriage return/line feed.

S

search element Used to identify an element which is a search item in an IMAGE master or detail data set

T

type The attribute for a data element (X, I, etc.), file type for a file (VPLS, MAST, KSAM, etc.), or a user-definable 4-character definition for other entities.

V

VPLS Refers to a VPLUS forms file.