DATATRIEVE

DATATRIEVE-11
Reference Manual

Order No. AA-U049A-TK

digital software

DATATRIEVE-11
Reference Manual

Order No. AA-U049A-TK

September 1983

This manual contains reference information for DATATRIEVE-11.

OPERATING SYSTEM AND VERSION:	RSX-11M RSX-11M PLUS RSTS/E
SOFTWARE VERSION:	DATATRIEVE-11 V3

digital equipment corporation, maynard, massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

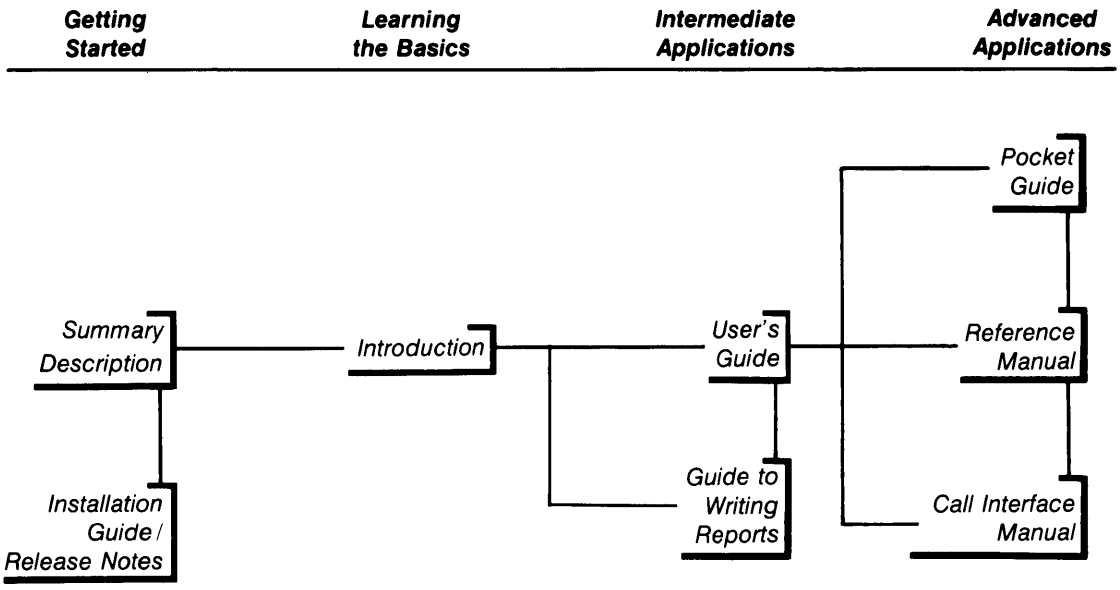
No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage-paid READER'S COMMENTS form on the last page of this document requests your critical evaluation to assist us in preparing future documentation.

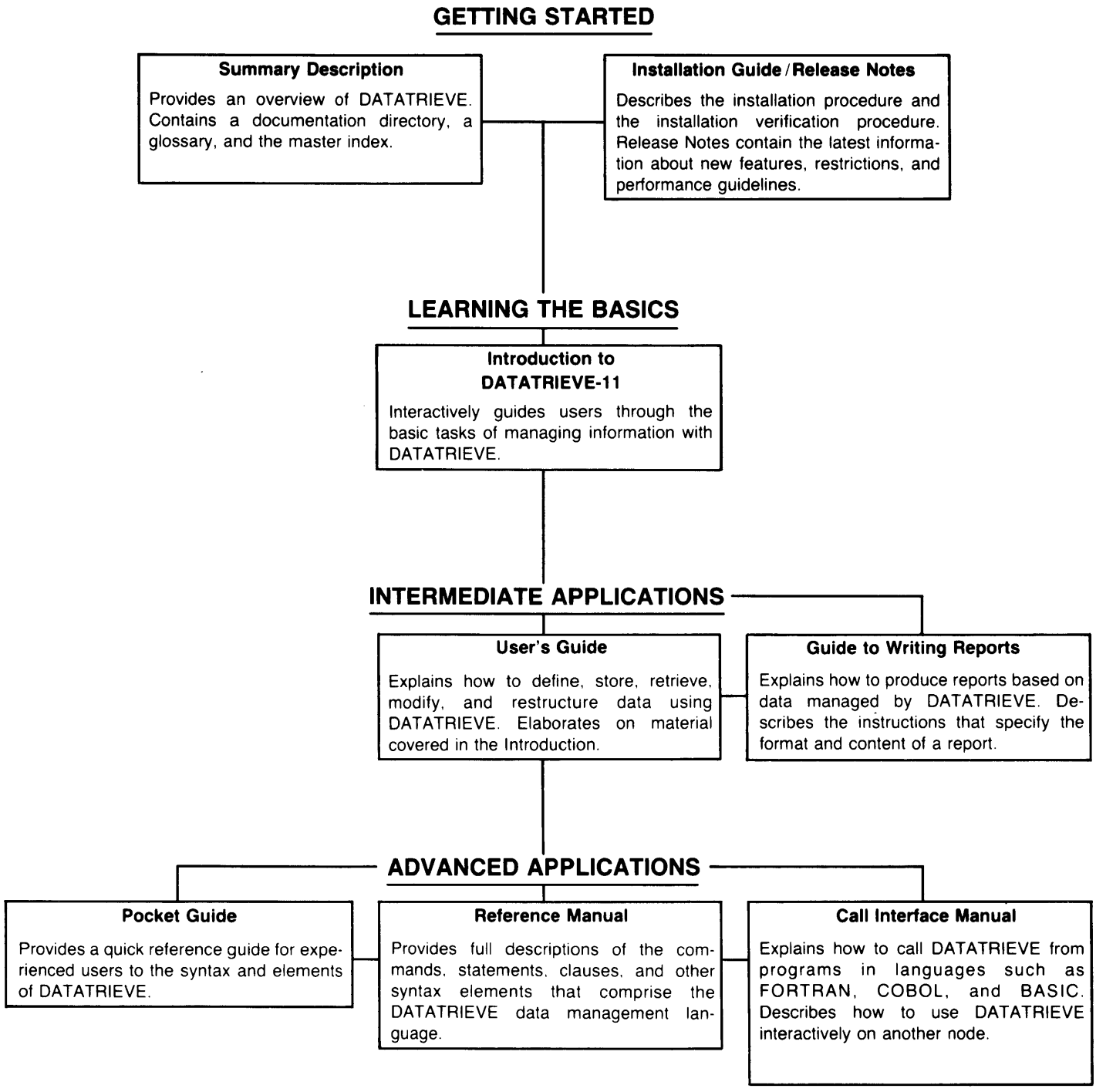
The following are trademarks of Digital Equipment Corporation:

digital™	DECwriter	RSTS
	DIBOL	RSX
MASSBUS	UNIBUS	DECmate
PDP	VAX	DECsystem-10
P/OS	VMS	DECSYSTEM-20
Professional	VT	DECUS
DEC	Rainbow	Work Processor

DATATRIEVE-11 DOCUMENTATION MAP



DATATRIEVE-11 DOCUMENTATION DESCRIPTION



Contents

How to Use This Manual

Use this manual to find reference information on DATATRIEVE-11 terms, concepts, syntax elements, commands, statements, and clauses.

Intended Audience

This manual assumes that you have prior experience with DATATRIEVE-11, are familiar with your operating system, and understand the basic concepts of data processing.

Structure

This manual contains five chapters and three appendixes:

Chapter 1	Presents the basic terms, concepts, and components of DATATRIEVE-11
Chapter 2	Describes DATATRIEVE value expressions and Boolean expressions
Chapter 3	Describes the record selection expression
Chapter 4	Describes field and record definition clauses
Chapter 5	Describes, in alphabetical order, all DATATRIEVE-11 commands, statements, and definition clauses
Appendix A	Lists DATATRIEVE-11 keywords
Appendix B	Lists ASCII values for printing characters
Appendix C	Summarizes DATATRIEVE-11 syntax formats

Related Manuals

For more information about the subjects discussed in this book, consult the following manuals in the DATATRIEVE-11 documentation set:

DATATRIEVE-11 Introduction

DATATRIEVE-11 User's Guide

DATATRIEVE-11 Guide to Writing Reports

DATATRIEVE-11 Call Interface

DATATRIEVE-11 Installation Guide/Release Notes

Conventions

Examples of DATATRIEVE commands and statements are printed in a dot matrix typeface. The DATATRIEVE or VAX/VMS output lines displayed on your terminal are printed in black. The commands and statements you enter from your terminal are printed in red.

Symbols and conventions used in syntax formats:

UPPERCASE WORDS	Uppercase words in syntax formats are DATATRIEVE keywords.
lowercase words	Lowercase words in syntax formats indicate entries you must provide.
{}	Braces in syntax formats mean you must choose one, but no more than one, of the enclosed entries.
[ ]	Brackets in syntax formats mean you have the option of choosing one, but no more than one, of the enclosed entries.
...	A horizontal ellipsis in syntax formats means you have the option of repeating the preceding element.

1 Introduction to the DATATRIEVE Language

1.1 Introduction

DATATRIEVE simplifies the tasks of defining, managing, and retrieving data. This manual is the primary source for information on the structure of this language. This chapter presents the basic concepts and terms of DATATRIEVE and describes the components of DATATRIEVE-11.

1.2 Starting and Ending a DATATRIEVE Session

When you run DATATRIEVE-11, you are running an interactive program that accepts input from the terminal and uses the terminal as the default output device.

To start a DATATRIEVE session, begin at the operating system command level and type DTR, RUN $DTR, or the symbol defined for DATATRIEVE-11 at installation. DATATRIEVE displays a startup banner that indicates you have successfully invoked DATATRIEVE. Note that the startup banner should look like this, except for the date:

```
DATATRIEVE-11, Query and Report System
Version V3.00 01-SEP-83
Type HELP for help
DTR>
```

If you do not get this startup banner when you invoke DATATRIEVE, consult the person responsible for DATATRIEVE-11 at your site.

To end your DATATRIEVE session and return to your system prompt, type EXIT and press RETURN or type CTRL/Z.

1.3 Commands and Statements

During an interactive DATATRIEVE session, you control DATATRIEVE by entering a series of commands and statements. Although intermingled in this manual, commands and statements have different functions and structures.

Commands deal with the data dictionary and perform the data description functions of DATATRIEVE. Commands let you define DATATRIEVE's basic data structures such as domains, records, files, tables, and procedures. DATATRIEVE stores this information in your current data dictionary and you can assign appropriate access privileges to different users. With commands, you can:

- Create, change, and display definitions in the data dictionary (DEFINE, EDIT, DELETE, EXTRACT, SHOW)
- Maintain the access control lists (ACL) associated with those definitions (DEFINEP, DELETEP, SHOWP)
- Get access to domains (READY)
- Release access to domains, variables, collections, and DATATRIEVE tables (RELEASE, FINISH)
- Get on-line information about DATATRIEVE (HELP)
- End your DATATRIEVE sessions (EXIT)

Statements deal with data and perform the query, report, and data manipulation functions of DATATRIEVE. Statements let you manage data by storing, modifying, and erasing records. You also use statements to retrieve and display selected data. With statements, you can:

- Store records (STORE)
- Form and manipulate groups of records (FIND, SORT, SELECT, DROP)
- Display data (PRINT, LIST, REPORT, SUM)
- Modify records (MODIFY)
- Erase records (ERASE)
- Declare variables (DECLARE)
- Assign values to fields and variables (Assignment)

Commands and statements differ in structure, as well as in function. A command consists of a keyword (the command name, such as READY or SHOW), and may include other elements, such as additional keywords, dictionary names, and definition clauses. You can enter commands only at DATATRIEVE command level, which is indicated by the DTR> prompt. You cannot join commands with statements or other commands to form compound commands.

A statement also consists of a keyword (the statement name, such as FIND or PRINT), and may include other elements, such as additional keywords, record selection expressions, value expressions, and Boolean expressions. Only statements may contain DATATRIEVE value expressions (see Chapter 2) and record selection expressions (see Chapter 3). Unlike commands, you can combine statements to form compound statements (BEGIN-END and THEN), complex logical structures (FOR, REPEAT, and WHILE), conditional transfers (IF-THEN-ELSE and ABORT), and procedures.

A compound statement is two or more statements combined in a BEGIN-END or THEN statement. You can use a compound statement anywhere you can use a simple statement. DATATRIEVE executes each statement of a compound statement in consecutive order.

You can perform complex or repetitive tasks by nesting statements within other statements. To create repeating loops, for example, nest statements in REPEAT, FOR, and WHILE statements. For example:

```
DTR> SET NO PROMPT
DTR> REPEAT 3
CON>                    PRINT "This is a test"
This is a test
This is a test
This is a test

DTR>
```

To perform conditional transfers or branching, nest statements in the IF-THEN-ELSE statement. For example:

```
DTR> SET NO PROMPT
DTR> READY YACHTS WRITE
DTR> FOR YACHTS WITH BUILDER = "PEARSON"
CON>                   BEGIN
CON>                    PRINT
CON>                    PRINT SKIP
CON>                    IF *,"Y to modify price, N to skip" CONT "Y"
CON>                     THEN
CON>                     BEGIN
CON>                      MODIFY PRICE
CON>                      PRINT SKIP
CON>                     END ELSE
CON>                     PRINT "No change", SKIP
CON>                    IF *,"Y to continue" NOT CONT "Y" THEN
CON>                     ABORT "End of price changes"
CON>                   END
```

(continued on next page)

```
                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG       ALL    WEIGHT BEAM   PRICE

 PEARSON        10M        SLOOP     33     12,441  11


Enter Y to modify price, N to skip: N(RET)
No change

Enter Y to continue, N to quit: N(RET)
ABORT: End of price changes
Execution terminated by "ABORT" statement
DTR>
```

Chapter 5 describes all DATATRIEVE language elements in alphabetical order. You can find alphabetical and functional summaries of DATATRIEVE commands and statements in Tables 5-1 and 5-2.

1.3.1 The DATATRIEVE Character Set

The DATATRIEVE character set consists of uppercase letters (A through Z), lowercase letters (a through z), digits (0 through 9), and the following special characters:

.	(period)	'	(single quotation mark)
,	(comma)	=	(equal sign)
;	(semicolon)	|	(vertical line)
:	(colon)	<	(less-than-sign)
(	(left parenthesis)	>	(greater-than-sign)
)	(right parenthesis)	$	(dollar sign)
[	(left bracket)	!	(exclamation point)
]	(right bracket)	_	(underscore)
+	(plus sign)	%	(percent sign)
-	(hyphen or minus sign)	?	(question mark)
*	(asterisk)	@	(at sign)
/	(slash)	(RET)	(RETURN)
##	(space)	"	(double quotation mark)
(TAB)	(TAB)		

DATATRIEVE accepts lowercase letters as input, but converts them to uppercase letters before analyzing the syntax of the input. DATATRIEVE preserves lowercase letters only in character string literals enclosed in quotation marks.

Similarly, DATATRIEVE accepts hyphens (-) as input but converts them to underscores (_) before analyzing the syntax of your input. To use a hyphen as a minus sign in an expression such as PRICE - PRICE * 0.1, include spaces on both sides of it. If you do not include spaces, DATATRIEVE converts the hyphen to an underscore and tells you that the field "PRICE_PRICE" is undefined or used out of context. You do not have to put spaces on both sides of the hyphen in a numeric expression such as 10-2*0.1, although it is a good idea to do so for readability and consistency.

1.3.2 Command and Statement Elements

Every DATATRIEVE command and statement consists of the command or statement name and one or more of the following elements:

- Other keywords
- Names, which identify items associated with values
- Expressions, which specify values or create record streams
- Continuation characters, which allow a command or statement to be continued on the next input line
- Terminator, which signals the end of a command or statement
- Comments, which allow you to enter text with a command or statement

The following sections describe all these elements except expressions. Chapter 2 discusses DATATRIEVE value expressions and Boolean expressions and Chapter 3 discusses the record selection expression (RSE).

1.3.2.1 Keywords — A keyword is an element of a command or statement. Most keywords are command or statement names, but some keywords are used in record definitions and dictionary tables. In this manual, DATATRIEVE keywords appear in uppercase letters.

You can use keywords where shown in syntax formats of commands and statements. Do not use keywords as names of domains, records, fields, dictionary tables, collections, procedures, variables or views. Appendix A contains a list of all DATATRIEVE keywords.

1.3.2.2 Names — A DATATRIEVE name is a character string used to identify the following items:

Collections	Fields	Procedures
Dictionary tables	Ports	Variables
Domains	Records	Views

A DATATRIEVE name can consist of 1 through 31 letters, digits, hyphens (-), and underscores (_) and must conform to the following rules:

- A name must begin with a letter.
- A name must end with a letter or digit.
- A name cannot be a keyword. (See Appendix A for a list of DATATRIEVE keywords.)

You can continue a name from one line to another by typing a hyphen (-) and pressing RETURN.

Some valid DATATRIEVE names are:

- TOTAL_SALARY
- YACHTS
- PRICE_PER_POUND
- YEAR_TO_DATE_EARNINGS_FOR_1980

Some invalid DATATRIEVE names are:

- TOTAL (Duplicates a keyword)
- 1980_EARNINGS (Does not begin with a letter)
- PRICE_PER_POUND($/LB) (Contains illegal characters)
- YEAR_TO_DATE_EARNINGS_FOR_FY_1980 (Too many characters)

1.3.2.3 Termination and Continuation Characters — DATATRIEVE uses special characters to terminate commands and statements and to continue commands and statements that are longer than one line.

A terminator signals the end of a command or statement. The formal terminator in DATATRIEVE is the semicolon (;). You can use a semicolon to separate commands and statements on the same input line. For example:

```
DTR> READY FAMILIES; FIND FAMILIES WITH NUMBER_KIDS = 2; SELECT 1RET
[6 records found]
```

DATATRIEVE does not begin processing the commands and statements on an input line until you press RETURN.

You can end most commands and statements (except the DEFINE and DELETE commands and the DECLARE statement) by pressing RETURN when the syntax of the command or statement is complete. If SET PROMPT is in effect and you press RETURN before a statement or command is complete, DATATRIEVE prompts you for the next element.

You can use the hyphen (-) at the end of an input line to continue your command or statement on the next line. DATATRIEVE does not check the syntax of your input until you press RETURN at the end of a line that does not end with a hyphen. The total number of characters in an input line extended by hyphens cannot exceed 132 or DATATRIEVE displays an error message.

If a line you are extending ends with a complete word, precede the hyphen with a space. Otherwise, DATATRIEVE treats the characters at the end of one line and the beginning of the next as a single character string. If you have to change lines in the middle of a character string literal, name, or keyword, do not precede the hyphen with a space.

Note that DATATRIEVE does not convert a hyphen at the end of a line to an underscore. For example:

```
RW> END-(RET)
CON> REPORT(RET)
Invalid report statement ("ENDREPORT")
```

This example is valid:

```
RW> END--(RET)
RW> REPORT(RET)
```

1.3.2.4 Comments — You can include a comment in an input line by preceding the comment with an exclamation point (!) and ending it by pressing RETURN. The comment can include any characters on your terminal keyboard, except escape and control characters.

You can use comments in procedures and command files to document their functions. DATATRIEVE stores comments in the dictionary as part of the procedure definition, but ignores the comments when you invoke the procedure or execute the command file.

1.3.3 DATATRIEVE Procedures

Most DATATRIEVE applications involve sequences of commands and statements that you use regularly. You can avoid retyping such a sequence by storing it in the data dictionary as a procedure. With the DEFINE PROCEDURE command, you give the procedure a name and enter both the name and the sequence of commands and statements into the dictionary. For example:

```
DTR> DEFINE PROCEDURE SUM_YACHTS(RET)
DFN>    READY YACHTS(RET)
DFN>    FIND FIRST *,"How many to find" YACHTS(RET)
DFN>    SUM 1 ("NUMBER"/"OF YACHTS") USING 9,(RET)
DFN>    PRICE USING $$$$,$$$ BY BUILDER(RET)
DFN> END_PROCEDURE(RET)
DTR> SHOW PROCEDURES(RET)
Procedures:
        SUM_YACHTS
DTR>
```

When you want to use the sequence of commands and statements in a procedure, invoke the procedure by entering a colon (:) and the procedure name. DATATRIEVE then executes the statements and commands in the procedure. For example:

```
DTR> :SUM_YACHTS(RET)
Enter How many to find: 6(RET)

              NUMBER                NUMBER
MANUFACTURER OF YACHTS  PRICE    OF YACHTS  PRICE

 ALBERG           1     $36,951
 ALBIN            3     $64,000
 AMERICAN         2     $28,790
                                      6     $129,741

DTR>
```

1.4 DATATRIEVE Prompts

Three types of DATATRIEVE prompts help you keep track of your status during a DATATRIEVE session:

- Input line prompts show that DATATRIEVE is ready for your input.
- Syntax prompts show what DATATRIEVE expects your next input to be.
- Prompts for storing and modifying values request you to enter values for fields in records.

1.4.1 Input Line Prompts

DATATRIEVE has six input line prompts:

- DTR> is the DATATRIEVE command level prompt. It shows that DATATRIEVE is ready for you to enter a DATATRIEVE command or statement.
- CON> is the continuation prompt. DATATRIEVE displays the CON> prompt when you press RETURN before completing a command or statement. You must finish the command or statement or enter a CTRL/Z to return to DATATRIEVE command level.
- DFN> is the definition prompt. DATATRIEVE displays the DFN> prompt after you enter a DEFINE command and continues to prompt with DFN> until you end the DEFINE command or enter CTRL/Z.
- QED> is the DATATRIEVE-11 Editor command mode prompt. DATATRIEVE displays the QED> prompt after you enter an EDIT command. You can then enter Editor commands to modify dictionary objects. To exit from the Editor, type CTRL/Z in response to the QED> prompt.
- IN> is the DATATRIEVE-11 Editor insert mode prompt. DATATRIEVE displays the IN> prompt after you enter an INSERT or REPLACE editing command in response to the QED> Editor command mode prompt. See Sections 5.25.3 and 5.25.5 for information on the INSERT and REPLACE editing commands.
- RW> is the Report Writer prompt. DATATRIEVE displays the RW> prompt after you enter a REPORT statement and continues to prompt with RW> until you end the report or enter CTRL/Z.

The CON> prompt is the most complex DATATRIEVE prompt. Because commands and statements have so many options, you may press RETURN, intending to continue the command or statement, and find that DATATRIEVE executes what you have entered. That is because DATATRIEVE interprets what you have entered as a complete command or statement. If you intend to continue the command or statement, press RETURN after a comma, in the middle of a string of required keywords, or in the middle of a value expression or Boolean expression. You can also use the hyphen (-) as a continuation character. See Section 1.3.2.3 for more information on the hyphen.

1.4.2 Syntax Prompts

When you press RETURN before completing a command or statement, DATATRIEVE prompts you with a phrase in square brackets that tells you what sort of input it expects:

```
DTR> READY(RET)
[Looking for Dictionary Element]
CON> YACHTS(RET)
DTR> FIND(RET)
[Looking for "FIRST", domain name, or collection name]
CON> YACHTS WITH(RET)
[Looking for Boolean expression]
CON> LOA(RET)
[Looking for relational operator (eq, gt, etc.)]
CON> GT(RET)
[Looking for a value expression]
CON> 24(RET)
[105 records found]
DTR>
```

DATATRIEVE displays these prompting messages by default. You can suppress syntax prompts by entering a SET NO PROMPT command at DATATRIEVE command level. To enable syntax prompts again, enter the SET PROMPT command.

1.4.3 Prompts for Storing and Modifying Values

When you enter STORE or MODIFY statements that do not contain the USING clause, DATATRIEVE prompts you to enter the values for the fields to be stored or modified. These prompts take the general form "Enter" followed by a field name:

```
DTR> READY FAMILIES WRITE(RET)
DTR> STORE FAMILIES(RET)
Enter FATHER: GEORGE(RET)
Enter MOTHER: MARTHA(RET)
Enter NUMBER_KIDS: 1(RET)
Enter KID_NAME: JENNIFER(RET)
Enter AGE: 12
DTR>
```

The SET NO PROMPT command does not affect these prompts.

1.5 Data Files and DATATRIEVE Record Definitions

DATATRIEVE stores information in data files. A data file contains a group of related records. A record contains data organized by and stored in fields. The two types of fields are elementary fields and group fields:

- An elementary field contains one item of data. This item can consist of one character ("1") or a number of characters that are to be interpreted as a unit ("10 Downing Street").
- A group field contains one or more elementary fields and may also contain other group fields.

You control the storage and interpretation of information with the DEFINE RECORD command. When you define a record, you define the fields of the record and specify how DATATRIEVE stores and retrieves data in the record.

You also specify how records are stored in the data file. The data file that contains your records has certain attributes that depend on the way you define the record fields. An individual record contains unique information, but all the records in a data file share a common organization as specified in the record definition.

To define the physical organization of a data file, use the DEFINE FILE command. DATATRIEVE data files are RMS files and can be organized in two ways:

- A sequential file contains records stored physically next to each other. The sequence of records in the file is completely independent of the information in any field of the record. Each new record in a sequential file is stored at the end of the last record written to the file. Thus, you can modify records in a sequential file but you cannot delete records.
- An indexed file contains records stored by index field values. When you define the file with the DEFINE FILE command, you can specify one field in the record, called the primary key, to be used as the index for the file organization. Records are stored in the file according to the order of values in the primary key field. The primary key for a personnel file, for example, might be the employee number. Because records are stored by index rather than by sequence, you can delete records from an indexed file. You cannot, however, modify the value of the primary key field.

DATATRIEVE does not support RSTS/E stream files.

See Chapter 4 in this manual for more information on DATATRIEVE records and record definitions.

1.6 DATATRIEVE Domains

To manipulate and manage data in a file, you must explicitly connect that data file to a record definition. DATATRIEVE makes this connection with a data structure called a domain. You use the DEFINE DOMAIN command to write a domain definition and to store that definition in a data dictionary.

A domain definition establishes a name for the domain and associates that name with the names of a record definition and a data file. When you use the name of a domain, you tell DATATRIEVE to use a particular record definition to interpret the data stored in a specific file.

1.7 ADT, the Application Design Tool

ADT, the Application Design Tool, is an interactive feature of DATATRIEVE-11 that helps you define a domain, a record, and a data file.

ADT uses your responses to a series of questions to build a command file containing the DATATRIEVE commands to define the domain, record, and data file.

When you invoke the command file, DATATRIEVE executes each command, stores the definitions of the domain and record in your current dictionary, and creates an RMS file.

Chapter 3 in *Introduction to DATATRIEVE-11* discusses the use of the Application Design Tool and contains a sample ADT session.

1.8 DATATRIEVE Tables

A DATATRIEVE table is a group of code and description pairs you create and store as a named table. With DATATRIEVE tables, you associate codes with translations, thereby saving record space and time. A code like "EB", for example, can be associated with a lengthy translation like "Employee Benefits". You can then store the code in a record, and print the descriptive character string associated with the code in the table. Because the code can be as short as a single character and its associated description much longer, tables can save a great deal of record space. Tables can also help you validate data according to the presence or absence of a data item in the table.

To create and name a table, use the DEFINE TABLE command. DATATRIEVE stores the table definition in the data dictionary. See the section in this manual on the DEFINE TABLE command and Chapter 13 of the *DATATRIEVE-11 User's Guide* for more information on defining and using tables.

1.9 The DATATRIEVE-11 Report Writer

The DATATRIEVE-11 Report Writer helps you display your data in easy-to-read formats. You can display a report on your terminal, print it on a hard-copy printer, or store it in an RMS file for display or printing at a later time.

With the Report Writer, you can specify the number of lines per page or the number of pages in your report. You can also create title pages and print special headings above and detail lines below groups of sorted data that share a common value in one or more fields. Your reports can have the title, page number, and date at the top of each page and can report statistical summaries of your data in columns and at the bottoms of the report, individual pages, or groups of sorted data. The Report Writer supplies default formatting values for elements such as columns-per-page that you do not supply. For more information, see the *DATATRIEVE-11 Guide to Writing Reports*.

1.10 The DATATRIEVE-11 Editor

With the DATATRIEVE Editor, you can modify and add to data dictionary objects already defined in your current data dictionary. The Editor has an edit mode and an insert mode:

- When you invoke the Editor with the EDIT command, DATATRIEVE responds with the QED> edit mode prompt. In edit mode, the Editor interprets all of your input as commands and you can display and alter the text of the dictionary object. Use CTRL/Z to return to DATATRIEVE command level.

- When you enter the INSERT or REPLACE command in response to the QED> prompt, DATATRIEVE displays the IN> insert mode prompt. In insert mode, the Editor interprets all of your input as new text to be entered into the dictionary object. Use CTRL/Z to return to edit mode.

See the section on the EDIT command in this manual and Chapter 17 in the *DATATRIEVE-11 User's Guide* for information on edit mode commands.

1.11 The DATATRIEVE-11 Remote Access Interface

The DATATRIEVE-11 Remote Access Interface allows you to call DATATRIEVE on your own system or on another DECnet node from an application program. It also allows you to run DATATRIEVE from a remote terminal. The Remote Access Interface consists of:

- The DATATRIEVE-11 Distributed Server
- The DATATRIEVE-11 Call Interface
- The DATATRIEVE-11 Remote Terminal Interface

The Distributed Server, also called DDMF (Distributed Data Manipulation Facility), is a "slave" program that allows users on other DECnet nodes to use DATATRIEVE for accessing data files and dictionaries on your node. A DATATRIEVE component sends commands to DDMF, and DDMF passes the results back to that component. Call Interface subroutines, for example, send commands to and receive information from a distributed server. DDMF can perform all of the DATATRIEVE functions with the exception of ADT and Guide Mode.

Both DATATRIEVE-11 and VAX–11 DATATRIEVE have distributed servers.

The Call Interface allows application programs to access data files and data dictionaries by calling DATATRIEVE subroutines. With the Call Interface, you can customize the appearance of DATATRIEVE, write programs to perform tasks that interactive DATATRIEVE cannot do, and access data through DATATRIEVE on other DECnet nodes. To use the Call Interface, include calls to external DATATRIEVE subroutines in your program. The subroutines pass information between the calling programs and a local or remote DATATRIEVE Distributed Server. You must have DECnet software to use the Call Interface.

The DATATRIEVE Remote Terminal Interface (REMDTR) allows you to access the DATATRIEVE Remote Call Interface from a terminal. The Terminal Interface uses the Remote Call Interface to establish a DECnet link to the Distributed Server. When the link is established, REMDTR displays the version of DATATRIEVE and the REMDTR> prompt. From this point on, you can type commands and statements as though you are running interactive DATATRIEVE on the remote node.

See the *DATATRIEVE-11 Call Interface Manual* for information on using the Remote Terminal Interface and on writing programs that use the Call Interface.

Value Expressions and Boolean Expressions 2

2.1 Introduction

This chapter describes the value expressions and Boolean expressions you can use in DATATRIEVE-11 statements.

A value expression is a string of symbols that specifies a value for DATATRIEVE to use when executing statements. A Boolean expression is the logical representation of a relationship between value expressions. The value of a Boolean expression is either true or false.

2.2 Value Expressions

Value expressions specify values for DATATRIEVE to use when executing statements. DATATRIEVE-11 provides you with the following types of value expressions:

- Character string literals
- Numeric literals
- Field names
- Global and local variables
- Date value expressions
- Prompting value expressions
- Values from dictionary tables
- Statistical expressions
- Arithmetic expressions including parentheses
- Concatenated expressions

The following sections describe these value expressions.

2.2.1 Character String Literals

A character string literal is a string of printing characters up to 130 characters long. The maximum size for an input line in DATATRIEVE-11 is 132 characters. In character string literals, however, two of those characters are used for opening and closing quotation marks. Printing characters consist of uppercase and lowercase letters, numbers, and the following special characters:

```
! @ # $ % ^ & * ( ) - _ = + ` [
{ ] } ~ ; : ' " \ | , < . > / ?
```

You can continue a literal on more than one line by typing a hyphen (-), the DATATRIEVE continuation character, and pressing RETURN. DATATRIEVE strips the hyphen from that part of the character string literal and waits for you to complete the literal by typing the closing quotation mark on the next line. As long as the total number of characters (including two double quotation marks) in the literal does not exceed 132, you can use any number of continuation characters between the quotation marks.

You can create a character string exceeding 132 characters in two ways:

- Use concatenated expressions (described in Section 2.2.9).
- Define a very long group field comprised of several elementary fields, each of which can be up to 132 characters long, then store character string values in each elementary field. The following record definition, for example, can store a string literal of 468 characters:

```
01  STRING LIT.
    05  LINE_1 PIC X(132).
    05  LINE_2 PIC X(132).
    05  LINE_3 PIC X(132).
    05  LINE_4 PIC X(80).
```

Use double quotation marks to enclose character string literals:

```
Literal: "MAXIMUM PRICE IS $1400.  PLEASE RE-ENTER PRICE."
Value:  MAXIMUM PRICE IS $1400.  PLEASE RE-ENTER PRICE.
```

To include double quotation marks in a character string literal, you must type two consecutive quotation marks for every one you want to include in the literal:

```
Literal: "They said, ""We're going."""
Value:  They said, "We're going."
```

Although DATATRIEVE usually converts all lowercase letters of your input to uppercase, it preserves lowercase letters in character string literals. Preserving the case of character string literals has important effects on comparisons using character string literals, because such comparisons are case-sensitive:

```
DTR> READY YACHTS↵
DTR> FIND YACHTS WITH BUILDER = "ALBIN"↵
[3 records found]
DTR> FIND YACHTS WITH BUILDER = "albin"↵
[0 records found]
DTR>
```

2.2.2 Numeric Literals

A numeric literal is a string of digits that DATATRIEVE interprets as a decimal number. A numeric literal may contain a decimal point and up to 18 digits. The decimal point is optional and does not count as a digit.

A numeric literal cannot begin or end with a decimal point. For example, .5 and 123. are not valid numeric literals, but 0.5 and 123.0 are valid.

With two restrictions, the data type of a field or variable determines the maximum numeric literal you can assign to the field or variable:

- You cannot specify more than 18 digits in the PICTURE (PIC) clause. If you do, DATATRIEVE displays an error message on your terminal.
- The format specified in the PICTURE (PIC) clause of a field or variable also limits the values you can assign with numeric literals. See the section on the PICTURE clause in Chapter 5 for a discussion of these restrictions.

2.2.3 Field Names

You can use these types of field names as value expressions:

- Elementary and REDEFINES field names
- Group field names
- Virtual field names (COMPUTED BY fields)
- Query names
- Qualified field names

2.2.3.1 Elementary and REDEFINES Field Names — The value specified by an elementary field name is the value stored in a field of a record.

If the field name you use refers to a REDEFINES field, the value associated with the name is determined by the clauses that define the REDEFINES field in the record definition. In the following example, PART_NUMBER is an elementary

field, and PART_NUMBER_PARTS and PART_NUMBER_GROUPS are REDEFINES fields that redefine the PART_NUMBER field. The PRINT statement shows how DATATRIEVE interprets values stored in the PART_NUMBER elementary field when you specify the elementary field name and the REDEFINES field names:

```
DTR> SET NO PROMPT↵
DTR> SHOW PART-REC↵
RECORD PART_REC
01 PART_RECORD.
   05 PART_NAME PIC X(10).
   05 PART_NUMBER PIC 9(10).
   05 PART_NUMBER_PARTS REDEFINES PART_NUMBER.
      07 PRODUCT_GROUP PIC 99.
      07 PRODUCT_YEAR PIC 99.
      07 ASSEMBLY_CODE PIC 9.
      07 SUP_ASSEMBLY PIC 9(5).
   05 PART_NUMBER_GROUPS REDEFINES PART_NUMBER.
      07 PRODUCT_GROUP_ID PIC 9(4).
      07 PART_DETAIL_ID PIC 9(6).
;
DTR> READY PARTS↵
DTR> FOR PARTS PRINT PART-RECORD, PART-NUMBER-PARTS, PART-NUMBER-GROUPS↵

                                                          PRODUCT  PART
   PART        PART     PRODUCT PRODUCT ASSEMBLY   SUP     GROUP  DETAIL
   NAME       NUMBER     GROUP    YEAR     CODE   ASSEMBLY   ID      ID

DODADS     1113335559    11       13       3      35559    1113   335559
GOFER      0987654321    09       87       6      54321    0987   654321
WALLTHING  5555599999    55       55       5      99999    5555   599999
WIDGET     1234567890    12       34       5      67890    1234   567890

DTR>
```

2.2.3.2 Group Field Names — When you use group field names in assignment statements, the value of the group field depends on the type of assignment. The usual assignment statement has this format:

group-field-name-1 = group-field-name-2

In this type of statement, DATATRIEVE assigns values on the basis of similar elementary field names. The value of group field name 2 includes the values of all elementary fields, all REDEFINES fields, and all COMPUTED BY fields.

DATATRIEVE assigns the values of elementary fields in group field 2 to the elementary fields in group field 1 that match group field 2 names.

DATATRIEVE ignores fields in group field 2 whose names do not match any field names in group field 1. If an elementary field name in group field 1 does not match a group field 2 field name, DATATRIEVE assigns the following values:

- Zero (numeric fields)
- Blank (alphabetic or alphanumeric fields)

DATATRIEVE stores no values in any COMPUTED BY or REDEFINES fields in group field 1, regardless of any matches between the names of those fields and fields in group field 2. The values of the elementary, REDEFINES, and

COMPUTED BY fields associated with group field name 2 are the values stored in or associated with the fields that constitute group field 2 of a record.

Because of problems that can arise from conflicting data types, assignments of the following type are not recommended:

elementary-field-name = group-field-name

If you make an assignment of this type, the value of the group field is the same as the value displayed on your terminal with a DISPLAY group field name statement. The value is the concatenation of the values in the elementary fields that constitute the group field. DATATRIEVE ignores REDEFINES and COMPUTED BY fields. You can reduce conflicts of data types by using an alphanumeric PICTURE string in the definition of the elementary field. The following example illustrates the results of assigning the value of the group field SPECS, which contains both alphanumeric and numeric elementary fields, to an elementary field with an alphanumeric PICTURE string and a numeric PICTURE string:

```
DTR> SET NO PROMPT(RET)
DTR> SHOW STRING_REC(RET)
RECORD STRING_REC
 USING
01 TOP.
   03 ELEM_FIELD  PIC X(18).
;
DTR> SHOW NUMERIC_REC(RET)
RECORD NUMERIC_REC
 USING
01 TOP.
   03 ELEM_FIELD  PIC 9(18).
;
DTR> SHOW YACHT(RET)
RECORD YACHT
 USING
01 BOAT.
  03 TYPE.
     06 MANUFACTURER PIC X(10)
        QUERY_NAME IS BUILDER.
     06 MODEL PIC X(10).
  03 SPECIFICATIONS
     QUERY_NAME SPECS.
     06 RIG PIC X(6)
        VALID IF RIG EQ "SLOOP","KETCH","MS","YAWL".
     06 LENGTH_OVER_ALL PIC XXX
        VALID IF LOA BETWEEN 15 AND 50
        QUERY_NAME IS LOA.
     06 DISPLACEMENT PIC 99999
        QUERY_HEADER IS "WEIGHT"
        EDIT_STRING IS ZZ,ZZ9
        QUERY_NAME IS DISP.
     06 BEAM PIC 99.
     06 PRICE PIC 99999
        VALID IF PRICE>DISP*1.3 OR PRICE EQ 0
        EDIT_STRING IS $$$,$$$.
;
DTR> READY YACHTS READ(RET)
DTR> !(RET)
DTR> ! Print the SPECS group field for the first yacht(RET)
DTR> !(RET)
DTR> PRINT SPECS OF FIRST 1 YACHTS(RET)
```

(continued on next page)

```
        LENGTH
         OVER
 RIG     ALL    WEIGHT BEAM  PRICE

KETCH    37     20,000  12  $36,951

DTR> READY STRING_TEST WRITE↵
DTR> READY NUMERIC_TEST WRITE↵
DTR> !↵
DTR> ! Store the SPECS value in the alphanumeric↵
DTR> ! elementary field, then print STRING_TEST↵
DTR> !↵
DTR> FOR FIRST 1 YACHTS ↵
CON>    STORE STRING_TEST USING ELEM_FIELD = SPECS↵
DTR> PRINT STRING_TEST↵

       ELEM
      FIELD

KETCH 37 200001236

DTR> !↵
DTR> ! Store the SPECS value in the numeric↵
DTR> ! elementary field, then print NUMERIC_TEST↵
DTR> !↵
DTR> FOR FIRST 1 YACHTS↵
CON>    STORE NUMERIC_TEST USING ELEM_FIELD = SPECS↵
Non-digit in string "KETCH 37 200001236951", ignoring character(s)
DTR> PRINT NUMERIC_TEST↵

       ELEM
      FIELD

000037200001236951

DTR>
```

2.2.3.3 COMPUTED BY Fields — COMPUTED BY fields are virtual fields. DATATRIEVE does not store the value of a COMPUTED BY field in a record. That value is calculated every time you refer explicitly or implicitly to a COMPUTED BY field. The value specified by a virtual field name is the value associated with that field name in a record.

2.2.3.4 Query Names — If the record definition contains a query name for a field, you can use the query name exactly as you use the associated field name.

2.2.3.5 Qualified Field Names — To clarify the context for recognizing names and associating values with those names, you can qualify field names with several optional elements:

- **Context variables** allow you to distinguish between fields in different record streams.
- **Record names** allow you to distinguish between fields in different domains that have the same field name.
- **Group field names** allow you to distinguish between fields that have the same field name, but are contained in different group fields.

Use the following format to qualify field names:

[context-variable][record-name][group-field-name]...[field-name]

To refer to fields within a record, you must create a context. If there is more than one valid context for a field name, DATATRIEVE uses the most recent valid context. If there is no valid context for a field name, DATATRIEVE issues an error message:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS↵
DTR> FIND TINIES IN YACHTS WITH LOA LE 25↵
[13 records found]
DTR> FIND BIGGIES IN YACHTS WITH LOA GE 40↵
[8 records found]
DTR> FOR A IN BIGGIES↵
CON> PRINT TINIES WITH BUILDER = A.BUILDER↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG   ALL   WEIGHT BEAM  PRICE

 ISLANDER     BAHAMA     SLOOP   24     4,200  08   $6,500

DTR> PRINT↵
No record selected, printing whole collection

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG   ALL   WEIGHT BEAM  PRICE

 CHALLENGER   41         KETCH   41    26,700  13  $51,228
 COLUMBIA     41         SLOOP   41    20,700  11  $48,490
 GULFSTAR     41         KETCH   41    22,000  12  $41,350
 ISLANDER     FREEPORT   KETCH   41    22,000  13  $54,970
 NAUTOR       SWAN 41    SLOOP   41    17,750  12
 NEWPORT      41 S       SLOOP   41    18,000  11
 OLYMPIC      ADVENTURE  KETCH   42    24,250  13  $80,500
 PEARSON      419        KETCH   42    21,000  13

DTR> PRINT A↵
Field "A" is undefined or used out of context
DTR>
```

This example uses three context variables: BIGGIES, TINIES, and A. BIGGIES and TINIES also become the collection names. The FOR statement uses the context variable A to establish a context for the first PRINT statement. The next PRINT statement does not specify a context, so DATATRIEVE uses the most recent context and displays all the records in the most recently formed collection, BIGGIES. An attempt to display the variable A without a context causes DATATRIEVE to display an error message.

2.2.4 Variables

With the DECLARE statement, you can define global and local variables and use them as value expressions. When you declare them, DATATRIEVE initializes numeric variables to zero and alphanumeric variables to a space.

2.2.4.1 Global Variables — You define global variables with DECLARE statements entered at DATATRIEVE command level. That is, any variable not declared in a BEGIN-END block is a global variable. You can then use a global variable as a value expression in any DATATRIEVE statement. Unless you define a global variable with a COMPUTED BY clause, the global variable retains the value you assign to it until you either assign it a new value or release it with the RELEASE command.

The value of a global variable defined with a COMPUTED BY clause depends on the value expression that controls the computation. The following example declares the COMPUTED BY variable to be 1.2 times the price of a boat in the YACHTS domain. The value of the variable changes according to the value of the PRICE field for different records:

```
DTR> SET NO PROMPT(RET)
DTR> READY YACHTS(RET)
DTR> DECLARE VAR COMPUTED BY PRICE * 1.2.(RET)
DTR> FOR FIRST 5 YACHTS PRINT PRICE, VAR USING $$$,$$$.99(RET)

 PRICE      VAR
$36,951 $44,341.20
$17,900 $21,480.00
$27,500 $33,000.00
$18,600 $22,320.00
$ 9,895 $11,874.00

DTR>
```

2.2.4.2 Local Variables — You can define local variables with DECLARE statements entered in BEGIN-END and THEN statements. A local variable stays in effect for subsequent statements of the compound statement in which it is declared, but has no meaning in any outer statements containing that compound statement. A local variable is released as soon as DATATRIEVE completes the execution of the clause or statement in which it was declared:

```
DTR> SET NO PROMPT(RET)
DTR> BEGIN(RET)
CON>     DECLARE X PIC XXX.(RET)
CON>     X = "TOP"(RET)
CON>     PRINT X(RET)
CON>                 BEGIN(RET)
CON>                     DECLARE X PIC 9.99.(RET)
CON>                     X = 1.23(RET)
CON>                     PRINT X(RET)
CON>                 END(RET)
CON>     PRINT X(RET)
CON> END(RET)

 X

TOP

 X

1.23

 X

TOP
```

(continued on next page)

```
DTR> PRINT X(RET)
Field "X" is undefined or used out of context
DTR>
```

In this example, the variable X declared in the inner BEGIN statement does not affect the variable X declared in the outer BEGIN statement, even though the inner X has a different data type and value. Neither local variable exists when DATATRIEVE completes the execution of the compound statements containing them.

Note

To avoid problems resolving names for variables and fields, do not use variable names that duplicate field names of domains you have readied.

2.2.5 DATE Value Expressions

To define a field as a date, use the USAGE IS DATE clause. The IS keyword is optional. You can then assign the current system date to the field with the DATATRIEVE date value expression, "TODAY":

```
DTR> DECLARE X USAGE DATE.(RET)
DTR> X = "TODAY"(RET)
DTR> PRINT X(RET)

     X

 18-May-83

DTR>
```

To use date value expressions, assign the DATE data type to the field or variable. Otherwise, DATATRIEVE treats the expression as a character string literal:

```
DTR> PRINT "TODAY"(RET)
TODAY

DTR>
```

DATATRIEVE returns the character string literal "TODAY" because the quoted expression is not associated with a variable or field of the DATE data type. However, if you supply an edit string containing date edit string characters, DATATRIEVE returns the current system date in the form specified by the edit string:

```
DTR> PRINT "TODAY" USING DD-MMM-YY(RET)

18-May-83

DTR>
```

You can also subtract one date from another to find the elapsed number of days. You might, for example, want to know how many days you have to complete a project. Define variables for today's date and the project due date, then subtract today's date from the project due date.

```
DTR> DECLARE T USAGE DATE,RET
DTR> T = "TODAY"RET
DTR> DECLARE PROJECT_DUE USAGE DATE,RET
DTR> PROJECT_DUE = "21-AUG-83"RET
DTR> PRINT (PROJECT_DUE - T)RET
          31

DTR>
```

DATATRIEVE indicates that the project is due in 31 days.

2.2.6 Prompting Value Expressions

If you want DATATRIEVE to prompt you for a value, use a prompting value expression. This feature is especially useful in a procedure because it allows you to use a different value each time you invoke the procedure. DATATRIEVE recognizes two types of prompting expressions:

*.prompt-string

**.prompt-string

The prompt string is a character string literal. If the prompt string contains no blanks and conforms to the rules for DATATRIEVE names, you do not have to enclose the literal in quotation marks. Otherwise, you must enclose it in quotation marks.

If you put a *.prompt value expression in a REPEAT loop or a FOR loop, DATATRIEVE prompts you for a value each time it executes the loop.

If you put a **.prompt value expression in a REPEAT loop or a FOR loop, DATATRIEVE prompts you for a value only once, the first time it executes the loop. If the **.prompt value expression assigns a value to a variable or a field, DATATRIEVE uses the same value each time through the loop. This feature is especially useful when storing or modifying a group of records that have a common value in one or more fields.

The following example shows the difference between the *.prompt and **.prompt value expressions:

```
DTR> SET NO PROMPTRET
DTR> READY OWNERS WRITERET
DTR> REPEAT 2RET
CON>    STORE OWNERS USINGRET
CON>    BEGINRET
CON>              NAME = *.NAMERET
CON>              BOAT_NAME = *.BOAT_NAMERET
CON>              BUILDER = **.BUILDERRET
CON>              MODEL = **.MODELRET
CON>    ENDRET
Enter NAME: KARENRET
Enter BOAT_NAME: SEASICKRET
Enter BUILDER: BEANRET
Enter MODEL: 12RET
Enter NAME: ANDREWRET
Enter BOAT_NAME: DINGHY-2RET
DTR>
```

The first time DATATRIEVE executes the BEGIN statement, it prompts for all four values. The second time, it only prompts for the *.NAME and *.BOAT_NAME values and uses the values previously entered for MODEL and BUILDER for the second record.

2.2.7 Values From a Table

You can use a value stored in a dictionary table anywhere the syntax of a DATATRIEVE statement allows a value expression. The format for retrieving a value from a dictionary table is:

```
value-expression VIA table-name
```

Dictionary table entries are stored as code string/translation string pairs. If the specified value expression is stored as a code string in the specified table, the value of the entire expression is the corresponding translation string in the table.

If the table contains an ELSE clause and the value expression you specify does not match any code string in the table, the value of the entire expression is the translation string stored in the ELSE clause of the table. If the table contains no ELSE clause and the value expression you specify does not match any code string in the table, DATATRIEVE tells you the value was not found.

See Chapter 13 of the *DATATRIEVE-11 User's Guide* for more information about creating and using dictionary tables.

2.2.8 Statistical Expressions

Statistical expressions consist of DATATRIEVE statistical function keywords (MAX, MIN, AVERAGE, TOTAL, and COUNT) and value expressions.

Table 2-1 lists DATATRIEVE statistical functions and the values they return.

Table 2-1: Values Derived with Statistical Functions

Function	Value Returned
AVERAGE	The average value of the value expression
COUNT	The number of records in the CURRENT collection or in a specified collection or record stream
MAX	The largest value of the value expression
MIN	The smallest value of the value expression
TOTAL	The total value of the value expression

You can use statistical expressions anywhere a DATATRIEVE statement allows a value expression. A statistical expression can itself be the value expression of another statistical expression. With the exception of TOTAL, DATATRIEVE uses the edit string specified in the EDIT_STRING clause of a field definition when printing a statistical value for a field.

With the exception of the COUNT function, you must supply a value expression when using statistical functions. Use this format for all statistical functions except COUNT:

function-name value-expression [OF rse]

A value expression is usually a field name or the name of a variable that the statistical function operates on:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS↵
DTR> PRINT MAX PRICE OF FIRST 10 YACHTS↵

 PRICE

$36,951

DTR>
```

When the value expression is more complex than a field name or variable name, enclose the expression in parentheses to avoid unexpected results:

```
DTR> READY YACHTS↵
DTR> FIND A IN YACHTS WITH BUILDER = "ALBIN" AND PRICE NE 0↵
[3 records found]
DTR> PRINT ALL↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG   ALL    WEIGHT BEAM  PRICE

 ALBIN         79         SLOOP   26      4,200  10  $17,900
 ALBIN         BALLAD     SLOOP   30      7,276  10  $27,500
 ALBIN         VEGA       SLOOP   27      5,070  08  $18,600

DTR> ! PRINT THE AVERAGE PRICE PER POUND WITHOUT PARENTHESES↵
DTR> !↵
DTR> PRINT AVERAGE PRICE/DISP OF A↵
        5.079
        2.932
        4.207

DTR> ! PRINT THE AVERAGE PRICE PER POUND WITH PARENTHESES↵
DTR> !↵
DTR> PRINT AVERAGE (PRICE / DISP) OF A↵
        3.903

DTR>
```

When DATATRIEVE executes the PRINT statement without parentheses around the expression, it divides the average PRICE by 4200 (the weight of the first yacht in the collection), then by 7276 (the weight of the second yacht in the collection), and then by 5070 (the weight of the third yacht) to give three unexpected, but accurate, values of 5.079, 2.932, and 4.207. The second PRINT statement yields the expected results, the average of 5.079, 2.932, and 4.207.

The COUNT function does not need a value expression, so use this format:

```
COUNT [OF rse]
```

Note the use of a record selection expression in the following:

```
DTR> SET NO PROMPT↵
DTR> READY FAMILIES↵
DTR> PRINT COUNT OF FAMILIES WITH NUMBER_KIDS EQ 2↵

  6
```

When you supply a record selection expression in a statistical expression, DATATRIEVE uses all the records in the resulting record stream to compute the value returned by the function:

```
DTR> PRINT AVERAGE PRICE OF YACHTS WITH BUILDER = "AMERICAN"↵

 PRICE

$14,395

DTR>
```

If you do not specify an RSE, DATATRIEVE uses all records in the CURRENT collection to compute the value returned by the function. DATATRIEVE displays the message "A current collection has not been established" if you do not specify an RSE and have no CURRENT collection.

When you use two or more statistical expressions in the same statement, be sure to specify an RSE in each expression to identify the record stream on which the function is to operate. If you do not, your results may be misleading. Because no RSE was specified for the AVERAGE function in the following example, DATATRIEVE displays the average LOA for the whole current collection. The RSE for the AVERAGE function in the second PRINT statement tells DATATRIEVE to compute the average LOA for only the first five yachts, thus producing a different average:

```
DTR> SET NO PROMPT↵
DTR> FIND YACHTS↵
[113 records found]
DTR> PRINT AVERAGE LOA, TOTAL LOA OF FIRST 5 YACHTS↵

LENGTH   LENGTH
 OVER     OVER
  ALL      ALL

  30.        146

DTR> PRINT AVERAGE LOA OF FIRST 5 YACHTS,↵
CON> TOTAL LOA OF FIRST 5 YACHTS↵

LENGTH   LENGTH
 OVER     OVER
  ALL      ALL

  29         146

DTR>
```

2.2.9 Arithmetic Expressions

An arithmetic expression consists of value expressions and arithmetic operators. The value expressions must evaluate to numbers. You can use an arithmetic expression anywhere the syntax of a DATATRIEVE statement allows a value expression.

DATATRIEVE provides four arithmetic operators. Table 2-2 shows these arithmetic operators and the operation each performs.

Table 2-2: Arithmetic Operators

Operator	Operation
+	Addition
–	Subtraction or negation
*	Multiplication
/	Division

You do not have to use spaces to separate arithmetic operators from value expressions, except in one case: if a field or variable name precedes a minus sign, you must put a space between the minus sign and the field or variable name. Otherwise, DATATRIEVE interprets the minus sign as a hyphen and converts it to an underscore:

```
DTR> DECLARE X PIC 99.(RET)
DTR> X = 8(RET)
DTR> !(RET)
DTR> ! Print X-1 without a space before the minus sign(RET)
DTR> !(RET)
DTR> PRINT X-1(RET)
"Field X_1" is undefined or used out of context
DTR> !(RET)
DTR> ! Print X -1 with a space before the minus sign(RET)
DTR> !(RET)
DTR> PRINT X -1(RET)
   7

DTR> !(RET)
DTR> ! Print 1-X without a space before the minus sign(RET)
DTR> !(RET)
DTR> PRINT 1-X(RET)
 -7

DTR>
```

You can control the order in which DATATRIEVE performs arithmetic operations by using parentheses. DATATRIEVE follows these rules of precedence when evaluating arithmetic expressions:

1. DATATRIEVE first evaluates any value expressions in parentheses.
2. DATATRIEVE performs multiplications and divisions from left to right in the arithmetic expression.

3. DATATRIEVE performs additions and subtractions from left to right in the arithmetic expression.

The following examples show how parentheses affect the evaluation of arithmetic expressions:

```
DTR> PRINT (6 * 7) + 5(RET)
  47

DTR> PRINT 6 * (7 + 5)(RET)
  72

DTR> PRINT 12 - 6 / 2(RET)
     9.000

DTR> PRINT (12 - 6) / 2(RET)
      3.000

DTR>
```

2.2.10 Concatenated Expressions

DATATRIEVE allows you to join expressions with character values to form a concatenated expression. You can use a concatenated expression anywhere the syntax of a DATATRIEVE statement allows a character string literal. When DATATRIEVE concatenates value expressions, it converts their values to character string literals.

DATATRIEVE provides two ways to concatenate expressions: a single or double bar separating value expressions. In each case, DATATRIEVE first converts the value expression to its character string literal, then joins the literals to form a longer literal. Both operators treat leading spaces in the second literal as spaces. The type of operator determines how DATATRIEVE treats trailing spaces in the first literal.

When you use the single bar (|), DATATRIEVE treats trailing spaces in the first string literal as spaces:

```
Expression: "ABC"|"DEF"   "ABC "|"DEF"   "ABC"|"  DEF"   "ABC  "|"  DEF"
Result:       ABCDEF        ABC  DEF        ABC  DEF         ABC    DEF
```

The double bar causes DATATRIEVE to suppress trailing spaces in the first literal:

```
Expression: "ABC"||"DEF"   "ABC  "||"DEF"   "ABC"||"  DEF"   "ABC  "||"  DEF"
Result:       ABCDEF         ABCDEF           ABC  DEF         ABC  DEF
```

Concatenated expressions provide the only method for assigning values to fields or variables whose lengths exceed 256 characters. For example:

```
DTR> DECLARE STR PIC X(300) EDIT_STRING IS T(50).(RET)
DTR> STR = *.L1|*.L2|*.L3|*.L4|*.L5(RET)
Enter L1: This string contains the first part of a long character (RET)
Enter L2: string.  This string is so long that you can't use just (RET)
Enter L3: one character string literal to assign a value to it. (RET)
Enter L4: You need concatenated expressions to increase the length (RET)
Enter L5: of this string beyond the limit of 132 characters.(RET)
```

(continued on next page)

```
DTR> PRINT STR(RET)

                              STR

This string contains the first part of a long
character string. This string is so long that you
can't use just one character string literal to
assign a value to it.  You need concatenated
expressions to increase the length of this string
beyond the limit of 132 characters.

DTR>
```

This example combines the T edit string indicating that a line of text can contain 50 characters, prompting value expressions, and character string literals to assign a value to a long variable. You can use similar assignment statements in the USING clauses of the STORE and MODIFY statements. Note that you must type a space before pressing RETURN. The T edit string prints up to 50 characters on a line, but does not break up words from one input line to another unless you type a space to separate the last word on a line from the first word on the next line.

If you want to format the printed output of a concatenated expression, you must specify an edit string for it. DATATRIEVE does not use any edit strings specified for the components of a concatenated expression when you print it.

Note

The length of concatenated expressions is limited only by the amount of DATATRIEVE pool space available. However, concatenated expressions are not intended for storing massive amounts of text. If DATATRIEVE pool space is exhausted, DATATRIEVE will stop executing the statement it is currently processing.

2.3 Boolean Expressions

A Boolean expression is the logical representation of a relationship between value expressions. The value of a Boolean expression is either true or false. The Boolean expressions you can use in DATATRIEVE-11 consist of value expressions, relational operators, and Boolean operators. Relational operators control the comparison of value expressions. Boolean operators enable you to join two or more Boolean expressions and to reverse the value of a Boolean expression. All Boolean expressions contain value expressions and relational operators, and some contain Boolean operators.

You can use Boolean expressions in the following DATATRIEVE clauses and statements:

- The WITH clause in a record selection expression
- The IF clause of an IF-THEN-ELSE statement
- The VALID IF clause in a record definition
- The WHILE statement

2.3.1 Relational Operators

With relational operators, you can compare value expressions, check whether a code string is contained in a table, and check whether a record stream is empty or not. Most Boolean expressions contain a field name, a relational operator, and a value expression. Table 2-3 shows the format for using each relational operator preceded by a field name.

Table 2-3: Relational Operators Preceded by Field Name

Boolean Expression Formats and Evaluations
field-name { = / EQ / EQUAL } {val-exp} [,...] True if the value of field name is equal to a value expression in the list.
field-name NOT EQUAL {val-exp} [,...] True if the value of field name is not equal to any value expression in the list.
field-name { NE / NOT-EQUAL / NOT_EQUAL } val-exp True if the value of field name is not equal to the value expression.
field-name { > / GT / GREATER-THAN / GREATER_THAN } val-exp True if the value of field name is greater than the value expression.
field-name { GE / GREATER-EQUAL / GREATER_EQUAL } val-exp True if the value of field name is greater than or equal to the value expression.
field-name { < / LT / LESS-THAN / LESS_THAN } val-exp True if the value of field name is less than the value expression.
field-name { LE / LESS-EQUAL / LESS_EQUAL } val-exp True if the value of field name is less than or equal to the value expression.
field-name { BT / BETWEEN } val-exp-1 [AND] val-exp-2 True if the value of field name is between value expression 1 and value expression 2. Value expression 1 must be less than value expression 2.

(continued on next page)

Table 2-3: Relational Operators Preceded by Field Name (Cont.)

Boolean Expression Formats and Evaluations
field-name {NOT BT / NOT BETWEEN} val-exp-1 [AND] val-exp-2 True if the value of field name is not between value expression 1 and value expression 2. Value expression 1 must be less than value expression 2.
field-name {CONT / CONTAINING} val-exp True if the value of field name contains the value expression as a substring.
field-name {NOT CONT / NOT CONTAINING} val-exp True if the value of field name does not contain the value expression as a substring.
field-name IN table-name True if the value of field name is a code string in table name.
field-name NOT IN table-name True if the value of field name is not a code table name.
ANY rse True if the record stream is not empty.

Note that you can specify more than one value expression only after the =, EQ, EQUAL, and NOT EQUAL operators. DATATRIEVE displays an error message if you specify more than one value expression after any other operator:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS↵
DTR> FIND A IN YACHTS WITH BEAM EQ 9, 10, 11↵
[70 records found]
DTR> FIND B IN YACHTS WITH BEAM NE 12, 13, 14↵
Expected end of statement, encountered ","
DTR>
```

In a Boolean expression, you can also use a prompting value expression or a variable name in place of a field name. You cannot, however, use any other value expressions in place of the field name. Generally, you use relational operators to compare field values to value expressions.

The order and value associated with numeric and alphanumeric characters is determined by the ASCII collating sequence. Appendix B lists printing ASCII characters and their corresponding values. Because lowercase letters have a higher ASCII value than uppercase letters, DATATRIEVE treats lowercase letters in character string literals as greater than uppercase letters.

In Boolean expressions using the CONTAINING relational operator, the comparison of the value expression and the field value is not case-sensitive. That is,

uppercase letters and lowercase letters are treated the same, whether or not you enclose a character string literal in quotation marks.

The following examples show the use of relational operators to compare field values to value expressions:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS↵
DTR> FIND YACHTS WITH BEAM EQ 9, 10, 14↵
[50 records found]
DTR> PRINT CURRENT WITH RIG NE "SLOOP"↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL    WEIGHT BEAM  PRICE

 EASTWARD       HO        MS     24      7,000  09  $15,900
 FISHER         30        KETCH  30     14,500  09
 GRAMPIAN       34        KETCH  33     12,000  10  $29,675

DTR> PRINT COUNT OF YACHTS WITH BUILDER CONTAINING "EE"↵
    2

DTR> FIND YACHTS WITH LOA < 20↵
[2 records found]
DTR> FIND YACHTS WITH PRICE BETWEEN 7000 AND 10000↵
[2 records found]
DTR>
```

The IN relational operator compares the contents of a field with the code strings in a dictionary table. This comparison is useful for validating data you assign to fields or variables. You can, for example, write a record definition that uses a table to validate the data before it is stored:

```
DTR> DEFINE RECORD PHONE_REC USING↵
DFN> 01 PHONE.↵
DFN>     02 NAME PIC X(20).↵
DFN>     02 NUMBER PIC 9(7) EDIT_STRING IS XXX-XXXX.↵
DFN>     02 LOCATION PIC X(9).↵
DFN>     02 DEPARTMENT PIC XX↵
DFN>             VALID IF DEPARTMENT IN DEPT_TABLE.↵
DFN> ;↵
DTR>
```

The relational operator ANY checks whether a record stream is empty. This operator is useful for work with lists in hierarchical records. The record selection expression following ANY generally specifies the name of a list or sublist:

```
DTR> SET NO PROMPT↵
DTR> READY FAMILIES↵
DTR> PRINT FAMILIES WITH ANY KIDS WITH AGE = 20↵

                        NUMBER     KID
  FATHER      MOTHER     KIDS      NAME     AGE

BASIL        MERIDETH      6    BEAU        28
                                BROOKS      26
                                ROBIN       24
                                JAY         22
                                WREN        17
```

(continued on next page)

```
                                    JILL        20
JEROME      RUTH              4     ERIC        32
                                    CISSY       24
                                    NANCY       22
                                    MICHAEL     20

DTR> PRINT FAMILIES WITH ANY KIDS WITH KID_NAME CONT "RAL"RET

                          NUMBER     KID
  FATHER      MOTHER       KIDS      NAME      AGE

JIM           ANN              2    URSULA      7
                                    RALPH       3

DTR>
```

For more information on lists and hierarchies, see Chapter 15 of the *DATATRIEVE-11 User's Guide*.

2.3.2 Boolean Operators

DATATRIEVE recognizes four Boolean operators: AND, OR, BUT, and NOT. With AND, OR, and BUT, you can join two or more Boolean expressions together to form a single Boolean expression. NOT allows you to reverse the value of a Boolean expression.

When you link Boolean expressions with AND or BUT, the resulting expression is true only if all the Boolean expressions linked with AND or BUT are true. When you link Boolean expressions with OR, the resulting expression is true if any one of the Booleans linked with OR is true. When you precede a Boolean expression with NOT, the resulting expression is true if the Boolean expression following NOT is false:

```
DTR> SET NO PROMPTRET
DTR> READY YACHTSRET
DTR> PRINT YACHTS WITH BUILDER = "PEARSON" AND LOA = 30RET

                                  LENGTH
                                   OVER
MANUFACTURER    MODEL      RIG     ALL    WEIGHT BEAM  PRICE

 PEARSON        30        SLOOP   30      8,320  09

DTR> FIND YACHTS WITH BUILDER = "PEARSON" OR LOA = 30RET
[21 records found]
DTR> PRINT FAMILIES WITH FATHER NOT EQ "JIM" ANDRET
CON> ANY KIDS WITH AGE GT 31RET

                          NUMBER     KID
  FATHER      MOTHER       KIDS      NAME      AGE

JEROME        RUTH             4    ERIC        32
                                    CISSY       24
                                    NANCY       22
                                    MICHAEL     20
HAROLD        SARAH            3    CHARLIE     31
                                    HAROLD      35
                                    SARAH       27

DTR>
```

2.3.3 Compound Boolean Expressions

Use parentheses to group Boolean expressions into compound Boolean expressions:

```
DTR> SET NO PROMPT(RET)
DTR> PRINT YACHTS WITH(RET)
CON> (MODEL = "28" AND BUILDER = "TANZER") OR(RET)
CON> (MODEL = "BALLAD" AND BUILDER = "ALBIN")(RET)

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG       ALL    WEIGHT BEAM  PRICE

 ALBIN         BALLAD     SLOOP     30      7,276  10  $27,500
 TANZER        28         SLOOP     28      6,800  10  $17,500

DTR>
```

DATATRIEVE evaluates Boolean expressions in parentheses before evaluating other Boolean expressions. If a Boolean expression contains Boolean operators as well as parentheses, DATATRIEVE evaluates the compound expression in the following order:

1. Expressions enclosed in parentheses
2. Expressions preceded by NOT
3. Expressions combined with AND
4. Expressions combined with OR

Table 2-4 shows how DATATRIEVE evaluates compound Boolean expressions.

Table 2-4: Compound Boolean Expressions

Expressions and Evaluations
bool-1 AND bool-2 AND bool-3 True if all three Boolean expressions are true
bool-1 AND (bool-2 OR bool-3) True if Boolean 1 is true and either Boolean 2 or Boolean 3 is true
(bool-1 AND bool-2) OR bool-3 True if Boolean 1 and Boolean 2 are true or if Boolean 3 is true
(bool-1 AND bool-2) OR (bool-3 AND bool-4) True if both Boolean 1 and Boolean 2 are true or if both Boolean 3 and Boolean 4 are true
NOT (bool-1 OR bool-2) AND bool-3 True if both Boolean 1 and Boolean 2 are false and Boolean 3 is true

The Record Selection Expression (RSE) 3

3.1 Introduction

This chapter describes how to use the DATATRIEVE-11 record selection expression (RSE) to form record streams and collections. In the RSE, you specify the conditions that DATATRIEVE uses to form subsets and combinations of records that you can output, report, and change. When you use RSEs to specify records, DATATRIEVE operations execute faster than when you use collections to specify records; see Chapter 18 in the *DATATRIEVE-11 User's Guide* for information on optimizing your DATATRIEVE applications and using RSEs and collections.

The group of records that satisfies the conditions you specify in the RSE is called a record stream.

3.2 Format of an RSE

The format of a record selection expression as it applies to records in RMS-11 files is:

Format

```
[ FIRST n ]                    { domain-name      }
[ ALL     ]  [context-var IN]  { collection-time  }
                               { list-name        }

    [WITH boolean-expression]

        [SORTED BY sort-key [,...]]
```

This format shows that the RSE contains one required element: the source of records as specified by the domain, collection, or list name. There are also four

optional elements that affect the characteristics of the record stream. The following sections describe each element of an RSE.

3.3 Specifying the Record Source

The name of the record source is the only required element in a record selection expression. The record source tells DATATRIEVE which domain, collection, or list to search when forming a record stream.

You can use the given name of any type of DATATRIEVE domain as a record source for DATATRIEVE to search when forming a record stream. Before you can use a domain name in an RSE, you must ready the domain with a READY command:

```
DTR> READY YACHTS
DTR> PRINT YACHTS

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL       RIG      ALL    WEIGHT BEAM  PRICE

 ALBERG       37 MK II    KETCH     37     20,000  12  $36,951
 ALBIN        79          SLOOP     26      4,200  10  $17,900
        .                   .                 .
        .                   .                 .
        .                   .                 .
 WRIGHT       SEAWIND II  SLOOP     33     14,900  09  $34,480

DTR>
```

To use a collection as the record source, you must first establish the collection with a FIND statement. To specify the most recently established collection, use the keyword CURRENT:

```
DTR> READY YACHTS
DTR> FIND YACHTS
[113 records found]
DTR> PRINT CURRENT

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL       RIG      ALL    WEIGHT BEAM  PRICE

 ALBERG       37 MK II    KETCH     37     20,000  12  $36,951
 ALBIN        79          SLOOP     26      4,200  10  $17,900
        .                   .                 .
        .                   .                 .
        .                   .                 .
 WRIGHT       SEAWIND II  SLOOP     33     14,900  09  $34,480

DTR>
```

To form a named collection to use as the record source, specify the collection name:

```
DTR> SET NO PROMPT↵
DTR> FIND BIG_ONES IN YACHTS WITH LOA > 40↵
[8 records found]
DTR> PRINT BIG_ONES WITH PRICE NE 0↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG    ALL    WEIGHT BEAM  PRICE

 CHALLENGER   41          KETCH   41     26,700  13  $51,228
 COLUMBIA     41          SLOOP   41     20,700  11  $48,490
 GULFSTAR     41          KETCH   41     22,000  12  $41,350
 ISLANDER     FREEPORT    KETCH   41     22,000  13  $54,970
 OLYMPIC      ADVENTURE   KETCH   42     24,250  13  $80,500

DTR>
```

To retrieve, modify, or report data in the items of a list in a hierarchical domain, use the name of the list as the record source. By using a list name to specify the source of records for an RSE, you can form a record stream from the list items in a single record or in multiple records. In the following example, the SELECT statement picks out the first record in the current collection. Then the RSE in the PRINT statement identifies the first item in the KIDS list in the selected record as the record stream:

```
DTR> READY FAMILIES↵
DTR> FIND FAMILIES↵
[14 records found]
DTR> SELECT↵
DTR> PRINT AGE OF FIRST 1 KIDS↵

AGE

 7

DTR>
```

To form a stream of records containing lists, use the FOR statement. In the following example, the FOR statement forms a record stream containing a list of families with 2 children. Then the RSE in the PRINT statement identifies the KIDS list in the record stream as the record source to search for children older than 20:

```
DTR> SET NO PROMPT↵
DTR> FOR FAMILIES WITH NUMBER_KIDS = 2↵
CON> PRINT KID_NAME, AGE OF KIDS WITH AGE GT 20↵

   KID
   NAME     AGE

ANN         29
JEAN        26
MARTHA      30
TOM         27

DTR>
```

For more information on using the PRINT statement to display items from lists, see the section on the PRINT statement in this manual. Refer also to Chapter 15 of the *DATATRIEVE-11 User's Guide* for additional examples illustrating the use of hierarchical records.

Note that you cannot use a list name as the record source for an ERASE statement. To change or erase fields in a list, you must establish a valid record context with a SELECT statement or a FOR statement and use the MODIFY statement. The following example shows how to establish a valid record context and how to use CTRL/Z to abort the modification without changing the record:

```
DTR> SET NO PROMPT(RET)
DTR> READY FAMILIES WRITE(RET)
DTR> SHOW FAMILY-REC(RET)
RECORD FAMILY_REC
01 FAMILY.
   03 PARENTS.
      06 FATHER PIC X(10).
      06 MOTHER PIC X(10).
   03 NUMBER_KIDS PIC 99 EDIT_STRING IS Z9.
   03 KIDS OCCURS 0 TO 10 TIMES DEPENDING ON NUMBER_KIDS.
      06 EACH_KID.
         09 KID_NAME PIC X(10) QUERY_NAME IS KID.
         09 AGE PIC 99 EDIT_STRING IS Z9.
;
DTR> ERASE ALL OF KIDS(RET)
Field "KIDS" is undefined or used out of context
DTR> MODIFY EACH_KID OF FIRST 1 KIDS(RET)
Field "KIDS" is undefined or used out of context
DTR> FOR FIRST 2 FAMILIES MODIFY EACH_KID OF KIDS(RET)
Enter KID_NAME: ^Z
Execution terminated by operator
DTR> FIND FIRST 2 FAMILIES(RET)
[2 records found]
DTR> SELECT(RET)
DTR> MODIFY EACH_KID OF KIDS(RET)
Enter KID_NAME: ^Z
Execution terminated by operator
DTR>
```

3.4 Specifying Record Stream Characteristics

Once you have identified the record source, you can use the four optional elements of the RSE to specify the characteristics of the record stream. You can use any or all of these optional elements in an RSE, but you must specify them in the order shown in the syntax diagram in Section 3.1. The following sections describe each optional element.

3.4.1 Restricting the Number of Records

The keywords FIRST and ALL allow you to specify how many records are to be in the record stream formed by the RSE. If you omit this element of the RSE or specify ALL, the record stream consists of all records that satisfy the conditions

of the RSE. If you specify FIRST n, where n is a value expression that evaluates to a positive integer, the record stream consists of n records that satisfy the conditions of the RSE:

```
DTR> PRINT FIRST 3 YACHTS↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG   ALL   WEIGHT BEAM  PRICE

 ALBERG       37 MK II   KETCH   37    20,000  12  $35,000
 ALBIN        79         SLOOP   26     4,200  10  $17,900
 ALBIN        BALLAD     SLOOP   30     7,276  10  $27,500

DTR>
```

If n does not evaluate to an integer, DATATRIEVE truncates the fractional part of the value and uses the integer part as the value for n. When n is greater than the number of records satisfying the conditions of the RSE, the record stream consists of as many records as satisfy the conditions. For example:

```
DTR> SET NO PROMPT↵
DTR> PRINT MODEL, PRICE OF FIRST 5 YACHTS WITH↵
CON> BUILDER = "AMERICAN"↵

  MODEL      PRICE
26           $9,895
26-MS       $18,895

DTR>
```

When you specify a sort order in the RSE, DATATRIEVE sorts the records as specified. In these examples, DATATRIEVE selects the first three records as the record stream:

```
DTR> READY YACHTS↵
DTR> PRINT FIRST 3 YACHTS↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG   ALL   WEIGHT BEAM  PRICE

 ALBERG       37 MK II   KETCH   37    20,000  12  $36,951
 ALBIN        79         SLOOP   26     4,200  10  $17,900
 ALBIN        BALLAD     SLOOP   30     7,276  10  $27,500

DTR> PRINT FIRST 3 YACHTS SORTED BY LOA↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG   ALL   WEIGHT BEAM  PRICE

 WINDPOWER    IMPULSE    SLOOP   16       650  07   $3,500
 CAPE DORY    TYPHOON    SLOOP   19     1,900  06   $4,295
 ENCHILADA    20         SLOOP   20     2,300  07

DTR>
```

3.4.2 Naming a Record Stream

You can give a record stream a name by including a context variable in the RSE. Using context variables to qualify field names also lets you refer to fields from different record streams in the same statement. Chapter 12 of the *DATATRIEVE-11 User's Guide* provides additional information on the use of context variables and qualified field names.

When you use a context variable in a FIND statement, the context variable becomes the name of the collection. Until you form another collection or release the one just formed, it has two names: the name of the context variable and CURRENT (because it is the most recently formed collection):

```
DTR> READY YACHTS
DTR> FIND BIGGIES IN YACHTS WITH LOA GT 40
[8 records found]
DTR> SHOW COLLECTIONS
Collections:
        BIGGIES (also CURRENT)

DTR>
```

When you use nested FOR loops to join domains sharing one or more field names, you can use context variables in RSEs to specify which of the identically named fields you want DATATRIEVE to act on. The RSE in the second FOR statement in the following example compares all yachts to find yachts that have the same specifications:

```
DTR> SET NO PROMPT
DTR> FOR A IN YACHTS
CON> FOR B IN YACHTS WITH B.TYPE NE A.TYPE AND B.SPECS = A.SPECS
CON> PRINT A.BOAT

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG   ALL   WEIGHT BEAM  PRICE

 CARIBBEAN      35         SLOOP 35    18,000  11  $37,850
 CHRIS-CRAF     CARIBBEAN  SLOOP 35    18,000  11  $37,850
 SCAMPI         30         SLOOP 30     6,600  10
 SOLNA CORP     SCAMPI     SLOOP 30     6,600  10

DTR>
```

3.4.3 Specifying Conditions for the Record Stream

The WITH clause allows you to specify conditions for the record stream. The Boolean expression in the WITH clause specifies:

- Values for DATATRIEVE to compare with values in records
- The type of comparison for DATATRIEVE to perform when searching through the record source

Any Boolean expression is valid in a WITH clause. See Chapter 2 for information on forming Boolean expressions.

When the values of a record satisfy the conditions specified in the Boolean expression (that is, when the Boolean expression is true), the record becomes part of the record stream formed by the RSE:

```
DTR> SET NO PROMPT↵
DTR> PRINT YACHTS WITH BUILDER = "ALBIN"↵

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG       ALL    WEIGHT BEAM  PRICE

 ALBIN          79         SLOOP     26      4,200  10  $17,900
 ALBIN          BALLAD     SLOOP     30      7,276  10  $27,500
 ALBIN          VEGA       SLOOP     27      5,070  08  $18,600

DTR> READY FAMILIES↵
DTR> FIND FAMILIES WITH ANY KIDS WITH AGE > 30↵
[3 records found]
DTR> PRINT CURRENT WITH ANY KIDS WITH↵
CON> KID_NAME = "ELLEN"↵

                         NUMBER     KID
  FATHER       MOTHER     KIDS      NAME      AGE

JIM          LOUISE          5    ANNE        31
                                  JIM         29
                                  ELLEN       26
                                  DAVID       24
                                  ROBERT      16

DTR>
```

You can also use a comma to form a compound Boolean expression having an implied OR:

```
DTR> FIND YACHTS WITH RIG = "MS", "KETCH"↵
[18 records found]
DTR>
```

3.4.4 Sorting the Records

The SORTED BY clause lets you sort the records in the record stream. You can sort the records according to the values in one or more fields or according to a related value expression. The SORTED BY clause has this format:

SORTED [BY] sort-key,...

In the sort key, you can specify both the order of the sort with a DATATRIEVE keyword and the field to be used as the basis of the sort. The sort-key has this format:

```
{ ASC[ENDING]  }
{ DESC[ENDING] } { field-name  [,...] }
{ INCREASING   }
{ DECREASING   }
```

If you do not include a keyword to specify the order of the sort with the first sort key, the default order is ascending. If you do not specify the sort order, DATATRIEVE uses the sort order that applied to the preceding field. As the format indicates, you must include at least one field name in the sort key. Additional sort keys must be separated by commas.

The keywords ASC, ASCENDING, and INCREASING are equivalent and specify that the sorted values begin with the smallest and end with the largest:

```
DTR> READY YACHTS↵
DTR> FIND FIRST 3 YACHTS WITH PRICE NE 0↵
[3 records found]
DTR> PRINT BUILDER, LOA, PRICE OF CURRENT SORTED BY ASC PRICE↵

              LENGTH
               OVER
MANUFACTURER   ALL     PRICE

 ALBIN         26     $17,900
 ALBIN         30     $27,500
 ALBERG        37     $36,951

DTR>
```

The keywords DESC, DESCENDING, and DECREASING are also equivalent and specify that the sorted values begin with the largest and end with the smallest:

```
DTR> PRINT BUILDER, LOA, PRICE OF CURRENT SORTED BY DESC PRICE↵

              LENGTH
               OVER
MANUFACTURER   ALL     PRICE

 ALBERG        37     $36,951
 ALBIN         30     $27,500
 ALBIN         26     $17,900

DTR>
```

The value of characters for alphanumeric sorts is determined by the ASCII values of the characters. See Chapter 2 and Appendix B for information on the ASCII sorting sequence.

When you specify multiple sort keys, DATATRIEVE treats the first field in the sort list as the major sort key and successive field or value expressions as minor keys:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS↵
DTR> FIND YACHTS WITH LOA BT 28 AND 30↵
[24 records found]
DTR> FIND Y IN CURRENT WITH PRICE NE 0↵
[11 records found]
DTR> PRINT BUILDER, LOA, PRICE OF Y SORTED BY↵
CON> ASC LOA, DESC PRICE↵
```

(continued on next page)

```
              LENGTH
               OVER
MANUFACTURER   ALL     PRICE

 SABRE          28     $22,000
 CAPE DORY      28     $21,990
 TANZER         28     $17,500
 ISLANDER       28     $15,908
 GRAMPIAN       28     $14,475
 NORTHERN       29     $20,975
 ALBIN          30     $27,500
 HUNTER         30     $21,500
 ISLANDER       30     $20,990
 IRWIN          30     $19,950
 GRAMPIAN       30     $17,775

DTR>
```

Chapter 3 of the *DATATRIEVE-11 Guide to Writing Reports* discusses sorting with multiple sort keys to form control groups.

Record Definitions 4

4.1 Introduction

In DATATRIEVE, you define a logical record with the DEFINE RECORD command. The logical record is the basic unit that helps you store and interpret data. To define the logical record, you combine field definition clauses that specify the characteristics of each field. This combination of fields determines the structure of the record and represents the relationships among the items of data you store in the fields.

For simple records, you can use the interactive Application Design Tool (ADT). Chapter 3 of the *Introduction to DATATRIEVE-11* explains the use of ADT. ADT does not allow you to use all the available field definition clauses, however, so you cannot use it to define all types of records.

This chapter briefly explains the elements of a record definition and the record and field definition clauses you can use to define records for DATATRIEVE domains. For more information about definition clauses, see the alphabetic listing for each clause in Chapter 5 in this manual. For more information on record and field definitions, see Chapter 5 of the *DATATRIEVE-11 User's Guide.*

4.2 Components of a Record Definition

A record definition consists of one or more field definitions. Each field definition describes the field and its relationship to other fields in the record.

Every field definition consists of three parts:

- A level number that determines the relationship between the field and other fields in the record

- A field name that identifies the field
- A period (.) that signals the end of the field definition

Most field definitions also contain one or more field definition clauses.

4.3 Types of Fields

A record definition can contain both elementary and group fields. An elementary field is the basic unit of data. It contains no other field within it. A group field, on the other hand, contains elementary or group fields.

Every record definition must conform to the following rules for including elementary and group fields:

- A record definition must contain at least one elementary field.
- If a record contains more than one elementary field, it must contain a group field that includes all other fields in the record.
- A group field must contain at least one other field.

4.3.1 Field Levels

Level numbers determine the relationship among fields in a record definition. The YACHT record definition is shown in Figure 4-1.

Figure 4-1 The YACHT Record Definition

```
01 BOAT
        03 TYPE
                06 MANUFACTURER
                06 MODEL
        03 SPECIFICATIONS
                06 RIG
                06 LENGTH_OVER_ALL
                06 DISPLACEMENT
                06 BEAM
                06 PRICE
```

A group field that contains all other fields in a record is at the first or top level. In the YACHT record, the group field BOAT contains all other fields in the record and is the top level field.

Second level fields are contained only in a top level field and in no other group field. TYPE and SPECIFICATIONS are both second level fields contained in and subordinate to the top level field BOAT. TYPE and SPECIFICATIONS are also group fields because they contain elementary fields. Fields subordinate only to a second level field are third level fields, and so on. MANUFACTURER and MODEL in the YACHT record, for example, are third level fields. MANUFACTURER and MODEL are also elementary fields since they do not contain any lower-level fields. RIG, LENGTH_OVER_ALL, DISPLACEMENT, BEAM, and PRICE are also third level elementary fields.

4.3.2 Level Numbers

Every field in a record definition must have a level number that specifies its relationship to the other fields in the record. DATATRIEVE recognizes field levels in a record definition according to the level numbers you assign to each field. Thus, the level number must be the first element of a field definition. The number of the highest possible level is 1, and the number of the lowest possible level is 65. Leading zeros, as in 01 and 05, do not affect the value of the level number.

The top level field must have the smallest level number assigned to any field in the record definition. No other field in the record can have the same level number as the top level field. The level number usually assigned to the top level field is 01. Any field with a higher level number is subordinate to the top level field.

In the YACHT record definition, BOAT is the top level field and is the only field with the level number 01. TYPE and SPECIFICATIONS are second level group fields and have the same 03 level number. The seven elementary fields are all third level fields and have a level number of 06. MANUFACTURER and MODEL are subordinate to the TYPE field, while the other five elementary fields are subordinate to SPECIFICATIONS.

The level numbers in the YACHT record definition illustrate several rules about assigning level numbers to fields:

- Only level numbers determine the relationships among fields.
- Level numbers need not be consecutive. Only the relative value of level numbers determines the relationship between fields. TYPE and SPECIFICATIONS, for example, could have level numbers of 02, 04, or 05 and the YACHT record structure would still be the same as defined in Figure 4-1.
- Differences in level numbers mean differences in levels and differences in relationship. Fields must have the same level number to be equivalent. If, for example, the elementary field LENGTH_OVER_ALL in the SPECIFICATIONS field had a level number of 07, the field immediately preceding it, RIG, would become a group field containing LENGTH_OVER_ALL.

Fields in the YACHTS record definition and in other examples in this book are indented to show their relationship to other fields. Indenting fields helps clarify the structure of the record, but has no effect on the relationship among fields.

4.4 Field Names

In addition to a level number, every field in a record must have a field name. You use the field name to identify the field in DATATRIEVE statements and DATATRIEVE uses the field name when printing the contents of the field. A field name must conform to DATATRIEVE's rules for names:

- A name can consist of from 1 to 31 letters, digits, hyphens (-), and underscores (_).
- A name must begin with a letter and must end with a letter or digit.

- A name cannot duplicate a DATATRIEVE keyword. See Appendix A for a list of DATATRIEVE keywords.
- A name can be continued from one line to another only by using a hyphen (-), the DATATRIEVE continuation character.
- A name can duplicate another field name in the same record definition only if the duplicate field names are in different group fields. Duplicating field names in a record is not a good data management practice.

See Chapter 5 of the *DATATRIEVE-11 User's Guide* for a more detailed list of rules for DATATRIEVE names.

4.4.1 QUERY_NAME and QUERY_HEADER Clauses

By using the QUERY_NAME clause in a field definition, you can specify an alternate name for a field. You can then use the query name in place of the field name in DATATRIEVE statements. The YACHT record definition, for example, specifies the query name BUILDER for the MANUFACTURER field and the query name SPECS for the SPECIFICATIONS field. See Chapter 5 for more information on the QUERY_NAME clause.

You can also specify a QUERY_HEADER for a field. If you do not include a QUERY_HEADER clause, DATATRIEVE uses the field name as the column header for the field when displaying data. If you include a QUERY_HEADER clause in the field definition, DATATRIEVE uses the query header, instead of the field name, for the column header. You can override the printing of the field name or query header by specifying a column header modifier as part of a print list element in the PRINT, SUM, and REPORT statements. See the sections in Chapter 5 on these statements for more information.

4.4.2 FILLER Field Name

By using the FILLER keyword as the name of an elementary or group field, you can mask fields in a data file:

- Use FILLER as an elementary field to reserve space in the physical record of the data file or to suppress the display of fields that you do not want to lose, but do not want or need to display.
- Use FILLER as a group field to access elementary fields in the group and to suppress display of the entire group field.

You cannot retrieve data from or store data in a FILLER field.

Like other fields, a field named FILLER must have a level number. It can also contain field definition clauses. You can use the name FILLER for more than one field at the same level in a group field. When you use the PRINT, MODIFY, STORE, REPORT, and SUM statements to retrieve, update, or store the contents of a record, DATATRIEVE ignores values in FILLER fields.

Do not use FILLER fields, however, to protect sensitive information stored in physical records. The DISPLAY statement displays all the contents of a group field, regardless of the field names in the record definition.

If the FILLER field is an elementary field, you cannot get access to data in the field with any statement except DISPLAY. If FILLER is the name of a group field, you can access data in the physical record by specifying the name of an elementary or group field in the FILLER group field. Each of those fields has its own valid name, and you can retrieve the value by specifying that name in a record selection expression, a print list, or a field list. You cannot access data in the physical record by using the group name (FILLER) or by retrieving for output whole records or group fields containing the group field named FILLER. DATATRIEVE stops accessing fields in a group when it encounters the name FILLER and moves to the next field at the same level or at a higher level.

The following example shows the use of FILLER fields to mask two elementary fields of the PERSONNEL data file, EMPLOYEE_STATUS and SALARY, for two different types of retrieval from the same data file:

```
DTR> SHOW PERSONNEL↵
DOMAIN PERSONNEL
 USING PERSONNEL_REC ON PERSON.DAT;
DTR> SHOW PERSONNEL_M↵
DOMAIN PERSONNEL_M
 USING PERSONNEL_MASK ON PERSON.DAT;
DTR> SHOW PERSONNEL_REC↵
RECORD PERSONNEL_REC
USING
01   PERSON.
     05 ID                      PIC IS 9(5).
     05 EMPLOYEE_STATUS         PIC IS X(11)
                                QUERY_NAME IS STATUS
                                QUERY_HEADER IS "STATUS"
                                VALID IF STATUS EQ "TRAINEE","EXPERIENCED".
     05 EMPLOYEE_NAME           QUERY_NAME IS NAME.
         10 FIRST_NAME              PIC IS X(10)
                                   QUERY_NAME IS F_NAME.
         10 LAST_NAME               PIC IS X(10)
                                   QUERY_NAME IS L_NAME.
     05 DEPT                    PIC IS XXX.
     05 START_DATE              USAGE IS DATE.
     05 SALARY                  PIC IS 9(5)
                                EDIT_STRING IS $$$,$$$.
     05 SUP_ID                  PIC IS 9(5).
;
DTR> SHOW PERSONNEL_MASK↵
RECORD PERSONNEL_MASK
USING
01   PERSON.
     05 ID                      PIC IS 9(5).
     05 FILLER                  PIC IS X(11)
                                QUERY_NAME IS STATUS
                                QUERY_HEADER IS "STATUS"
                                VALID IF STATUS EQ "TRAINEE","EXPERIENCED".
     05 EMPLOYEE_NAME           QUERY_NAME IS NAME.
         10 FIRST_NAME              PIC IS X(10)
                                   QUERY_NAME IS F_NAME.
         10 LAST_NAME               PIC IS X(10)
                                   QUERY_NAME IS L_NAME.
     05 DEPT                    PIC IS XXX.
```

(continued on next page)

```
    05 START_DATE            USAGE IS DATE.
    05 FILLER                PIC IS 9(5)
                             EDIT_STRING IS $$$,$$$.
    05 SUP_ID                PIC IS 9(5).
;
DTR> READY PERSONNEL
DTR> READY PERSONNEL_M
DTR> FIND FIRST 5 A IN PERSONNEL
[5 records found]
DTR> FIND FIRST 5 B IN PERSONNEL_M
[5 records found]
DTR> PRINT A

                    FIRST        LAST             START              SUP
 ID      STATUS      NAME        NAME     DEPT    DATE      SALARY   ID

00012 EXPERIENCED CHARLOTTE  SPIVA       TOP  12-Sep-72   $75,892 00012
00891 EXPERIENCED FRED       HOWL        F11  9-Apr-76    $59,594 00012
02943 EXPERIENCED CASS       TERRY       D98  2-Jan-80    $29,908 39485
12643 TRAINEE     JEFF       TASHKENT    C82  4-Apr-81    $32,918 87465
32432 TRAINEE     THOMAS     SCHWEIK     F11  7-Nov-81    $26,723 00891

DTR> PRINT B

        FIRST        LAST             START      SUP
 ID      NAME        NAME     DEPT    DATE       ID

00012 CHARLOTTE  SPIVA       TOP  12-Sep-72   00012
00891 FRED       HOWL        F11  9-Apr-76    00012
02943 CASS       TERRY       D98  2-Jan-80    39485
12643 JEFF       TASHKENT    C82  4-Apr-81    87465
32432 THOMAS     SCHWEIK     F11  7-Nov-81    00891

DTR>
```

4.5 Field Classes

DATATRIEVE classifies fields in a record by the type of data contained in the field and by the way data is stored. Table 4-1 summarizes field classes and their content.

Table 4-1: Field Classes and Content

Field Type	Class	Content
Elementary	Alphanumeric	Any valid ASCII character
	Numeric	Any combination of digits and an optional sign (+ or –)
	DATE	A date
	COMPUTED BY	None; the field definition specifies a value expression, but no value is stored in the record
Group	Alphanumeric	The values of the fields contained in the group field

4.6 Field Definition Clauses

When you define a field, you specify its characteristics with one or more optional field definition clauses. A field definition clause consists of a keyword, such as PICTURE or QUERY_NAME, followed by a character string or value expression.

Use field definition clauses to specify the following field characteristics:

- The class and length of the data and the format in which the data is stored (PICTURE, USAGE, COMPUTED BY, OCCURS)
- The format for DATATRIEVE to use when writing the data to a file or output device (EDIT_STRING, QUERY_HEADER, SIGN)
- The values accepted when you store data in the field (VALID IF)
- The way DATATRIEVE computes numeric values when you refer to the field (USAGE, COMPUTED BY)
- An alternate and equivalent name for the field (QUERY_NAME)
- An alternate way to DEFINE another field in the record (REDEFINES)

DATATRIEVE-11 field definition clauses are summarized in Table 4-2.

Table 4-2: Summary of Field Definition Clauses

Clause	Type of Field	Purpose
COMPUTED BY	Elementary	Describes a COMPUTED BY field
EDIT_STRING	Elementary	Specifies the format of a value when DATATRIEVE writes a field value to a file or output device
OCCURS	Elementary or group	Defines multiple occurrences of a field or group of fields (see Chapter 5 of the *DATATRIEVE-11 User's Guide*)
PICTURE	Elementary	Specifies the data type, length, and format of values stored in the field
QUERY_HEADER	Elementary	Specifies the column header for a field when DATATRIEVE writes one or more field values to a file or output device
QUERY_NAME	Elementary or group	Specifies an alternate name for a field
REDEFINES	Elementary or group	Creates an alternate definition for a field in the record
SIGN	Numeric elementary	Specifies the location and representation of the sign in a numeric field
USAGE	Numeric or date elementary	Specifies the length and data type of a numeric field or specifies a date field
VALID IF	Elementary	Tests a value against conditions specified in a Boolean expression before storing the value in the field

When you write field definitions, you should observe these rules and guidelines for using field definition clauses:

- You must include a level number and a field name in the definition of a field.
- You must include a PICTURE, COMPUTED BY, or USAGE clause in the definition of an elementary field.
- You do not have to include a PICTURE clause for any DATATRIEVE data type. When you do specify a PICTURE clause in a field definition, DATATRIEVE treats it as an edit string if no EDIT_STRING clause is present.
- You can use all the field definition clauses listed in Table 4-2 for elementary fields. However, you can use only REDEFINES, QUERY_NAME, and OCCURS clauses for group fields. DATATRIEVE ignores any PICTURE or USAGE clause included in a group field definition.
- You can put one or more field definition clauses on the same input line as the level number and field name. You can also put each field definition clause on one or more input lines. Separate each field definition clause from the next by entering a space, a tab, or a carriage return.
- You can enter field definition clauses in any order.
- You can indent field definition clauses with spaces or tabs or enter them on separate lines. This does not change the characteristics of the field, but does make record definitions easier to read and understand.
- You can use only one OCCURS...DEPENDING clause in a record definition. No other field definition can follow the last elementary field in a group field containing this clause. That is, only subordinate fields can be defined after the OCCURS...DEPENDING clause.
- You must end each field definition with a period. If the field is a group field with no field definition clause, place the period immediately after the field name. If the field definition contains one or more clauses, place the period after the last clause.

You can find detailed descriptions of field definition clauses in the alphabetical listings of Chapter 5.

Commands, Statements, and Definition Clauses 5

5.1 Introduction

This chapter describes all DATATRIEVE-11 commands, statements, record definition clauses, and field definition clauses. Table 5-1 lists the commands, statements, and clauses in alphabetical order. Table 5-2 lists frequently performed functions and the commands, statements, and clauses you can use to perform them.

The remainder of the chapter describes each command, statement, and clause, in alphabetical order. Each section of the chapter has the same format and consists of the following categories of information:

- **Function**

  The function statement briefly describes the function or effect of the command, statement, or clause.

- **Format**

  The format indicates the spelling and placement of keywords and the placement of required and optional syntax elements for the entry. As a rule, you cannot abbreviate command and statement names and other keywords. The sequence of syntax elements is also critical. You must follow the sequence shown in the format. If you omit an optional element, leave its relative position in the command, statement, or clause empty and proceed to the remaining elements in a left-to-right order. If necessary, review the documentation conventions listed at the end of this section.

- **Arguments**

  Arguments describe the components of the command, statement, or clause.

- **Restrictions**

  Restrictions tell you what requirements and limits there are on the use and action of the command, statement, or clause. They also list the access privileges you must have to use the entry.

- **Results**

  Results tell you what action DATATRIEVE takes when you use the command, statement, or clause.

- **Usage Notes**

  Usage Notes describe common uses of the command, statement, or clause and indicate what other language elements you can use in conjunction with the command, statement, or clause.

- **Examples**

  Examples show the use of representative sequences of commands, statements, and clauses. In the case of clauses, they show sample record or field definitions.

The following symbols and conventions are used in syntax formats:

Convention	Meaning
UPPERCASE WORDS	Uppercase words in the text are DATATRIEVE keywords.
lowercase words	Lowercase words are generic terms that indicate entries you must provide.
{}	Braces mean you **must** choose one, but **no more than one**, of the enclosed entries.
[]	Brackets mean you have the **option** of choosing one, but no more than one, of the enclosed entries.
...	A horizontal ellipsis means you have the option of repeating the preceding element of the syntax format.
. . .	A vertical ellipsis means you have the option of repeating an element of the syntax format on the succeeding line.

Table 5-1: Alphabetical Summary of Commands, Statements, and Clauses

Language Element	Function
(CO = Command) (CL = Clause) (S = Statement) (RW = Report Writer)	
:(Invoke Procedure) (CO)	Invokes a DATATRIEVE procedure
@(Invoke Command File) (CO)	Invokes a command file
ABORT (S)	Ends statement, procedure, or command file execution
ADT (CO)	Invokes Application Design Tool (ADT)
ALLOCATION (CL)	Specifies type of word boundary alignment for storing and retrieving records
Assignment (S)	Assigns a value to an elementary field, group field, or variables
AT BOTTOM (RW)	Prints summary lines at the bottom of a report, a page, or a control group
AT TOP (RW)	Prints header lines or summary lines at the top of a report, a page, or a control group
BEGIN-END (S)	Groups statements into one compound statement
CLOSE (CO)	Closes the file created by the OPEN command
COMPUTED BY (CL)	Describes a COMPUTED BY field
DECLARE (S)	Defines a variable
DECLARE PORT (S)	Creates a temporary port
DEFINE DICTIONARY (CO)	Creates a private data dictionary
DEFINE DOMAIN (CO)	Creates a domain definition
DEFINE FILE (CO)	Creates a data file for a domain
DEFINE PORT (CO)	Creates a port definition
DEFINE PROCEDURE (CO)	Creates a procedure definition
DEFINE RECORD (CO)	Creates a record definition
DEFINE TABLE (CO)	Creates a dictionary table definition
DEFINEP (CO)	Adds an entry to an access control list (ACL)
DELETE (CO)	Removes a definition from a data dictionary
DELETEP (CO)	Removes an entry from an access control list (ACL)
DISPLAY (CO)	Displays on your terminal the unformatted value of a single value expression

(continued on next page)

Table 5-1: Alphabetical Summary of Commands, Statements, and Clauses (Cont.)

Language Element	Function
DROP (S)	Removes a record from a collection but not from its data file
EDIT (CO)	Invokes the DATATRIEVE Editor
EDIT_STRING (CL)	Specifies the format of a field value when it is printed
END_REPORT (RW)	Ends a report specification
ERASE (S)	Removes a record from an indexed or relative data file
EXIT (CTRL/Z) (CO)	Ends a DATATRIEVE session
EXTRACT (CO)	Copies the definition of a dictionary element into a command file
FIND (S)	Retrieves a record and establishes a current collection
FINISH (CO)	Ends control over and frees space occupied by a domain and its associated collections
FOR (S)	Causes execution of one or a series of statements for a specified record
HELP (CO)	Provides online assistance for a command, a statement, or a clause
IF-THEN-ELSE (S)	Executes either of two statements, depending on the evaluation of a Boolean expression
MODIFY (S)	Changes field contents of a specified record
OCCURS (CL)	Defines multiple occurrences of a field or a group of fields
OPEN (CO)	Creates a file that records user/DATATRIEVE dialogue
PICTURE (CL)	Specifies the format of a field value as it is stored in a record
PRINT (S)	Displays a value expression according to a specified format
PRINT (RW)	Specifies format of the reported data
QUERY_HEADER (CL)	Specifies the column header for the field value when it is printed
QUERY_NAME (CL)	Specifies an alternate name for a field
READY (CO)	Gives access to a domain
REDEFINES (CL)	Creates an alternate definition for a field
RELEASE (CO)	Ends control over and frees space occupied by a specified collection, table, or global variable
REPEAT (S)	Executes a statement a specified number of times
REPORT (S)	Invokes the Report Writer
SELECT (S)	Selects a record in a collection

(continued on next page)

Table 5-1: Alphabetical Summary of Commands, Statements, and Clauses (Cont.)

Language Element	Function
SET (CO)	Establishes the current data dictionary, sets the maximum number of columns per page, controls the use of forms, the display of prompts and the effect of ABORT, and starts Guide Mode
SET (RW)	Specifies the report header, name, date, length, and width of a report
SHOW (CO)	Displays information about the data dictionary and its contents
SHOWP (CO)	Displays the access control list (ACL) associated with a dictionary object
SIGN (CL)	Specifies the location and representation of the sign in a numeric field
SORT (S)	Sorts records in a collection
STORE (S)	Creates and stores a record in a domain
SUM (S)	Provides a summary of totals for a specified numeric field
THEN (S)	Joins two or more statements into a compound statement
USAGE (CL)	Specifies the internal storage format of a field or specifies a date field
VALID IF (CL)	Tests a value before storing it in a field
WHILE (S)	Causes repetition of a statement while a specified condition is true

Table 5-2: Summary of Commands, Statements, and Clauses by Function

Function	Language Element
Combining/repeating statements	
Creating a compound statement	THEN BEGIN-END
Repeating a statement	REPEAT
Executing a statement conditionally	IF-THEN-ELSE WHILE
Applying a statement to a record	FOR
Data dictionaries	
Creating a data dictionary	DEFINE DICTIONARY
Establishing a current dictionary	SET DICTIONARY
Displaying the file specification of the current dictionary	SHOW DICTIONARY
Deleting the definition of a dictionary element	DELETE
Dictionary tables	
Creating a dictionary table	DEFINE TABLE
Modifying a dictionary table	EDIT
Copying a dictionary table to a command file	EXTRACT
Releasing a dictionary table	RELEASE
Displaying dictionary table names and definitions	SHOW

(continued on next page)

Table 5-2: Summary of Commands, Statements, and Clauses by Function (Cont.)

Function	Language Element
Domains	
Readying a domain for use	READY
Creating a domain definition	DEFINE DOMAIN ADT
Creating a data file for a domain	DEFINE FILE
Deleting a domain definition	DELETE
Copying a domain definition to a command file	EXTRACT
Releasing control of a domain	FINISH
Displaying a domain name or definition	SHOW
Ending execution	ABORT
Handling records	
Printing a record/field value	PRINT REPORT SUM
Adding a record to a domain	STORE
Forming a collection of records	FIND
Selecting a record	SELECT
Sorting a record	SORT
Removing a record from a collection	DROP
Deleting a record from a file	ERASE
Changing a field value	MODIFY
Recording a DATATRIEVE session	OPEN CLOSE

(continued on next page)

Table 5-2: Summary of Commands, Statements, and Clauses by Function (Cont.)

Function	Language Element
Online assistance	
Defining a domain, record, and data file	ADT
Enabling/disabling statement prompting	SET PROMPT
Printing command/statement information	HELP
Starting Guide Mode	SET GUIDE
Access control lists (ACL)	
Deleting an entry	DELETEP
Adding an entry	DEFINEP
Printing contents on a terminal	SHOWP
Ports	
Creating a temporary port	DECLARE PORT
Defining a port	DEFINE PORT
Procedures	
Creating a procedure	DEFINE PROCEDURE
Deleting a procedure	DELETE
Modifying a procedure	EDIT
Copying a procedure to a command file	EXTRACT
Displaying procedure names and definitions	SHOW

(continued on next page)

Table 5-2: Summary of Commands, Statements, and Clauses by Function (Cont.)

Function	Language Element
Record Definitions	
Creating a record definition	DEFINE RECORD ADT
Copying a record definition to a command file	EXTRACT
Displaying a record name and definition	SHOW
Specifying a record word boundary in storage	ALLOCATION
Describing a COMPUTED BY field	COMPUTED BY
Specifying a print edit string	EDIT_STRING
Defining a multiple field occurrence	OCCURS
Specifying the format of a field value	PICTURE
Specifying a column header in a print statement	QUERY_HEADER
Specifying an alternate field name in a print statement	QUERY_NAME
Creating an alternate definition for a field	REDEFINES
Specifying a sign in a numeric field	SIGN
Specifying the internal storage format of a numeric or date field	USAGE
Specifying a VALID IF test during storage	VALID IF
Terminating DATATRIEVE	EXIT (CTRL/Z)
Variables	
Defining a variable	DECLARE
Assigning a value to a variable	Assignment
Releasing a variable	RELEASE

5.2 : (Invoke Procedure)

Function

Invokes a DATATRIEVE procedure.

Format

```
{:} procedure-name
```

Arguments

procedure-name

Is the name of the DATATRIEVE procedure you want to invoke.

Restrictions

- You must have DATATRIEVE E (EXECUTE/EXTEND) access to a procedure before you can invoke it.
- You cannot invoke a procedure during an ADT, Edit or Guide Mode session.
- You cannot invoke a procedure in the definition of a domain, record, or table.
- You should not allow a procedure to invoke itself, either directly or indirectly; you may create an infinite loop.
- A procedure invoked from a BEGIN-END block cannot include a FIND, SELECT, or DROP statement.

Results

- If the procedure consists of full commands or statements, DATATRIEVE executes each command or statement in the procedure in order.
- If the procedure contains only a clause or an argument for a command or statement, DATATRIEVE includes the procedure within the command or statement invoking the procedure.

Usage Notes

- You can nest procedures by invoking a procedure within another procedure, but you must be careful not to allow the procedure to invoke itself.
- You can invoke a procedure a specified number of times by including it in a REPEAT statement. You can also apply a procedure to a collection of records by invoking the procedure in a FOR statement. You must, however, use care when invoking a procedure in these statements. For example, the following syntax can be used, but the results may be unexpected:

```
REPEAT n :procedure-name
```

This statement **does not** execute the procedure n times. When DATATRIEVE encounters the first complete statement in the procedure, it assumes that the REPEAT statement is also complete. Therefore, it executes the **first** statement in the procedure n times. DATATRIEVE then executes the remaining commands or statements in the procedure once.

- To repeat the entire procedure n times, enclose the procedure invocation in a BEGIN-END block:

```
REPEAT n
    BEGIN
        :procedure-name
    END
```

Use a similar technique for controlling procedures invoked in a FOR loop. For example:

```
FOR rse
    BEGIN
        :procedure-name
    END
```

Examples

Define and then invoke a procedure to find the employee in PERSONNEL with the largest salary:

```
DTR> DEFINE PROCEDURE MAX_SALARY↵
DFN>     READY PERSONNEL↵
DFN>     PRINT PERSONNEL WITH SALARY = MAX SALARY OF PERSONNEL↵
DFN> END_PROCEDURE↵
DTR> SHOW MAX_SALARY↵
PROCEDURE MAX_SALARY
         READY PERSONNEL
         PRINT PERSONNEL WITH SALARY = MAX SALARY OF PERSONNEL
END_PROCEDURE
DTR> :MAX_SALARY↵

                    FIRST       LAST            START              SUP
 ID     STATUS      NAME        NAME    DEPT    DATE      SALARY   ID

00012 EXPERIENCED CHARLOTTE  SPIVA      TOP  12-Sep-1972 $75,892 00012

DTR>
```

Define a procedure that displays employee records in a given department with salaries greater than $40,000, then invoke this procedure three times:

```
DTR> SET NO PROMPT↵
DTR> DEFINE PROCEDURE BIG_SALARY↵
DFN>     FOR PERSONNEL WITH DEPT  = *,"the department"↵
DFN>     BEGIN↵
DFN>       IF SALARY GT 40000↵
DFN>       THEN PRINT PERSON↵
DFN>     END↵
DFN> END_PROCEDURE↵
DTR> REPEAT 3↵
CON> BEGIN↵
CON>    :BIG_SALARY↵
CON> END↵
Enter the department: F11↵

                    FIRST       LAST            START              SUP
 ID     STATUS      NAME        NAME    DEPT    DATE      SALARY   ID

00891 EXPERIENCED FRED        HOWL       F11    9-Apr-1976 $59,594 00012
78923 EXPERIENCED LYDIA       HARRISON   F11   19-Jun-1979 $40,747 00891
Enter the department: T32↵
38462 EXPERIENCED BILL        SWAY       T32    5-May-1980 $54,000 00012
83764 EXPERIENCED JIM         MEADER     T32    4-Apr-1980 $41,029 87289
Enter the department: TOP↵
00012 EXPERIENCED CHARLOTTE   SPIVA      TOP   12-Sep-1972 $75,892 00012

DTR>
```

Invoke a procedure that specifies an edit string clause for a variable:

```
DTR> SET NO PROMPT↵
DTR> SHOW E_S↵
PROCEDURE E_S
         EDIT_STRING IS $$,$$$.99
END_PROCEDURE
DTR> DECLARE PRICE_PER_FT COMPUTED BY PRICE/LOA :E_S.↵
DTR> PRINT TYPE, PRICE_PER_FT OF FIRST 5 YACHTS↵

                          PRICE
                           PER
MANUFACTURER   MODEL        FT

 ALBERG       37 MK II     $998.67
 ALBIN        79           $688.46
 ALBIN        BALLAD       $916.66
 ALBIN        VEGA         $688.88
 AMERICAN     26           $380.57

DTR>
```

5.3 @ (Invoke Command File) Command

Function

Invokes a command file.

Format

```
@ file-spec
```

Arguments

file-spec

Is the file specification of the file you want to execute.

Restrictions

- The @ command must be on a line by itself.
- You must have operating system read access to a command file before you can invoke it.
- You cannot invoke a command file from a procedure, but you can include a command file in a procedure. When you enter @ file-spec in response to the DFN> prompt, DATATRIEVE includes the statements of the command file in the procedure as though you had just typed them yourself. The resulting procedure succeeds or fails depending on the logic of the statements in the command file.
- You cannot invoke command files while you are in ADT or Guide Mode.
- You cannot invoke a command file in a loop created by the FOR, REPEAT, or WHILE statements.
- You cannot invoke a command file within a BEGIN-END block.
- You should not allow a command file to invoke itself, either directly or indirectly; you can create an infinite loop.
- You should not put incomplete commands or statements in the middle of a command file. DATATRIEVE looks at the next entry in the command file for the completion of the command or statement and displays an error message if the command or statement is not completed.

Results

- If you do not specify a file type for the command file, DATATRIEVE uses a .CMD file type as a default.
- If the command file consists of full commands or statements, DATATRIEVE executes each command or statement in the command file in order.
- If the command file ends with an incomplete DATATRIEVE command or statement, DATATRIEVE returns a CON> prompt, waiting for you to supply the missing elements.

Usage Note

To include comments in a command file, place an exclamation mark (!) before each comment line.

Example

Invoke a command file, BUILDR.CMD, to produce a report on YACHTS by a specified builder:

```
DTR> @BUILDR
READY YACHTS
REPORT YACHTS WITH BUILDER = *.BUILDER
SET COLUMNS_PAGE = 70
PRINT BOAT
END_REPORT
Enter BUILDER: ALBIN

                                                    11-Oct-1983
                                                    Page 1

                                        LENGTH
                                         OVER
 MANUFACTURER       MODEL        RIG     ALL     WEIGHT   BEAM    PRICE

  ALBIN             79           SLOOP   26       4,200    10    $17,900
  ALBIN             BALLAD       SLOOP   30       7,276    10    $27,500
  ALBIN             VEGA         SLOOP   27       5,070    08    $18,600

DTR>
```

5.4 ABORT Statement

Function

Stops the execution of either a single statement or an entire procedure or command file.

Format

```
ABORT [value-expression]
```

Arguments

value-expression

Is a DATATRIEVE value expression, usually a character string literal.

Restriction

The ABORT statement affects the outermost statement that contains it. When a statement contains nested FOR loops, you cannot use an ABORT statement to transfer control from an inner loop to an outer loop. Similarly, when a statement contains nested BEGIN-END blocks, you cannot use an ABORT statement to transfer control from an inner block to an outer one.

Results

- The ABORT statement does not end an interactive DATATRIEVE session. During an interactive session, DATATRIEVE stops executing the statement that contains the ABORT statement, displays a message containing the value of the value expression specified in the ABORT statement, and returns control to DATATRIEVE command level (indicated by the DTR> prompt):

```
DTR> SET NO PROMPT(RET)
DTR> DECLARE X PIC XXX.(RET)
DTR> X = "DEF"(RET)
DTR> IF X = "ABC" THEN PRINT X ELSE (RET)
CON>          ABORT "X does not equal ABC"(RET)
ABORT: X does not equal ABC
Execution terminated by "ABORT" statement
DTR>
```

- During an interactive session, if the ABORT statement is in a procedure or command file, the result depends on whether SET ABORT is in effect:
  - If SET ABORT is in effect, DATATRIEVE returns to command level without executing the rest of the procedure or command file when the ABORT statement is encountered.
  - If SET NO ABORT is in effect, DATATRIEVE stops execution of the statement that contains the ABORT statement and then executes any statements and commands remaining in the procedure or command file.

- If you invoke DATATRIEVE from operating system level with an invocation command line, the effect of an ABORT statement depends on whether SET ABORT is in effect:
  - If SET ABORT is in effect, DATATRIEVE returns to the operating system command level without executing the rest of the command file.
  - If SET NO ABORT is in effect, DATATRIEVE stops executing the statement containing the ABORT statement and then processes any statements and commands remaining in command file.
- If the ABORT statement occurs in a STORE or MODIFY statement, DATATRIEVE does not store a new record or modify the target record.
- When you submit a batch stream containing a DATATRIEVE command file, any ABORT statements in the command file behave as they would if you had entered them during an interactive session:
  - Whether SET ABORT is in effect or not, DATATRIEVE prints a message containing the value of the value expression specified in the ABORT statement in the log file of the batch job.
  - If SET ABORT is in effect and DATATRIEVE processes an ABORT statement in a procedure or command file, DATATRIEVE returns to DATATRIEVE command level without executing the rest of the procedure or command file.
  - If SET NO ABORT is in effect and DATATRIEVE processes an ABORT statement in a procedure or command file, DATATRIEVE stops executing the statement containing the ABORT statement and then processes any statements and commands remaining in the procedure or command file.
  - If you invoke DATATRIEVE with an invocation command line in the batch stream and DATATRIEVE processes an ABORT statement, SET ABORT ends the DATATRIEVE session and returns control of the process to the batch stream. SET NO ABORT aborts the current statement, and DATATRIEVE processes the remaining statements in the command file.

If the batch stream contains other command files queued after the DATATRIEVE command file, the processing of a DATATRIEVE ABORT statement does not end the batch job.

Usage Note

Use an IF-THEN-ELSE statement to establish conditions for the ABORT statement (see Section 5.35.) The Boolean expression in the IF clause establishes the conditions that control the ABORT statement. The ABORT statement executes when:

- The Boolean expression is true and the ABORT statement is in the THEN clause.
- The Boolean expression is false and the ABORT statement is in the ELSE clause.

Examples

Store a record in the YACHTS domain, using an ABORT statement to stop the operation if the value of the BEAM field is zero:

```
DTR> SET NO PROMPT↵
DTR> STORE YACHTS VERIFY USING↵
DTR> IF BEAM EQ 0 THEN ABORT "Bad value for BEAM"↵
Enter MANUFACTURER: AMERICAN↵
Enter MODEL: 1980↵
Enter RIG: SLOOP↵
Enter LENGTH_OVER_ALL: 25↵
Enter DISPLACEMENT: 7500↵
Enter BEAM: 0↵
Enter PRICE: 10000↵
ABORT: Bad value for BEAM
Execution terminated by "ABORT" statement
DTR>
```

Define a procedure to write a report on the current collection. Use the ABORT statement to stop the entire procedure if there is no current collection to report on:

```
DTR> SET NO PROMPT↵
DTR> DEFINE PROCEDURE YACHT_REPORT↵
DFN> SET ABORT↵
DFN> PRINT "Have you established a current collection?"↵
DFN> IF *,"YES or NO" CONTAINING "N" THEN↵
DFN>      ABORT "Sorry--no collection, no report."↵
DFN> REPORT↵
DTR> PRINT BOAT↵
DFN> AT BOTTOM OF REPORT PRINT COUNT, TOTAL PRICE↵
DFN> END_REPORT↵
DFN> END_PROCEDURE↵
DTR> :YACHT_REPORT↵

Have you established a current collection?
Enter YES or NO: NO↵
ABORT: Sorry--no collection, no report.
Execution terminated by "ABORT" statement
DTR>
```

Define a procedure to update the OWNERS domain after the sale of a boat. Check the boat in question against the inventory in the YACHTS domain and abort the procedure if the boat is not in YACHTS. Then, check the OWNERS domain for a record of the boat. If a record exists, change the owner's name and update the boat name if desired. If no record exists, store a new record in the OWNERS domain. The procedure requires MODIFY or WRITE access to the OWNERS domain for the update and EXTEND or WRITE access for the entry of a new record. The example aborts because there is no YACHTS record for an ALBERG 42. The value expression specified in the ABORT statement includes the variables BLD and MOD:

```
DTR> SHOW SALE_BOAT↵
PROCEDURE SALE_BOAT
READY YACHTS WRITE
READY OWNERS WRITE
SET ABORT
DECLARE BLD PIC X(10),
DECLARE MOD PIC X(10),
BLD = *,"BUILDER'S NAME"
MOD = *,"MODEL"
IF NOT ANY YACHTS WITH (BUILDER = BLD AND MODEL = MOD) THEN
          BEGIN
                   PRINT "Record not found in yachts."
                   ABORT "Possible inventory error for--"|!BLD!MOD
          END ELSE
          PRINT "Yachts record found for "!BLD!MOD
IF ANY OWNERS WITH (BUILDER = BLD AND MODEL = MOD) THEN
          BEGIN
                   FOR OWNERS WITH (BUILDER = BLD AND MODEL = MOD)
                   MODIFY USING
                            BEGIN
                            PRINT COL 10, NAME
                            NAME = *,"New owner's name"
                            PRINT COL 10, BOAT_NAME
                            IF *,"Change boat name? Y/N"
                                   CONTAINING "Y" THEN PRINT SKIP THEN
                                   BOAT_NAME = *,"New boat name"
                            END
          END ELSE
          STORE OWNERS USING
                   BEGIN
                            NAME = *,"New owner's name"
                            BOAT_NAME = *,"Boat name"
                            BUILDER = BLD
                            MODEL = MOD
                   END
END_PROCEDURE
DTR> :SALE_BOAT↵
Enter Builder's name: ALBERG↵
Enter Model: 42↵
Record not found in yachts.
ABORT: Possible inventory error for--ALBERG 42
Execution terminated by "ABORT" statement
DTR>
```

5.5 ADT Command

Function

Invokes the Application Design Tool (ADT), an interactive aid that helps you define a domain, its associated record, and its data file.

Format

```
ADT
```

Arguments

None.

Restrictions

- You cannot invoke ADT from an application program using the DATATRIEVE-11 Call Interface.
- Do not use the ADT command in a command file.
- The record definition and data file definition created by ADT do not contain all the features, clauses, and options available when using the DATATRIEVE DEFINE commands.
- See the following commands for the access privileges needed to add the domain and record definitions to the dictionary and to define the data file for the domain: DEFINE DOMAIN, DEFINE RECORD, DEFINE FILE, and DELETE.

Results

- DATATRIEVE invokes the Application Design Tool, which asks you a series of questions about the domain, the size and type of the fields you want in the record, and the name and type of the data file.
- ADT writes a command file you can invoke at DATATRIEVE command level.
- If you respond to any of ADT's questions with a CTRL/Z, DATATRIEVE displays the message "ADT exited by user request" and returns control to DATATRIEVE command level.

Example

Invoke the Application Design Tool.

```
DTR> ADT↵
Do you want help? (YES or NO) :
```

See Chapter 3 of *Introduction to DATATRIEVE-11* for a sample ADT session and more information on ADT.

5.6 ALLOCATION Clause

Function

Specifies the type of word boundary alignment DATATRIEVE uses when storing records in a data file associated with the record definition. It also controls the way DATATRIEVE retrieves data from files created by user programs or other applications.

Format

```
ALLOCATION IS { MAJOR_MINOR
              { ALIGNED_MAJOR_MINOR
              { LEFT_RIGHT
```

Arguments

MAJOR_MINOR

Causes DATATRIEVE to force word boundary alignments according to data types for elementary fields defined with the USAGE IS COMP SYNC clause and to force group fields to the maximum alignment of the elementary fields contained within the group field. MAJOR_MINOR is the default for VAX-11 DATATRIEVE and VAX-11 COBOL.

ALIGNED_MAJOR_MINOR

Causes DATATRIEVE to force word boundary alignments according to data types for all elementary fields in the record and group fields to the maximum alignment of the elementary fields they contain.
ALIGNED_MAJOR_MINOR is the default for COBOL-81.

LEFT_RIGHT

Causes DATATRIEVE to force word boundary alignment for elementary fields defined as COMP, COMP_1, COMP_2, and DATE. LEFT_RIGHT is the default alignment for DATATRIEVE-11, COBOL-11, and COBOL-74.

Restriction

When defining a record for an existing data file, the alignment type of the record definition must match the alignment type of the data file.

Results

- For records with ALIGNED_MAJOR_MINOR allocation and for fields with USAGE IS COMP SYNC clauses in records with MAJOR_MINOR allocation, DATATRIEVE aligns fields on word boundaries. When DATATRIEVE shifts a field to the next word boundary, the bytes skipped (filler bytes) are considered part of the record.
- The number of filler bytes DATATRIEVE inserts when ALIGNED_MAJOR_MINOR or MAJOR_MINOR is in effect depends on the data type and/or the length of the field:
  - Fields declared COMP (INTEGER) always begin on one-word boundaries.
  - Fields declared COMP_1 (REAL) always begin on two-word boundaries, while fields declared COMP (INTEGER) that are from five to nine digits long also begin on two-word boundries.
  - Fields declared DATE or COMP_2 (DOUBLE) always begin on four-word boundaries. Fields declared COMP (INTEGER) that are from 10 to 18 digits long are also aligned on four-word boundries.
- When the allocation is either MAJOR_MINOR or ALIGNED_MAJOR_MINOR, the group field boundaries are aligned according to the maximum alignment of the elementary fields that comprise the group.
- When the allocation is LEFT_RIGHT, fields begin on word boundaries if they are declared DATE, COMP_2 (DOUBLE), or COMP_1 (REAL).

Usage Notes

- The ALLOCATION clause is an optional record definition clause. When you include it in a record definition, it affects the way DATATRIEVE handles the internal storage and retrieval of data in some or all of the fields in your data file.
- If you want to use DATATRIEVE-11 on data files created with VAX-11 DATATRIEVE, you must add an ALLOCATION MAJOR_MINOR clause to the DATATRIEVE-11 record definition. This ensures that DATATRIEVE-11 can interpret the VAX-11 data correctly.

Example

Define records without an ALLOCATION clause (the default is LEFT_RIGHT) and with MAJOR_MINOR alignment. Specify USAGE IS COMP SYNC for one MAJOR_MINOR record and USAGE IS COMP for another MAJOR_MINOR record.

```
DTR> !RET
DTR> ! Define a record with default allocationRET
DTR> !RET
DTR> DEFINE RECORD A_REC USINGRET
DFN> 01 TOP,RET
DFN>     03 CHAR PIC X,RET
DFN>     03 NUM USAGE IS COMP,RET
DFN> ;RET
[Record A_REC is 4 bytes long]
DTR> !RET
DTR> ! Define a record with MAJOR_MINOR allocationRET
DTR> ! and no SYNC clauseRET
DTR> !RET
DTR> DEFINE RECORD B_REC USINGRET
DFN> ALLOCATION IS MAJOR_MINORRET
DFN> 01 TOP,RET
DFN>     03 CHAR PIC X,RET
DFN>     03 NUM USAGE IS COMP,RET
DFN> ;RET
[Record B_REC is 3 bytes long]
DTR> !RET
DTR> ! Define a record with MAJOR_MINOR allocationRET
DTR> ! and a USAGE IS COMP SYNC clauseRET
DTR> !RET
DTR> DEFINE RECORD C_REC USINGRET
DFN> ALLOCATION IS MAJOR_MINORRET
DFN> 01 TOP,RET
DFN>     03 CHAR PIC X,RET
DFN>     03 NUM USAGE IS COMP SYNC,RET
DFN> ;RET
[Record C_REC is 4 bytes long]
DTR>
```

5.7 Assignment Statement

The assignment statement assigns a value to an elementary field, a group field, or a variable.

5.7.1 Assigning a Value to an Elementary Field

Function

Assigns a value to an elementary field in a MODIFY or STORE statement.

Format

```
field-name = value-expression
```

Arguments

field-name

Is the name of an elementary field.

value-expression

Is the value to be assigned to the field. This argument can be any DATATRIEVE value expression. (See Chapter 2.)

Restrictions

- The equal sign (=) is **not** the same as the relational operators EQ or EQUAL; it is the assignment operator that tells DATATRIEVE to store the value on the right in the field specified on the left.
- This type of assignment statement can be used only in the USING clause in a MODIFY or STORE statement (see Sections 5.35 and 5.53).
- To use the assignment statement in a USING clause in a STORE statement, you must ready the domain for WRITE or EXTEND access. To use this statement in a USING clause in a MODIFY statement, you must ready the domain for WRITE or MODIFY access (see the READY command, Section 5.42.)
- You cannot assign a value to a COMPUTED BY field.

Results

- DATATRIEVE stores the value of the value expression in the specified field, performing any datatype conversions necessary.

- If the value expression is a prompting value expression(*.prompt-name), DATATRIEVE prompts for the value of the field. DATATRIEVE reprompts for the value if any of the following occurs:
  - Truncation error: you have entered more characters than the field definition allows.
  - Conversion error: you have entered a character that is inappropriate for the field, such as a letter in a numeric field.
  - Sign error: you have entered a minus sign in an unsigned numeric field.
  - VALID IF error: you have entered an invalid value in the field. That is, the Boolean expression specified in the VALID IF clause in the record definition is evaluated as false because the value specified in the assignment statement is not a valid value for the field. See Chapter 4 for more information on record definitions and field definition clauses.
- If the value expression is not a prompting value expression and truncation, conversion, or sign errors occur, DATATRIEVE accepts the value with a warning:
  - If you enter too many digits for a numeric field, DATATRIEVE truncates the high-order digits and stores the remaining digits in the field.
  - If you enter too many characters for an alphanumeric field, DATATRIEVE truncates the rightmost characters and stores the remaining characters in the field.
  - If a VALID IF failure occurs, DATATRIEVE does not execute the assignment statement and does not execute the STORE or MODIFY statement containing the assignment statement.

Usage Notes

- If you want DATATRIEVE to check your values for validity and give you the opportunity to recover from an error caused by invalid input, use a prompting value expression in the USING clause of a MODIFY or STORE statement:

  ```
  field-name = *.prompt-string
  ```

  This type of assignment statement provides the only way to recover from truncation, conversion, sign errors, or VALID IF failures and is especially useful when you are creating records with the STORE statement.
- If you are unsure of the name of a field or the type of data it contains, use the SHOW FIELDS command (see Section 5.49). SHOW FIELDS displays a brief description of each field in a readied domain.

Examples

Ready the YACHTS domain for WRITE access, then store a record in the domain for the manufacturer CHALLENGER:

```
DTR> READY YACHTS WRITE↵
DTR> STORE YACHTS USING MANUFACTURER = "CHALLENGER"↵
DTR> PRINT YACHTS WITH BUILDER EQ "CHALLENGER"↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL    WEIGHT BEAM  PRICE

 CHALLENGER                                  0  00
 CHALLENGER   32          SLOOP   32    12,800  11  $31,835
 CHALLENGER   35          SLOOP   35    14,800  12  $39,215
 CHALLENGER   41          KETCH   41    26,700  13  $51,228

DTR>
```

Ready the YACHTS domain for MODIFY access, then change the value of the PRICE field in the first record to a new value:

```
DTR> READY YACHTS MODIFY↵
DTR> FIND YACHTS↵
[113 records found]
DTR> SELECT↵
DTR> PRINT PRICE THEN MODIFY USING PRICE = *,"NEW PRICE"↵

 PRICE

$36,951
Enter NEW PRICE: 39000↵
DTR> PRINT PRICE↵

 PRICE

$39,000

DTR>
```

5.7.2 Assigning a Value to a Group Field

Function

Assigns a value to a group field in a MODIFY or STORE statement.

Format

```
group-field-name-1 = group-field-name-2
```

Arguments

group-field-name-1

Is the name of a group field to which you want to assign a value.

group-field-name-2

Is the name of a group field containing the values you want to assign to group field 1.

Restrictions

- The equal sign (=) is **not** the same as the relational operators EQ or EQUAL. It is the assignment operator that tells DATATRIEVE to store the value on the left in the field specified on the right.
- This type of assignment statement can be used only in a USING clause in a MODIFY or STORE statement (see Section 5.35 and 5.53).
  - To use the assignment statement in the USING clause of a MODIFY statement, you must ready the domain containing group field 1 for MODIFY or WRITE access.
  - To use the assignment statement in the USING clause of a STORE statement, you must ready the domain for WRITE or EXTEND access (see the READY command, Section 5.42).
- Both group field 1 and group field 2 must have at least one elementary field with the same field name or query name.

Result

DATATRIEVE changes the values of all fields in group field 1 to the values of identically named fields in group field 2. All unmatched elementary fields in group field 1 are set to zero (numeric fields) or blank (string fields) in a STORE statement that does not contain explicit assignments. Unmatched fields remain unchanged in a MODIFY statement. Use explicit assignments in the STORE statement to assign values to unmatched fields in group field 1.

Examples

Store a record in the YACHTS domain with the same values in the group field SPECIFICATIONS (query name SPECS) as the selected record:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS WRITE↵
DTR> FIND YACHTS↵
[113 records found]
DTR> SELECT↵
DTR> PRINT SPECS↵
```

(continued on next page)

```
       LENGTH
        OVER
 RIG     ALL    WEIGHT BEAM  PRICE

KETCH    37     20,000  12  $36,951

DTR> STORE YACHTS USING(RET)
CON> BEGIN(RET)
CON>      BUILDER = *.BUILDER(RET)
CON>      SPECS = SPECS(RET)
CON> END(RET)
Enter BUILDER: HUGHES(RET)
DTR> PRINT YACHTS WITH BUILDER = "HUGHES"(RET)

                                  LENGTH
                                   OVER
MANUFACTURER    MODEL      RIG     ALL    WEIGHT BEAM  PRICE

 HUGHES                   KETCH    37     20,000  12  $36,951

DTR>
```

Store records in a new domain called PRICED_YACHTS. Store only boats that have a price:

```
DTR> SET NO PROMPT(RET)
DTR> DEFINE DOMAIN PRICED_YACHTS USING YACHT ON PYACHT.DAT;(RET)
DTR> DEFINE FILE FOR PRICED_YACHTS(RET)
DTR> READY PRICED_YACHTS WRITE(RET)
DTR> FOR YACHTS WITH PRICE NE 0(RET)
CON>       STORE PRICED_YACHTS USING BOAT=BOAT(RET)
DTR> FIND PRICED_YACHTS(RET)
[50 records found]
DTR>
```

5.7.3 Assigning a Value to a Variable

Function

Assigns a value to a variable.

Format

```
variable-name = value-expression
```

Arguments

variable-name

Is the name of a variable that has been defined with a DECLARE statement.

value-expression

Is the value to be assigned to the specified variable.

Restrictions

- The equal sign (=) is **not** the same as the relational operators EQ or EQUAL. It is the assignment operator that tells DATATRIEVE to store the value on the left in the field specified on the right.
- You can use this type of assignment statement anywhere a DATATRIEVE statement is allowed.
- You must define the variable name with the DECLARE statement (see Section 5.11) before you can assign a value to it.

Results

- DATATRIEVE assigns the specified value to the variable.
- If the value expression is a prompting value expression(*.prompt-name), DATATRIEVE prompts for the value of the field. DATATRIEVE reprompts for the value if any of the following occurs:
  - Truncation error: you have entered more characters than the field definition allows.
  - Conversion error: you have entered a character that is inappropriate for the field, such as a letter for a numeric field.
  - Sign error: you have entered a minus sign for an unsigned numeric field.
  - VALID IF error: you have entered an invalid value for the field. That is, the Boolean expression specified in the VALID IF clause in the record definition is evaluated as false if the value specified in the assignment statement is not a valid value for the field. See Chapter 4 for more information on record definitions and field definition clauses.
- If the value expression is not a prompting value expression and truncation, conversion, or sign errors occur, DATATRIEVE accepts the value with a warning:
  - If you enter too many digits for a numeric field, DATATRIEVE truncates the high-order digits, stores the remaining digits in the field.
  - If you enter too many characters for an alphanumeric field, DATATRIEVE truncates the rightmost characters and stores the remaining characters in the field.
  - If a VALID IF failure occurs, DATATRIEVE does not execute the assignment statement.

Usage Notes

- If you want DATATRIEVE to check your values for validity and give you the opportunity to recover from an error caused by invalid input, use a prompting value expression:

  field-name = *.prompt-string

  This type of assignment statement provides the only way to recover from truncation, conversion, sign errors, or VALID IF failures and is especially useful when you are creating records with the STORE...USING statement.

- Use the assignment statement to assign the current system date to a date field. Define a date field with the USAGE IS DATE clause and then assign the current system date to the field with the DATATRIEVE date value expression, "TODAY":

```
DTR> DECLARE CURRENT_DATE USAGE DATE;RET
DTR> CURRENT_DATE = "TODAY"RET
DTR> PRINT CURRENT_DATERET

  CURRENT
   DATE

16-Feb-83

DTR>
```

Examples

Declare the global variable NEW_PRICE. Assign a value to it:

```
DTR> DECLARE NEW_PRICE PIC 9(5) EDIT_STRING IS $$$,$$$,RET
DTR> NEW_PRICE = "$25,000"RET
DTR> PRINT NEW_PRICERET

  NEW
 PRICE

$25,000
DTR>
```

Declare two global variables. Assign the value in the PRICE field of the first boat in YACHTS, and then assign the value of OLD_PRICE to NEW_PRICE:

```
DTR> DECLARE NEW_PRICE PIC 9(5).RET
DTR> DECLARE OLD_PRICE PIC 9(5).RET
DTR> FIND YACHTS; SELECT; PRINT PRICERET

 PRICE

$36,951

DTR> OLD_PRICE = PRICERET
DTR> PRINT OLD_PRICERET

  OLD
 PRICE

$36,951

DTR> NEW_PRICE = OLD_PRICERET
DTR> PRINT NEW_PRICERET

  NEW
 PRICE

$36,951

DTR>
```

5.8 BEGIN-END Statement

Function

Groups DATATRIEVE statements into a single compound statement called a BEGIN-END block.

Format

```
BEGIN
      statement-1
     [statement-2]
      .
      .
      .
END
```

Arguments

statement

Is a DATATRIEVE statement. Within the BEGIN-END block, terminate each statement with a semicolon, a RETURN, or both.

Restrictions

- Do not use FIND, SELECT, or DROP statements in a BEGIN-END block.
- Observe all restrictions on statements included in the BEGIN-END block. This manual lists these restrictions in the descriptions of the various statements.
- Do not use DATATRIEVE commands in a BEGIN-END block. You cannot include a procedure that contains DATATRIEVE commands, nor can you invoke a command file that contains DATATRIEVE commands in a BEGIN-END block.
- Do not store a command file invocation in a BEGIN-END block. When you type @file-name within a BEGIN-END block, DATATRIEVE immediately executes the command file, and all the statements in the file become part of the BEGIN-END block just as though you had typed them yourself.

Results

- DATATRIEVE executes statements in a BEGIN-END block in sequential order.
- When the BEGIN-END block includes a DECLARE statement, the variable it defines is a local variable. You cannot refer to a local variable from outside the BEGIN-END block. DATATRIEVE automatically releases all local variables when it finishes executing the BEGIN-END block in which they are defined.

- If a BEGIN-END block that defines a local variable is in a FOR loop or REPEAT statement, DATATRIEVE initializes the local variable each time it executes the BEGIN-END block. The variable is initialized to zero if numeric or blank if string. See Section 5.11 for information about the DECLARE statement.
- When you create a single BEGIN-END block interactively, DATATRIEVE prompts you, after you press RETURN, for the elements needed to complete an individual statement or to complete the block. The CON> prompt indicates that DATATRIEVE is ready for you to continue entering statements or elements of statements into the BEGIN-END block. After you enter END, DATATRIEVE executes all the statements in the BEGIN-END block. When DATATRIEVE completes the last statement in the BEGIN-END block, you see the DTR> prompt indicating you have returned to DATATRIEVE command level.
- When you are entering nested BEGIN-END blocks interactively, DATATRIEVE does not execute any of the statements in the nested blocks until you enter the END that completes the outermost BEGIN-END block. DATATRIEVE continues to prompt with CON> until you enter the END that completes the outermost BEGIN-END block.
- If you enter CTRL/C while DATATRIEVE is executing the statements in a BEGIN-END block, DATATRIEVE does not execute any of the remaining statements that follow in the block. DATATRIEVE treats the whole BEGIN-END block as one statement, regardless of the number of statements or nested BEGIN-END blocks it contains. Because CTRL/C cancels the execution of the current statement, it cancels the execution of the remaining outermost BEGIN-END block, regardless of the number of levels of other BEGIN-END blocks nested in it.
- If a statement in a BEGIN-END block prompts for a value and you enter a CTRL/C, DATATRIEVE reprompts for the value. To end the execution of a BEGIN-END block, enter CTRL/Z in response to the prompt. DATATRIEVE then stops executing the outermost BEGIN-END block and returns you to DATATRIEVE command level.

Usage Notes

- You can use a BEGIN-END block anywhere you can use a DATATRIEVE statement.
- You can nest BEGIN-END blocks. The only limit to the number of levels of nested BEGIN-END blocks you can form is the amount of system pool available at any given time. If you exceed this limit, DATATRIEVE stops executing statements and signals an error. Your only option at this point is to reduce the levels of BEGIN-END block nesting.
- To repeat an entire sequence of DATATRIEVE statements, put the statements in a BEGIN-END block, and put the BEGIN-END block in a REPEAT statement.

- To repeat all statements in a procedure, invoke the procedure in a BEGIN-END block, and put the BEGIN-END block in a REPEAT statement. If you invoke a procedure in a REPEAT statement (REPEAT n :procedure-name), DATATRIEVE repeats the first statement of the procedure n times and then executes the other statements in the procedure one time each.
- Use a BEGIN-END block to include more than one DATATRIEVE statement in a FOR loop.
- Use a BEGIN-END block to include more than one DATATRIEVE statement in the THEN and ELSE clauses in an IF-THEN-ELSE statement.
- When storing or modifying records, use BEGIN-END blocks to include more than one statement in the USING and VERIFY USING clauses of the STORE and MODIFY statements.
- When you invoke a procedure that uses consecutive PRINT statements to format output, the line spacing changes when you invoke the procedure in a BEGIN-END block. The one blank line between the result of each PRINT statement disappears. To preserve that spacing when you include the procedure in a BEGIN-END block, edit the procedure and insert a SKIP print list element at the beginning of the second and each succeeding PRINT statement.

Examples

Store five records in the domain PHONES, each having LOCATION MB1-H2 and DEPARTMENT CE. The SET NO PROMPT command suppresses the "[Looking for...]" prompts that precede each CON> prompt. This example also shows how DATATRIEVE responds to CTRL/Z when prompting for input:

```
DTR> SHOW PHONES(RET)
DOMAIN PHONES
 USING PHONE_REC ON PHONE.DAT;
DTR> SHOW PHONE-REC(RET)
RECORD PHONE_REC
 USING
01 PHONE_REC.
   03 NAME                PIC X(10).
   03 NUMBER              PIC X(8).
   03 LOCATION            PIC X(10).
   03 DEPARTMENT  PIC X(3).
;
DTR> READY PHONES WRITE(RET)
DTR> SET NO PROMPT(RET)
DTR> REPEAT 5 STORE PHONES USING(RET)
CON> BEGIN(RET)
CON>     NAME =           *.NAME(RET)
CON>     NUMBER =         *.NUMBER(RET)
CON>     LOCATION =       "MB1-H2"(RET)
CON>     DEPARTMENT =     "CE"(RET)
CON> END(RET)
Enter NAME: FRED(RET)
Enter NUMBER: 555-1243(RET)
Enter NAME: KAREN(RET)
Enter NUMBER: 423-9981(RET)
Enter NAME: ^Z
Execution terminated by operator
DTR>
```

Use a BEGIN-END block to put three statements in the USING clause of a MODIFY statement. Print a YACHTS record, modify the price, and print the result of the modification and the next record to modify:

```
DTR> READY YACHTS WRITE↵
DTR> SET NO PROMPT↵
DTR> FOR YACHTS WITH PRICE = 0↵
CON>     MODIFY USING↵
CON>               BEGIN↵
CON>                       PRINT↵
CON>                       PRICE = *,"NEW PRICE"↵
CON>                       PRINT↵
CON>               END↵

                                    LENGTH
                                     OVER
MANUFACTURER  MODEL      RIG        ALL    WEIGHT BEAM  PRICE

 BLOCK I.     40         SLOOP      39     18,500  12
Enter NEW PRICE: 30000

                                    LENGTH
                                     OVER
MANUFACTURER  MODEL      RIG        ALL    WEIGHT BEAM  PRICE

 BLOCK I.     40         SLOOP      39     18,500  12   $30,000
 BUCCANEER    270        SLOOP      27      5,000  08
Enter NEW PRICE: ^Z
Execution terminated by operator

DTR>
```

Use a BEGIN-END block in a REPEAT statement to repeat an entire procedure:

```
DTR> SET NO PROMPT↵
DTR> DEFINE PROCEDURE LOOP_EXAMPLE↵
DFN>     PRINT "Show how a BEGIN-END works with REPEAT"↵
DFN>     PRINT "and more than one statement"↵
DFN> END_PROCEDURE↵
DTR> REPEAT 2↵
CON>     :LOOP_EXAMPLE↵
Show how a BEGIN-END works with REPEAT
Show how a BEGIN-END works with REPEAT

and more than one statement

DTR> REPEAT 2↵
CON> BEGIN↵
CON>     :LOOP_EXAMPLE↵
CON> END↵
Show how a BEGIN-END works with REPEAT
and more than one statement
Show how a BEGIN-END works with REPEAT
and more than one statement

DTR>
```

5.9 CLOSE Command

Function

Closes an RMS log file created with an OPEN command. A log file records your interactive dialogue with DATATRIEVE.

Format

```
CLOSE
```

Arguments

None.

Restriction

You can use the CLOSE command only at DATATRIEVE command level in response to the DTR> prompt.

Results

- DATATRIEVE closes the log file created by the OPEN command. The CLOSE command has no effect on the location of the log file, which is catalogued in the directory determined by the file specification entered in the OPEN command (see Section 5.37).
- DATATRIEVE also closes a log file when you exit from DATATRIEVE.

Usage Note

The CLOSE command allows you to close a log file without ending an interactive DATATRIEVE session.

Example

Open a log file, display a record, then close the file:

```
DTR> OPEN LOG(RET)
DTR> SHOW FAMILY_REC(RET)
RECORD FAMILY_REC
01 FAMILY.
   03 PARENTS.
      06 FATHER PIC X(10).
      06 MOTHER PIC X(10).
   03 NUMBER_KIDS PIC 99 EDIT_STRING IS Z9.
   03 KIDS OCCURS 0 TO 10 TIMES DEPENDING ON NUMBER_KIDS.
      06 EACH_KID.
         09 KID_NAME PIC X(10) QUERY_NAME IS KID.
         09 AGE PIC 99 EDIT_STRING IS Z9.
;

DTR> CLOSE(RET)
DTR>
```

5.10 COMPUTED BY Clause

Function

Describes a COMPUTED BY field.

Format

```
COMPUTED BY value-expression
```

Arguments

value-expression

Is a value expression consisting of one or more field or query names, numeric literals, and arithmetic operators. This includes concatenation expressions, table expressions, and statistical expressions.

Restrictions

- This clause is valid for elementary fields only.
- You cannot use an OCCURS clause or a VALID IF clause in a COMPUTED BY field.
- You cannot use a COMPUTED BY field as a sort key in SORT and SUM statements or in the SORTED BY clause of record selection expressions.
- Because it does not exist in the record, you cannot assign a value to a COMPUTED BY field. STORE and MODIFY statements cannot refer to a COMPUTED BY field.
- You cannot redefine a COMPUTED BY field with the REDEFINES clause.
- Because a COMPUTED BY field is a virtual expression, you cannot use a clause such as PICTURE or USAGE that specifies how the value is stored.

Result

When you refer to the COMPUTED BY field in a statement, DATATRIEVE resolves the field name to the nearest single record context and evaluates the value expression after the field using the field value(s) in that record. See Chapter 12 in the *DATATRIEVE-11 User's Guide* for more information on COMPUTED BY fields.

Usage Notes

- COMPUTED BY fields are useful if you display arithmetic computations frequently. A COMPUTED BY field allows you to specify the computation once in the record definition, then print its value simply by referring to its field name. Generally, the computation includes the name of one or more fields in the record definition.

- A COMPUTED BY field does not occupy space in a record. It exists solely in the record definition as a virtual field. DATATRIEVE calculates the value of a COMPUTED BY field only when you use the field in a statement.
- If you include a COMPUTED BY clause in a field definition, you do not have to include any other field definition clause. However, you may want to specify the format of the computation with an EDIT_STRING clause.

Examples

Compute the price per pound of a yacht as the price divided by the displacement. In this case, both the PRICE and DISP fields are defined in the record definition:

```
06 PRICE_PER_POUND
   EDIT_STRING $$$,$$9.99
   COMPUTED BY PRICE/DISP.
```

When you use the PRICE_PER_POUND field in a command or statement, DATATRIEVE divides the value of the record's PRICE field by the value of its DISP field. The result of the computation is then the value of the PRICE_PER_POUND field.

Derive the value of the SALESMAN field from a dictionary table named SALESMEN:

```
06 SALESMAN
   EDIT_STRING IS X(20)
   COMPUTED BY MANUFACTURER VIA SALESMEN.
```

In this example, DATATRIEVE uses the value of the MANUFACTURER field in the current record to search the dictionary table SALESMEN for a matching code. If one is found, DATATRIEVE uses its translation as the value of the SALESMAN field.

5.11 DECLARE Statement

Function

Defines a global or a local variable.

Format

```
DECLARE variable-name variable-definition.
```

Arguments

variable-name

Is the name of the variable being defined. The name must conform to the rules for names listed in Chapter 4.

variable-definition

Is the definition of the variable, which consists of field definition clauses. If you include more than one such clause, separate them with spaces or tabs.

. (period)

Ends the DECLARE statement.

Restrictions

- You must include at least one COMPUTED BY, PICTURE (PIC), or USAGE clause in the variable definition to specify the data type, length, scale, and edit string of the variable.
- Although you can include other field definition clauses, you cannot use an OCCURS clause or REDEFINES clause in the variable definition. A variable can have only the properties of an elementary, nonrepeating field in a DATATRIEVE record definition.
- You must explicitly initialize all global variables. DATATRIEVE does not initialize global variables.

Results

- When you use a DECLARE statement at DATATRIEVE command level, DATATRIEVE creates a global variable. A global variable exists until you redeclare it, end the DATATRIEVE session, or explicitly release the variable with the RELEASE command.

- Unless you define a global variable with a COMPUTED BY clause, the global variable retains the value you assign it until you assign a new value to it, redeclare it, end the DATATRIEVE session, or explicitly release the variable with a RELEASE command. The value of a global variable defined with a COMPUTED BY clause depends on the value expression that controls the computation. If the value expression is based on the value of a field in a record, the variable has a value only when there is a valid single record context in which to resolve the value expression.
- When you use a DECLARE statement in another DATATRIEVE statement, such as a BEGIN-END or THEN statement, DATATRIEVE creates a local variable. A local variable exists only within the statement in which it is defined. DATATRIEVE releases all local variables created within a statement when it encounters the end of that statement.

  For example, if you define a local variable in the third BEGIN-END block in a series of four nested BEGIN-END blocks, you can refer to, assign values to, and retrieve values from the variable in the third and fourth blocks. That is, after DATATRIEVE executes the DECLARE statement in the third block, the value of the variable can be changed or retrieved by any of the remaining statements in the third block, including any of the statements in the fourth and innermost block. That local variable, however, has no meaning for any statement outside those two inner blocks. No statement in the first or second block and outside the two inner blocks can refer to the local variable defined in the third one.
- If the statement that defines a local variable is in a FOR loop or REPEAT statement, the local variable is initialized each time DATATRIEVE executes the statement. The initial value DATATRIEVE assigns to the variable depends on the field definition clauses included in the DECLARE statement that defined the variable. Numeric variables are initialized to zero, and alphanumeric and alphabetic variables are initialized as blanks.

Usage Note

To assign a value to a variable, use the following type of assignment statement:

variable-name = value-expression

See Chapter 2 and Section 5.7.3 in this manual for more information on assigning a value to a variable.

Examples

Declare the global variable NEW_BEAM as a two-digit numeric field with an edit string of Z9:

```
DTR> DECLARE NEW_BEAM PIC 99 EDIT_STRING IS Z9.RET
DTR> NEW_BEAM = 9RET
DTR> PRINT NEW_BEAMRET

NEW
BEAM

 9

DTR>
```

Declare the global variable X as an integer number:

```
DTR> DECLARE X USAGE IS COMP.RET
DTR> X = 36RET
DTR> PRINT XRET
  X

   36

DTR> SHOW FIELDSRET
Global variables:
   X      [Number]

DTR> RELEASE XRET
DTR> SHOW FIELDSRET
No Domains Readied or Global Variables Declared
DTR>
```

Declare the variable DUE as a date. Assign today's date to DUE and suppress the header with a hyphen in parentheses:

```
DTR> DECLARE DUE USAGE IS DATE.RET
DTR> DUE = "TODAY"RET
DTR> PRINT DUE (-)RET
22-Sep-83
DTR>
```

Use the SHOW FIELDS command to display the names and types of currently defined global variables:

```
DTR> DECLARE X USAGE COMP.RET
DTR> DECLARE Y PIC 9(5).RET
DTR> DECLARE A PIC XX.RET
DTR> DECLARE B PIC AAA.RET
DTR> SHOW FIELDSRET
Global variables :
     B     [Character string]
     A     [Character string]
     Y     [Number]
     X     [Number]

DTR>
```

Use the RELEASE command (see Section 5.44) to remove global variables from your workspace and from the context stack:

```
DTR> SHOW FIELDSRET
Global variables:
     B     [Character string]
     A     [Character string]
     Y     [Number]
     X     [Number]

DTR> RELEASE X, ARET
DTR> SHOW FIELDSRET
Global variables:
     B     [Character string]
     Y     [Number]

DTR>
```

5.12 DECLARE PORT Statement

Function

Creates a temporary DATATRIEVE port with the name you specify and readies the port for WRITE access. DATATRIEVE does not enter a definition of the port in the data dictionary.

Format

```
DECLARE PORT port-name USING
        level-number-1 field-definition-1.
        [level-number-2 field-definition-2.]
                .               .
                .               .
                .               .
        [level-number-n field definition-n.]
;
```

Arguments

port-name

Is the name of the port. It cannot duplicate a DATATRIEVE keyword or the given name of any domain you may bring into your workspace.

level-number

Is the level number for the field in the port declaration that indicates the relationship of the field to the other fields of the port.

field-definition

Is a field definition. A field definition must end with a period. A port declaration must have at least one field definition.

; (semicolon)

Ends the port declaration.

Restrictions

- You cannot invoke a procedure in a port declaration.
- You must include at least one PICTURE (PIC) clause or one USAGE clause in the port declaration.

Results

- The semicolon (;) is required to end a port declaration. DATATRIEVE continues to prompt with CON> until you type a semicolon and press RETURN or until it detects a syntax error. If you make a syntax error when declaring a port, DATATRIEVE returns to command level without creating the port.
- If your application program declares a port at DATATRIEVE command level, DATATRIEVE creates a global port that is available to your application program until you end your access to it with the FINISH command or until you end the DATATRIEVE session. A global port is similar to a global variable.
- If your application program declares the port in a compound statement, DATATRIEVE creates a local port that is available only for the remainder of the statement in which it is declared, and for those statements contained in the statement in which it is declared. A local port is similar to a local variable.

Usage Notes

- You cannot ready a declared port.
- See the *DATATRIEVE-11 Call Interface Manual* for examples and information about using a port to transfer data between DATATRIEVE and your application program.

5.13 DEFINE DICTIONARY Command

Function

Creates a file for DATATRIEVE to use as a data dictionary and connects you to the newly created dictionary.

Format

```
DEFINE DICTIONARY file-spec
```

Arguments

file-spec

Is the file specification for the data dictionary in the following format:

For RSTS/E systems:	dev:[PPN]file-name.type
For RSX systems:	dev:[UIC]file-name.type;ver

You must name at least one field in the file specification. DATATRIEVE uses the following defaults for fields not specified:

Field	Default
dev:	SY: (the system device)
[UIC/PPN]	Your default UIC/PPN
file-name	QUERY
.type	.DIC
;ver	1 or next higher version (non-RSTS/E systems only)

Restriction

You must have W (WRITE) access to the specified dictionary.

Results

- DATATRIEVE creates an empty indexed file for use as a data dictionary.
- If you do not enter a file specification on the same input line as DEFINE DICTIONARY, DATATRIEVE prompts with DFN>. After you specify a file, you return to the DATATRIEVE command level, indicated by the DTR> prompt.
- DATATRIEVE makes the newly created dictionary your current dictionary.

Usage Notes

- To verify the creation of the new dictionary, use the SHOW DICTIONARY command. DATATRIEVE displays the name of the new dictionary as the current dictionary. To return to your login default dictionary, use the SET DICTIONARY command.
- Use the following commands to define objects in your dictionary:
  - DEFINE DOMAIN
  - DEFINE PORT
  - DEFINE PROCEDURE
  - DEFINE RECORD
  - DEFINE TABLE

For information on defining a database instance, see the *DATATRIEVE-11/DBMS-11 Interface Manual.*

Examples

Define a file named NEWDIC on the system device for use as a data dictionary. The file will have the same UIC/PPN as your default UIC/PPN and a type of .DIC. The current dictionary is OLDDIC.DIC:

```
DTR> SHOW DICTIONARY(RET)
The current dictionary is SY:[56,22]OLDDIC.DIC;1
DTR> DEFINE DICTIONARY NEWDIC(RET)
DTR> SHOW DICTIONARY(RET)
The current dictionary is SY:[56,22]NEWDIC.DIC;1

DTR>
```

Define a data dictionary on a given device in a given directory, assign the dictionary the name MYDATA.DIC, confirm its creation, and reset the current dictionary:

```
DTR> DEFINE DICTIONARY(RET)
DFN> DB2:[33,17]MYDATA.DIC(RET)
DTR> SHOW DICTIONARY(RET)
The current dictionary is DB2:[33,17]MYDATA.DIC;1
DTR> SET DICTIONARY NEWDIC(RET)
DTR> SHOW DICTIONARY(RET)
The current dictionary is SY:[33,17]NEWDIC.DIC;1

DTR>
```

5.14 DEFINE DOMAIN Command

The DEFINE DOMAIN command stores a domain definition in your current dictionary. You can define domains for single RMS files or domains for a view of one or more domains.

5.14.1 Defining an RMS Domain

Function

Stores a domain definition in your current dictionary and creates an access control list (ACL) for the domain.

Format

```
DEFINE DOMAIN domain-name USING record-name [(passwd) | (*)] ON file-spec
```

Arguments

domain-name

Is the name of the domain being defined. It must conform to rules for DATATRIEVE-11 dictionary object names.

record-name

Is the name of the record definition to be associated with the domain. Before you can ready the domain, use the DEFINE RECORD command (See Section 5.18) to enter this record definition in the same dictionary.

(password)
(*)

Is the password DATATRIEVE will use to check that you have E (EXECUTE/EXTEND) privilege for the record definition. Use a password only if the record definition already exists. Use (*) to have DATATRIEVE prompt you for the password.

file-spec

Is the file specification of the RMS file containing the data for the domain. This file does not have to exist until you try to ready the domain. The file specification has the following format:

For RSTS/E systems:	dev:[PPN]file-name.type
For RSX systems:	dev:[UIC]file-name.type;ver

You must specify at least one field. On RSTS/E systems, you must specify a file name. All other fields are optional. If you omit fields in the file specification, DATATRIEVE uses the following defaults:

Field	Default
dev:	SY: (the system device)
[UIC/PPN]	Your default UIC/PPN
file-name	No default
.type	.DAT
;ver	1 or next higher version number (non-RSTS/E systems only)

; (semicolon)

Ends the domain definition.

Restrictions

- You cannot invoke a procedure in a domain definition.
- Do not assign a domain a name that duplicates a DATATRIEVE keyword.

Results

- DATATRIEVE prompts you with the DFN> prompt until you type a semicolon and press RETURN or until it detects a syntax error. If you make a syntax error, DATATRIEVE returns to command level (indicated by the DTR> prompt) without creating a domain definition.
- DATATRIEVE enters the domain definition in your current dictionary and creates an access control list (ACL) for the domain by entering a UIC/PPN in the ACL with full access privileges (RWMEC). The actual UIC/PPN stored in the list depends on your installation. Refer to Chapter 20 in the *DATATRIEVE-11 User's Guide* for more information.

Usage Notes

- The record and the data file associated with the domain need not be defined when you issue a DEFINE DOMAIN command.
- You cannot modify a domain definition with the DEFINE DOMAIN command. Make the desired changes. When you exit from the Editor, DATATRIEVE installs the updated domain definition in your current dictionary. See Chapter 17 of the *DATATRIEVE-11 User's Guide* and Section 5.25 of this manual for information on using the DATATRIEVE Editor.

Examples:

Define the domain PHONES. Use the record definition PHONE_REC. Specify PHONE.DAT as the input data file:

```
DTR> DEFINE DOMAIN PHONES USING(RET)
DFN> PHONE_REC ON PHONE.DAT(RET)
DFN> ;(RET)
DTR>
```

Define the domain HOUSE. Use HOUSE_REC as the associated record and store the data in the file DB2:[202,202]HOUSE.DAT.

```
DTR> DEFINE DOMAIN HOUSE USING(RET)
DFN> HOUSE_REC ON DB2:[202,202]HOUSE.DAT(RET)
DFN> ;(RET)
DTR>
```

5.14.2 Defining a View Domain

Function

Stores a view domain definition in your current dictionary and creates an ACL entry for the domain.

Format

```
DEFINE DOMAIN view-name OF domain-name [,...]      USING
        level-number-1 field-name-1      OCCURS FOR rse-1 .
        level-number-2 field-name-2    { OCCURS FOR rse-2 }
                                       { FROM domain-name-2 }  .
              .              .              .
              .              .              .
              .              .              .
;
```

Arguments

view-name

Is the name of the view being defined. The name cannot duplicate the name of any other object in the same dictionary.

domain-name

Is the name of a domain containing the records to be included in the view. You cannot use the name of the view. When specifying more than one domain, use a comma to separate each domain name from the next.

level-number

Is the level number for a field in the view definition.

field-name

Is the name of a field in the view definition.

A field-name followed by an OCCURS FOR rse clause has no relationship to any field in the domain or domains specified in the RSE.

A field-name followed by a FROM domain-name clause must be the name of a field in a domain specified in the OF domain-name clause.

OCCURS FOR rse

Establishes the field as a list, and indicates that the associated field is to be included in the view only for those records specified by the RSE. The RSE must contain a reference to one of the domains listed in the OF domain-name clause.

FROM domain-name

Indicates that the definition of the associated field is identical to that of the field of the same name in the domain specified by domain-name. The argument domain-name must be the same as that used in the preceding OCCURS FOR rse clause.

. (period)

Ends a field definition.

; (semicolon)

Ends the domain definition.

Restrictions

- The view definition must contain an OCCURS FOR rse clause as the top-level field and at least one FROM domain-name clause.
- The restrictions on defining RMS domains apply to defining view domains.

Results

- DATATRIEVE prompts you with the DFN> prompt until you type a semicolon and press RETURN or until it detects a syntax error. If you make a syntax error, DATATRIEVE returns to command level (indicated by the DTR> prompt) without creating a view domain definition.
- DATATRIEVE enters the view domain definition in your current dictionary and creates an access control list (ACL) for the domain by entering a UIC/PPN in the ACL with full access privileges (RWMEC).

Usage Notes

To include an entire record in a view field, specify the top-level field as the field name. To include only selected fields from a record, name each field you want in the view on a separate line.

See Chapter 14 of the *DATATRIEVE-11 User's Guide* for information on using views.

Example

Define a view of big yachts, ready the view domain, and print the first five records in the domain:

```
DTR> DEFINE DOMAIN BIGGEST_YACHTS OF YACHTS USING
DFN> 01 BIGGEST_YACHT     OCCURS FOR YACHTS WITH LOA GT 40.
DFN>    03 BUILDER        FROM YACHTS.
DFN>    03 MODEL          FROM YACHTS.
DFN>    03 PRICE          FROM YACHTS.
DFN> ;
DTR> READY YACHTS
DTR> SET NO PROMPT
DTR> READY BIGGEST-YACHTS
DTR> PRINT FIRST 5 BIGGEST-YACHTS

MANUFACTURER    MODEL      PRICE

 CHALLENGER   41          $51,228
 COLUMBIA     41          $48,490
 GULFSTAR     41          $41,350
 ISLANDER     FREEPORT    $54,970
 NAUTOR       SWAN 41

DTR>
```

See Chapter 14 of the *DATATRIEVE-11 User's Guide* for more examples of view definitions.

5.15 DEFINE FILE Command

Function

Creates an RMS sequential or indexed sequential data file for the DATATRIEVE RMS domain specified in the command.

Format

For Sequential Files:

```
DEFINE FILE [FOR] domain-name [,]
    [ ALLOCATION = n ]
    [ SUPERSEDE      ]   [,...]
    [ MAX            ]
```

For Indexed Sequential Files:

```
DEFINE FILE [FOR] domain-name [,]
    [ ALLOCATION = n ]
    [ SUPERSEDE      ]   [,...]
    [ MAX            ]
    { KEY = field-name-1  [ ( [NO] CHANGE [,]  [NO] DUP ) ] }  [,...]
```

Arguments

domain-name

Is the name of the DATATRIEVE RMS domain for which you want to create a data file.

ALLOCATION = n

Specifies an unsigned, nonzero integer that determines the number of disk blocks initially allocated for the data file. If you omit this argument, zero blocks are allocated for the file.

SUPERSEDE

Deletes any data file with a complete file specification exactly duplicating that specified in the RMS domain definition and replaces it with the new file DATATRIEVE creates. If you do not specify a version number for the file in your domain definition, the new file does not replace the old one but is assigned the next higher version number (non-RSTS/E systems only).

MAX

Causes DATATRIEVE to create a fixed-length RMS file for a domain whose record definition contains an OCCURS...DEPENDING clause. The length of every record in the data file has the maximum possible size, as determined by the value of the max argument in the OCCURS...DEPENDING clause:

```
OCCURS min TO max TIMES DEPENDING ON field-name
```

Each record can then store the maximum number of items in the list defined by the OCCURS...DEPENDING clause. If you omit this argument, DATATRIEVE does not create a file with fixed-length records of the maximum possible size. The size of each record is determined when you store it. If you define the file to be a sequential file, you cannot increase the size of a record to include more list items than you first stored in the record.

KEY = field-name

Causes DATATRIEVE to create an RMS indexed sequential file and specifies a field in the domain's record definition to be used as an index key for the domain's data file. When you omit this clause, DATATRIEVE creates an RMS sequential file. The first key field specified in the DEFINE FILE command is the primary key, and all subsequent ones are alternate keys. If you specify more than one KEY clause, use a comma (,) to separate each clause from the next. If you are defining a file for a hierarchical record, do not make a list field the primary key.

(NO) CHANGE

Determines whether or not you can modify the content of the associated key field. See Tables 5-3 and 5-4 for defaults and allowed combinations of key field attributes.

(NO) DUP

Determines whether or not you can assign the same value to the specified key fields of two or more records. See Tables 5-3 and 5-4 for defaults and allowed combinations of key field attributes.

Table 5-3: Default Values for KEY fields

Primary Key	Alternate Key(s)
NO CHANGE NO DUP	CHANGE DUP

Table 5-4: Allowed Combinations of Key Field Attributes

Combinations	Key Type	
	Primary	Alternate
CHANGE + DUP	Not allowed	Default
CHANGE + NO DUP	Not allowed	Not allowed
NO CHANGE + DUP	Allowed	Allowed
NO CHANGE + NO DUP	Default	Allowed

Restrictions

- To define a data file for an RMS domain, you must have A (ACCESS) privilege for the domain and E (EXECUTE) privilege for the record.
- You cannot assign the CHANGE attribute to a primary key. See Table 5-4 for the allowed combinations of key field attributes.
- You cannot designate a list field as the primary key when defining a file for a hierarchical record.
- The domain specified in the command must be an RMS domain, and not a view domain, DBMS-11 domain, or port.

Results

- If you do not include a KEY clause, DATATRIEVE creates an RMS sequential data file for the specified domain. If you include one or more KEY clauses, DATATRIEVE creates an RMS indexed sequential data file for the domain.
- The file specification for the RMS file created by this command is the same as the file specification defined in the DEFINE DOMAIN command that defined the domain name. If the domain definition omits a field in the file specification, DATATRIEVE uses the following defaults:

Field	Default
dev:	SY: (the system device)
[UIC/PPN]	Your default UIC/PPN
file-name	No default
.type	.DAT
;ver	1 or next higher version number (non-RSTS/E systems only)

- If you omit the ALLOCATION = n clause, DATATRIEVE sets the initial disk space allocation for the data file to zero blocks. When you store records in the data file, RMS automatically extends the data file according to the cluster size on RSTS/E systems and the default extension size on RSX systems.
- If the record definition associated with the specified domain contains no OCCURS...DEPENDING clause, DATATRIEVE creates a data file with a fixed-length record format.
- If the record definition associated with the specified domain contains an OCCURS...DEPENDING clause, DATATRIEVE creates a data file as follows:
  - If the clause does not include the MAX argument, DATATRIEVE creates a file with a variable-length record format.
  - If the clause includes the MAX argument, DATATRIEVE creates a data file with a fixed-length record format.
- If you include the SUPERSEDE argument and the file specification in the domain definition specifies a version number, DATATRIEVE deletes any existing data file having that file specification and version number and replaces it with the new data file created by the DEFINE FILE command. The new file has the same file specification including version number, if applicable, as that of the deleted file.

Usage Notes

- If you define a sequential file, you cannot delete records from it with the ERASE statement. You can, however, change the value of any field in a record in a sequential file.
- If you define an indexed sequential file, you can delete records from it with the ERASE statement. You cannot, however, change the value of the primary key field of a record or the value of any secondary key field with the NO CHANGE attribute.
- If you change the size of a record, you need to define a new file to agree with the new record definition. Otherwise, you receive an error message indicating "bad record size" when you try to ready the domain.
- If you define a sequential file for a hierarchical record(OCCURS...DEPENDING) and do not include a MAX clause, you cannot extend the length of the list without defining a new file.
- If you define an indexed sequential file for a hierarchical record (OCCURS...DEPENDING) and do not use MAX, you can extend the length of the list.
- See Chapter 6 in the *DATATRIEVE-11 User's Guide* for more information on defining data files.

Examples

Define an indexed file for the domain PHONES. Use the field NAME as the primary key:

```
DTR> DEFINE FILE FOR PHONES KEY = NAME(RET)
DTR>
```

Define a sequential file for the domain FAMILIES:

```
DTR> DEFINE FILE FOR FAMILIES(RET)
DTR>
```

Define a new indexed file for the domain YACHTS. Use the group field TYPE as the primary key, and allow duplicate values for this key. This command replaces the previous data file for YACHTS:

```
DTR> DEFINE FILE FOR YACHTS SUPERSEDE KEY=TYPE (DUP)(RET)
DTR>
```

5.16 DEFINE PORT Command

Function

Enters the definition of a DATATRIEVE port in your current dictionary and creates an access control list (ACL) for the port.

Format

```
DEFINE PORT port-name [USING] record-name ;
```

Arguments

port-name

Is the name of the port being defined. The name of the port cannot duplicate the name of any other object in the same dictionary.

record-name

Is the name of the record definition to be associated with the port. The name of the record cannot duplicate the name of any other object in the same dictionary.

; (semicolon)

Ends the port definition.

Restrictions

- Do not use a DATATRIEVE keyword as the name of a port or as the name of the record definition associated with the port.
- You cannot use a DEFINE PORT command as part of a compound statement.
- You cannot invoke a procedure in a port definition.
- You must enter the port definition in your current dictionary before using the port definition to transfer data between DATATRIEVE and your application program.

Results

- DATATRIEVE prompts you with the DFN> prompt until you type a semicolon and press RETURN or until it detects a syntax error. If you make a syntax error, DATATRIEVE returns to command level (indicated by the DTR> prompt) without entering the port definition in the dictionary.
- DATATRIEVE enters the domain definition in your current dictionary and creates an access control list (ACL) for the domain by entering a UIC/PPN in the ACL with full access privileges (RWMEC).

Usage Notes

- Before you can use the port, you must enter the record definition associated with the port in the data dictionary with the DEFINE RECORD command (see Section 5.18).
- Use the SHOW command to display the definition of the port on your terminal.
- Use the SHOW DOMAINS command to display the names of all the ports in your current dictionary.
- See the *DATATRIEVE-11 Call Interface Manual* for information about using a port to transfer data between DATATRIEVE and your application program:

Example

Define a port for transferring records between the YACHTS domain and an application program.

```
DTR> DEFINE PORT YPORT USING YACHT;RET
DTR>
```

5.17 DEFINE PROCEDURE Command

Function

Enters a procedure definition in your current dictionary and creates an access control list (ACL) for the procedure.

Format

```
DEFINE PROCEDURE procedure-name
                .
                .
                .
END_PROCEDURE
```

Arguments

procedure-name

Is the name of the procedure you want to define. That procedure name cannot duplicate the name of any other object in the same dictionary.

END_PROCEDURE

Ends the procedure.

Restrictions

- You must enter the DEFINE PROCEDURE command at DATATRIEVE command level (indicated by the DTR> prompt). It cannot be part of a DATATRIEVE statement.
- To define a procedure, you must have operating system write access privilege to the dictionary file.
- The procedure being defined cannot contain any DEFINE commands.

Results

- After you type DEFINE PROCEDURE procedure-name and press RETURN, DATATRIEVE displays the DFN> prompt. You then define the procedure by entering commands and statements in response to this prompt. DATATRIEVE continues to prompt with the DFN> prompt until you type END_PROCEDURE to end the definition.
- DATATRIEVE enters the procedure in your current dictionary and creates an access control list (ACL) for the domain by entering aUIC/PPN in the ACL with full access privileges (RWMEC).

Usage Notes

- To invoke a procedure during an interactive DATATRIEVE session, enter a colon followed by the name of the procedure. You can invoke a procedure in response to any DATATRIEVE prompt, except those of ADT, Guide Mode, and the Editor. You can also invoke procedures in the midst of input lines.
- To invoke a procedure with an invocation command line from your operating system level, type the command you use to invoke DATATRIEVE, a colon, and the procedure name.
- The DEFINE PROCEDURE command creates procedures. To modify a procedure after you have stored it in the dictionary, use the DATATRIEVE Editor. When you exit from the Editor, DATATRIEVE deletes the old procedure definition and stores the updated procedure definition in your current dictionary. See Chapter 17 of the *DATATRIEVE-11 User's Guide* and Section 5.25 of this manual for information on using the DATATRIEVE Editor.
- When you invoke a procedure from a REPEAT or FOR loop, enclose the procedure in a BEGIN-END block to ensure that DATATRIEVE executes all the statements in the procedure each time through the loop. A procedure invoked in this way cannot include any commands or FIND, SELECT, DROP, or RELEASE statements.
- For more information on procedures, see Chapter 9 of the *DATATRIEVE-11 User's Guide.*

Examples

Define a procedure to set your current dictionary to DEMO.DIC:

```
DTR> DEFINE PROCEDURE DEMO↵
DFN> SET DICTIONARY DB2:[53,27]DEMO.DIC↵
DFN>     SHOW DICTIONARY↵
DFN> END_PROCEDURE↵
DTR> :DEMO↵
The default dictionary is DB2:[53,27]DEMO.DIC
DTR>
```

Define a procedure that displays a group of boats with a price less than a figure you supply when you run the procedure.

```
DTR> DEFINE PROCEDURE PRICE_LIST↵
DFN>     READY YACHTS↵
DFN>     PRINT SKIP, COL 20, "*** Price List of YACHTS ***", SKIP↵
DFN>     FOR YACHTS WITH PRICE NE 0 AND PRICE LE *,"the ceiling price"↵
DFN>     PRINT BOAT↵
DFN>     PRINT SKIP, COL 10, "See anything interesting?"↵
DFN> END_PROCEDURE↵
DTR> :PRICE_LIST↵

                    *** Price List of YACHTS ***

Enter the ceiling price: 5,000

                                  LENGTH
                                   OVER
MANUFACTURER    MODEL      RIG     ALL    DISPLACEMENT BEAM  PRICE

 CAPE DORY    TYPHOON     SLOOP    19         1,900     06   $4,295
 VENTURE      21          SLOOP    21         1,500     07   $2,823
 VENTURE      222         SLOOP    22         2,000     07   $3,564
 WINDPOWER    IMPULSE     SLOOP    16           650     07   $3,500

          See anything interesting?

DTR>
```

Use an invocation command line to invoke the DEMO procedure:

```
$ DTR :DEMO↵
The default dictionary is DB2:[53,27]DEMO.DIC
```

5.18 DEFINE RECORD Command

Function

Enters a record definition in your current dictionary and creates an access control list (ACL) for the record definition.

Format

```
DEFINE RECORD record-name [USING]
    [ ALLOCATION IS { MAJOR-MINOR
                    { ALIGNED-MAJOR-MINOR
                    { LEFT-RIGHT                } ]
      level-number-1 field-name-1 [field-definition-1] .
     [level-number-2 field-name-2 field-definition-2 .]
          .          .          .
          .          .          .
          .          .          .
;
```

Arguments

record-name

Is the name of the record being defined. The record name cannot duplicate the name of any other object in the same dictionary.

ALLOCATION

Specifies the type of word-boundary alignment DATATRIEVE uses when storing records in the data file. It also controls the way DATATRIEVE retrieves data from data files created by user programs or other application software. The default allocation is LEFT_RIGHT. See the *PDP-11 COBOL Language Reference Manual* or the *COBOL-81 Language Reference Manual* for more information on word-boundary alignment and allocation of fill bytes. See Section 5.6 in this manual for more information on the ALLOCATION clause.

level-number

Is the level number for the field in the record definition. It indicates the relationship of the field to the other fields in the record definition.

field-name

Is the name of the field. Every field must have a name. The keyword FILLER is a special field name that can be repeated at the same level in the record definition.

field-definition

Is a field definition. A record definition must contain at least one field definition. Elementary fields must have at least one field definition clause. Group fields do not have to have any field definition clauses.

. (period)

Ends the field definition.

; (semicolon)

Ends the record definition.

Restrictions

- You cannot invoke a procedure in a record definition.
- The level number must be an integer between 1 and 65.
- A record definition must contain the field definition of at least one elementary field.
- The field definition of an elementary field must contain at least one field definition clause.
- No field name should duplicate the domain name.

Results

- DATATRIEVE prompts with the DFN> prompt until you type a semicolon and press return, or until it detects a syntax error. If you make a syntax error, DATATRIEVE returns to command level (indicated by the DTR> prompt) without creating the record definition.
- When you end the record definition, DATATRIEVE displays a message on your terminal that indicates the length of the new record:

```
DTR> DEFINE RECORD A_REC USING(RET)
DFN> 01 TOP.(RET)
DFN>    03 CHAR PIC X.(RET)
DFN>    03 NUM USAGE IS COMP.(RET)
DFN> ;(RET)
[Record A_REC is 4 bytes long]
DTR>
```

- DATATRIEVE enters the record definition in your current dictionary and creates an ACL for the record definition by entering a UIC/PPN with full access privileges of RWMEC.

Usage Notes

- The DEFINE RECORD command creates records. To modify or replace a record after you have stored it in the dictionary, use the DATATRIEVE Editor. When you exit from the Editor, DATATRIEVE deletes the old record definition and stores the updated record definition in your current dictionary. See Chapter 17 of the *DATATRIEVE-11 User's Guide* and Section 5.25 of this manual for information on using the DATATRIEVE Editor.
- If you change a record definition, you may not be able to use old data files. If the new record definition is not the same length, or if the new field definitions change to data types incompatible with the previous definition, you have to redefine and restructure the data as described in Chapter 16 of the *DATATRIEVE-11 User's Guide.*

  The safest method for changing record definitions is to define a new domain and a new file to accompany the new record definition, and use the STORE statement to transfer the values from the data file of the old domain to the data file of the new domain.

Examples

Define the record PHONE_REC:

```
DTR> DEFINE RECORD PHONE_REC USING
DFN> 01 PHONE.
DFN>     02 NAME PIC X(20).
DFN>     02 NUMBER PIC 9(7) EDIT_STRING IS XXX-XXXX.
DFN>     02 LOCATION PIC X(9).
DFN>     02 DEPARTMENT PIC XX.
DFN> ;
[Record PHONE_REC is 38 bytes long.]
DTR>
```

Define the record FAMILY:

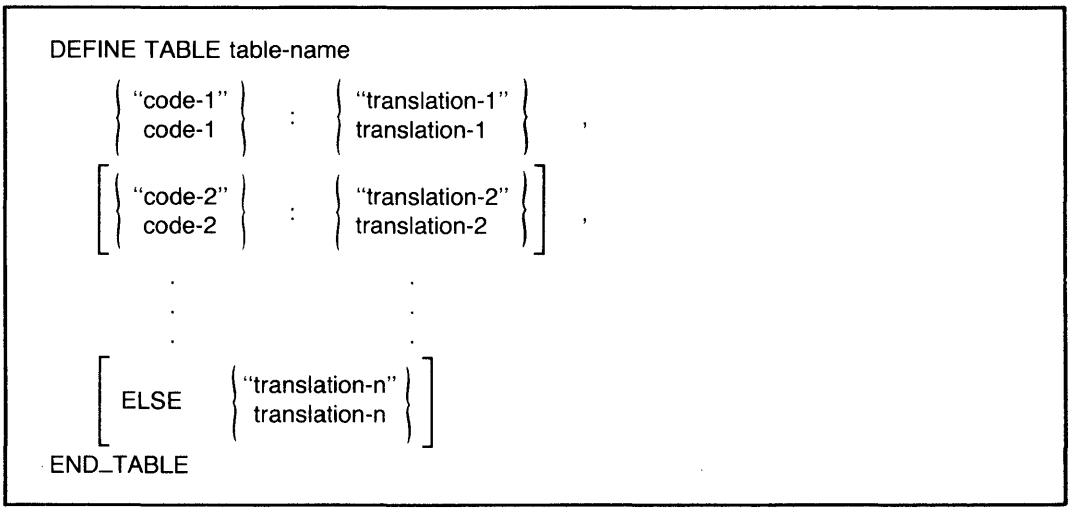
```
DTR> DEFINE RECORD FAMILY USING
DFN> 01 FAMILY.
DFN>     03 PARENTS.
DFN>         06 FATHER PIC X(10).
DFN>         06 MOTHER PIC X(10).
DFN>     03 NUMBER_KIDS PIC 99 EDIT_STRING IS Z9.
DFN>     03 KIDS OCCURS 0 TO 10 TIMES DEPENDING ON NUMBER_KIDS.
DFN>         06 EACH_KID.
DFN>             09 KID_NAME PIC X(10) QUERY_NAME IS KID.
DFN>             09 AGE PIC 99 EDIT_STRING IS Z9.
DFN> ;
[Record FAMILY is 142 bytes long.]
DTR>
```

5.19 DEFINE TABLE Command

Function

Enters a dictionary table in your current dictionary and creates an access control list (ACL) for the table.

Format



Arguments

table-name

Is the name of the dictionary table being defined. The name of the table cannot duplicate the name of any other object in the same dictionary.

"code" : "translation"
code : translation

Are code-translation pairs. You must separate the code and its translation with a colon. The comma after each pair is required except for the last pair if there is no ELSE clause. If the code or translation string conforms to the rules for DATATRIEVE names, you do not have to use quotation marks. DATATRIEVE converts lowercase letters in unquoted code or translation strings to uppercase letters.

If the code or translation string does not conform to the rules for DATATRIEVE names (especially if it contains spaces), or if you want to preserve lowercase letters, you must use quotation marks and follow the rules for character string literals (see Chapter 2).

ELSE "translation"
translation

Is the translation to be used if you specify a code not defined in the dictionary table. The rules for specifying this translation string are the same as those for codes and translations.

END_TABLE

Ends the dictionary table definition.

Restrictions

- You cannot include the invocation of a procedure (:procedure-name) in a dictionary table definition.
- The name of a dictionary table cannot duplicate a DATATRIEVE keyword.

Results

- DATATRIEVE prompts with the DFN> prompt until you end the table definition with the keyword END_TABLE.
- DATATRIEVE enters the dictionary table in your current dictionary and creates an ACL for the table by entering a UIC/PPN with full access privileges of RWMEC.

Usage Notes

- When you invoke a dictionary table with an IN or VIA clause, the table is loaded into your workspace and remains available to you until you remove it with a RELEASE command or exit from DATATRIEVE. Changing your current dictionary does not affect dictionary tables loaded in your workspace.
- Use the SHOW READY command to display the names of dictionary tables loaded in your workspace.
- DATATRIEVE uses a default edit string of 10 characters when it displays the translation string on your terminal or writes the value to another output device. You can specify an edit string each time you use a VIA value expression. For example:

```
DTR> PRINT "CE" VIA DEPT_TABLE↵
Commercial

DTR> PRINT "CE" VIA DEPT_TABLE USING T(25)↵
Commercial Engineering

DTR>
```

- You can specify a column header each time you use a VIA value expression.
- For more information on using dictionary tables, see Chapter 13 of the *DATATRIEVE-11 User's Guide*.

- The definition of a dictionary table differs from the definitions of domains and records because it contains values, not just a data description.
- To modify a dictionary table, use the DATATRIEVE Editor to modify, delete, or add codes and translations. When you exit from the Editor, DATATRIEVE deletes the old table definition and stores the updated definition in your current dictionary. See Chapter 17 of the *DATATRIEVE-11 User's Guide* and Section 5.25 of this manual for information on using the DATATRIEVE Editor.

Examples

Define a table of department codes and display the translation for the RD code string:

```
DTR> DEFINE TABLE DEPT_TABLE(RET)
DFN> CE : "Commercial Engineering",(RET)
DFN> PE : "Plant Engineering",(RET)
DFN> CS : "Customer Support",(RET)
DFN> RD : "Research and Development",(RET)
DFN> SD : "Sales Department",(RET)
DFN> ELSE "UNKNOWN DEPARTMENT"(RET)
DFN> END_TABLE(RET)
DTR> PRINT "RD" VIA DEPT_TABLE USING T(25)(RET)
Research and Development

DTR>
```

Define a table with a translation for each possible rig:

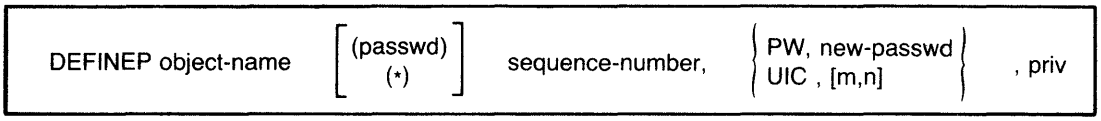
```
DTR> DEFINE TABLE RIG_TABLE(RET)
DFN> SLOOP : "ONE MAST",(RET)
DFN> KETCH : "TWO MASTS, BIG ONE IN FRONT",(RET)
DFN> YAWL  : "SIMILAR TO KETCH",(RET)
DFN> MS    : "SAILS AND A BIG MOTOR",(RET)
DFN> ELSE    "SOMETHING ELSE"(RET)
DFN> END_TABLE(RET)
DTR> PRINT "KETCH" VIA RIG_TABLE USING T(30)(RET)
TWO MASTS, BIG ONE IN FRONT

DTR>
```

5.20 DEFINEP Command

Function

Adds an entry to the access control list (ACL) for a dictionary object.

Format



Arguments

obj-name

Is the name of the domain, record, procedure, or table whose ACL you want to modify.

(passwd)
(*)

Is the password necessary to gain C (control) access to the dictionary object or an asterisk enclosed in parentheses (*). If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts for the password, but does not print your response on the terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have C (control) access privilege.

seq-no

Is the sequence number of the entry to be added to the ACL. This argument must be an unsigned integer and must be followed by a comma (,).

PW, new-passwd
UIC, [m,n]

Specifies the lock type and key for the entry. If the lock type is PW, specify a 1- to 10-character password. If it is UIC, specify a UIC/PPN, enclosed in square brackets. You must put a comma after the lock type and another comma after the key.

priv

Is a letter or string of letters indicating the type of access privilege to be granted. You can also use a space enclosed in quotation marks to indicate that the user is not granted any access privileges. Table 5-5 lists access privilege codes.

Table 5-5: Access Control Privilege Codes

Letter	Privilege
R	Read
E	Execute or Extend
M	Modify
W	Write
C	Control
""	No privilege

Restriction

You must have C (control) access privilege to the dictionary object.

Result

DATATRIEVE creates an entry in the ACL. Depending on the sequence number you supply, DATATRIEVE may change the sequence number of other entries in the ACL:

- If the sequence number already exists in the ACL, DATATRIEVE adds the entry immediately before the existing entry with the same number and increases by one the sequence number of all entries after the new entry.
- If the sequence number is greater than the last sequence number plus one, DATATRIEVE ignores your sequence number and adds the entry to the end of the ACL. Its sequence number becomes the next sequential number in the ACL.
- You can specify more than one privilege with a string of letters, in any order, such as RWM for read, write, and modify access. Do not put spaces between letters in a string of letters.

Usage Notes

- When designing an ACL, put entries with the most specific user identification criteria at the top of the list and entries with the most general user identification criteria at the bottom of the list. When you access a dictionary object, DATATRIEVE begins at the top of the list and applies the first entry in which all the user identification criteria apply to you.

  If, for example, the first ACL entry specifies a number as the only user identification criterion in the entry and you are using that number, you match all the user identification criteria for that entry. Your UIC/PPN and any password you supply do not matter. That you also match other entries in the ACL is of no consequence; your other user identification characteristics would not be checked against other ACL entries until you used another number to access the dictionary object in question.

- You can enter user identification criteria and privilege specifications in any order. You need not put the user identification criteria before the privilege specifications.

- To avoid errors when making an addition, display on your terminal a copy of the current ACL with the SHOWP command before issuing the DEFINEP command. See the description in this chapter of the SHOWP command. Because DATATRIEVE can change sequence numbers, a new entry can affect the numbering of other ACL entries.

- To remove an entry from an ACL, use the DELETEP command (see Section 5.23).

- Chapter 20 of the *DATATRIEVE-11 User's Guide* discusses the use of ACLs.

Example

Define an ACL entry for a dictionary that uses all the user identification criteria and all the privilege specifications. Use the SHOWP command to show the new access control list:

```
DTR> DEFINEP MONTHLY_DATA 1,PW, "GELSEY","C"RET
DTR> DEFINEP MONTHLY_DATA 2,UIC, "[150,*]","R"RET
DTR> SHOWP MONTHLY_DATARET
        1,PW, "GELSEY", "C"
        2,UIC, [150,*], "R"
DTR>
```

5.21 DELETE Command

Function

Deletes a dictionary object and its associated access control lists (ACLs) from your current dictionary.

Format

```
DELETE object-name ;
```

Arguments

object-name

Is the name of the object you want to remove from the dictionary.

; (semicolon)

Ends the DELETE command.

Restrictions

- You must have C (control) access privilege to the dictionary object before you can delete it.
- You cannot delete a dictionary with the DELETE command, even if the dictionary is empty. To delete a dictionary, you must use the operating system DELETE command. See the *RSTS/E DCL User's Guide* or the *RSX-11M/M-PLUS Command Language Manual*, as appropriate.

Results

- If you specify the name of a domain definition or a record definition in the DELETE command, DATATRIEVE deletes the definition of the domain or record, but does not delete the associated record or domain definition or the associated data file.
- A readied domain is not affected by the deletion of the domain definition or the definition of its associated record. However, after you release a domain whose definition or record definition has been deleted from the dictionary, you cannot ready the domain again.
- If the dictionary object you specify in a DELETE command is a procedure, DATATRIEVE deletes the procedure from the data dictionary. You cannot invoke a procedure after it has been deleted from the dictionary.
- If the dictionary object you specify in a DELETE command is a dictionary table, DATATRIEVE deletes the dictionary table from the data dictionary. A dictionary table loaded in your workspace remains there until you release it, even if you delete its definition. However, after you have released the table, it cannot be loaded again.
- When DATATRIEVE deletes a dictionary object, it also deletes the ACL associated with the object.

Examples

Delete a procedure from your current dictionary:

```
DTR> SHOW HELO↵
PROCEDURE HELO
          PRINT "Good morning"
END_PROCEDURE

DTR> DELETE HELO;↵
DTR> SHOW HELO↵
"HELO" has not been defined in the dictionary
DTR>
```

Delete the domain and record definitions of a readied domain:

```
DTR> SHOW YACHTS_SEQ↵
DOMAIN YACHTS_SEQ
 USING YACHT_SEQ_REC ON YACHTS.DAT;
DTR> READY YACHTS_SEQ↵
DTR> FIND YACHTS_SEQ↵
[113 records found]
DTR> DELETE YACHTS_SEQ↵
DTR> SHOW YACHT_SEQ↵
"YACHTS_SEQ" has not been defined in the dictionary
DTR> DELETE YACHT_SEQ_REC↵
DTR> SHOW YACHT_SEQ_REC↵
"YACHT_SEQ_REC" has not been defined in the dictionary
DTR> SHOW READY↵
Ready domains:
        YACHTS_SEQ:  RMS INDEXED, PROTECTED READ

DTR> SHOW FIELDS↵
YACHTS_SEQ
   BOAT
      TYPE [Indexed field]
          MANUFACTURER (BUILDER)      [Character string, indexed key]
          MODEL       [Character string, indexed key]
      SPECIFICATIONS (SPECS)
          RIG [Character string]
          LENGTH_OVER_ALL (LOA)       [Character string]
          DISPLACEMENT (DISP) [Number]
          BEAM        [Number]
          PRICE       [Number]

DTR> FINISH YACHTS_SEQ↵
DTR> SHOW READY↵
No Domains Readied
DTR> SHOW YACHTS_SEQ↵
"YACHTS_SEQ" has not been defined in the dictionary
DTR> SHOW YACHT_SEQ_REC↵
"YACHT_SEQ_REC" has not been defined in the dictionary
DTR> SHOW FIELDS↵
No Domains Readied or Global Variables Declared
DTR>
```

5.22 DELETEP Command

Function

Deletes an entry from the access control list (ACL) of a dictionary object.

Format

```
DELETEP object-name sequence-number
```

Arguments

object-name

Is the name of the dictionary object whose ACL you want to change.

sequence-number

Is a nonzero integer indicating the entry's position in the ACL.

Restrictions

- You must have C (control) access privilege to the dictionary object before you can delete an entry from its ACL.
- You can delete only one ACL entry at a time with the DELETEP command.

Results

- DATATRIEVE deletes the entry with the specified sequence number from the ACL of the dictionary object.
- If the sequence number you specify is greater than the number of entries in the ACL, DATATRIEVE displays the message "Protection table element doesn't exist" and does not delete anything.
- When you remove an entry from any position in the ACL, except the last one, DATATRIEVE renumbers the remaining entries so that the sequence numbers begin at one and increase in steps of one.

Usage Notes

- To ensure that you delete the correct entry, display the ACL on your terminal with the SHOWP command before you enter the DELETEP command.
- If you need to remove many entries from a long ACL, begin with the entries at the bottom of the list and work your way toward the top. This method preserves the original sequence numbers of the entries you want to remove. If you start removing entries at the top of the list, every time you remove an entry the numbers of all the other ones you want to remove change.

- DATATRIEVE allows you to delete the last ACL entry having C access, so be careful when deleting ACL entries. The only way you can gain access to a dictionary object that does not allow you C access is to log into a privileged account from the system command level before invoking DATATRIEVE. A privileged account automatically grants you C (control) access.

Examples

Show the ACL of the YACHTS domain and delete an entry from it:

```
DTR> SHOWP YACHTS↵
        1, UIC, [32, 56], "RWMEC"
        2, UIC, [150, 150], "R"

DTR> DELETEP YACHTS 2↵
DTR> SHOWP YACHTS↵
        1, UIC, [32, 56], "RWMEC"

DTR>
```

See Chapter 20 of the *DATATRIEVE-11 User's Guide* for other examples of working with ACLs.

5.23 DISPLAY Statement

Function

Displays on your terminal the value of a single DATATRIEVE value expression. The value displayed is not formatted by any edit string associated with the value expression.

Format

```
DISPLAY value-expression
```

Arguments

value-expression

Is a DATATRIEVE value expression.

Restrictions

- To display the value of a field in a record, the domain containing that record must be readied for read, write, or modify access. If you specify a field name from a domain readied for extend access, DATATRIEVE displays an error message.
- You must establish a valid context for the record or records containing the field value or values you want to display. See Chapter 12 in the *DATATRIEVE-11 User's Guide* for information on record context.

Results

- DATATRIEVE displays the current, unformatted value of the specified value expression, ignoring the COLUMNS_PAGE setting and any edit string associated with the value.
- If the specified value expression is a group field, DATATRIEVE concatenates the values of the elementary fields and leaves no spaces between fields except the blanks that pad character string fields.

Examples

Declare a numeric variable with a money edit string, assign it a value, and use the PRINT and DISPLAY statements to display the value:

```
DTR> DECLARE SALARY PIC Z(5)9V99 EDIT_STRING $$$$$$.99.RET
DTR> SALARY = 15753.67RET
DTR> PRINT SALARYRET

 SALARY

$15753.67

DTR> DISPLAY SALARYRET
DISPLAY: 15753.67
DTR>
```

Declare the SALARY variable as a character variable, assign it a value, and use the PRINT and DISPLAY statements to display the value:

```
DTR> DECLARE SALARY PIC X(15).RET
DTR> SALARY = "MUCH TOO LOW"RET
DTR> PRINT SALARYRET

    SALARY

MUCH TOO LOW

DTR> DISPLAY SALARYRET
DISPLAY: MUCH TOO LOW
DTR>
```

Display the group fields TYPE, SPECS, and BOAT of a record in the YACHTS domain:

```
DTR> READY YACHTS RET
DTR> FIND FIRST 1 YACHTSRET
[1 Record found]
DTR> SELECT; PRINTRET

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG   ALL   WEIGHT BEAM  PRICE

 ALBERG       37 MK II   KETCH   37    20,000  12  $36,951

DTR> DISPLAY TYPERET
DISPLAY: ALBERG     37 MK II
DTR> DISPLAY SPECSRET
DISPLAY: KETCH 37 200001236951
DTR> DISPLAY BOATRET
DISPLAY: ALBERG     37 MK II  KETCH 37 200001236951
DTR>
```

5.24 DROP Statement

Function

Removes the selected record from a collection, but does not remove that record from the data file in which it resides.

Format

```
DROP [collection-name]
```

Arguments

collection-name

Is the name of a collection. If you do not specify a collection, DATATRIEVE removes the selected record in the most recent single record context.

Restrictions

- You must use the SELECT statement to establish a selected record in a collection before using a DROP statement. DATATRIEVE displays the message "No collection with selected record for DROP" if you have no established collections, or if no collection has a selected record.
- You cannot drop a record that has been erased or dropped. If you have already removed the selected record, DATATRIEVE displays a message and the DROP statement does not execute.
- You can use the DROP statement only to remove records from collections. DATATRIEVE displays the message "DROP can only be used for collections" if you try to use DROP to remove anything other than a record selected from a collection.

Results

- Once a record has been dropped from a collection, it is no longer available to you in the collection. The dropped record is not erased from the data file. You can retrieve it again by forming a record stream or another collection that contains it.

- When you enter a DROP statement without specifying a collection name, DATATRIEVE drops the selected record in the nearest single record context:
  - If the CURRENT collection has a selected record, DATATRIEVE drops it from the CURRENT collection.
  - If the CURRENT collection has no selected record but other named collections do, DATATRIEVE drops the selected record from the most recently formed of those collections.
  - If the selected record in the nearest single record context has been previously dropped, DATATRIEVE displays the message "No collection with selected record for DROP" and the DROP statement does not execute.
- When you specify a collection name, DATATRIEVE drops the selected record in the specified collection. If the named collection has no selected record, or if the selected record has been erased, DATATRIEVE displays an error message and does not execute the DROP statement.

Usage Notes

- Use the DROP statement to refine a collection until it contains exactly the records you want.
- Before dropping a selected record in the nearest single record context, display the record on your terminal by typing PRINT and pressing RETURN.

Examples

Form a collection, select a record, and drop the selected record. PRINT statements show the established collection before and after the DROP statement executes:

```
DTR> READY YACHTS↵
DTR> FIND A IN YACHTS WITH BUILDER = "AMERICAN"↵
[2 records found]
DTR> PRINT A↵

                                  LENGTH
                                   OVER
MANUFACTURER    MODEL      RIG     ALL    WEIGHT BEAM   PRICE

 AMERICAN     26          SLOOP    26      4,000  08    $9,895
 AMERICAN     26-MS       MS       26      5,500  08   $18,895

DTR> SELECT↵
DTR> DROP↵
DTR> PRINT A↵

                                  LENGTH
                                   OVER
MANUFACTURER    MODEL      RIG     ALL    WEIGHT BEAM   PRICE

 AMERICAN     26-MS       MS       26      5,500  08   $18,895

DTR>
```

Store a record in YACHTS, form a series of collections, and use the SELECT, DROP, ERASE, and PRINT statements to illustrate the difference between the DROP and ERASE statements:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS WRITE↵
DTR> STORE YACHTS USING BUILDER = "AMERICAN"↵
DTR> FIND A IN YACHTS WITH BUILDER = "AMERICAN"↵
[3 records found]
DTR> PRINT A↵
                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM   PRICE

 AMERICAN                                       0  00
 AMERICAN     26          SLOOP    26       4,000  08    $9,895
 AMERICAN     26-MS       MS       26       5,500  08   $18,895

DTR> SELECT FIRST; PRINT↵

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM   PRICE

 AMERICAN                                       0  00

DTR> DROP↵
DTR> PRINT A↵

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM   PRICE

 AMERICAN     26          SLOOP    26       4,000  08    $9,895
 AMERICAN     26-MS       MS       26       5,500  08   $18,895

DTR> FIND B IN YACHTS WITH BUILDER = "AMERICAN"↵
[3 records found]
DTR> PRINT B↵

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM   PRICE

 AMERICAN                KETCH                  0  00
 AMERICAN     26          SLOOP    26       4,000  08    $9,895
 AMERICAN     26-MS       MS       26       5,500  08   $18,895

DTR> SELECT FIRST↵
DTR> ERASE↵
DTR> DROP↵
No collection with selected record for DROP
DTR> RELEASE A, B↵
DTR> FIND YACHTS WITH BUILDER = "AMERICAN"↵
[2 records found]
DTR> PRINT CURRENT↵
```

(continued on next page)

```
                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM   PRICE

 AMERICAN       26         SLOOP    26      4,000  08    $9,895
 AMERICAN       26-MS      MS       26      5,500  08   $18,895

DTR>
```

Form a collection and use a FOR statement to drop all records in the collection. PRINT statements show the established collection and the empty collection after the DROP statement:

```
DTR> FIND A IN YACHTS WITH BUILDER = "CHALLENGER"RET
[3 records found]
DTR> PRINT CURRENTRET

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM   PRICE

 CHALLENGER     32         SLOOP    32     12,800  11   $31,835
 CHALLENGER     35         SLOOP    35     14,800  12   $39,215
 CHALLENGER     41         KETCH    41     26,700  13   $51,228

DTR> FOR CURRENT DROPRET
DTR> PRINT CURRENTRET
DTR>
```

Form a collection and use the FOR statement to selectively drop records. PRINT statements show the established collection and the refined collection:

```
DTR> SET NO PROMPTRET
DTR> FIND A IN YACHTS WITH BUILDER = "CHALLENGER" OR RET
CON> BUILDER = "AMERICAN"RET
[5 records found]
DTR> PRINT ARET

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM   PRICE

 AMERICAN       26         SLOOP    26      4,000  08    $9,895
 AMERICAN       26-MS      MS       26      5,500  08   $18,895
 CHALLENGER     32         SLOOP    32     12,800  11   $31,835
 CHALLENGER     35         SLOOP    35     14,800  12   $39,215
 CHALLENGER     41         KETCH    41     26,700  13   $51,228

DTR> FOR A WITH BUILDER = "CHALLENGER" DROPRET
DTR> PRINT ARET

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM   PRICE

 AMERICAN       26         SLOOP    26      4,000  08    $9,895
 AMERICAN       26-MS      MS       26      5,500  08   $18,895

DTR>
```

Form a collection, select and drop a record, then select the record before the dropped record. When you try to select the next record (the record dropped previously,) an error message informs you that the selected record has been dropped:

```
DTR> FIND A IN YACHTS WITH BUILDER = "CHALLENGER"↵
[3 records found]
DTR> PRINT↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL    WEIGHT BEAM  PRICE

 CHALLENGER   32          SLOOP   32     12,800  11  $31,835
 CHALLENGER   35          SLOOP   35     14,800  12  $39,215
 CHALLENGER   41          KETCH   41     26,700  13  $51,228

DTR> SELECT 3↵
DTR> DROP↵
DTR> PRINT A↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL    WEIGHT BEAM  PRICE

 CHALLENGER   32          SLOOP   32     12,800  11  $31,835
 CHALLENGER   35          SLOOP   35     14,800  12  $39,215

DTR> SELECT 2↵
DTR> SELECT NEXT↵
Record has been dropped from collection
Execution failed
DTR>
```

Try to use the DROP statement to remove records from a domain. An error message informs you that you cannot use a domain name in a DROP statement:

```
DTR> FOR YACHTS WITH RIG = "SLOOP" DROP↵
DROP can only be used for collections
DTR>
```

5.25 EDIT Command

Function

Invokes the DATATRIEVE Editor to edit a dictionary object.

Format

```
EDIT object-name [(passwd)] [ADVANCED]
                 [(*)    ]
```

Arguments

object-name

Is the name of a DATATRIEVE domain, record, procedure, or table definition you want to edit.

(passwd)
(*)

Is an asterisk enclosed in parentheses (*) or the password necessary to gain C (control) access to the dictionary object. If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts for the password, but does not display your response on the terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have C (control) access privilege to the object you want to edit.

ADVANCED

Specifies that you want to edit a domain or record definition. You must include this keyword before you can edit a domain or record definition.

Restrictions

- You must have W (write) access to your current dictionary before you can use the EDIT command.
- To edit a DATATRIEVE domain, record, procedure, or table definition, you must have C (control) access privilege to the dictionary object you want to edit.

Results

- When you use the EDIT command, DATATRIEVE invokes the Editor, loads the specified definition into the main text buffer of the Editor, and prompts with QED>, the editing command mode prompt. You can type any Editor command in response to the prompt. Editor commands are summarized in Table 5-6.

Table 5-6: Summary of DATATRIEVE Editor Commands

Command	Format and Functions
CTRL/Z	CTRL/Z Ends an editing session and returns you to DATATRIEVE command level when typed in response to the QED> prompt. The edited dictionary object replaces the previous version in the data dictionary. When typed in response to the IN> prompt, CTRL/Z exits from insert mode.
DELETE	D[ELETE] [range] Deletes the current line or the specified range and sets the current line as the line following the last deleted line.
EXIT	EX[IT] Ends an editing session and returns you to DATATRIEVE command level; the edited dictionary object replaces the previous version in the data dictionary.
INSERT	I[NSERT] [range] Enters insert mode. You type lines to be inserted before the current line or before the first line of the specified range. Type CTRL/Z to exit from insert mode. If you do not specify a range, the current line does not change when you leave insert mode; if you specify a range, the Editor sets the line specified by the range as the current line. Do not use ALL, AND, BEFORE, FOR, REST, or WHOLE when specifying the range in this command.
QUIT	QUIT Returns you to DATATRIEVE command level and leaves the dictionary object unchanged by the editing session.
REPLACE	R[EPLACE] [range] Deletes the current line or the specified range and enters insert mode. Type CTRL/Z to exit from insert mode. The Editor sets the current line to the line following the last deleted line. You can use all range specifiers with this command.
SUBSTITUTE	S/str-1/[str-2][/[range]] Substitutes string 2 for all occurrences of string 1 in the specified range or in the current line if you omit the range. The Editor sets the line in which the last substitution occurred as the current line. If you omit string 2, the Editor deletes the specified string and makes no substitution. The search for the first occurrence of string 1 starts with the current line and proceeds toward the end of the text buffer except when you specify WHOLE as the range; when you specify WHOLE, the Editor searches and substitutes from the beginning to the end of the text buffer. Do not use END to specify the range in this command.
TYPE	[T[YPE]] [range] Displays the specified range of lines or the line following the current line if you omit the range. Sets the first line displayed as the current line, with one exception: if you specify E[ND] in the range, the Editor sets the current line at the end-of-buffer marker ([EOB]).

- To end an editing session, type EXIT, CTRL/Z, or QUIT in response to the QED> prompt:
  - EXIT or CTRL/Z causes DATATRIEVE to update the definition. If the object name is a procedure, DATATRIEVE does not invoke the procedure.
  - QUIT causes DATATRIEVE to ignore the contents of the Editor's text buffer, display the message "Execution failed", and return you to DATATRIEVE command level.
- The DATATRIEVE Editor does not check the syntax of any DATATRIEVE commands or statements in the text buffer. If you make a syntax error when correcting your definition, DATATRIEVE responds to the error only when it executes the commands or statements after you exit from the Editor, or when you invoke the edited procedure or ready the domain.

Usage Notes

- The DATATRIEVE Editor has two modes: command mode (the edit mode) and insert mode.
  - When you use the EDIT command to invoke the Editor, the Editor prompts with QED>, the command mode prompt. In command mode, the Editor interprets all your inputs as commands, and you can display and alter the text of the dictionary object.
  - In insert mode, you can enter text directly into the dictionary object. To enter insert mode, use the INSERT and REPLACE commands. The Editor then prompts with IN>, the insert mode prompt. In insert mode, the Editor interprets all your input as new text to be entered into the dictionary object. To exit from insert mode, type CTRL/Z.
- The Editor uses a line pointer to keep track of the current line. Some editing commands move you through the text from one line to another, thus changing the current line. Other commands display or alter lines at various places in the text but leave the current line unchanged. The line pointer keeps track of your position in the text.
  - The line pointer points to the entire current line, not to any part of the line. To display the current line, type a period (.) and carriage return in response to the QED> prompt.
  - The maximum line size you can edit is 132 characters.
  - The current line pointed to by the line pointer can be a blank line at the end of the text buffer, thus allowing you to add text to the end of the dictionary object. The symbol [EOB] marks the end of the text buffer.

- The DELETE, INSERT, REPLACE, SUBSTITUTE, and TYPE Editor commands strings all contain an optional argument that specifies the range of lines on which the command operates. The range may be a single line, a series of consecutive lines, or a group of nonsequential lines. Table 5-7 summarizes the range specifiers you can use with DATATRIEVE Editor commands. See Chapter 17 in the *DATATRIEVE-11 User's Guide* for examples of using range specifiers in Editor commands.

Table 5-7: Range Specifications for Editing Commands

Range Keyword and Format	Range Specified
ALL [%]ALL "string" 'string'	All lines containing the specified string. The search does not distinguish between upper- and lowercase letters.
AND BE[GIN] BEGIN E[ND] [%]AND END [,...] "string" , "string" 'string' 'string'	All lines as specified by BEGIN, END, and "string". BEGIN specifies the first line of the dictionary object and END specifies the last line ([EOB]).
BEFORE [%]BEF[ORE]	All lines before the current line and the current line.
BEGIN [%]BE[GIN]	First line of the dictionary object.
END [%]E[ND]	Last line of the dictionary object.
FOR BE[GIN] FOR "string" ; n 'string'	Number of lines specified by n, starting with the the line specified by BEGIN or "string". BEGIN specifies the first line of the dictionary object.
REST [%]R[EST	The current line and all remaining lines in the dictionary object.
string 'string' "string"	The current line or the next line containing the specified string. The search starts with the current line and continues toward the end of the text buffer. The Editor does not distinguish between upper- and lowercase letters.
WHOLE [%]WH[OLE]	All lines of the dictionary object.
+ BE[GIN] "string" + n 'string'	Line that is n lines from the line specified by BEGIN or "string". BEGIN specifies the first line of the dictionary object. When you specify a string, the command applies to the line that is n lines from the next occurrence of the specified string.
• •	The current line.

Note that if you want to display the rest of a dictionary object using the abbreviated form of the REST specifier without a TYPE command, you must precede the R with a percent sign to distinguish the R from the abbreviated form of the REPLACE command.

- Be careful when editing record definitions:
  - If you change the length of the record definition, you have to create a new data file and transfer the old information from the old file to the new one.
  - If you change the name of any fields in a record definition, you have to edit the procedures and reports that refer to those fields.
- DATATRIEVE does not use any new dictionary definitions until you explicitly release control of the domain with the FINISH command. Thus, if you change the definition of a readied domain or its associated record definition, the changes do not have any effect until you enter a FINISH command and ready the domain again.
- See the following sections for more information on DATATRIEVE Editor commands. See Chapter 17 of the *DATATRIEVE-11 User's Guide* for examples of using the DATATRIEVE Editor.

Examples

Edit a table, display the table with the WHOLE range specifier, search for a line with the TYPE command, display that line as the current line, change the line, and exit from the Editor.

```
DTR> EDIT DEPT_TABLE
QED> WHOLE
          CE : "Commercial Engineering",
          PE : "Plant Engineering",
          CS : "Customer Support",
          RD : "Research and Development",
          SD : "Sales Department",
          ELSE "UNKNOWN DEPARTMENT"
          END_TABLE
QED> TYPE "ELSE"
          ELSE "UNKNOWN DEPARTMENT"
QED> S /UNKNOWN DEPARTMENT/Unknown Department/
          ELSE "Unknown Department"
QED> EXIT
DTR>
```

Edit the PERSONNEL_REC record definition, display the definition with the WHOLE range specifier, set the START_DATE line as the current line with the TYPE command, and change the line with the REPLACE command. Use the QUIT command to exit from the Editor without changing the record definition:

```
DTR> EDIT PERSONNEL_REC ADVANCED↵
QED> WHOLE↵
          USING
          01  PERSON.
              05 ID                    PIC IS 9(5).
              05 EMPLOYEE_STATUS       PIC IS X(11)
                                       QUERY_NAME IS STATUS
                                       QUERY_HEADER IS "STATUS"
                                       VALID IF STATUS EQ "TRAINEE","EXPERIENCED".
              05 EMPLOYEE_NAME         QUERY_NAME IS NAME.
                  10 FIRST_NAME            PIC IS X(10)
                                          QUERY_NAME IS F_NAME.
                  10 LAST_NAME             PIC IS X(10)
                                          QUERY_NAME IS L_NAME.
              05 DEPT                  PIC IS XXX.
              05 START_DATE            USAGE IS DATE.
              05 SALARY                PIC IS 9(5)
                                       EDIT_STRING IS $$$,$$$.
              05 SUP_ID                PIC IS 9(5).
          ;
QED> TYPE "DATE"↵
              05 START_DATE            USAGE IS DATE.
QED> REPLACE↵
IN>           05 START_DATE            USAGE IS DATE↵
IN>                                   EDIT_STRING IS MMMBDDBY(4),↵
IN>       ^Z
QED> WHOLE↵
          USING
          01  PERSON.
              05 ID                    PIC IS 9(5).
              05 EMPLOYEE_STATUS       PIC IS X(11)
                                       QUERY_NAME IS STATUS
                                       QUERY_HEADER IS "STATUS"
                                       VALID IF STATUS EQ "TRAINEE","EXPERIENCED".
              05 EMPLOYEE_NAME         QUERY_NAME IS NAME.
                  10 FIRST_NAME            PIC IS X(10)
                                          QUERY_NAME IS F_NAME.
                  10 LAST_NAME             PIC IS X(10)
                                          QUERY_NAME IS L_NAME.
              05 DEPT                  PIC IS XXX.
             05 START_DATE             USAGE IS DATE
                                      EDIT_STRING IS MMMBDDBY(4).
              05 SALARY                PIC IS 9(5)
                                       EDIT_STRING IS $$$,$$$.
              05 SUP_ID                PIC IS 9(5).
          ;
QED> QUIT↵
Execution failed
DTR>
```

5.25.1 DELETE Command

Function

Deletes one or more lines from the dictionary object.

Format

```
D[ELETE] [range]
```

Arguments

range

Specifies the range of lines to be deleted. If you omit the range, DATATRIEVE deletes only the current line.

Restrictions

You can issue the DELETE command only in response to the QED> prompt.

Results

The Editor deletes the current line or the specified range of lines and sets the line pointer to the line following the last line deleted.

Examples

Display the TEST procedure and the current line, then delete the current line. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT TEST↵
QED> WHOLE↵
          READY PERSONNEL READ
          FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEPT
          REPORT CURRENT
          SET REPORT_NAME = "DETAILED SALARY REPORT"
          AT TOP OF DEPT PRINT DEPT
          AT TOP OF STATUS PRINT STATUS
          PRINT ID, FIRST_NAME|1|" "|LAST_NAME ("NAME"), SALARY
          AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL:"
                  TOTAL SALARY USING $,$$$,$$$, SKIP
          AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
                  TOTAL SALARY USING $,$$$,$$$, COL 42, "----------", SK
          AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
                  TOTAL SALARY USING $,$$$,$$$
          SET NO DATE
          SET COLUMNS_PAGE = 80
          END_REPORT
QED> .↵
          READY PERSONNEL READ
QED> DELETE↵
QED> .↵
          FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEPT
QED> QUIT↵
Execution failed
DTR>
```

Delete all the AT TOP and AT BOTTOM statements in the TEST procedure. To do this, use two DELETE commands to delete lines that contain the string "AT" and lines that contain the string "TOTAL". Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT TEST<RET>
QED> D ALL "AT"<RET>
QED> D ALL "TOTAL"<RET>
QED> WHOLE<RET>
          READY PERSONNEL READ
          REPORT CURRENT
          SET REPORT_NAME = "DETAILED SALARY REPORT"
          PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
          SET NO DATE
          SET COLUMNS_PAGE = 80
          END_REPORT
QED> QUIT<RET>
Execution failed
DTR>
```

Delete the last two SET statements in the TEST procedure. Use the QUIT command to exit from the Editor without changing the procedure.

```
DTR> EDIT TEST<RET>
QED> <RET>
          FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEPT
QED> <RET>
          REPORT CURRENT
QED> <RET>
          SET REPORT_NAME = "DETAILED SALARY REPORT"
QED> <RET>
          AT TOP OF DEPT PRINT DEPT
QED> .<RET>
          AT TOP OF DEPT PRINT DEPT
QED> DELETE "SET" FOR 2<RET>
QED> WHOLE<RET>
          READY PERSONNEL READ
          FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEPT
          REPORT CURRENT
          SET REPORT_NAME = "DETAILED SALARY REPORT"
          AT TOP OF DEPT PRINT DEPT
          AT TOP OF STATUS PRINT STATUS
          PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
          AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL:",
                  TOTAL SALARY USING $,$$$,$$$, SKIP
          AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
                  TOTAL SALARY USING $,$$$,$$$, COL 42, "----------", SKIP
          AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
                  TOTAL SALARY USING $,$$$,$$$
          END_REPORT
QED> QUIT<RET>
Execution failed
DTR>
```

Establish the PRINT statement that precedes the first AT BOTTOM statement as the current line, then delete all lines from the beginning of the TEST procedure through the current line:

```
DTR> EDIT TEST↵
QED> ↵
        READY PERSONNEL READ
QED> ↵
        FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEPT
QED> ↵
        REPORT CURRENT
QED> ↵
        SET REPORT_NAME = "DETAILED SALARY REPORT"
QED> ↵
        AT TOP OF DEPT PRINT DEPT
QED> ↵
        AT TOP OF STATUS PRINT STATUS
QED> ↵
        PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
QED> .↵
        PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
QED> DELETE BEFORE↵
QED> WHOLE↵
        AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL:
                TOTAL SALARY USING $,$$$,$$$, SKIP
        AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
                TOTAL SALARY USING $,$$$,$$$, COL 42, "----------", SK
        AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
                TOTAL SALARY USING $,$$$,$$$
        SET NO DATE
        SET COLUMNS_PAGE = 80
        END_REPORT
QED> QUIT↵
Execution failed
DTR>
```

5.25.2 EXIT Command

Function

Ends an editing session, places the edited dictionary object in the data dictionary, and returns you to DATATRIEVE command level. The edited dictionary object replaces the previous version of the object.

Format

```
{ EXIT   }
{ CTRL/Z }
```

Arguments

None

Restrictions

- You must issue the EXIT or CTRL/Z command in response to the QED> prompt to exit from the Editor.
- The Editor treats an EXIT command typed in response to the IN> insert mode prompt as a literal to be inserted in the dictionary object.

Results

- EXIT or CTRL/Z typed in response to the QED> Editor prompt stops an editing session, places the edited dictionary object in the data dictionary, and returns you to DATATRIEVE command level.
- CTRL/Z typed in response to the IN> insert mode prompt ends the insert operation and returns you to Editor command level (indicated by the QED> prompt).

Usage Notes

Changes made with the Editor affect only dictionary objects in the data dictionary. They do not affect dictionary objects such as readied domains and loaded tables in your workspace. If, for example, you change the record definition of a readied domain, you cannot see the effect of this change until you finish the associated domain and ready it again. Similarly, to use an edited table, you must release the loaded version of the table and refer to it again to obtain the edited version.

Examples

Use CTRL/Z to exit from insert mode and the EXIT command to end an editing session:

```
DTR> EDIT A(RET)
QED> .(RET)
        READY YACHTS
QED> DELETE(RET)
QED> INSERT(RET)
IN>     READY YACHTS READ(RET)
IN>     ^Z
QED> EXIT(RET)
DTR> SHOW A(RET)
PROCEDURE A
READY YACHTS READ
FIND A IN YACHTS WITH BUILDER = "GRAMPIAN" AND
         BUILDER = "AMERICAN"
REPORT A SORTED BY BUILDER
PRINT BUILDER, MODEL, PRICE USING $$$,$$$
END-REPORT
END_PROCEDURE
DTR>
```

5.25.3 INSERT Command

Function

Enters insert mode, which allows you to enter text directly into the dictionary object.

Format

```
I[NSERT] [range]
```

Arguments

range

Specifies a range of lines to follow the inserted lines. If you omit the range, DATATRIEVE puts the inserted lines before the current line.

Restrictions

Do not use the following range specifiers in the INSERT command:

- ALL
- AND (,)
- BEFORE
- FOR (;)
- REST
- WHOLE

Results

- The Editor uses the IN> prompt after you enter the INSERT command to indicate that you are in insert mode.
- The Editor inserts the lines you specify before the line specified in the range or before the current line if you omit a range.
- If you do not specify a range, the current line does not change when you exit from insert mode. If you do specify a range, the line specified by the range becomes the current line.

Usage Notes

To exit from insert mode, type CTRL/Z.

Examples

Insert comments in a procedure before the current line. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT A(RET)
QED> INSERT(RET)
IN>       !(RET)
IN>       ! THIS IS A COMMENT(RET)
IN>       !(RET)
IN>      ^Z
QED> WHOLE(RET)
          !
          ! THIS IS A COMMENT
          !
         READY YACHTS
         FIND A IN YACHTS WITH BUILDER = "GRAMPIAN" AND
                 BUILDER = "AMERICAN"
         REPORT A SORTED BY BUILDER
         PRINT BUILDER, MODEL, PRICE USING $$$,$$$
         END-REPORT
QED> QUIT(RET)
DTR>
```

Insert comments after the last line, before the first line, and before the first AT BOTTOM statement of the TEST procedure. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT TEST↵
QED> INSERT BEGIN↵
IN>       !↵
IN>       ! THIS PROCEDURE WRITES A REPORT↵
IN>       !↵
IN>       ^Z
QED> INSERT END↵
IN>       !↵
IN>       ! THIS IS THE END OF THE PROCEDURE↵
IN>       !↵
IN>       ^Z
QED> INSERT "AT BOTTOM"↵
IN>       !↵
IN>       ! THIS SECTION FORMATS THE DETAIL LINES OF THE REPORT↵
IN>       !↵
IN>       ^Z
QED> WHOLE↵
          !
          ! THIS PROCEDURE WRITES A REPORT
          !
          READY PERSONNEL READ
          FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEP
          REPORT CURRENT
          SET REPORT_NAME = "DETAILED SALARY REPORT"
          AT TOP OF DEPT PRINT DEPT
          AT TOP OF STATUS PRINT STATUS
          PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
          !
          ! THIS SECTION FORMATS THE DETAIL LINES OF THE REPORT
          !
          AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL
                TOTAL SALARY USING $,$$$,$$$, SKIP
          AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
                TOTAL SALARY USING $,$$$,$$$, COL 42, "----------", SI
          AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
                TOTAL SALARY USING $,$$$,$$$
          SET NO DATE
          SET COLUMNS_PAGE = 80
          END_REPORT
          !
          ! THIS IS THE END OF THE PROCEDURE
QED> QUIT↵
Execution failed
DTR>
```

Insert lines between the fourth and fifth lines from the current line of the TEST procedure. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT TEST↵
QED> INSERT .+5↵
IN>       !↵
IN>       ! THIS IS BEWEEN THE FOURTH AND FIFTH LINES FROM↵
IN>       ! THE CURRENT LINE↵
IN>       !↵
IN>       ^Z
```

(continued on next page)

```
QED> WHOLE↵
          READY PERSONNEL READ
          FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEPT
          REPORT CURRENT
          SET REPORT_NAME = "DETAILED SALARY REPORT"
          AT TOP OF DEPT PRINT DEPT
          !
          ! THIS IS BEWEEN THE FOURTH AND FIFTH LINES FROM
          ! THE CURRENT LINE
          !
          AT TOP OF STATUS PRINT STATUS
          PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
          AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL:",
                  TOTAL SALARY USING $,$$$,$$$, SKIP
          AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
                  TOTAL SALARY USING $,$$$,$$$, COL 42, "----------", SKIP
          AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
                  TOTAL SALARY USING $,$$$,$$$
          SET NO DATE
          SET COLUMNS_PAGE = 80
          END_REPORT
QED> QUIT↵
Execution failed
DTR>
```

5.25.4 QUIT Command

Function

Returns you to DATATRIEVE command level and leaves the dictionary object unchanged by the editing session.

Format

```
QUIT
```

Arguments

None

Restrictions

You cannot abbreviate the QUIT command.

Results

The QUIT command ends an editing session without making any changes to the dictionary object.

Examples

Use the QUIT command to end an editing session, then show the edited object to confirm that no changes were made:

```
DTR> EDIT A(RET)
QED> WHOLE(RET)
          READY YACHTS
          PRINT FIRST 2 YACHTS
          FIND A IN YACHTS WITH BUILDER = "GRAMPIAN" AND
                  BUILDER = "AMERICAN"
          REPORT A SORTED BY BUILDER
          PRINT BUILDER, MODEL, PRICE USING $$$,$$$
          END-REPORT
QED> INSERT BEGIN(RET)
IN>       !(RET)
IN>       ! THIS IS A COMMENT(RET)
IN>       !(RET)
IN>       ^Z
QED> WHOLE(RET)
          !
          ! THIS IS A COMMENT
          !
          READY YACHTS
          PRINT FIRST 2 YACHTS
          FIND A IN YACHTS WITH BUILDER = "GRAMPIAN" AND
                  BUILDER = "AMERICAN"
          REPORT A SORTED BY BUILDER
          PRINT BUILDER, MODEL, PRICE USING $$$,$$$
          END-REPORT
QED> QUIT(RET)
Execution failed
DTR> SHOW A(RET)
PROCEDURE A
READY YACHTS
PRINT FIRST 2 YACHTS
FIND A IN YACHTS WITH BUILDER = "GRAMPIAN" AND
          BUILDER = "AMERICAN"
REPORT A SORTED BY BUILDER
PRINT BUILDER, MODEL, PRICE USING $$$,$$$
END-REPORT
END_PROCEDURE
DTR>
```

5.25.5 REPLACE Command

Function

Deletes the current line or a specified range of lines in a dictionary object and enters insert mode.

Format

R[EPLACE] [range]

Arguments

range

Specifies a range of lines to be deleted. If you omit the range, DATATRIEVE deletes the current line.

Restrictions

There are no range specification restrictions for the REPLACE command.

Results

When you enter the REPLACE command, the Editor deletes the current line or the lines specified by the range and then prompts with IN> to show that you are in insert mode.

Usage Notes

To exit from insert mode, type CTRL/Z.

Examples

Replace the current line in the TEST procedure with a comment. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT A(RET)
QED> .(RET)
         READY YACHTS
QED> REPLACE(RET)
IN>      READY YACHTS READ(RET)
IN>      ^Z
QED> WHOLE(RET)
         READY YACHTS READ
         PRINT FIRST 2 YACHTS
         FIND A IN YACHTS WITH BUILDER = "GRAMPIAN" AND
                 BUILDER = "AMERICAN"
         REPORT A SORTED BY BUILDER
         PRINT BUILDER, MODEL, PRICE USING $$$,$$$
         END-REPORT
QED> QUIT(RET)
Execution failed
DTR>
```

Delete all lines in a procedure and enter insert mode to create a new procedure. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT A(RET)
QED> WHOLE(RET)
         READY YACHTS
         PRINT FIRST 2 YACHTS
         FIND A IN YACHTS WITH BUILDER = "GRAMPIAN" AND
                 BUILDER = "AMERICAN"
         REPORT A SORTED BY BUILDER
         PRINT BUILDER, MODEL, PRICE USING $$$,$$$
         END-REPORT
```

(continued on next page)

```
QED> REPLACE WHOLE↵
IN>       !↵
IN>       ! THIS IS A TEST↵
IN>       !↵
IN>       ^Z
QED> WHOLE↵
          !
          ! THIS IS A TEST
          !
QED> QUIT↵
Execution failed
DTR>
```

Delete the first line containing the string "PRINT" and replace it with a new line. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT A↵
QED> REPLACE "PRINT"↵
IN>       PRINT FIRST 5 YACHTS↵
IN>       ^Z
QED> WHOLE↵
          READY YACHTS
          PRINT FIRST 5 YACHTS
          FIND A IN YACHTS WITH BUILDER = "GRAMPIAN" AND
                   BUILDER = "AMERICAN"
          REPORT A SORTED BY BUILDER
          PRINT BUILDER, MODEL, PRICE USING $$$,$$$
          END-REPORT
QED> QUIT↵
Execution failed
DTR>
```

5.25.6 SUBSTITUTE Command

Function

Substitutes a character string for all instances of another character string in the current line or in the specified range of lines.

Format

```
S /string-1/[string-2][/[range]]
```

Arguments

string-1

Is the string of characters to be replaced.

string-2

Is the string of characters to replace string 1. If you omit string 2, string 1 is deleted from the specified line.

range

Specifies the range of lines within which the substitution takes place.

/

Is a delimiter. It can be any printing character not in string 1 or string 2. The examples in this manual use the slash (/) as the delimiter.

Restrictions

- The DATATRIEVE Editor recognizes only the first letter (S) of the SUBSTITUTE command. If you type SUBSTITUTE, the Editor displays an error.
- All delimiters in a SUBSTITUTE command must be identical. The Editor interprets the first character after the S command as the delimiter and treats all characters that follow it as part of string 1 until it encounters a character identical to the first character.
- If you do not specify a range, you need to specify only the first two delimiters.
- If you specify a range, you must use all three delimiters as shown in the format.
- Do not use END to specify a range for the SUBSTITUTE command.

Results

- The Editor searches for string 1 from the current line towards the end of the text buffer except when the WHOLE range is specified. When you specify WHOLE, the Editor begins the search at the beginning of the text buffer rather than at the current line.
- If you specify a range, the Editor replaces all occurrences of string 1 in that range with string 2.
- If you do not specify a range, the Editor replaces only the first occurrence of string 1. The first occurrence need not be in the current line.
- The Editor displays all lines in which a substitution occurs. If no substitution takes place, no lines are displayed.
- When a substitution takes place, the current line is set to the last line in which a substitution occurred. If no substitution takes place, the current line remains unchanged.

Examples

Substitute the string YAHCT for the string YACHT in the current line of a procedure. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT A(RET)
QED> .(RET)
          READY YACHTS
QED> S /YACHT/YAHCT(RET)
          READY YAHCTS
QED> QUIT(RET)
Execution failed
DTR>
```

Substitute the string YAHCT for all occurrences of YACHT in a procedure. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT A(RET)
QED> S *YACHT*YAHCT* WH(RET)
          READY YAHCTS
          PRINT FIRST 2 YAHCTS
          FIND A IN YAHCTS WITH BUILDER = "GRAMPIAN" AND
QED> QUIT(RET)
Execution failed
DTR>
```

Substitute the string YAHCT for the occurrences of YACHT from the current line to the end of the procedure. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT A(RET)
QED> WHOLE(RET)
          READY YACHTS
          PRINT FIRST 2 YACHTS
          FIND A IN YACHTS WITH BUILDER = "GRAMPIAN" AND
                  BUILDER = "AMERICAN"
          REPORT A SORTED BY BUILDER
          PRINT BUILDER, MODEL, PRICE USING $$$,$$$
          END-REPORT
QED> .(RET)
          READY YACHTS
QED> (RET)
          PRINT FIRST 2 YACHTS
QED> .(RET)
          PRINT FIRST 2 YACHTS
QED> S /YACHT/YAHCT/ REST(RET)
          PRINT FIRST 2 YAHCTS
          FIND A IN YAHCTS WITH BUILDER = "GRAMPIAN" AND
QED> QUIT(RET)
Execution failed
DTR>
```

5.25.7 TYPE Command

Function

Displays the line following the current line or a specified range of lines.

Format

```
[T[YPE]] [range]
```

Arguments

range

Specifies the range of lines to be displayed.

Restrictions

None

Results

- If you omit a range, the Editor displays the first line following the current line. If you specify a range, the Editor displays the range of lines.
- The first line displayed becomes the current line, with one exception: if the range contains END, the line pointer points to the end-of-text buffer ([EOB]).

Usage Notes

- Both the command name and range are optional. Entering RETURN is the same as typing T and pressing RETURN. Typing WHOLE and pressing RETURN is the same as typing T WHOLE and pressing RETURN.
- You can use the TYPE command to search for text strings in dictionary objects by enclosing the string to search for in pairs of single or double quotation marks.

Examples

Edit the TEST procedure and use the TYPE command to display the current line, the line following the current line, the first and last lines of the procedure, and the entire procedure. The TYPE command does not change text, so use the EXIT command to exit from the Editor:

```
DTR> EDIT TEST(RET)
QED> T(RET)
        FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEPT
QED> (RET)
        REPORT CURRENT
```

(continued on next page)

```
QED> T WH(RET)
        READY PERSONNEL READ
        FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEPT
        REPORT CURRENT
        SET REPORT_NAME = "DETAILED SALARY REPORT"
        AT TOP OF DEPT PRINT DEPT
        AT TOP OF STATUS PRINT STATUS
        PRINT ID, FIRST_NAME|1" "|LAST_NAME ("NAME"), SALARY
        AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL:",
                TOTAL SALARY USING $,$$$,$$$, SKIP
        AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
                TOTAL SALARY USING $,$$$,$$$, COL 42, "----------", SKIP
        AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
                TOTAL SALARY USING $,$$$,$$$
        SET NO DATE
        SET COLUMNS_PAGE = 80
        END_REPORT
QED> WHOLE(RET)
        READY PERSONNEL READ
        FIND PERSONNEL WITH DEPT = "D98", "T32" SORTED BY STATUS, DEPT
        REPORT CURRENT
        SET REPORT_NAME = "DETAILED SALARY REPORT"
        AT TOP OF DEPT PRINT DEPT
        AT TOP OF STATUS PRINT STATUS
        PRINT ID, FIRST_NAME|1" "|LAST_NAME ("NAME"), SALARY
        AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL:",
                TOTAL SALARY USING $,$$$,$$$, SKIP
        AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
                TOTAL SALARY USING $,$$$,$$$, COL 42, "----------", SKIP
        AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
                TOTAL SALARY USING $,$$$,$$$
        SET NO DATE
        SET COLUMNS_PAGE = 80
        END_REPORT
QED> T BEGIN, END(RET)
        READY PERSONNEL READ
        [EOB]
QED> EXIT(RET)
DTR>
```

Edit the TEST procedure and set the current line to the first AT BOTTOM statement. Then use the TYPE command to display all lines from the current line to the end of the procedure:

```
DTR> EDIT TEST(RET)
QED> T "AT BOTTOM"(RET)
        AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL:",
QED> .(RET)
        AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL:",
QED> T REST(RET)
        AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE, "TOTAL:",
                TOTAL SALARY USING $,$$$,$$$, SKIP
        AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
                TOTAL SALARY USING $,$$$,$$$, COL 42, "----------", SKIP
        AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
                TOTAL SALARY USING $,$$$,$$$
        SET NO DATE
        SET COLUMNS_PAGE = 80
        END_REPORT
QED> ^Z
DTR>
```

5.26 EDIT_STRING Clause

Function

Specifies the output format for an elementary field value.

Format

```
EDIT_STRING [IS] edit-string
```

Arguments

edit-string

Is one or more edit string characters that define the output format for the field value.

Restrictions

- This clause is valid only for elementary fields.
- You must immediately precede the R edit string character with the C edit string character. Do not use R alone as an edit string character.
- Do not specify one or more D edit string characters for a date field without specifying another valid date edit string character. DATATRIEVE interprets a D without another date edit string character as the DB edit string and interprets the field as numeric.

Result

DATATRIEVE uses the specified edit string as the default format when writing a field value to a file or output device.

Usage Notes

- If you do not include an EDIT_STRING clause in a field definition, DATATRIEVE uses the PICTURE clause as the default output format. However, DATATRIEVE does not display a sign specified in a PICTURE clause unless you specify those characters in an edit string.
- To override the default output format specified by either a PICTURE or EDIT_STRING clause, specify an edit string in the PRINT statement.
- The edit string specifies the format of a field value and can consist of one or more edit characters. In general, each edit character corresponds to one character position in the printed output. For example, 999999 specifies that the output will be six digits in six character positions. Do not use spaces in the edit string.

- When you need several identical edit characters in an edit string, shorten the edit string by placing a repeat count in parentheses following the edit character. The edit string 9(6), for example, is the same as 999999.
- Table 5-8 contains a list of edit string characters grouped by character type and function. Edit string characters function as replacement characters, insertion characters, or, for numeric fields, floating characters:
  - Replacement characters are place holders and are replaced by characters from the field's content.
  - Insertion characters are inserted in the output of the field's content.
  - Floating characters replace all but the last leading zero with spaces. The last leading zero is then replaced by the specified floating character.

The characters you can use in an edit string depend on the class of the field (alphanumeric, numeric, or date).

Table 5-8: Edit String Characters

Character Type	Edit String Character	Description
Alphanumeric Replacement	X	Each X is replaced by one character from the field's content.
	T	Indicates text. The number of Ts is the length of a line. DATATRIEVE prints the entire field using lines of the length indicated. Words are not broken; if a word is longer than the line length specified, it is printed on a line of its own. Edit strings containing a T cannot contain other characters.
Numeric Replacement	9	Each 9 is replaced by one digit from the field's content. Nondigit characters are ignored, digits are right justified in the output, and leading digit positions (if any) are filled with zeros.
	Z	If a Z matches a leading zero in the field's content, it is replaced by a space. If not, Z is replaced by a digit from the field's content.
	* (asterisk)	If an asterisk (*) matches a leading zero in the field's content, an asterisk is placed in that character position. If not, it is replaced by a digit from the field's content.
Alphanumeric Insertion	+ (plus)	If only one plus sign is specified for an alphanumeric field, it is inserted in that position.
	- (hyphen)	A hyphen is inserted in that character position.
	. (period)	A period is inserted in that character position.
	, (comma)	A comma is inserted in that character position.

(continued on next page)

Table 5-8: Edit String Characters (Cont.)

Character Type	Edit String Character	Description
Numeric Insertion	+ (plus)	If only one plus sign is specified, it is replaced by a plus sign if the field's content is positive or a minus sign if it is negative.
	– (minus)	If only one minus sign is specified, it is replaced by a blank if the field's content is positive or a minus sign if it is negative.
	. (decimal)	A decimal point is inserted in that character position. Put only one decimal point in a numeric edit string.
	, (comma)	If all the digits to the left of the comma are suppressed zeros, the comma is replaced by a blank. If not, a comma is inserted in that character position.
	CR	If the field's content is negative, the letters CR are inserted. If the field's content is positive, CR is replaced by 2 blanks. Put only one CR in an edit string, either at the far right or the far left.
	DB	If the field's content is negative, the letters DB are inserted. If the field's content is positive, DB is replaced by two blanks. Put only one DB in an edit string, either at the far right or the far left.
Alphanumeric and Numeric Insertion	B	A space is inserted in that character position.
	0 (zero)	A zero is placed in that character position.
	$	If only one dollar sign is specified, it is printed in that character position.
	%	A percent sign is inserted in that character position.
	/ (slash)	A slash is inserted in that character position.
Numeric Floating Insertion	$	If more than one dollar sign is specified to the left of the other edit string characters, leading zeros are suppressed, and one dollar sign is displayed to the immediate left of the leftmost digit.
	+ (plus)	If more than one plus sign is specified to the left of the other edit string characters, any leading zeros are suppressed, the sign of the field's value (plus or minus) is displayed to the immediate left of the leftmost character position determined by the other edit string characters.
	– (minus)	If more than one minus sign is specified to the left of the other edit string characters, any leading zeros that the minus sign matches are suppressed. If the value of the field is negative, a minus sign is displayed to the immediate left of the leftmost character position determined by the other edit string characters.

(continued on next page)

Table 5-8: Edit String Characters (Cont.)

Character Type	Edit String Character	Description
Date Replacement	D	Each D is replaced by the corresponding digit of the day of the month. Put no more than two Ds in a date edit string; the use of DD is recommended. Days of the month between 1 and 9 are displayed with leading zeros.
	M	Each M is replaced by the corresponding letter of the name of the month. An edit string of M(9) prints the entire name of the month.
	N	Each N is replaced by a digit of the number of the month. Put no more than two Ns in a date edit string; the use of NN is recommended.
	Y	Each Y is replaced by the corresponding digit of the numeric year. Put no more than four Ys in a date edit string; the use of YY or YYYY is recommended.
	J	Each J is replaced by the corresponding digit of the Julian date. Put no more that three Js in a date edit string; the use of JJJ is recommended.
	W	Each W is replaced by the corresponding letter from the name of the day of the week. An edit string of W(9) prints the entire day. Put no more than 9 Ws in a date edit string.
	B	Each B is replaced by a space in that character position.
	/ (slash)	A slash is inserted in that character position.
	- (hyphen)	A hyphen is inserted in that character position.
	. (period)	A period is inserted in that character position.

5.26.1 Formatting Alphanumeric Fields

Usage Notes for Replacement Characters

- The character X specifies the number of characters to be printed. Each X is replaced by one character from the field's content, in left-to-right order. If the field value has more characters than the edit string has Xs, only the leftmost characters are printed. If it has fewer characters than its edit string, DATATRIEVE pads the output with spaces on the right.
- If an alphanumeric field contains only digits and you use the field in any arithmetic computations, you should define it as a numeric field. You can perform computations with alphanumeric fields that contain only digits; but, because alphanumeric fields are padded with spaces on the right, the results may not be valid. Edit string characters for numeric fields are explained in Section 5.27.2.

- The character T allows you to print alphanumeric field values on one or more lines. The primary use of T is to print fields containing large amounts of text. The number of Ts in the edit string specifies the maximum number of characters to be printed on one line. The edit string T(20), for example, indicates that a line of output will contain no more than 20 characters (unless a single word contains more than 20 characters). Place the T edit string at the end of the command to avoid nonstaged output.
- If a field contains more characters than specified in a T edit string, DATATRIEVE prints as many full words on the line as possible. (A word, in this sense, is a string of characters delimited by a space.) DATATRIEVE then prints the remaining characters on the next line and following lines, if necessary. DATATRIEVE does not print out trailing spaces when you use a T edit string.
- When you specify a T edit string, DATATRIEVE prints only full words. It does not divide words. If a word is longer than the T edit string, DATATRIEVE prints the word in full on that line.

Usage Notes for Insertion Characters

- With insertion characters, you can insert a space, a zero, a dollar sign, a percent sign, a slash, a plus sign, or a hyphen in the output of a field value. DATATRIEVE inserts the specified characters or spaces in the positions indicated in the edit string.
- The B (space) and slash characters are also valid for numeric edit strings, but cannot be used in T edit strings.
- The hyphen is only valid in an X edit string.

Examples

Use a T(20) edit string to display the value of the EVALUATION field from the first three records in the EMP_REVIEW domain:

```
DTR> SET NO PROMPT↵
DTR> READY EMP_REVIEW↵
DTR> SHOW EMP_REV_REC↵
RECORD EMP_REV_REC
 USING
01 EMP_REC.
   03 BADGE      PIC 9(5).
   03 NAME       PIC X(10).
   03 JOB                PIC X(15).
   03 EVALUATION PIC X(60).
   03 EVAL_DATE  USAGE IS DATE
                EDIT_STRING IS DD-MMM-YY.
;
DTR> FOR EMP_REVIEW↵
CON> PRINT NAME, EVAL_DATE, EVALUATION USING T(20), SKIP↵
```

(continued on next page)

```
                EVAL
   NAME         DATE          EVALUATION

BRAD H.      29-Apr-83 Brad did a fine job
                       in implementing
                       staged output.

TERRY C.     21-Mar-83 Terry is an
                       exceptional system
                       manager and
                       developer.

DAVID D.     12-Mar-83 David is a master of
                       developing report
                       specifications.


DTR>
```

Print the values of the alphanumeric field ALPHA in the EDIT_TEST domain without an edit string and with a variety of edit strings:

```
DTR> READY EDIT_TEST↵
DTR> SHOW EDIT_REC↵
RECORD EDIT_REC
 USING
01 TEST.
         03 ALPHA        PIC X(10).
;
DTR> FIND A IN EDIT_TEST↵
[2 records found]
DTR> FOR A PRINT ALPHA↵

  ALPHA

CHALLENGER
123

DTR> FOR A PRINT ALPHA USING X(10)↵

  ALPHA

CHALLENGER
123

DTR> FOR A PRINT ALPHA USING X(3)↵

ALPHA

 CHA
 123

DTR> FOR A PRINT ALPHA USING XX/X(8)↵

   ALPHA

CH/ALLENGER
12/3
```

(continued on next page)

```
DTR> FOR A PRINT ALPHA USING X(5)/X(5)RET

   ALPHA

CHALL/ENGER
123   /

DTR> FOR A PRINT ALPHA USING X(5)-XXRET

 ALPHA

CHALL-EN
123   -

DTR> FOR A PRINT ALPHA USING XX+XXRET

ALPHA

CH+AL
12+3

DTR>
```

5.26.2 Formatting Numeric Fields

By using EDIT_STRING clauses in record definitions, you can format numeric field values in a way that is meaningful and easy to read. With edit strings you can suppress leading zeros and print dollar signs, percent signs, commas, decimal points, and plus or minus signs. For example, in the YACHT record, the EDIT_STRING clause for the PRICE field ($$$,$$$) causes DATATRIEVE to print a PRICE field value of 09870 as $9,870.

DATATRIEVE provides three types of edit characters for numeric fields: replacement characters, insertion characters, and floating characters.

Usage Notes for Replacement Characters

- The 9 replacement character specifies how many digits from the field value are to be displayed. Each 9 in the edit string is replaced by one digit from the field value. DATATRIEVE ignores nondigit characters, right-justifies the output, and fills any leading character positions that do not contain digits with zeros.
- The Z replacement character also specifies how many digits from the field value are to be displayed. Each Z in the edit string is replaced by one digit from the field value. Leading zeroes in the field value are replaced with spaces.
- The asterisk replacement character specifies how many asterisks or digits from the field value are to be displayed.

- A leading zero is any zero in a field value that has only zeros to the left of it. If there is an implied decimal place to the left of a zero, then it is not a leading zero. If a Z corresponds to a leading zero in a field, a space is printed instead of a zero. If an asterisk corresponds to a leading zero in a field, then an asterisk is printed instead of a zero.
- The following table illustrates the differences between numeric replacement characters. A number sign (#) represents a space:

PICTURE String	Edit String	Output for Field Values	
		04092	00055
9(5)	None	04092	00055
	9(5)	04092	00055
	Z(5)	#4092	###55
	*(5)	*4092	***55
		0100	**0010**
99V99	None	0100	0010
	99.99	01.00	00.10
	ZZ.ZZ	#1.00	##.10
	** **	*1.00	**.10

- Before printing a field, DATATRIEVE computes the number of digits to the left of the decimal point. If there is no V in the picture string, all digits are to the left of the decimal. If there are more digits to the left of the decimal than the edit string specifies with either replacement characters or floating characters, DATATRIEVE prints an asterisk in each character position specified by the edit string, indicating that the edit string is not long enough for the field value. For example, a PICTURE clause could specify four digits for a field, and only two digits for an edit string:

```
03 MODEL_NUMBER
   PICTURE IS 9999
   EDIT_STRING IS 99.
```

  If the field value is 1234, DATATRIEVE prints two asterisks (**) for the field value. Therefore, be careful to specify edit strings that are long enough for numeric fields.

- If the field value has fewer digits than the edit string specifies, DATATRIEVE pads the output with leading zeros. You can suppress leading zeros by using the Z replacement character or floating characters. If there are more leading zeros than Zs or floating characters, the remaining zeros are displayed at the left of the field value, but to the right of any floating character.

- If you define a numeric field with the clauses PIC 9(4) and USAGE IS COMP, then the field can contain numbers as high as 32,768. To print the field when it contains a five digit number, you must use an edit string that specifies five digits.

Usage Notes for Insertion Characters

- With insertion characters, you can print the signs of fields, decimal points, dollar signs, DB or CR for negative values, percent signs, commas, zeros, slashes, and character string literals.
- A PRINT statement does not display the sign of a numeric field unless you specify an EDIT_STRING clause in the field definition or an edit string in the PRINT statement:
  - To print a sign (+ or −) in a field value (indicated by an S in the field's PICTURE clause), specify a + or − in the edit string for that field. The sign must be the first or last character in the edit string.
  - If you specify only one sign in the edit string, DATATRIEVE prints the sign in the position you indicate. If you specify more than one sign in the leftmost part of the edit string, the sign is a floating character.
  - You can also use the edit characters DB and CR to indicate the sign of a field value. You can only use one DB or CR in an edit string, and it must be the leftmost or rightmost element of the edit string.

The following table shows the use of the edit characters +, −, DB, and CR. A number sign (#) represents a space:

PICTURE String	Edit String	Output for Field Values	
		−1234	+4321
S9(4)	None	1234−	4321
	−9999	−1234	#4321
	9999−	1234−	4321
	9999+	1234−	4321+
	+9999	−1234	+4321
	9999DB	1234DB	4321
	CR9999	CR1234	##4321

- For a field defined with a V in its PICTURE clause, you must use an edit string containing a decimal point (.) to print a value that has a decimal point in it. DATATRIEVE matches the decimal point in the edit string with the implied decimal place in the field.
  - If the edit string contains fewer digits to the right of the decimal than the field contains, the extra digits are not printed.
  - If the edit string contains fewer digits to the left of the decimal than the field contains, DATATRIEVE prints asterisks. It is best to place the period in the edit string in the same position as the V in the picture string.

  The following table illustrates the use of the decimal point in formatting numeric fields. A number sign (#) represents a space.

PICTURE String	Edit String	Output for Field Values	
		0321	1234
99V99	None	0321	1234
	Z9.99	#3.21	12.34
	999.9	003.2	012.3
	9.999	3.210	* ***
	Z(4)	###3	##12

- If the last character of the edit string is a period not followed by other input on the same line, DATATRIEVE treats the period as the termination of the field definition and not as part of the edit string. If the EDIT_STRING clause is the last part of the field definition, specify two periods: the first period is part of the edit string and the second period ends the field definition. If the EDIT STRING clause is not the last part of the field definition, place the next clause on the same line or place a hyphen at the end of the line.
- To print a comma, slash, percent sign, dollar sign, or zero, specify that character in the edit string. If there are only spaces to the left of a comma, DATATRIEVE prints a space instead of a comma. If you include more than one dollar sign, it is a floating character. The following table shows how these characters format numeric field values. A number sign (#) represents a space:

PICTURE String	Edit String	Output for Field Values	
		123456	000040
9(6)	$999,999	$123,456	$000,040
	$$$$,$$$	$123,456	#####$40
	ZZZ,ZZZ	123,456	#####40
	999/999	123/456	000/040
	99%	**%	40%

Usage Notes for Floating Characters

- You can specify three edit string characters ($, −, and +) as floating characters. A floating character replaces all but the last leading zero with spaces. The floating character ($, −, or +) replaces the last leading zero. Floating characters that correspond to digits are replaced by those digits. Since one character in the field is replaced by a floating character, specify one more character in the edit string than the total number of digits in the field.
- To use a floating character, you must specify two or more of the same character. You can specify only one floating character in an edit string. For example, you cannot have both a floating minus sign and a floating dollar sign in the same edit string. The floating character must be the leftmost character in an edit string.
- The following table shows how floating characters format field values. A number sign (#) represents a space:

PICTURE String	Edit String	Output for Field Values	
		−1234	+0021
S9(4)	++++9	−1234	##+21
	−−−−9	−1234	###21
		0015786	**0500001**
9(5)V99	$9(5).99	$00157.86	$05000.01
	$$$,$$$.99	###$157.86	#$5,000.01
	$$$,$$$.00	###$157.00	#$5,000.00

Usage Notes for Date Fields

- A field defined as USAGE IS DATE is stored internally as a binary value. To print this field, DATATRIEVE must convert it to another format. If you do not include an EDIT_STRING clause in the field definition for a date field, DATATRIEVE prints the field value in the following format:

  DD-MMM-YYYY

- To print the date in any other format, you must specify an EDIT_STRING clause in the field's definition or use an edit string in your output statement. By using an edit string for a date field, you can print the date in a different format. For example, you can have DATATRIEVE print January 1, 1980 as:

  1 Jan 80

  1980/001

  Tuesday /January 01

  Jan 01 80

  1/01/80

  1-Jan-1980 Tue

- With date edit strings you can have the date include:
  - The name of the day of the week (such as Monday, Tuesday) or just the first characters of the name (such as Mon, Tue)
  - The day of the month (such as 1, 2, 3)
  - The name of the month (such as January, February) or just the first characters of the name (such as Jan, Feb)
  - The number of the month (for example, 1 for January and 12 for December)
  - The year (for example, 1980) or just the last two digits of the year (80)
  - The Julian date (for example, 001 for January 1 and 366 for December 31 in a leap year)
  - Delimiters to separate the parts of the date (such as a slash or period after the month and day)
- In the date edit string, you specify the characters to be printed (letters, digits, spaces, slashes, hyphens, or periods) and the order in which the parts of the date are to appear (such as month, day, year). Table 5-7 shows the edit string characters you can use to specify the format of date values.
- DATATRIEVE always outputs the total number of characters you specify with an alphabetic edit string such as M(9) or W(9). If the content of the field is shorter than its edit string specifies, DATATRIEVE pads the output with blanks. If, for example, the edit string specifies a nine-letter month with M(9) and the field contains a month with a name shorter than nine letters (such as June), DATATRIEVE prints the four-character month name and then five spaces.

- The following table illustrates the use of date edit strings for two field values: June 4, 1980 and November 15, 1983. A number sign (#) represents a space:

Edit String	Output for Field Values	
	June 4, 1980	November 15, 1985
DD-MMM-YY	#4-Jun-80	15-Nov-85
MMMBDDBY(4)	Jun##4#1980	Nov#15#1985
M(9)BDDBY(4)	June#######4#1980	November##15#1985
NN/DD/YY	#6/04/80	11/15/85
W(9)	Wednesday	Friday
YYYY/JJJ	1980/156	1985/319
DDBMMMBYY/WWW	#4#Jun#80/Wed	15#Nov#85/Fri
DD.NN.YY	#4.06.80	15.11.85

5.27 ERASE Statement

Function

Permanently removes one or more data records from an indexed sequential or relative data file.

Format

```
ERASE [ALL [OF rse]]
```

Arguments

ALL

Causes DATATRIEVE to permanently remove from the data file every record in the current collection.

ALL OF rse

Causes DATATRIEVE to permanently remove from the data file every record identified by the record selection expression.

Restrictions

- The domain containing the targeted record or records must be readied for WRITE access (see the READY Command, Section 5.44).
- The data file containing the targeted record or records must be an indexed sequential file or a relative file. You cannot delete records from a sequential file.
- You cannot delete a record from a view domain or a port.
- You cannot use the ERASE statement to change or remove fields from a list in a hierarchical record.

Results

- DATATRIEVE permanently deletes the targeted record or records from the data file:
  - If you use the argument ALL, DATATRIEVE deletes from the data file every record in the current collection.
  - If you use the argument ALL OF rse, DATATRIEVE permanently deletes from the data file every record identified by the record selection expression.
  - If you put the keyword ERASE by itself in a FOR statement, DATATRIEVE permanently deletes from the data file each record specified by the FOR statement.

- If you erase a collection or record stream that contains a selected record, the field values of the selected record are still available in your workspace, even though the record has been removed from the data file by the ERASE statement. You can display the fields of that selected record, and you can use those field values in value expressions. The values remain in your workspace until you change the single record context with another SELECT statement or with a DROP statement, or until you release control with a RELEASE or FINISH command.
- The ERASE statement permanently removes the selected record from the data file if you do not establish a target record stream with a FOR statement, with the OF rse clause, or with the keyword ALL. DATATRIEVE displays the message "No context for ERASE" if you have no selected record in any collection.

Usage Note

Before using the ERASE statement, check the current collection and selected record with the SHOW CURRENT command or with a PRINT statement.

Examples

Erase all the yachts built by Albin:

```
DTR> READY YACHTS
DTR> FIND YACHTS WITH BUILDER EQ "ALBIN"
[3 records found]
DTR> ERASE ALL
DTR> PRINT ALL
DTR>
```

Define a procedure to erase selected yachts:

```
DTR> DEFINE PROCEDURE SELL-BOAT
DFN> READY YACHTS WRITE
DFN> FIND YACHTS WITH BUILDER EQ *.BUILDER AND MODEL EQ *.MODEL
DFN> PRINT ALL
DFN> IF *."Y IF BOAT SOLD" EQ "Y"
DFN>    THEN ERASE ALL ELSE
DFN>    PRINT "Sell it now!"
DFN> END_PROCEDURE
DTR> :SELL-BOAT
Enter BUILDER: AMERICAN
Enter MODEL: 26

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL   WEIGHT BEAM  PRICE

 AMERICAN       26        SLOOP  26     4,000  08   $9,895

Enter Y IF BOAT SOLD: N
Sell it now!

DTR>
```

5.28 EXIT Command

Function

Stops a DATATRIEVE session.

Format

```
{ EXIT    }
{ CTRL/Z  }
```

Arguments

None.

Restrictions

You must issue the EXIT or CTRL/Z command in response to the DTR> prompt to stop your DATATRIEVE session.

Results

- EXIT stops a DATATRIEVE session and returns you to system command level.
- When you stop your DATATRIEVE session by typing either EXIT or CTRL/Z, DATATRIEVE automatically finishes all readied domains and releases all collections, global variables, and tables.
- CTRL/Z does not stop your DATATRIEVE session when you are in ADT, Guide Mode, or the DATATRIEVE Editor.
- CTRL/Z also does not stop your DATATRIEVE session when you enter it in response to an Enter prompt during the execution of a STORE or MODIFY statement. Entering a CTRL/Z to an Enter prompt returns you to DATATRIEVE command level (indicated by the DTR> prompt).
- Entering CTRL/Z in response to the DFN>, CON>, or RW> prompts returns you to DATATRIEVE command level. Typing EXIT in response to these prompts causes a syntax error.

Example

End a DATATRIEVE session.

```
DTR> EXIT(RET)
```

5.29 EXTRACT Command

Function

Copies the dictionary definition of one or more dictionary objects from your current dictionary to a command file.

Format

```
EXTRACT [ON] file-spec object-name [,...]
```

Arguments

file-spec

Is the specification of the RMS file to contain the definition or definitions in the following format:

For RSTS/E systems: dev:[PPN]file-name.type

For RSX systems: dev:[UIC]file-name.type;ver

You must specify at least one field in the file specification. On RSTS/E systems, you must specify a file name. DATATRIEVE uses the following defaults for fields not specified:

Field	Default
dev:	SY: (the system device)
[UIC/PPN]	Your default UIC/PPN
file-name	No default
.type	.CMD
;ver	1 or next higher version (non-RSTS/E systems only)

object-name

Is the name of the dictionary object you want to extract.

Restrictions

- To extract the definition of a dictionary object from the dictionary, you must have read access privilege to the domain, record, procedure, or table.
- An ON file-spec clause must precede the list of dictionary object names.

Results

- DATATRIEVE creates a command file with the file specification you provide. DATATRIEVE enters into the command file a DELETE command and a DEFINE command for each dictionary object specified in the EXTRACT command. The DEFINE command contains a copy of the object specified by the dictionary name.
- The EXTRACT command copies the text of the definition exactly as it was entered in the dictionary by DATATRIEVE. For example, when you extract a domain definition, the specification for the data file is extracted just as it was originally entered. If only the file name (and not the device, directory, type, or version number) was stored in the dictionary, the EXTRACT command copies only the file name into the new DEFINE command.
- When the EXTRACT command copies the definition into the command file, DATATRIEVE does not delete the definition from the dictionary.
- When you invoke the command file, DELETE command in that file causes DATATRIEVE to delete from the dictionary the definition of any dictionary object with the same name as that of the extracted dictionary object.

Usage Notes

- The EXTRACT command enables you to create backup copies of your dictionary objects and to transport data definitions to other dictionaries.
- To invoke the command file, type @ and the command file name in response to the DTR> prompt. If the file type is not .CMD, you must include the file type in the file specification.
- When you are moving a definition from one dictionary to another, use a SHOW object-name command to see if an object with the same name as that in the DELETE and DEFINE commands exists in the new dictionary.

  If the object exists, you must decide whether or not you want it deleted when you invoke the command file. If you do not want that object deleted, you must edit the command file to change the names in the DELETE and DEFINE commands.

  Be sure to anticipate the problems that might arise when you change current dictionaries. No two objects in the same dictionary can have the same name. If you shift to a new current dictionary, however, that new one may legitimately contain an object with the same name.

  For example, you extract the definition of procedure ABC from the dictionary WIDGETS.DIC. You then set your current dictionary to THINGS.DIC, which contains a dictionary table called ABC. If you invoke the command file, the DELETE ABC command in the command file deletes the table definition from THINGS.DIC, and the DEFINE PROCEDURE ABC command enters the procedure definition.

- To invoke a command file produced by the EXTRACT command, you must have system read access privilege to the file and you must have DATATRIEVE C (control) access to any dictionary objects to be deleted from the command file.
- The EXTRACT command does not copy the ACL of the dictionary object. After you invoke the command file, the ACL for the dictionary object is the default ACL assigned by the dictionary. Use the DEFINEP command to add other entries to the ACL.
- To extract all objects in a dictionary, use the QXTR utility provided with your DATATRIEVE-11 installation kit. Run it at system command level by typing RUN $QXTR, then answer the on-line questions. The QXTR utility:
  - Provides a backup of the whole dictionary
  - Creates DEFINEP commands to reproduce the ACL of each object
  - Gives you the option of extracting an object in a form suitable for using with VAX-11 DATATRIEVE

  See Chapter 21 in the *DATATRIEVE-11 User's Guide* for information on using QXTR.

Example

Extract the definitions of the domain YACHTS and the record YACHT from the dictionary OLDDIC.DIC. Change the current dictionary to a new dictionary, NEWDIC.DIC, and use the SHOW command to check for objects with the names YACHTS and YACHT in NEWDIC. Then invoke the command file created with the EXTRACT command to copy the YACHTS and YACHT definitions to NEWDIC:

```
DTR> EXTRACT ON MOVING YACHTS, YACHT(RET)
DTR> SET DICTIONARY NEWDIC(RET)
DTR> SHOW DICTIONARY(RET)
The current dictionary is SY:[150,150]NEWDIC.DIC
DTR> SHOW DOMAINS, RECORDS(RET)
Domains:
Records:
```

(continued on next page)

```
DTR> @MOVING (RET)
DELETE YACHTS;
"YACHTS" has not been defined in the dictionary
DEFINE DOMAIN YACHTS
 USING YACHT ON YACHT.DAT;
DELETE YACHT;
"YACHT" has not been defined in the dictionary
DEFINE RECORD YACHT
USING
ALLOCATION IS LEFT_RIGHT
01 BOAT.
  03 TYPE.
     06 MANUFACTURER PIC X(10)
        QUERY_NAME IS BUILDER.
     06 MODEL PIC X(10).
  03 SPECIFICATIONS
     QUERY_NAME SPECS.
     06 RIG PIC X(6)
        VALID IF RIG EQ "SLOOP","KETCH","MS","YAWL".
     06 LENGTH_OVER_ALL PIC XXX
        VALID IF LOA BETWEEN 15 AND 50
        QUERY_NAME IS LOA.
     06 DISPLACEMENT PIC 99999
        QUERY_HEADER IS "WEIGHT"
        EDIT_STRING IS ZZ,ZZ9
        QUERY_NAME IS DISP.
     06 BEAM PIC 99.
     06 PRICE PIC 99999
        VALID IF PRICE>DISP*1.3 OR PRICE EQ 0
        EDIT_STRING IS $$$,$$$.
;
[Record YACHT is 41 bytes long]
DTR> SHOW DOMAINS, RECORDS(RET)
Domains:
        YACHTS
Records:
        YACHT
DTR>
```

5.30 FIND Statement

Function

Establishes a collection of records from a domain, view, collection, or list. The collection formed with the FIND statement becomes the current collection.

Format

```
FIND rse
```

Arguments

rse

Is a record selection expression specifying the records to be included in the collection.

Restrictions

- The domain or view containing the records specified in the RSE must be readied for READ, WRITE, or MODIFY access. The source domain cannot be readied for EXTEND access.
- The source domain cannot be a port.
- Use the FIND statement only at DATATRIEVE command level, indicated by the DTR> prompt. Do not use it in a BEGIN-END block, a FOR statement, a REPEAT statement, or a THEN statement.

Results

- DATATRIEVE forms a current collection consisting of the records specified in the RSE. If you specify a context variable in the RSE, the collection has two names: CURRENT and that of the context variable.
- When the collection has been formed, DATATRIEVE displays the message "[n records found]", where n is the number of records in the collection:
  - If the FIND statement is inside a procedure, DATATRIEVE does not display this message unless the FIND statement is the last statement in the procedure.
  - If you put the FIND statement on the same input line with other DATATRIEVE statements or commands, using semicolons to separate them, DATATRIEVE does not display this message unless the FIND statement is the last one on the line.
- DATATRIEVE collects the records in the order it finds them in the data file. There is no order to the collection unless you include the SORTED BY clause in the record selection expression, or unless the domain uses an indexed file.

Usage Note

To establish single record context in compound statements, use the RSE clauses of FOR, PRINT, MODIFY, WHILE, and ERASE. These RSEs can establish the record streams needed to control the name recognition and single record context inside compound statements.

Examples

Form a collection of yachts longer than 30 feet. Use the context variable BIG_ONES to name the collection:

```
DTR> READY YACHTS↵
DTR> FIND BIG_ONES IN YACHTS WITH LOA GT 30↵
[57 records found]
DTR> SHOW COLLECTIONS↵
Collections:
          BIG_ONES (also CURRENT)
DTR>
```

Form a collection of the 10 most expensive yachts sorted by descending price:

```
DTR> FIND FIRST 10 YACHTS SORTED BY DESC PRICE↵
[10 records found]
DTR> SHOW COLLECTIONS↵
Collections:
          CURRENT
          BIG_ONES
DTR>
```

5.31 FINISH Command

Function

Ends your control over domains and releases any collections associated with the domains.

Format

```
FINISH [domain-name] [,...]
```

Arguments

domain-name

Is the name of a readied domain you want to finish. Separate multiple domain names with commas (,).

Restriction

You can use the FINISH command only at DATATRIEVE command level, indicated by the DTR> prompt.

Results

- If you do not specify the names of any domains in the FINISH command, DATATRIEVE ends your control over all readied domains.
- If you specify the names of one or more domains in the FINISH command, DATATRIEVE ends your control only over the specified domains.
- DATATRIEVE ends your control over all collections associated with domains you finish.
- The FINISH command does not remove dictionary tables from your workspace. You must use the RELEASE command to remove dictionary tables from your workspace.

Usage Notes

- Use the FINISH command to clear your workspace of unneeded domains and to end your control of access to the data file associated with a readied domain. For example, if you ready a domain for exclusive use, no one else can get access to the data file associated with that domain until you use the FINISH command or ready the domain again with a less restrictive access control option.
- You need not use the FINISH command before exiting from DATATRIEVE. The EXIT command automatically ends your control over all readied domains, collections and dictionary tables.

- DATATRIEVE does not use any new dictionary definitions until you explicitly end your control over the domain with the FINISH command. Thus, if you change the definition of a readied domain or its associated record definition, the changes do not have any effect until you enter a FINISH command and ready the domain again.
- If you delete the domain or record definition associated with a readied domain from the dictionary, you can continue to work with the domain, performing any operations consistent with your mode of access to the domain. You can also ready the domain again to change your access mode, as long as you avoid any conflicts of access control options. However, when you use the FINISH command to end your control over the domain, the domain or record definition is no longer in the dictionary, and you can no longer get any access to the domain or record.

Examples

Use the FINISH command to end your control of the domain YACHTS:

```
DTR> SHOW READY(RET)
Ready domains:
        YACHTS:  RMS INDEXED, PROTECTED READ

DTR> FINISH YACHTS(RET)
DTR> SHOW READY(RET)
No ready domains
DTR>
```

Define a domain and record, then delete and redefine them to show how new domain definitions do not take effect until the old domain is finished and the new domain is readied:

```
DTR> DEFINE DOMAIN D1 USING D1REC ON D1;(RET)
DTR> DEFINE RECORD D1REC USING(RET)
DFN> 01 TOP PIC X.(RET)
DFN> ;(RET)
[Record D1REC is 1 bytes long]
DTR> DEFINE FILE FOR D1(RET)
DTR> READY D1 WRITE; STORE D1(RET)
Enter TOP: A(RET)
DTR> DELETE D1;(RET)
DTR> DELETE D1REC;(RET)
DTR> SHOW D1(RET)
"D1" has not been defined in the dictionary
DTR> SHOW D1REC(RET)
"D1REC" has not been defined in the dictionary
DTR> SHOW READY(RET)
Ready domains:
        D1: RMS SEQUENTIAL, PROTECTED WRITE
```

(continued on next page)

```
DTR> DEFINE DOMAIN D1 USING D1REC ON D1;RET
DTR> DEFINE RECORD D1REC USINGRET
DFN> 01 TOP PIC XX,RET
DFN> ;RET
[Record D1REC is 2 bytes long]
DTR> STORE D1RET
Enter TOP: BBRET
Truncation during assignment
Re-enter TOP: BRET
DTR> FIND D1RET
[2 records found]
DTR> PRINT ALLRET

TOP

 A
 B

DTR> DEFINE FILE FOR D1RET
DTR> FIND D1RET
[2 records found]
DTR> PRINT ALLRET

TOP

 A
 B

DTR> FINISH D1RET
DTR> SHOW READYRET
No ready domains
DTR> READY D1 WRITERET
DTR> FIND D1RET
[0 records found]
DTR> REPEAT 2 STORE D1RET
Enter TOP: AARET
Enter TOP: BBRET
DTR> FIND D1RET
[2 records found]
DTR> PRINT ALLRET

TOP

AA
BB

DTR> SHOW READYRET
Ready domains:
        D1: RMS SEQUENTIAL, PROTECTED WRITE
DTR> FINISH D1RET
DTR> SHOW READYRET
No ready domains
DTR>
```

5.32 FOR Statement

Function

Causes DATATRIEVE to execute a statement or group of statements once for each record in the record stream formed by a record selection expression. This statement provides repeating loops for DATATRIEVE operations.

Format

```
FOR rse statement
```

Arguments

rse

Is a record selection expression that forms the record stream that controls 1) the number of times DATATRIEVE executes the statement, and 2) the single record context for the statement. (See Chapter 12 of the *DATATRIEVE-11 User's Guide* for a discussion of DATATRIEVE context.)

statement

Is a simple or compound statement you want DATATRIEVE to execute once for each record in the record stream formed by the RSE. You can form compound DATATRIEVE statements with the BEGIN-END, IF-THEN-ELSE, and THEN statements.

Restrictions

- The domain containing the records specified in the RSE must be readied for READ, WRITE, or MODIFY access.
- Do not include FIND, SELECT, SORT, DROP, or RELEASE statements in a FOR statement.
- You cannot include a DATATRIEVE command in a FOR statement.

Results

- DATATRIEVE executes the statement once for each record in the record stream unless an ABORT statement, a CTRL/C, or a CTRL/Z typed in response to an Enter prompt ends the repetitions.
- DATATRIEVE executes in the order you entered them, the statements of a BEGIN-END block, repeating the process for each record in the record stream.

- Each time DATATRIEVE executes the statement in the FOR loop, it establishes a single record context for one record from the record stream. However, a statement in a BEGIN-END block in a FOR loop can create its own single record context. The new context lasts until DATATRIEVE completes the execution of the nested statement. The single record context then returns to its state prior to the execution of the statement. Similarly, when DATATRIEVE completes the execution of the FOR loop, the single record context returns to its state prior to the execution of the FOR loop.

Usage Notes

- Use the FOR statement to repeat sequences of DATATRIEVE statements. You do not have to know how many records the record stream contains to control the number of times DATATRIEVE loops through the statement. You can use an ABORT statement in the FOR statement to end the loop when certain conditions are met, regardless of the number of records remaining in the record stream.
- Use nested FOR loops to work with lists (the repeating fields contained in hierarchical records). Establish the single record context in the outer loop, and the single list item context in the inner loop. The statement acts on all list items in one target record before acting on any in the next record in the record stream.
- You can force an exit from a FOR loop in several ways:
  - Include an IF-THEN-ELSE statement in the FOR loop. Specify the exit conditions in the IF clause and include an ABORT statement in the THEN clause.
  - Enter a CTRL/C during the execution of a statement, or enter a CTRL/Z in response to an Enter prompt. Use a variable in the FOR loop as a counter to force an exit from the loop before all records in the record stream have been acted upon. To do this:
    1. Declare the variable outside the FOR loop.
    2. Inside the FOR loop, increase the value of the variable by one for each repetition of the loop.
    3. Specify the conditions for the exit, based on the value of the variable in the IF portion of the IF-THEN-ELSE statement, and include an ABORT statement in the THEN or ELSE clause.

Examples

Assign a value to the field PRICE for three yachts with prices equal to zero:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS MODIFY↵
DTR> FIND FIRST 3 A IN YACHTS WITH PRICE = 0↵
[3 records found]
DTR> PRINT A↵
```

(continued on next page)

```
                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM  PRICE

 BLOCK I.       40         SLOOP    39     18,500  12
 BUCCANEER      270        SLOOP    27      5,000  08
 BUCCANEER      320        SLOOP    32     12,500  10

DTR> FOR A↵
CON>     MODIFY USING PRICE = DISP * 1.3 + 5000↵
DTR> PRINT A↵

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM  PRICE

 BLOCK I.       40         SLOOP    39     18,500  12  $29,050
 BUCCANEER      270        SLOOP    27      5,000  08  $11,500
 BUCCANEER      320        SLOOP    32     12,500  10  $21,250

DTR>
```

Use a variable to force an end to a FOR loop before all records in the record stream have been acted upon:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS↵
DTR> DECLARE A PIC 9.↵
DTR> A = 0↵
DTR> PRINT A↵

A

0

DTR> FOR YACHTS↵
CON> BEGIN↵
CON>     A = A + 1↵
CON>     PRINT A, BOAT↵
CON>     IF A = 5 THEN ABORT "End of loop"↵
CON> END↵

                                      LENGTH
                                       OVER
A MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM  PRICE

1  ALBERG         37 MK II   KETCH    37     20,000  12  $36,951
2  ALBIN          79         SLOOP    26      4,200  10  $17,900
3  ALBIN          BALLAD     SLOOP    30      7,276  10  $27,500
4  ALBIN          VEGA       SLOOP    27      5,070  08  $18,600
5  AMERICAN       26         SLOOP    26      4,000  08   $9,895
ABORT: End of loop
Execution terminated by "ABORT" statement
DTR>
```

Use nested FOR loops to increase by one year the age of each child in the first two records of the domain FAMILIES:

```
DTR> SET NO PROMPT↵
DTR> READY FAMILIES MODIFY↵
DTR> PRINT FIRST 2 FAMILIES↵

                        NUMBER     KID
  FATHER      MOTHER     KIDS      NAME     AGE

JIM          ANN           2     URSULA      7
                                 RALPH       3
JIM          LOUISE        5     ANNE       31
                                 JIM        29
                                 ELLEN      26
                                 DAVID      24
                                 ROBERT     16

DTR> SET NO PROMPT↵
DTR> FOR FIRST 2 FAMILIES↵
CON>     FOR KIDS↵
CON>             MODIFY USING AGE = AGE + 1↵
DTR> PRINT FIRST 2 FAMILIES↵

                        NUMBER     KID
  FATHER      MOTHER     KIDS      NAME     AGE

JIM          ANN           2     URSULA      8
                                 RALPH       4
JIM          LOUISE        5     ANNE       32
                                 JIM        30
                                 ELLEN      27
                                 DAVID      25
                                 ROBERT     17

DTR>
```

5.33 HELP Command

Function

Provides on-line information about the use of DATATRIEVE commands, statements, and language elements.

Format

```
HELP [ADVANCED] [topic] [,...]
```

Arguments

topic

Is a DATATRIEVE command, statement, or statement element. Use commas to separate each topic from the next.

ADVANCED

Specifies help text for advanced DATATRIEVE language elements and concepts.

Restriction

You can use the HELP command only at DATATRIEVE command level, in response to the DTR> prompt.

Results

- DATATRIEVE displays on your terminal information for each topic you specify.
- The HELP messages for each command or statement contain on-line information about optional arguments associated with the command or statement.
- Once you have specified ADVANCED, it applies to all subsequent entries. For example, if you specify HELP ADVANCED WOMBAT, HELP; you get advanced help on both WOMBAT and HELP.
- When you specify HELP with no topic, DATATRIEVE displays an explanation of the HELP command. If you type HELP HELP, DATATRIEVE displays a list of topics on which you can obtain help. If you enter HELP ADVANCED HELP, DATATRIEVE displays a list of advanced DATATRIEVE topics on which you can obtain help.

Examples

Use the HELP command to obtain information on using the HELP command:

```
DTR> HELP↵
Two levels of assistance are available, basic and advanced.
To get help for the elementary Datatrieve statements, type
HELP followed by the names of the statements for which help
is required. To get a list of the topics for which help is
available, type

                        HELP HELP

To get assistance with advanced Datatrieve statements, type

                        HELP ADVANCED

followed by the names of the statements for which help is
required. To get a list of the topics for which advanced
help is available, type

                     HELP ADVANCED HELP

Paired square brackets indicate that the enclosed clause is
optional. Capital letters indicate Datatrieve keywords.
Words in lower case indicate that a user supplied name or
value is required.
DTR>
```

Use the HELP command to obtain a list of HELP topics.

```
DTR> HELP HELP↵
The topics of greatest interest to the beginning user are
the following:

GUIDE           READY           FIND            SORT
PRINT           MODIFY          STORE           EXIT

Help is available for the following topics:

ABORT           ADT             ASSIGNMENT      BEGIN
COMPUTED        CONDITION       DATE            DECLARE
DEFINE          DEFINEP         DELETE          DELETEP
DICTIONARY      DISPLAY         DOMAIN          EDIT
EDIT-STRING     ERASE           EXIT            FILE
FIND            FINISH          FOR             GUIDE
HELP            IF              MODIFY          OCCURS
PIC             PRINT           PROCEDURE       RANGE
READY           RECORD          RELEASE         REPEAT
RSE             SELECT          SET             SHOW
SHOWP           SORT            STORE           SUM
TABLE           THEN            USAGE           VALUE
VIEW
DBMS            OWNER            WITHIN
DTR>
```

Use the HELP command to get general and advanced information on the FIND command:

```
DTR> HELP FIND↵
The FIND statement is used to establish a collection of
records from a domain, a list or a previously established
collection. The forms of the statement are:

                FIND domain-name WITH condition

                  FIND CURRENT WITH condition

where "domain-name", if used, must be the name of a readied
domain (see HELP READY) and "condition" is a conditional
expression.

The FIND statement establishes a collection of the records
from the domain, the list or the current collection that
satisfy the given condition. The number of records found
will be printed on your terminal.

Examples:

     FIND YACHTS WITH LENGTH-OVER-ALL BETWEEN 26 AND 30

     FIND CURRENT WITH PRICE < 15000

DTR> HELP ADVANCED FIND↵
The full form of the FIND statement is

                            FIND rse

where "rse" is a record selection expression. If "rse"
contains a name for the new collection, the resulting
collection is established under that name.
DTR>
```

5.34 IF-THEN-ELSE Statement

Function

Causes DATATRIEVE to execute one of two statements or compound statements depending on the evaluation of a conditional (Boolean) expression.

Format

```
IF boolean-expression [THEN] statement-1 [ELSE statement-2]
```

Arguments

boolean-expression

Is a Boolean expression (see Chapter 2).

statement-1

Specifies the simple or compound statement you want DATATRIEVE to execute when the Boolean expression evaluates to true.

ELSE statement-2

Specifies the simple or compound statement you want DATATRIEVE to execute when the Boolean expression evaluates to false.

Restriction

You must observe all restrictions on the individual statements used in the IF-THEN-ELSE statement.

Results

- When the Boolean expression evaluates to true, DATATRIEVE executes statement 1 in the THEN clause. The THEN keyword is optional; you can use it to clarify the syntax of the IF statement.
- If you specify an ELSE clause and the Boolean expression evaluates to false, DATATRIEVE executes statement 2 in the ELSE clause.
- If you do not specify an ELSE clause and the Boolean expression evaluates to false, DATATRIEVE does not execute statement 1 in the THEN clause and is ready to execute the next command or statement it encounters.

Usage Notes

- You can press RETURN before any element of the IF statement except ELSE, and DATATRIEVE will prompt you to continue the statement. If you press RETURN just before typing ELSE, DATATRIEVE considers the IF statement complete and executes or ignores the THEN clause, depending on the evaluation of the Boolean expression. DATATRIEVE then tries to execute the ELSE clause as though it were a separate statement and displays an error message on your terminal because the ELSE clause is not a statement. When you have to break lines in an IF-THEN-ELSE statement, put ELSE at the end of a line rather than at the beginning of the next. This practice is especially important when you are writing a procedure, because DATATRIEVE does not check the syntax of the statements in the procedure until you invoke it.
- Use an IF-THEN-ELSE statement to force an exit from a BEGIN-END block or from a FOR loop. Put an ABORT statement in either the THEN clause or the ELSE clause, and put the IF-THEN-ELSE statement in an appropriate place in the BEGIN-END block or FOR loop.
- You can nest IF-THEN-ELSE statements by using an IF-THEN-ELSE statement as either statement 1 or statement 2, or both.

Examples

Print each yacht built by Pearson, and modify the price:

```
DTR> SET NO PROMPT(RET)
DTR> READY YACHTS WRITE(RET)
CON> FOR YACHTS WITH BUILDER = "PEARSON"(RET)
CON>     BEGIN(RET)
CON>             PRINT(RET)
CON>             PRINT SKIP(RET)
CON>             IF *,"Y to modify price, N to skip" CONT "Y"(RET)
CON>                     THEN(RET)
CON>                     BEGIN(RET)
CON>                             MODIFY PRICE(RET)
CON>                             PRINT SKIP(RET)
CON>                     END ELSE(RET)
CON>                     PRINT "No change", SKIP(RET)
CON>             IF *,"Y to continue" NOT CONT "Y" THEN(RET)
CON>                     ABORT "End of price changes"(RET)
CON>     END(RET)

                                 LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL   WEIGHT BEAM  PRICE

 PEARSON      10M        SLOOP   33    12,441  11


Enter Y to modify price, N to skip: N(RET)
No change

Enter Y to continue: Y(RET)
 PEARSON      26         SLOOP   26     5,400  08
```

(continued on next page)

```
Enter Y to modify price, N to skip: Y(RET)
Enter PRICE: 34,521(RET)


Enter Y to continue: Y(RET)
 PEARSON      26W          SLOOP    26      5,200   09


Enter Y to modify price, N to skip: N(RET)
No change

Enter Y to continue: N(RET)
ABORT: End of price changes
Execution terminated by "ABORT" statement
DTR>
```

Use the IF statement to select families with fathers named Jim, and list the kids in those families:

```
DTR> SET NO PROMPT(RET)
DTR> READY FAMILIES(RET)
DTR> FOR FAMILIES WITH ANY KIDS(RET)
CON>     IF FATHER EQ "JIM" THEN(RET)
CON>             PRINT "The kids of JIM and "|MOTHER,(RET)
CON>             ALL KID_NAME ("Kids with Fathers"/(RET)
CON>                     "Named JIM") OF KIDS, SKIP(RET)

                                Kids with Fathers
                                    Named JIM

The kids of JIM and ANN             URSULA
                                    RALPH

The kids of JIM and LOUISE          ANNE
                                    JIM
                                    ELLEN
                                    DAVID
                                    ROBERT


DTR>
```

5.35 MODIFY Statement

Function

Changes the value of one or more fields in a selected record or in any or all records in a collection or record stream.

Format

```
MODIFY [ALL]  [ field-name [,...] ]
              [ USING statement-1 ]
      [VERIFY [USING] statement-2]
      [OF rse]
```

Arguments

ALL

Specifies that you want to modify either all records in the CURRENT collection or all records in the record stream specified by the OF rse clause.

field-name

Specifies the name of a field to modify in the target record or records. If you specify more than one field name, use a comma to separate field names. DATATRIEVE prompts you to supply a value for each specified field.

USING statement-1

Specifies a simple or compound statement that assigns a value or values to one or more fields in the target record you want to modify. This clause can also contain other DATATRIEVE statements, including PRINT, STORE, and MODIFY statements.

VERIFY [USING] statement-2

Specifies a statement that DATATRIEVE executes before modifying the target record.

OF rse

Is a record selection expression that forms a record stream containing the records you want to modify.

Restrictions

- The domain containing the records you want to modify must be readied for modify or write access.
- You cannot modify a primary or alternate key that has the NO CHANGE attribute in an RMS indexed sequential file.

- You cannot modify a COMPUTED BY field or a REDEFINES field.
- When entering a complex MODIFY statement, use the hyphen continuation character if you want to press RETURN before typing the key words USING, VERIFY, or OF rse. If you must break an input line near one of these key words, you can also type the keyword at the end of a line and press RETURN.
- You cannot use a prompting value expression in a USING clause to assign a value to a group field.
- You must establish a context to specify the records on which the MODIFY statement acts. You can establish context in four ways:
  - Form a collection with a FIND statement and use a SELECT statement to identify a selected record (see Table 5-9).
  - Form a CURRENT collection with a FIND statement and specify ALL in the absence of an OF rse clause in the MODIFY statement (see Table 5-10).
  - Form a record stream with the OF rse clause in the MODIFY statement (see Table 5-11).
  - Use a FOR loop and form a record stream with the RSE in the FOR statement (see Table 5-12).
- Do not use ALL in the absence of the OF rse clause in a MODIFY statement that is part of a FOR loop. The result is a series of redundant changes to the records; all the records in the CURRENT collection get modified each time through the FOR loop.
- Avoid redundant loops when using the OF rse clause in MODIFY statements used in FOR loops. The OF rse clause should contain a Boolean expression that matches each record to be modified with one record from the record stream providing the data (see Example 3).
- If you specify one or more field names in a MODIFY statement, they must be the names of group or elementary fields that DATATRIEVE can recognize in the context established for the MODIFY statement. If you specify more than one field name, use a comma to separate field names.
- If you specify a record stream with an OF rse clause and omit the field list or the USING clause, you must include the keyword ALL: MODIFY ALL OF rse. This restriction helps remind you that this statement makes all the records in the record stream identical.
- If you enter CTRL/Z while being prompted for a field value, DATATRIEVE will end the modify operation. The field that was being modified when you typed CTRL/Z will contain the modified value, but the data file will not be modified. You must reestablish the collection to restore the original value of the field you were modifying.

Results

- When DATATRIEVE executes a MODIFY statement, it immediately changes the information in the data file. See the first Usage Note.
- DATATRIEVE prompts you for field values with this message:

  Enter field-name:

  You can enter any of the following responses:

  - To leave the value unchanged, press TAB and then press RETURN.
  - To change the value of the field, type a new value, then press RETURN. The new value should conform to the data description of the field, as defined in the record definition.
  - To change the value of an alphanumeric field to spaces, type a space and press RETURN.
  - To change the value of a numeric field to zero, type a zero (0) or space and press RETURN. To end the prompting cycle, press CTRL/Z. DATATRIEVE discards the changes made to the record being modified when you entered CTRL/Z. That record is unchanged, but if you have already modified records from the collection or record stream, those changes remain in effect. Each of those records was changed in the data file as soon as you entered your response to the last prompt for that record.
- If you press RETURN in response to a prompt for a field value, DATATRIEVE reprompts for the value.
- If you enter a value in response to a prompt that is longer than the length of the field to which it is being assigned, DATATRIEVE displays an error message and reprompts for the value.
- If you omit both a list of field names and a USING clause, DATATRIEVE prompts for each elementary field in the record.
- If you include a list of field names, DATATRIEVE prompts for each elementary field specified or implied by the list. If you include a group field name in the list, DATATRIEVE prompts for each elementary field contained in the group field.
- If you specify a USING clause, DATATRIEVE uses any assignment statements in statement 1 to modify the values of the specified fields. DATATRIEVE does not then prompt for any field values unless the USING clause contains an assignment statement with a prompting value expression (value-expression = *.prompt-name). See Tables 5-9 through 5-12 for syntax examples of the USING clause.

- If you do not specify ALL or OF rse and do not put the statement in a FOR loop, DATATRIEVE modifies only the selected record:
  - DATATRIEVE prompts for values for each specified or implied field in the selected record unless you specify a USING clause that contains no prompting value expressions.
  - If you specify a USING clause that prompts for a value that is used in an arithmetic calculation (USING PRICE = PRICE * .EXCHANGE_RATE), DATATRIEVE uses your response to the prompt to calculate the value of the arithmetic expression. The value of that arithmetic expression is then the value stored in the updated field.
  - The context in which the MODIFY statement executes is determined by the selected record of the most recently established collection. If no selected record exists for the CURRENT collection, then the context is determined by the most recently established collection that has a selected record. If there is no selected record for any existing collection, DATATRIEVE displays an error message.

Table 5-9 lists the syntax, prompts, and results of the MODIFY statements that change selected records.

Table 5-9: Modifying the Selected Record

Syntax	Result
MODIFY	Prompts once for each elementary field in the record definition. Changes each elementary field in the selected record to the value you supply to the corresponding prompt.
MODIFY field-list	Prompts once for each elementary field specified or implied by the list. Changes each elementary field specified or implied by the list to the value you supply to the corresponding prompt.
MODIFY USING statement-1	No prompts unless statement 1 contains prompting value expressions. Changes each specified or implied elementary field in the selected record to the values supplied by the assignment statements in statement 1.

- If you specify the argument ALL but do not include an OF rse clause in a MODIFY statement, DATATRIEVE changes all the records in the CURRENT collection:
  - DATATRIEVE assigns the same value to each specified and implied field of every record in the CURRENT collection except when the MODIFY statement includes a USING clause that assigns a value to the field with an arithmetic calculation that uses the old value of the field. In this case, DATATRIEVE uses the value of each calculation to modify the field value in the corresponding record (see the fourth Usage Note).
  - If the USING clause contains a prompting value expression, DATATRIEVE prompts only once for a value for each specified or implied field. If the USING clause contains no prompting value expressions, DATATRIEVE does not prompt for the field values.

Table 5-10 lists the syntax, prompts, and results of MODIFY statements that change all the records in the CURRENT collection.

Table 5-10: Modifying All Records in the CURRENT Collection

Syntax	Result
MODIFY ALL	Prompts once for each field in the record definition. Changes each field of every record in the CURRENT collection to the value you supply to the corresponding prompt.
MODIFY ALL field-list	Prompts once for each elementary field specified or implied by the list. Changes each specified or implied field of every record in the CURRENT collection to the value you supply to the corresponding prompt.
MODIFY ALL USING statement-1	No prompts unless statement 1 contains prompting value expressions. Changes each specified or implied elementary field of every record in the CURRENT collection to the value supplied by the assignment statements in statement 1.

- If you include an OF rse clause in a MODIFY statement, DATATRIEVE changes all the records in the record stream specified by the RSE as follows:
  - DATATRIEVE assigns the same value to each specified and implied field of every record in the record stream, except when the MODIFY statement includes a USING clause that assigns a value to the field with an arithmetic calculation that uses the old value of the field. In this case, DATATRIEVE uses the value of each calculation to modify the field value in the corresponding record (see Usage Note 4).

- If the USING clause contains a prompting value expression, DATATRIEVE prompts only once for a value for each specified or implied field. If the USING clause contains no prompting value expressions, DATATRIEVE does not prompt for the field values.
- The ALL keyword is optional when you specify an OF rse clause in the MODIFY statement and has no effect on the result of the modify operation.

Table 5-11 lists the syntax, prompts, and results of MODIFY statements that change all the records in a target record stream.

Table 5-11: Modifying All Records in a Record Stream

Syntax	Result
MODIFY ALL OF rse	Prompts once for each field in the record definition. **Use with care**. Changes each field of every record in the record stream to the value you supply to the corresponding prompt. MODIFY ALL of domain-name changes **all** the records in the domain.
MODIFY [ALL] field-list OF rse	Prompts once for each elementary field specified or implied by the list. Changes each specified or implied field of every record in the record stream to the value you supply to the corresponding prompt.
MODIFY [ALL] USING statement-1 OF rse	No prompts unless statement 1 contains prompting value expressions. Changes each specified or implied elementary field of every record in the record stream to the value supplied by the assignment statement in statement 1.

- When you want DATATRIEVE to prompt for field values for each record in a record stream, include the MODIFY statement in a FOR statement. When you include a MODIFY statement in a FOR loop, DATATRIEVE modifies the records specified by the RSE in the FOR statement and prompts for field values as indicated in Table 5-12.
- When you include a VERIFY clause in a MODIFY statement, DATATRIEVE executes statement 2 for each target record before changing that record. If statement 2 contains an ABORT statement and a target record causes the abort conditions to be true, DATATRIEVE aborts the MODIFY statement without changing that record and returns you to command level, indicated by the DTR> prompt. Any previous records changed by that MODIFY statement remain changed.

Table 5-12: Modifying Each Record in a Record Stream

Syntax	Result
FOR rse MODIFY	Prompts once for every elementary field in the record definition for each record in the record stream. Changes each field of each record to the value you supply to the corresponding prompt.
FOR rse MODIFY field-list	Prompts once for every elementary field specified or implied by the list for each record in the record stream. Changes each specified or implied field of each record to the value you supply to the corresponding prompt.
FOR rse MODIFY USING statement-1	No prompts unless statement 1 contains prompting value expressions: • A *.prompt value expression prompts once for each record in the record stream. • A **.prompt value expression prompts only once, during the first execution of the FOR loop. Changes each specified or implied elementary field of each record in the record stream to the value supplied by the assignment statements in statement 1.

- SET ABORT and SET NO ABORT affect a MODIFY statement with an ABORT statement in the VERIFY clause as follows:
  - If the MODIFY statement is not part of a procedure or command file, SET ABORT and SET NO ABORT do not affect the result of the ABORT.
  - If the MODIFY statement is part of a procedure or command file and SET ABORT is in effect, DATATRIEVE aborts the MODIFY statement without changing the record that caused the abort and returns you to command level without executing the rest of the procedure or command file. If SET NO ABORT is in effect, DATATRIEVE aborts the MODIFY statement without changing the record that caused the abort and executes the next statement in the procedure or command file.

Usage Notes

- Use the MODIFY statement with care, as it changes information in the data file. DATATRIEVE catches any syntax errors you might make in entering the statement, but even correct syntax may contain a logical error that causes the wrong records to be changed or the wrong values to be entered into the fields.

  If DATATRIEVE prompts you for a field you did not expect, enter CTRL/Z to prevent changing records or fields you do not wish to change. **It is good practice to print each record before you modify it.**

- If you enter CTRL/Z in response to a prompt, DATATRIEVE aborts the modify operation. The field in the record that was being modified will contain the modified value, but the data file is not modified. To restore the original value of the field, reestablish the collection.
- Once you have modified the value or values in one or more fields, you can recover the previous information only by explicitly changing the new values back to the old ones. You may have to use several MODIFY statements to make the necessary changes.
- Statement 1 in the USING clause is usually an assignment statement or a BEGIN-END block containing more than one assignment statement.
- If you do not want all the records in the CURRENT collection or in a record stream created by an OF rse clause to have the same new field value, specify an assignment statement in the USING clause that makes the new value of a field in the target record depend on the old value of that field. For example:

```
DTR> READY YACHTS MODIFY(RET)
DTR> FIND YACHTS WITH BUILDER = "PEARSON"(RET)
[10 records found]
DTR> MODIFY ALL USING PRICE = PRICE + 500(RET)
DTR>
```

  The specified calculation applies to each target record, but the new values derived for each record are independent of one another. The operation performed on a field value, however, must be appropriate to the data type of that field. Do not, for example, try to do arithmetic with nonnumeric data in an alphanumeric field.

- You can use the MODIFY statement to transfer information from one domain to another, from one group field to another, and from one elementary field to another by specifying one of the following assignment statements in the USING clause:

  field-name-1 = field-name-2

  group-field-name-1 = group-field-name-2

  In both these statements, the field on the left of the assignment statement (1) specifies the target field and the field on the right specifies the source field (2). Each field name must be adequately qualified to establish the proper context for each side of the assignment statement.

  - Transfer of data is easiest when the field definitions are exactly the same on both sides of the assignment statement. If the field definitions differ, truncations may result if the lengths of the fields do not match.
  - You must also anticipate any conflicts between data types. You can, for example, modify an alphanumeric field with a numeric value, but modifying a numeric field with a value that contains nondigit characters causes DATATRIEVE to display a warning message.

- To include other DATATRIEVE statements in the USING clause of a MODIFY statement, put them in a BEGIN-END block. For example, to see the target record before changing it, put a PRINT statement before the assignment statement in the BEGIN-END block.
- Use a BEGIN-END block in the USING clause to create an audit trail of the previous values of the records you modify. Define and ready another domain that uses the same record definition as the one whose records you intend to modify. Then put an appropriate STORE statement in a BEGIN-END block in the USING clause of the MODIFY statement.
- Statement 2 in the VERIFY clause typically contains an IF-THEN-ELSE statement and an ABORT statement to control the modification operation.
  - DATATRIEVE first checks the value you supply to a MODIFY statement prompt against any validation conditions specified for that field in the record definition. If the value passes this validation test, it is then checked against the conditions in the VERIFY clause of the MODIFY statement.
  - If you always use the same validation conditions for modifying and storing data, put those conditions in VALID IF clauses in the record definition. That way, DATATRIEVE reprompts you for another value for the same field if the value you entered does not pass the validation test. When you use an ABORT statement in the VERIFY clause to validate a value, you are returned to DATATRIEVE command level and must reenter the MODIFY statement when the value does not pass the validation test.

Examples

Increase the price of all yachts built by "AMERICAN" by 10 percent:

```
DTR> READY YACHTS MODIFY(RET)
DTR> FIND A IN YACHTS WITH BUILDER = "AMERICAN"(RET)
[2 records found]
DTR> PRINT A(RET)

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL   WEIGHT BEAM  PRICE

 AMERICAN     26         SLOOP   26     4,000  08   $9,895
 AMERICAN     26-MS      MS      26     5,500  08  $18,895

DTR> MODIFY USING PRICE = PRICE * 1.1 OF A(RET)
DTR> PRINT A(RET)

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL   WEIGHT BEAM  PRICE

 AMERICAN     26         SLOOP   26     4,000  08  $10,884
 AMERICAN     26-MS      MS      26     5,500  08  $20,784

DTR>
```

Form a collection of yachts. Modify all the elementary fields within the group field SPECIFICATIONS for the records in the CURRENT collection:

```
DTR> SET NO PROMPT(RET)
DTR> READY YACHTS MODIFY(RET)
DTR> FIND YACHTS WITH BEAM = 0(RET)
[5 records found]
DTR> FOR CURRENT MODIFY USING(RET)
CON>     BEGIN(RET)
CON>             PRINT SPECS(RET)
CON>             RIG = **.RIG(RET)
CON>             LOA = **.LOA(RET)
CON>             DISP = *.WEIGHT(RET)
CON>             BEAM = *.BEAM(RET)
CON>             PRICE = PRICE * 1.1(RET)
CON>     END(RET)

       LENGTH
        OVER
 RIG    ALL   WEIGHT BEAM  PRICE

SLOOP   32     9,500  00
Enter RIG: (TAB)(RET)
Enter LOA: 33(RET)
Enter WEIGHT  12000(RET)
Enter BEAM: 10(RET)
SLOOP   32    11,000  00  $29,500
Enter WEIGHT: (TAB)(RET)

Enter BEAM: 11(RET)
SLOOP   31    13,600  00  $32,500
Enter WEIGHT: 15000(RET)
Enter BEAM: 12(RET)
SLOOP   35    23,200  00
Enter WEIGHT: (TAB)(RET)
Enter BEAM: 13(RET)
SLOOP   32    14,900  00  $34,480
Enter WEIGHT: (TAB)(RET)
Enter BEAM: 9(RET)
DTR> PRINT ALL(RET)

                                LENGTH
                                 OVER
MANUFACTURER   MODEL      RIG    ALL   WEIGHT BEAM  PRICE

 METALMAST   GALAXY     SLOOP   33    12,000  10
 O'DAY       32         SLOOP   33    11,000  11  $32,450
 RYDER       S. CROSS   SLOOP   33    15,000  12  $35,750
 TA CHIAO    FANTASIA   SLOOP   33    23,200  13
 WRIGHT      SEAWIND II SLOOP   33    14,900  09  $37,928

DTR>
```

Create a domain (UPDATES) that stores the original prices of all yachts in the YACHTS domain. UPDATE_REC contains four fields pertinent to the price of each yacht that are identical to YACHT fields: BUILDER, MODEL, DISPLACEMENT, and PRICE. Modify the prices of the first five yachts in the YACHTS domain and use MODIFY statements in nested FOR loops to change the prices back to their original values.

```
DTR> SET NO PROMPT
DTR> DEFINE DOMAIN UPDATES USING UPDATE_REC ON UPDATE.DAT;
DTR> DEFINE RECORD UPDATE-REC USING
01 BOAT.
  03 TYPE.
     06 MANUFACTURER PIC X(10)
        QUERY_NAME IS BUILDER.
     06 MODEL PIC X(10).
  03 DISPLACEMENT PIC 99999
     QUERY_HEADER IS "WEIGHT"
     EDIT_STRING IS ZZ,ZZ9
     QUERY_NAME IS DISP.
  03 PRICE PIC 99999
     VALID IF PRICE>DISP*1.3 OR PRICE EQ 0
     EDIT_STRING IS $$$,$$$.
;
DTR> DEFINE FILE FOR UPDATES KEY = BUILDER (DUP),
DFN>     KEY = MODEL (CHANGE, DUP)
DTR> READY UPDATES WRITE
DTR> READY YACHTS WRITE
DTR> FOR A IN YACHTS
CON>     STORE UPDATES USING BOAT = A.BOAT
DTR> PRINT FIRST 5 UPDATES

MANUFACTURER   MODEL     WEIGHT  PRICE

 ALBERG        37 MK II  20,000 $36,951
 ALBIN         79         4,200 $17,900
 ALBIN         BALLAD     7,276 $27,500
 ALBIN         VEGA       5,070 $18,600
 AMERICAN      26         4,000  $9,895

DTR> FOR FIRST 5 YACHTS MODIFY PRICE
Enter PRICE:  40000
Enter PRICE:  20000
Enter PRICE:  30000
Enter PRICE:  20000
Enter PRICE:  10000
DTR> PRINT FIRST 5 YACHTS

                                LENGTH
                                 OVER
MANUFACTURER   MODEL     RIG     ALL   WEIGHT BEAM  PRICE

 ALBERG        37 MK II  KETCH   37    20,000  12  $40,000
 ALBIN         79        SLOOP   26     4,200  10  $20,000
 ALBIN         BALLAD    SLOOP   30     7,276  10  $30,000
 ALBIN         VEGA      SLOOP   27     5,070  08  $20,000
 AMERICAN      26        SLOOP   26     4,000  08  $10,000
```

(continued on next page)

```
DTR> FOR FIRST 5 A IN UPDATES↵
CON>     FOR B IN YACHTS WITH BUILDER = A.BUILDER AND↵
CON>             MODEL = A.MODEL↵
CON>     MODIFY USING B.PRICE = A.PRICE↵
DTR> PRINT FIRST 5 YACHTS↵

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG       ALL    WEIGHT BEAM   PRICE

 ALBERG       37 MK II    KETCH     37     20,000  12   $36,951
 ALBIN        79          SLOOP     26      4,200  10   $17,900
 ALBIN        BALLAD      SLOOP     30      7,276  10   $27,500
 ALBIN        VEGA        SLOOP     27      5,070  08   $18,600
 AMERICAN     26          SLOOP     26      4,000  08    $9,895

DTR>
```

5.36 OCCURS Clause

The OCCURS clause defines multiple occurrences (or repetitions) of a field or group of fields. The multiple occurrences, called a list, create a hierarchy in the domain. (See Chapter 15 of the *DATATRIEVE-11 User's Guide* for more information on hierarchies.)

The OCCURS clause has two formats: one format for a fixed number of occurrences and one for a variable number of occurrences.

5.36.1 Fixed Number of Occurrences

Function

Defines a fixed number of occurrences for a field or group of fields.

Format

```
OCCURS n TIMES
```

Arguments

n

Is a positive integer specifying the number of occurrences for the field.

Restriction

A field definition cannot contain both an OCCURS and a COMPUTED BY clause. You cannot specify multiple occurrences of a COMPUTED BY field.

Result

This format of the OCCURS clause defines a list with a fixed number of occurrences. It reserves enough space in each record to contain all the occurrences of the field (or fields), whether or not they contain data.

Usage Notes

- A record definition can contain any number of OCCURS clauses that specify a fixed number of occurrences.
- A field in a group field with an OCCURS clause can contain an OCCURS clause. The result is a a list nested within a list (a sublist). See example 3.

Examples

The following field definition reserves enough space in every record for two occurrences of the elementary field KIDS_NAMES; each occurrence is 10 characters long. You can store up to two names (containing up to 10 characters each) in the KIDS_NAMES field:

```
03 KIDS_NAMES PIC X(10)
   OCCURS 2 TIMES.
```

The following definition specifies that the group field KIDS_NAMES is repeated twice. Each group field contains two fields: FIRST_NAME (10 characters long) and MIDDLE_INIT (1 character). A total of 22 characters is reserved in every record for the group field:

```
03 KIDS_NAMES OCCURS 2 TIMES.
   05 FIRST_NAME PIC X(10).
   05 MIDDLE_INIT PIC X.
```

The fields are stored in the record in the following order:

```
KIDS_NAMES
   FIRST_NAME
   MIDDLE_INIT
KIDS_NAMES
   FIRST_NAME
   MIDDLE_INIT
```

By nesting a sublist within a list, reserve enough space in a record to store up to three nicknames for each KIDS_NAMES:

```
03 KIDS_NAMES OCCURS 2 TIMES.
   05 FIRST_NAME PIC X(10).
   05 MIDDLE_INIT PIC X.
   05 NICKNAME PIC X(10)
      OCCURS 3 TIMES.
```

The fields are stored in the following order:

```
KIDS_NAMES
   FIRST_NAME
   MIDDLE_INIT
   NICKNAME
   NICKNAME
   NICKNAME
KIDS_NAMES
   FIRST_NAME
   MIDDLE_INIT
   NICKNAME
   NICKNAME
   NICKNAME
```

5.36.2 Variable Number of Occurrences

Function

Defines a variable number of occurrences of a group of fields.

Format

```
OCCURS min TO max TIMES DEPENDING ON field-name
```

Arguments

min

Is zero or a positive integer specifying the minimum number of occurrences of the field. DATATRIEVE does does not check this value.

max

Is a positive integer specifying the maximum number of occurrences of the field. This value must be greater than 0.

field-name

Is the name of a field in the same record definition. The value of the field determines the number of occurrences of this field. The field must be a numeric field containing a positive integer; it cannot be defined with digits to the right of the decimal point.

Restrictions

- No other field definition can follow the last elementary field in the group field containing this clause.
- A record definition can contain only one OCCURS clause in this format.
- A field definition cannot contain both an OCCURS and a REDEFINES clause or an OCCURS and a COMPUTED BY clause.

Result

This format of the OCCURS clause defines a list with a variable number of occurrences. The number of occurrences of the group field in any record is equal to the value of the field specified in the OCCURS clause. Therefore, the sizes of records in the domain can vary. If you use the MAX argument when you define the data file, all records in the file have the same length. See the section in this manual on the DEFINE FILE command.

Usage Notes

A group field containing this format of the OCCURS clause can contain one or more fields with an OCCURS clause. The nested OCCURS clause, however, must specify a fixed number of occurrences of the field.

Example

Define a record that specifies a variable number of occurrences.

```
01 FAMILY.
   03 PARENTS.
      06 FATHER PIC X(10).
      06 MOTHER PIC X(10).
   03 NUMBER_KIDS PIC 99 EDIT_STRING IS Z9.
   03 KIDS OCCURS 0 TO 10 TIMES DEPENDING ON NUMBER_KIDS.
      06 EACH_KID.
         09 KID_NAME PIC X(10) QUERY_NAME IS KID.
         09 AGE PIC 99 EDIT_STRING IS Z9.
```

The number of occurrences of the KIDS field depends on the value of the NUMBER_KIDS field. If the value is 0, there are no occurrences of the field; if it is 1, there is one occurrence, and so on. Each occurrence of KIDS contains three fields: the group field EACH_KID and the elementary fields KID_NAME and AGE.

5.37 OPEN Command

Function

Opens an RMS file to serve as a log of your interactive dialogue with DATATRIEVE. DATATRIEVE copies both your input and its own output to the file.

Format

```
OPEN file-spec
```

Argument

file-spec

Is the file specification of the file to be opened. The file specification has the following format:

For RSTS/E systems: dev:[PPN]file-name.type

For RSX systems: dev:[UIC]file-name.type;ver

You must specify at least a file-name. All other fields are optional. If you omit fields in the file specification, DATATRIEVE uses the following defaults:

Field	Default
dev:	SY: (the system device)
[UIC/PPN]	Your default UIC/PPN
file-name	No default
.type	.DAT
;ver	1 or next higher version number (non-RSTS/E systems only)

Restrictions

- Do not try to create a log file of a Guide Mode session. DATATRIEVE-11 does not write your input to the file.
- If you use the OPEN command in a procedure, no statements or commands in the procedure are written to the log file. The output of a procedure, however, is written to the file.
- If you invoke a procedure after you have used the OPEN command, none of the commands or statements in the procedure are written to the log file.
- You cannot have two log files open at the same time. If you enter a second OPEN command without closing the first log file, DATATRIEVE ignores the second OPEN command and the first log file remains open.

Results

- Except for dialogue with the Editor, Guide Mode, ADT, and procedures, DATATRIEVE writes both your input and its own output to the file you specify in the command.
- If you have a log file open when you invoke a command file, DATATRIEVE does not display the various DATATRIEVE prompts (such as DTR> and CON>) on the terminal or in the log file.
- DATATRIEVE closes the file when you enter a CLOSE command, or when you exit from DATATRIEVE with the EXIT command or CTRL/Z.
- If you close the file with a CLOSE or EXIT command, that command is also included in the file. If you close the file by exiting from DATATRIEVE with a CTRL/Z, neither the control character nor any of the input line is included in the file.
- DATATRIEVE puts a carriage return/line feed between prompts (DTR>, CON>, DFN>) and your input. The file requires minor editing to make it an exact copy of the terminal display.

Usage Notes

- Keeping log files of your dialogue with DATATRIEVE can provide you with a transaction log and can aid in developing and debugging DATATRIEVE applications.
- A log file of your dialogue with DATATRIEVE is essential when submitting a Software Problem Report (SPR) to DIGITAL. See *DATATRIEVE-11 User's Guide* for information on how to submit an SPR.
- To include the contents of a procedure in a log file, enter a SHOW procedure-name command before invoking the procedure.

Example

Open a log file, display the contents of a procedure, invoke the procedure, and close the log file. Exit from DATATRIEVE and use the operating system TYPE command to display the contents of the log file:

```
DTR> OPEN LOG(RET)
DTR> !  This is a test of the OPEN command(RET)
DTR> SHOW SELL_BOAT(RET)
PROCEDURE SELL_BOAT
READY YACHTS WRITE
FIND YACHTS WITH BUILDER EQ *.BUILDER AND MODEL EQ *.MODEL
PRINT ALL
IF *."Y IF BOAT SOLD" EQ "Y"
         THEN ERASE ALL ELSE
         PRINT "Sell it now!"
END_PROCEDURE
DTR> :SELL_BOAT(RET)
Enter BUILDER: ALBIN(RET)
Enter MODEL: VEGA(RET)
```

(continued on next page)

```
                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM  PRICE

 ALBIN         VEGA        SLOOP    27      5,070  08   $18,600

Enter Y IF BOAT SOLD: N(RET)
Sell it now!

DTR> EXIT(RET)
DTR> TYPE LOG.LST(RET)
DTR>
! This is a test of the OPEN command
DTR>
READY YACHTS
DTR>
SHOW SELL_BOAT
PROCEDURE SELL_BOAT
READY YACHTS WRITE
FIND YACHTS WITH BUILDER EQ *.BUILDER AND MODEL EQ *.MODEL
PRINT ALL
IF *."Y IF BOAT SOLD" EQ "Y"
          THEN ERASE ALL ELSE
          PRINT "Sell it now !"
END_PROCEDURE

DTR>
:SELL_BOAT
Enter BUILDER:
ALBIN
Enter MODEL:
VEGA

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG      ALL    WEIGHT BEAM  PRICE

 ALBIN         VEGA        SLOOP    27      5,070  08   $18,600

Enter Y IF BOAT SOLD:
N
Sell it now!
DTR>
EXIT
DTR>
```

5.38 PICTURE Clause

Function

Specifies the storage format for field values.

Format

```
PIC[TURE] [IS] picture-string
```

Arguments

picture-string

Is one or more picture string characters that describe the storage format for the field value.

Restrictions

- This clause is valid for elementary fields only.
- Do not use spaces in the picture string.

Result

The picture string controls the storage format of the field's values. In general, each character in the string corresponds to one character position in the field value. For example, for a numeric field without a USAGE clause, 999999 specifies six digits in six character positions, occupying six bytes of storage.

Usage Notes

- For numeric fields, you can also include a USAGE clause to specify the internal format of the digits.
- To enter a series of identical picture string characters, place a repeat count in parentheses following the picture character. For example, the picture string 9(6) is equivalent to 999999.
- Table 5-13 contains a list of picture string characters. The picture string characters you specify for a field depend on whether the field is alphanumeric or numeric.

Table 5-13: Picture String Characters

Field Class	Picture Character	Meaning
Alphanumeric	X	Each X represents one character in the field.
Numeric	9	Each 9 represents one digit in the field. You can specify from 1 to 18 digits for a numeric field.
	S	An S indicates that a sign (+ or –) is stored in the field. A picture string can have only one S and it must be the leftmost character. If there is no SIGN clause for the field, the sign shares the rightmost character position with the lowest-valued digit, with one exception: the sign bit is used for the sign for COMP values.
	V	A V indicates an implied decimal point. The decimal point does not occupy a character position in the field, although DATATRIEVE uses its location to align data in the field. A picture string can contain only one V.
	P	Each P specifies a decimal scaling position. Each P represents a "distance" in digits from an implied decimal point. (A P does not count toward the limit of 18 digits per field.) A P can appear at the right or left of the picture string. A V is unnecessary for any picture string containing a P.

- The picture string for an alphanumeric field specifies the number of characters in the field. Only the picture string character X is valid in the picture string for an alphanumeric field. Each X corresponds to a single character position in the field. For example, the following field definition specifies that the MODEL field contains ten alphanumeric characters:

```
06 MODEL PIC X(10).
```

- You can use the characters 9, S, V, and P in a picture string for a numeric field to specify the number of digits in the field, a sign, an implied decimal point, and a decimal scaling factor.
- The picture string character 9 represents one digit in the field value. For example, the picture string 9(4) indicates four digits, so the field value can range from 0 to 9999, with one exception: if you specify both PIC 9(4) and USAGE IS COMP, then it is possible to store numbers as large as 32,768. You can specify from 1 to 18 digits for a numeric field.

- To specify that a numeric field can contain a sign (+ or −), you must include an S in the picture string:
  - A picture string can contain only one S and it must be the leftmost character in the string. For example, the picture string S9(4) indicates a signed field, four digits in length; thus, the field value can range from –9999 to + 9999.
  - The sign specified with the S picture clause character is not printed unless you include an EDIT_STRING clause with the field definition. For example, the following field definition specifies that a sign in the picture string is to be printed following the last digit of the field value:

```
03 CURRENT_BALANCE
PICTURE IS S9999V99
EDIT_STRING IS $$$$9.99-.
```

- The picture character V specifies the position of an "implied" decimal point:
  - The picture string 9(5)V99 specifies a seven-digit field; the last two digits of the field value follow the decimal point. The decimal point does not occupy a character position in the record; DATATRIEVE uses the implied decimal point in computations, Boolean expressions, and other arithmetic operations.
  - If there is no V in the picture string, DATATRIEVE treats the field value as an integer (that is, as if a V were specified to the right of the rightmost digit). Thus, the picture strings 999 and 999V are equal.
  - DATATRIEVE displays the implied decimal point when you retrieve the value from a field with a V in the picture string if you use an edit string that contains a decimal point (.).
- The picture string character P specifies a decimal scaling position. Each P represents one decimal position between the value stored in the field and the implied decimal point of the value. A P does not occupy a character position in the field:
  - If you specify a P in a picture string, the V character is optional. For example, the picture strings 99PPP and 999PPV are equivalent.
  - P (or multiple Ps) must be the leftmost or rightmost character in the picture string. If leftmost, the decimal point is assumed to be to the left of the leftmost P; if rightmost, the decimal point is assumed to be to the right of the rightmost P.
  - DATATRIEVE treats each P in a picture string as a zero.

5.39 PRINT Statement

Function

Causes DATATRIEVE to format and write values to your terminal, a file, or a unit record device.

Format

```
PRINT ALL ALL print-list OF rse-1 [,print-list] OF rse-2
    [ ON { file-spec  } ]
         { *.prompt   }
```

Arguments

print-list

Is a list of value expressions and formatting specifications. Table 5-14 describes the print list elements. Table 5-15 describes the modifiers you can use to control the column header and format for each data field in the output of the PRINT statement.

ALL

When used alone following PRINT, ALL causes all the records in the CURRENT collection to be displayed on your terminal or written to the specified file or device.

When used with a print list, ALL causes the print list to be evaluated for each record in the current collection.

When the print list begins with an inner print list, ALL is required to establish the proper context in which to resolve references to the items in the hierarchical list.

rse

Is a record selection expression that creates the record stream DATATRIEVE uses to evaluate the elements of the print list.

file-spec

Is the file specification for the output file in the following format:

For RSTS/E systems: dev:[PPN]file-name.type

For RSX systems: dev:[UIC]file-name.type;ver

You must specify a file name. If you omit other fields in the file specification, DATATRIEVE uses the following defaults:

Field	Default
dev:	SY: (the system device)
[UIC/PPN]	Your default UIC/PPN
file name	No default
.type	.LST
;ver	1 or next higher version (non-RSTS/E systems only)

*.prompt-name

Is a prompting value expression that prompts you for a file specification for the output file.

Restrictions

- To print data from records in a domain, you must ready the domain for read, write, or modify access. You cannot print data from domains readied for extend access because you cannot establish collections or record streams from domains readied for extend access.
- If you specify a device name in a PRINT statement, the device must be one to which you have access, such as a line printer, a tape drive, your own terminal, or another terminal. You cannot cause the output of the PRINT statement to be displayed on another terminal that is logged in.
- To print character string literals, enclose the string in quotation marks. To include a quotation mark in a character string literal, type two quotation marks for every one you want in the output of the statement. For example:

```
DTR> PRINT "They said, "We're going.""RET
They said, "We're going."
DTR>
```

Results

- DATATRIEVE evaluates the print list and writes the resulting output to the specified or implied file or device. The format and content of the output depend on the print list elements and modifiers included in the statement. Table 5-14 describes print list elements and Table 5–15 describes print list modifiers.

Table 5-14: Print List Elements

Print List Element	Function and Results
field-name group-field-name [modifier] list-field-name	Specifies the field whose contents are to be formatted and printed. The optional modifier describes the column header for the field, or the format of the output, or both (see Table 5-15). If the field is a group field, DATATRIEVE displays all the elementary fields contained in that group field. If you omit this element, all the elementary fields are printed.
literal *.prompt-name **.prompt-name [modifier] arithmetic-exp statistical-exp	Specifies a value expression to be evaluated and printed. The optional modifier describes the column header for the value expression, or the format of the display, or both (see Table 5-15). Chapter 2 discusses these value expressions.
SPACE [n]	Inserts n horizontal spaces before the next print list element. If you omit n, DATATRIEVE inserts one space before the next print list element.
TAB [n]	Inserts the space of n tab characters before the next print list element. If you omit n, DATATRIEVE inserts the space of one tab before the next print list element. DATATRIEVE assumes that tabs are set every eight spaces and inserts enough spaces (not actually tab characters) in the print line to start the next print list element in the appropriate column.
COL n	Specifies that the following print list element begins in column n of the detail line. If n is less than the current column number, DATATRIEVE skips a line and begins the next print list element in column n. The first column in the line is column 1.
SKIP [n]	Begins the output of the next print list element at the beginning of the nth line from the current line. If n is greater than 1, the intervening lines are blank. If you omit n, DATATRIEVE moves the cursor to the beginning of the next line. If you omit this print list element, DATATRIEVE displays multilined output on consecutive lines.
NEW_PAGE	Moves the cursor to the top of a new print page. Column headers are suppressed, and output begins at column 1 unless another print list element changes the position of the cursor.
ALL print-list OF rse	Specifies an inner print list. DATATRIEVE evaluates the inner print list once for each record specified by the outer RSE. This print list element is generally used with a list in a hierarchical record to display all the values in the list for each of the records. DATATRIEVE uses the list name as the source of the RSE in the print list element. If an inner print list is the first element in a print list, you must add the keyword ALL before the inner print list (ALL ALL print-list OF rse OF rse). This additional ALL is not required if another print list element precedes the inner print list.

Table 5-15: Print List Modifiers

Print List Modifier	Function and Results
("header-segment"[/...])	Specifies one or more character string literals to be displayed as a column header or headers above the first line of output from the PRINT statement. The entire modifier must be enclosed in parentheses and must immediately follow its associated field name or value expression.
(–)	Suppresses the printing of column headers for all the elementary fields in a group field. When used with a field name or a dictionary table value expression, this modifier suppresses the printing of the query header associated with the field in its record definition or with the dictionary table in its dictionary definition.
USING edit-string	Imposes the characteristics of the specified edit string on the preceding field or value expression. The edit string must conform to the rules for the EDIT_STRING clause. If you follow an edit string with other print list items, put a space between the last character of the edit string and the comma that separates the edit string from the next print list item.

- If you do not put the PRINT statement in a FOR loop and do not include a record selection expression or the argument ALL, DATATRIEVE uses the data from the selected record of the most recently formed collection with a selected record to evaluate the print list. DATATRIEVE evaluates the print list once and creates one or more lines of output, depending on the formatting options you specify.
- If you specify the argument ALL and do not include an RSE, DATATRIEVE uses the data in the records of the CURRENT collection to evaluate the value expressions in the print list. DATATRIEVE evaluates the print list once for each record in the CURRENT collection and creates one or more lines of output for each record, depending on the formatting options you specify.
- If you include a record selection expression in a PRINT statement, DATATRIEVE uses the data in each record in the record stream to evaluate the value expressions in the print list. DATATRIEVE evaluates the print list once for each record in the record stream and creates one or more lines of output for each record, depending on the formatting options you specify.
- If you put a PRINT statement in a FOR loop, DATATRIEVE uses the data in each record in the record stream created by the rse in the FOR statement. DATATRIEVE evaluates the value expressions in the print list once for each record, and creates one or more lines of output for each record, depending on the formatting options you specify.
- Unless a COL, SPACE, or TAB print list element changes the position of the cursor, all output begins at column 1.

- If you do not specify a column header or column suppression modifier after a field name, variable, or dictionary table value expression, DATATRIEVE uses the query header for the element, if one has been defined:
  - If no query header has been defined for an elementary field, the field name is used. If the field name has underscores in it, DATATRIEVE suppresses the underscores and converts the field name to a multiline header. The LENGTH_OVER_ALL field in the YACHT record, for example, is converted to a three-line header:

```
DTR> SET NO PROMPT↵
DTR> READY YACHTS↵
DTR> FOR FIRST 5 YACHTS PRINT BUILDER, LOA↵
              LENGTH
               OVER
MANUFACTURER   ALL

 ALBERG         37
 ALBIN          26
 ALBIN          30
 ALBIN          27
 AMERICAN       26

DTR>
```

  - If no query header has been defined for a variable, DATATRIEVE uses the variable name.
- When you specify only one header segment, the specified header is printed on one line above the first line of output from the PRINT statement. If the header is shorter than the field reserved for the value, the header is centered above the field. If the header is longer, the field is centered under the header. In this case, however, DATATRIEVE determines the placement of the other output fields relative to the length of the header, not the length of the field.
- If you specify more than one header segment ("header-1"/"header-2"[/...]), the specified headers are printed on successive lines, centered above the associated field. If the edit string for the field is longer than any header segment, the width of the field is equivalent to the length of the edit string. If the edit string is shorter than any header segment, the width of the field is equivalent to the longest header segment.
- If you do not include the USING edit string modifier, DATATRIEVE formats print list items as follows:
  - If the print list item is a field name, DATATRIEVE uses the edit string specified for the field in the record definition or the PICTURE (PIC) clause.
  - If the print list item is a variable, DATATRIEVE uses either the edit string specified in the DECLARE statement that created the variable or the PICTURE (PIC) clause.
  - If the print list item is a prompting value expression, DATATRIEVE uses a default alphanumeric edit string 10 characters long, X(10).

- If you use a prompting value expression to specify the output file or device, DATATRIEVE prompts you for the name when it executes the PRINT statement. If you omit the ON clause, DATATRIEVE displays the output on your terminal.
- If you are using an RSX operating system and end your file specification with the name of a line printer or another terminal that is not a spooled device, the output of a PRINT statement can be immediately displayed on the device. Consult *RSX-11M/M-PLUS System Managers Guide,* for details regarding spooled devices.

Usage Notes

- PRINT statement arguments allow you to specify the following information:
  - The **data** to be printed to a terminal or file. Data can be the contents of a field, the value of a variable, or any other value expression.
  - The **format** of the data. You can specify an edit string to override any edit string in the field or variable definition or to format a value expression.
  - The **spacing** (both horizontal and vertical) for the output. You can insert tabs or spaces between columns or skip lines between lines of output.
  - **Column headers** for each column of data in the output. You can also specify that no header is to be printed above a column.
- When you omit the print list from a PRINT statement, DATATRIEVE uses the following defaults for data display and formatting:
  - The **data** displayed on the terminal or written to a file is the contents of all fields in the selected record (PRINT), the records in the CURRENT collection (PRINT ALL), or the records in the record stream formed by the rse in the PRINT statement (PRINT rse).
  - The **format** of the field's contents is determined by the record definition.
  - The horizontal **spacing** is determined by the longest of three items: 1) the edit string, if specified, 2) the longest header segment, if specified, or 3) the length of the value of the print list element. Output begins in column 1 and is single-spaced, with a single blank line following the header line.
  - **Column headers** for fields are the query headers specified in the record definition. If no query headers were specified, DATATRIEVE uses the field names for headers. If the field name contains an underscore, DATATRIEVE suppresses the underscore, places each part of the field name on a separate line, and centers each part of the field name above the column of data. If the query header contains only a hyphen, DATATRIEVE does not print any header.
- Use inner print lists to display data in the lists of hierarchical records.

Examples

Write to a file the builder and length-over-all of each yacht:

```
DTR> PRINT BUILDER, LOA OF YACHTS ON REPORT.REP↵
DTR>
```

Display on your terminal the data on five yachts that are 30 or 31 feet long:

```
DTR> READY YACHTS↵
DTR> PRINT FIRST 5 YACHTS WITH LOA BETWEEN 30 AND 31↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG    ALL    WEIGHT BEAM  PRICE

 ALBIN          BALLAD     SLOOP   30      7,276  10  $27,500
 BOMBAY         CLIPPER    SLOOP   31      9,400  11  $23,950
 C&C            CORVETTE   SLOOP   31      8,650  09
 CAL            3-30       SLOOP   30     10,500  10
 FISHER         30         KETCH   30     14,500  09

DTR>
```

Use the PRINT statement to display the group field SPECS from a selected record:

```
DTR> FIND YACHTS WITH BUILDER = "PEARSON"↵
[10 records found]
DTR> SELECT↵
DTR> PRINT SPECS↵

        LENGTH
         OVER
 RIG     ALL    WEIGHT BEAM  PRICE

SLOOP    33     12,441  11

DTR>
```

With an inner print list, display the father and kids of the first two families:

```
DTR> READY FAMILIES↵
DTR> PRINT FATHER, ALL EACH_KID OF KIDS OF FIRST 2 FAMILIES↵

               KID
  FATHER       NAME     AGE

JIM          URSULA       7
             RALPH        3
JIM          ANNE        31
             JIM         29
             ELLEN       26
             DAVID       26
             ROBERT      16

DTR>
```

Display the first two records of a target stream formed by a FOR loop:

```
DTR> SET NO PROMPT↵
DTR> FOR FIRST 2 YACHTS↵
CON> PRINT TYPE, LOA, PRICE↵

                          LENGTH
                           OVER
MANUFACTURER    MODEL      ALL     PRICE

 ALBERG        37 MK II    37     $36,951
 ALBIN         79          26     $17,900

DTR>
```

Display the hierarchical records of a target record stream formed by nested FOR loops:

```
DTR> SET NO PROMPT↵
DTR> FOR FIRST 2 FAMILIES↵
CON> FOR KIDS↵
CON> PRINT MOTHER, FATHER, KID_NAME↵

                          KID
  MOTHER      FATHER      NAME

ANN          JIM         URSULA
ANN          JIM         RALPH
LOUISE       JIM         ANNE
LOUISE       JIM         JIM
LOUISE       JIM         ELLEN
LOUISE       JIM         DAVID
LOUISE       JIM         ROBERT

DTR>
```

Display the records from a record stream formed by a PRINT statement containing two RSEs (inner print list precedes any other print list):

```
DTR> SET NO PROMPT↵
DTR> PRINT ALL ALL KID_NAME OF FIRST 1 KIDS,↵
CON>     MOTHER OF FIRST 2 FAMILIES↵

   KID
   NAME       MOTHER

URSULA        ANN
ANNE          LOUISE

DTR>
```

5.40 QUERY_HEADER Clause

Function

Specifies the column header DATATRIEVE uses when it formats the display of the field value for the PRINT statement or for the Report Writer AT and PRINT statements.

Format

```
QUERY_HEADER [IS] {"header-segment"} [/...]
```

Arguments

"header-segment"

Is the column header displayed above a column of data. If you specify only one character string literal, that string is printed on one line above the column. If you specify more than one character string literal, you must separate them by a slash (/). DATATRIEVE prints the literals on successive lines, centered above the column.

Restrictions

- This clause is valid for elementary fields only.
- The column header can include any character except a carriage return, line feed, or control character. To include a quotation mark in a column header, precede it with a second quotation mark. (See the third example.)

Results

- If you include this clause, DATATRIEVE uses the specified query header as the default column header when printing the field.
- If you omit this clause, DATATRIEVE uses the field name as the default column header when printing the field.

Usage Note

You can override the QUERY_HEADER clause by using the column header modifier for print list items in the PRINT statement and Report Writer AT and PRINT statements. The column header modifier overrides the default query header specified in the field definition (see Table 5-15 for more information on print list modifiers).

Examples

Give the DISPLACEMENT field of YACHTS a default query header of WEIGHT:

```
06 DISPLACEMENT PIC 99999
        QUERY_HEADER IS "WEIGHT"
        EDIT_STRING IS ZZ,ZZ9
        QUERY_NAME IS DISP.
```

Give the LENGTH_OVER_ALL field in YACHTS a column header of LENGTH (IN FEET), printed on two separate lines:

```
06 LENGTH_OVER_ALL PIC XXX
        QUERY_HEADER IS "LENGTH"/"(IN FEET)".
```

When you display the LOA field, DATATRIEVE prints the column header as:

```
LENGTH
(IN FEET)
```

Give the LENGTH_OVER_ALL field a three-line column header with one letter per line:

```
06 LENGTH_OVER_ALL PIC XXX
        QUERY_HEADER IS "L"/"O"/"A"
```

DATATRIEVE prints the column header as:

```
L
O
A
```

5.41 QUERY_NAME Clause

Function

Specifies an alternate name for the field.

Format

```
QUERY_NAME [IS] query-name
```

Arguments

query-name

Is the query name. The rules for forming and using a query name are the same as those for a field name.

Restriction

The query name must conform to rules for naming DATATRIEVE fields.

Result

You can use the the query name anywhere you can use the field name.

Usage Notes

- This clause is valid for both group and elementary fields.
- Use the QUERY_NAME clause when a field name is too long to use with ease.
- Like a field name, a query name can duplicate another query name (or a field name) in the record. A query name can also be qualified by other query names or field names.

Examples

Define DISP as a alternate name for the DISPLACEMENT field:

```
06 DISPLACEMENT PIC 99999
        QUERY_NAME IS DISP.
```

Define SPECS as an alternate name for the group field SPECIFICATIONS:

```
03 SPECIFICATIONS
        QUERY_NAME SPECS.
```

For the field DELINQUENT_ACCOUNT_STATUS, define the query name SPECIAL_HANDLING.

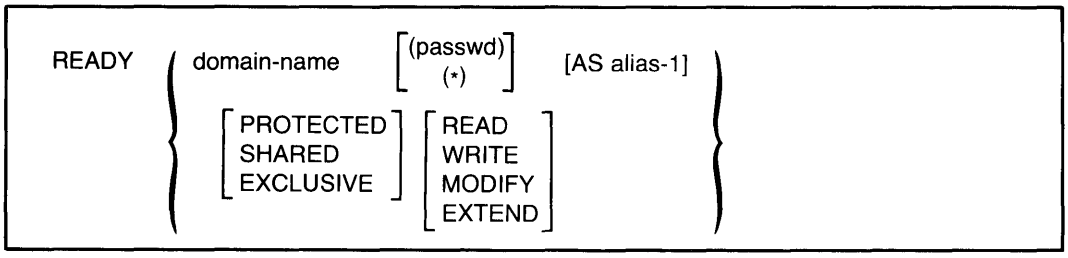
```
09 DELINQUENT_ACCOUNT_STATUS
        PIC X
        QUERY_NAME IS SPECIAL_HANDLING.
```

5.42 READY Command

Function

Gives you access to one or more domains and controls the access of other users to those domains.

Format



Arguments

domain-name

Is the name of a domain you want to access.

alias

Is an alternative name for the domain name specified in the command. You can use the alias in place of the name of the domain where the syntax of a statement calls for a domain name.

(passwd)
(*)

Is the password necessary to gain access to the domain. The parentheses are required. Use the asterisk to have DATATRIEVE prompt for the password. If you omit this argument, DATATRIEVE uses your login UIC/PPN to verify your access privileges.

PROTECTED
SHARED
EXCLUSIVE

Are options you can select to control the access of other users to a domain you ready. Access options are described in table 5-16.

Table 5-16: Access Options

Option	Access Constraints
PROTECTED	Any other user can have only read access to records in the domain. No other user can have write, modify, or extend access to the records in the domain. This option is the default for domains based on RMS files.
SHARED	Any other user can have access to the domain at the same time, in protected or shared mode. Individual records, however, cannot be shared at the same time.
EXCLUSIVE	No other user can have access to the domain at the same time, in any access mode. The file containing the data is locked by RMS.

READ
MODIFY
WRITE
EXTEND

Are options you can select to request a mode of access to a domain. Privileges assigned to you in the domain's access control list determine the mode of access you can specify. Table 5-17 describes access modes and lists the privileges needed for each access mode.

Table 5-17: Access Modes

Mode	Type of Access	Privileges Needed
READ	You can only retrieve records (the default).	Read, write, or modify
MODIFY	You can retrieve and modify records.	Modify or write
WRITE	You can retrieve, modify, store, and erase records.	Write
EXTEND	You can only store records.	Extend, write, or modify

Restrictions

- You must have E (execute) access privilege to the associated record definition before you can ready a domain.
- You must have privilege to the domain appropriate for the type of access specified. Table 5-17 lists privileges needed for access modes.
- You can ready only domains catalogued in your current dictionary.
- You can ready only domains whose associated record definitions are catalogued in your current dictionary.

- You cannot ready a domain unless the data file specified in the domain definition exists. When the data file exists:
  - You must have at least operating system read access to the file before you can ready the domain for read access.
  - You must have operating system write access to the file before you can ready the domain for write, modify, or extend access.
- If another user has readied the domain for exclusive use, you cannot ready the domain.
- If another user has readied the domain for protected use, you can ready the domain only for read access.
- You cannot specify the shared access option for a domain that uses an RMS sequential data file.
- If a conflict occurs between the access mode and the access option specified (such as trying to ready an RMS sequential file for protected write), DATATRIEVE automatically readies the domain with the exclusive access option.
- RMS does not enforce the exclusive access option when you combine it with the read access mode.

Results

- DATATRIEVE gives you access to the domain with the access option and access mode specified.
- If you do not specify an alias for the domain, you must use the domain name in commands and statements that operate on the domain.
- If you specify an alias for the domain name, DATATRIEVE assigns the alias to the domain. You must then use the alias in commands and statements that operate on the domain. Once you have assigned an alias to a domain, you cannot access the domain by its domain name (the name by which it is catalogued in your dictionary) until you finish the alias with the FINISH command or exit from DATATRIEVE.
- If you do not specify an access option for a domain, DATATRIEVE readies the domain with the default option, protected. With protected access, other users can have read access to the domain, but not write, modify, or extend access.
- If you do not specify an access mode for a domain, DATATRIEVE readies the domain for default access, read. With read access, you can retrieve records, but you cannot store, modify, or erase them.

- If you want to ready a domain that is already readied, the following rules apply:
  - You must use the same alias to ready the domain again.
  - The access option and access mode specified in the new READY command replace those previously in effect.
  - DATATRIEVE preserves your collections for domains based on RMS files.
  - Any changes you have made to the record definition associated with the domain do not take effect until you finish the domain and ready it again.

Usage Notes

- Use the SHOW READY command to show readied domains. The SHOW READY command displays the name, file type, access option, access mode, and dictionary name of all readied domains. The most recently readied domain is at the top of the displayed list.
- The access mode specified in the READY command determines the operations you can perform on the domain. Table 5-18 lists the access modes needed for statements that operate on domains. Use the SHOW READY command to make sure you have readied the target domain with the appropriate access mode.

Table 5-18: Access Modes Required by DATATRIEVE Statements

Statement	Access Mode Required
ERASE	Write
FIND	Modify, read, or write
MODIFY	Modify or write
PRINT	Modify, read, or write
REPORT	Modify, read, or write
SORT	Modify, read, or write
STORE	Write or extend
SUM	Modify, read, or write

- When specifying access options for a domain, impose as few restrictions on other users as the needs of your task allow. If you specify exclusive access, no other user can access the domain.
- If you must specify a password to get access to the domain, use the asterisk to prevent the password from being displayed on your terminal. After you enter the READY command, DATATRIEVE prompts for the password but does not echo the characters on your terminal.

- If you specified a password prompt in the domain definition, DATATRIEVE prompts for the password when you ready the domain and uses your response to search the entries of the access control list of the record definition to determine what privileges you have to the record definition (see Example 3).

  If a password is used in the record definition, it must match the password in the dictionary. The password in the domain definition is independent of the password in the record definition.

- A domain stays ready until you release it with the FINISH command or end your DATATRIEVE session with EXIT or CTRL/Z. FINISH ends your control over a domain, releases the collections of records from that domain, allows other users access to the domain in any access mode, and frees the computer resources used by the domain.

- If you redefine the format of the record associated with a readied domain, the change in the record definition does not take effect until you use the FINISH command to finish the domain and the READY command to ready it again. If you just ready the domain again, the new record definition does not take effect.

- If you want to change a record definition or the type of file organization of a domain's data file without redefining the domain, follow these steps to define a new data file, and transfer the data with the STORE statement:

  1. Ready the domain as an alias:

```
DTR> READY YACHTS AS OLD_YACHTS(RET)
DTR> SHOW READY(RET)
Ready domains:
   OLD_YACHTS: RMS INDEXED, PROTECTED READ

DTR>
```

  2. Change the record definition with the DEFINE RECORD command, if you wish.

  3. Define a new data file for the domain, but do not use the SUPERSEDE option; this creates a new version of the file associated with the readied domain. This does not interfere with the link between the readied domain and the original version of the data file:

```
DTR> DEFINE FILE FOR YACHTS KEY = TYPE (NODUP)(RET)
DTR>
```

  4. Ready the domain as a different alias and specify WRITE access mode. This READY command uses the new record definition, if you made one, and opens the new data file created by the DEFINE FILE command:

```
DTR> READY YACHTS AS NEW_YACHTS WRITE(RET)
DTR> SHOW READY(RET)
Ready domains:
   NEW_YACHTS:  RMS INDEXED, PROTECTED WRITE
   OLD_YACHTS:  RMS INDEXED, PROTECTED READ
DTR>
```

5. Use a FOR loop to move the data from the original data file associated with OLD_YACHTS to the new one associated with NEW_YACHTS. DATATRIEVE transfers data from fields in the original data file into fields with the same names in the new data file:

```
DTR> SET NO PROMPT↵
DTR> FOR A IN OLD_YACHTS↵
DTR>              STORE B IN NEW YACHTS USING B.BOAT = A.BOAT↵
DTR>
```

Examples

Ready the domain YACHTS for WRITE access:

```
DTR> READY YACHTS WRITE↵
```

Ready the domain PHONES for EXTEND access. The domain definition includes a password:

```
DTR> READY PHONES (*) EXTEND↵
Enter password for PHONES:        ↵
DTR>
```

Define a domain with a password prompt. Then ready the domain and respond to the password prompt. Note that DATATRIEVE does not echo the password you enter:

```
DTR> DEFINE DOMAIN PROMPT_YACHTS USING YACHT(*) ON YACHT;↵
DTR> READY PROMPT_YACHTS AS PYTS↵
Enter password for YACHT:      ↵

DTR>
```

5.43 REDEFINES Clause

Function

Provides an alternate way to define a field.

Format

```
level-no field-name-1 REDEFINES field-name-2
```

Arguments

level-no

Is the level number of field name 1. Although not a part of the REDEFINES clause, the level number is shown in the format to clarify its position relative to the clause.

field-name-1

Is the name of the REDEFINES field. Use this name when you want to refer to this field. Although not a part of the REDEFINES clause, the field name is shown in the format to clarify its function and its position relative to the clause.

field-name-2

Is the name of the field being redefined.

Restrictions

- The field to be redefined (field name 2) must appear in the record definition before its REDEFINES field (field name 1). Both fields must have the same level number.
- The definition of field name 2 cannot contain a REDEFINES clause. However, it can be subordinate to a group field with a REDEFINES clause.
- Neither field name 1 nor field name 2 can be defined with, or contain, a field defined with an OCCURS...DEPENDING clause.
- Neither field name 1 nor field name 2 can contain a COMPUTED BY clause. You cannot redefine a COMPUTED BY field.
- In the definition of field name 1, the REDEFINES clause must immediately follow the field name. No other clause can be used between the field name and the keyword REDEFINES.
- The REDEFINES field cannot describe an area larger than the area of field name 2. However, the area can be smaller than that of field name 2.

Result

The REDEFINES clause provides an alternate definition for an elementary or group field. The redefinition refers to the same area of the record as the original definition, but it uses the content of the field in a different way.

Usage Note

If you need to refer to parts of a numeric field as well as the field itself, you can redefine the field as a group field. The subordinate fields of the group fields would contain the parts of the numeric field value that you will refer to. Thus, the REDEFINES clause allows you to redefine a numeric field as a group field. A group field cannot be numeric; a group field is always alphanumeric.

Example

The following record definition shows a redefinition of the field PART_NUMBER, a numeric field containing 10 digits. Two group fields redefine PART_NUMBER: PART_NUMBER_PARTS and PART_NUMBER_GROUPS. Each redefinition specifies a group field containing a total of 10 digits (the total number of digits in all subordinate fields):

```
05 PART_NUMBER PIC 9(10).
05 PART_NUMBER_PARTS REDEFINES PART_NUMBER.
   07 PRODUCT_GROUP PIC 99.
   07 PRODUCT_YEAR PIC 99.
   07 ASSEMBLY_CODE PIC 9.
   07 SUB_ASSEMBLY PIC 99.
   07 PART_DETAIL PIC 999.
05 PART_NUMBER_GROUPS REDEFINES PART_NUMBER.
   07 PRODUCT_GROUP_ID PIC 9(4).
   07 PART_DETAIL_ID PIC 9(6).
```

In this example, the field PRODUCT_GROUP refers to the most significant digits of PART_NUMBER and PART_DETAIL to the least significant digits.

5.44 RELEASE Command

Function

Ends your control over one or more collections, tables, or global variables and frees the workspace they occupy.

Format

```
RELEASE  [ collection-name ]
         [ table-name      ]  [,...]
         [ variable-name   ]
```

Arguments

collection-name
table-name
variable-name

Is the name of a collection, a dictionary table, or a variable you want to release.

Restrictions

- You must use this command at DATATRIEVE command level, indicated by the DTR> prompt.
- You must specify at least one collection, table, or variable. There are no defaults.

Results

- DATATRIEVE releases the specified collections, dictionary tables, and variables, freeing the workspace they occupied. The effect is very much like the FINISH command.
- Records and domains associated with collections named in the RELEASE command are not affected by the command.
- If you specify more than one item in the command, DATATRIEVE releases the items in left-to-right order. If the command fails before all items are released, DATATRIEVE prints a message indicating which item could not be released. In such a case, DATATRIEVE releases all items in the command preceding the one that failed and does not release the ones that follow it.

Usage Notes

- Before you use the RELEASE command, use the SHOW command to see what collections, tables, and variables you have in your workspace:
  - SHOW COLLECTIONS shows the collections you have established and the order in which you created them.
  - SHOW TABLES shows dictionary tables you have accessed.
  - SHOW FIELDS shows global variables you have created.
- When you have two or more collections in your workspace and you release the CURRENT one, the collection you formed most recently becomes the new CURRENT collection.
- The RELEASE command is implicit in the following cases:
  - When a FIND statement successfully forms a collection, DATATRIEVE releases an existing collection with the same name. If the CURRENT collection has no other name, a new collection formed by a FIND statement releases and replaces the previous CURRENT collection. DATATRIEVE releases the previous collection, even if the new collection contains no records.
  - When you type EXIT or CTRL/Z to end a DATATRIEVE session, DATATRIEVE releases all collections, dictionary tables, and global variables.
  - When you use a FINISH command, DATATRIEVE releases all collections associated with the specified domain or domains.
  - When you declare a global variable that has the same name as one that exists, DATATRIEVE releases the old global variable.
- You cannot assign a value to or retrieve the value from a global variable that you have released. You can redefine the variable with the DECLARE statement. If you do, the previous value is lost and the variable is initialized to zero if numeric or blank if alphanumeric.

Examples

Release one of two named collections, then release the other:

```
DTR> SET NO PROMPT
DTR> FIND SMALL_ONES IN YACHTS WITH LOA < 20
[2 records found]
DTR> FIND BIG_ONES IN YACHTS WITH LOA > 40
[8 records found]
DTR> SHOW COLLECTIONS
Collections:
        BIG_ONES (also CURRENT)
        SMALL_ONES
```

(continued on next page)

```
DTR> RELEASE BIG_ONES↵
DTR> SHOW COLLECTIONS↵
Collections:
        SMALL_ONES (also CURRENT)
DTR> RELEASE SMALL_ONES↵
DTR> SHOW COLLECTIONS↵
No established collections
DTR>
```

Release the dictionary table DEPT_TABLE and the global variables, DEPT and NEW_DEPT:

```
DTR> DECLARE DEPT PIC XX.↵
DTR> DECLARE NEW-DEPT PIC XX.↵
DTR> DEPT = "CE"↵
DTR> NEW_DEPT = "SD"↵
DTR> PRINT DEPT VIA DEPT-TABLE USING T(20)↵

        DEPT

Commercial
Engineering

DTR> PRINT NEW-DEPT VIA DEPT_TABLE USING T(20)↵

        NEW
        DEPT

Sales Department

DTR> SHOW TABLES↵
Tables loaded:
         DEPT_TABLE
Tables:
        DEPT_TABLE      RIG_TABLE

DTR> SHOW TABLES↵
Tables:
        DEPT_TABLE      RIG_TABLE
DTR> SHOW FIELDS↵
Global variables:
    NEW_DEPT     [Character string]
    DEPT [Character string]
DTR> RELEASE DEPT↵
DTR> SHOW FIELDS↵
Global variables:
    NEW_DEPT     [Character string]
DTR> RELEASE NEW-DEPT↵
DTR> SHOW FIELDS↵
No Domains Readied or Global Variables Declared
DTR>
```

5.45 REPEAT Statement

Function

Causes DATATRIEVE to execute a simple or compound statement a specified number of times.

Format

```
REPEAT value-expression statement
```

Arguments

value-expression

Is a value expression indicating the number of times to execute the statement. This argument must evaluate to a positive, integer less than or equal to 32,767.

statement

Is any simple or compound DATATRIEVE statement except a FIND, SELECT, DROP, or SORT statement.

Restrictions

- Do not use a FIND, SELECT, DROP, or SORT statement in a REPEAT statement.
- You must observe all restrictions on the statements you use in a REPEAT statement.
- If the REPEAT statement invokes a procedure (for example, REPEAT n :procedure-name), the procedure cannot contain a command or a FIND, SELECT, DROP, or SORT statement as its first element.
- If a compound statement in a REPEAT statement invokes a procedure (for example, REPEAT n BEGIN :procedure-name; END), that procedure cannot contain a DATATRIEVE command or a FIND, SELECT, DROP, or SORT statement as its first element.

Results

- DATATRIEVE executes the statement the number of times specified by the value expression. Then DATATRIEVE executes the command or statement following the REPEAT statement.
- If a REPEAT statement invokes a procedure, DATATRIEVE executes only the first statement (whether simple or compound) in the procedure the number of times specified in the value expression. Each succeeding statement in the procedure is executed only once and is not part of the REPEAT statement. This is why the first element of a procedure cannot be a command or a FIND, SELECT, DROP, or SORT statement.

Usage Notes

- Use the REPEAT statement to repeat a simple or compound statement a fixed number of times.
- To force an exit from a loop created by a REPEAT statement, take one of the following actions:
  - Type CTRL/Z in response to any prompt within the loop.
  - Type CTRL/C at any time during the execution of the statement (but not in response to a prompt).
- To control a REPEAT statement loop, use an IF-THEN-ELSE statement with an ABORT statement in the THEN or ELSE clause. Put the IF statement in a BEGIN-END block in the REPEAT statement.
- Having SET ABORT or SET NO ABORT in effect does not change DATATRIEVE's response to an ABORT statement in a REPEAT statement loop. When the conditions for the ABORT statement are true, DATATRIEVE ends the REPEAT statement, does not execute any statements following the ABORT statement, and returns you to DATATRIEVE command level.
- You can nest REPEAT statements. DATATRIEVE executes each inner REPEAT statement the specified number of times each time it loops through the outer REPEAT statement.

Examples

Print "TEST REPEAT" three times:

```
DTR> REPEAT 3 PRINT "TEST REPEAT"RET
TEST REPEAT
TEST REPEAT
TEST REPEAT

DTR>
```

Exit from a REPEAT statement by responding to a prompt with a CTRL/Z:

```
DTR> READY YACHTS WRITERET
DTR> REPEAT 5 STORE YACHTSRET
Enter MANUFACTURER: HOBIERET
Enter MODEL: CATRET
Enter RIG: SLOOPRET
Enter LENGTH-OVER-ALL: 22RET
Enter DISPLACEMENT:  4000RET
Enter BEAM: 8RET
Enter PRICE: 6500RET
Enter MANUFACTURER: ^Z
Execution terminated by operator
DTR> FIND YACHTS WITH BUILDER = "HOBIE"RET
[1 record found]
DTR>
```

Show the effect of nesting REPEAT statements in procedures. The procedure NUM1 contains two PRINT statements. The procedure NUM2 contains two REPEAT statements, one nested in the other. The inner REPEAT statement causes DATATRIEVE to execute the first PRINT statement in NUM1 twice each time DATATRIEVE loops through the outer REPEAT statement:

```
DTR> SET NO PROMPT↵
DTR> SHOW NUM1↵
PROCEDURE NUM1
PRINT SKIP, "ONE, TWO, THREE"
PRINT "ONE, TWO, THREE, FOUR, FIVE"
END_PROCEDURE
DTR> :NUM1↵

ONE, TWO, THREE

ONE, TWO, THREE, FOUR, FIVE

DTR> SHOW NUM2↵
PROCEDURE NUM2
REPEAT 2
        BEGIN
                REPEAT 2 :NUM1
        END
:NUM1
END_PROCEDURE
DTR> :NUM2↵

ONE, TWO, THREE

ONE, TWO, THREE
ONE, TWO, THREE, FOUR, FIVE

ONE, TWO, THREE

ONE, TWO, THREE
ONE, TWO, THREE, FOUR, FIVE


ONE, TWO, THREE

ONE, TWO, THREE, FOUR, FIVE

DTR>
```

5.46 REPORT Statement

Function

Invokes the Report Writer and is the first entry in a report specification. You specify the data to go in the report and the output device for the report. You specify attributes of the report with Report Writer statements, listed in Table 5-19 and described in this section, and end the report specification with the END_REPORT statement.

Format

```
REPORT  [rse]   ON    file-spec
                      *.prompt
```

Arguments

rse

Specifies the data for the report. You can report on readied domains, collections, and lists.

ON

Specifies that the report is to be written to a file or displayed on a terminal. When you specify ON, the Report Writer inserts a form feed at the end of the report. If you specify ON when displaying a report on a video terminal, the last page of the report scrolls off the screen.

file-spec

Is the file to which you want to write the report. The file specification must have this format:

For RSTS/E systems: dev:[PPN]file-name.type

For RSX systems: dev:[UIC]file-name.type;ver

You must specify at least one field in the file specification. On RSTS/E systems, you must specify a file name. DATATRIEVE uses the following defaults for fields not specified:

Field	Default
dev:	SY: (the system device)
[UIC/PPN]	Your default UIC/PPN
file name	Null string
.type	.LST
;ver	1 or next higher version (non-RSTS/E systems only)

*.prompt

Is a prompting value expression that allows you to specify a file specification when DATATRIEVE processes the report specification.

Restrictions

- If you specify another terminal as the device to display your report, that terminal cannot be logged in to the system or attached by another user.
- To send your report to a tape drive, you must mount a tape and allocate the tape drive before specifying the tape drive in the REPORT statement.

Results

- DATATRIEVE writes the report to the specified file and prompts for Report Writer statements with the RW> prompt. Table 5-19 describes Report Writer statements. To end the report specification, use the END_REPORT statement.

Table 5-19: Summary of Report Writer Statements

Statement	Function
AT BOTTOM	Specifies the value, position, and format of the print objects in the header lines and summary lines displayed at the bottom of reports, report pages, and control groups
AT TOP	Specifies the value, position, and format of the print objects in the header lines and summary lines displayed at the top of reports, report pages, and control groups
END_REPORT	Ends a report specification
PRINT	Controls the content, format, and column headers for the detail lines in a report
SET	Controls the report header (name, date, format of data, and page numbering), and defines the size of report pages and the length of the report

- When you omit the ON clause in a REPORT statement, DATATRIEVE displays the report on your terminal.
- When you omit the RSE, the Report Writer uses the data in your current collection for the report. If there is no current collection, DATATRIEVE displays the error message "A current collection has not been established. "

Usage Notes

- You can write a report to a file in any directory to which you have write access.
- To report on sorted records, form a sorted collection and specify the name of the collection in the REPORT statement. If you specify a SORTED BY clause in an RSE in the REPORT statement, DATATRIEVE-11 may display the error "Sort work space exhausted" without creating the report.

- You can report on data only in domains readied for READ, WRITE, or MODIFY access. Because you cannot establish collections or record streams in domains readied for extend access, you cannot report on data in domains readied for extend access.
- The data you report must be contained in the current collection or in the record stream established by the RSE in the REPORT statement.
- If you make a typing error or syntax error in a Report Writer statement, DATATRIEVE will display an error message and return you to DATATRIEVE command level. If you have entered the Report Writer statements interactively in response to the RW> prompt, you must retype all the statements again to correct the error. To avoid having to retype Report Writer statements, create a command file or define a procedure that contains your Report Writer statements. If DATATRIEVE detects typing or syntax errors in Report Writer statements in a command file or procedure, you can simply edit the command file or procedure and correct the errors without retyping the entire sequence of statements.

Examples

Write a report to display on your terminal:

```
DTR> SET NO PROMPT(RET)
DTR> READY YACHTS(RET)
DTR> REPORT YACHTS WITH BUILDER = "ALBIN"(RET)
RW>  SET REPORT_NAME = "YACHTS BY ALBIN"(RET)
RW>  PRINT BOAT(RET)
RW>  END_REPORT(RET)
DTR>
```

Define a procedure that writes a report on a group of yachts to a file in your default directory:

```
DTR> SET NO PROMPT(RET)
DTR> READY YACHTS(RET)
DTR> SHOW YACHT_REPORT(RET)
PROCEDURE YACHT_REPORT
     DECLARE B PIC X(10),
     REPORT YACHTS WITH BUILDER = *,"Builder to report on"ON *,"file name"
     SET REPORT_NAME = *,"Report name"
     SET COLUMNS_PAGE = 70
     PRINT BOAT
     AT BOTTOM OF REPORT PRINT SKIP, COL 10,
             "BOAT COUNT:", SPACE, COUNT (-) USING Z9,
             COL 45, "AVERAGE PRICE:", AVERAGE PRICE
     END_REPORT
END_PROCEDURE
DTR> :YACHT_REPORT(RET)
Enter Report name: "YACHTS BY GRAMPIAN"(RET)
Enter Builder to report on: GRAMPIAN(RET)
Enter file name: GRAMPIAN.DAT(RET)
DTR>
```

The data file produced by this report specification looks like this:

```
                          YACHTS BY GRAMPIAN                    15-Feb-83
                                                                Page 1

                                                 LENGTH
                                                  OVER
MANUFACTURER        MODEL          RIG            ALL       WEIGHT   BEAM    PRICE

 GRAMPIAN           2-34           SLOOP          34        11,800    10    $29,675
 GRAMPIAN           26             SLOOP          26         5,600    08    $11,495
 GRAMPIAN           28             SLOOP          28         6,900    10    $14,475
 GRAMPIAN           30             SLOOP          30         8,600    09    $17,775
 GRAMPIAN           34             KETCH          33        12,000    10    $29,675

       BOAT COUNT:  5                                AVERAGE PRICE:      $20,619
```

See the *DATATRIEVE*-11 Guide to Writing Reports for more information on writing reports and for more examples.

5.46.1 AT BOTTOM Statement (Report Writer)

Function

Specifies the value, position, and format of the print objects in the header lines and summary lines displayed at the bottom of reports, report pages, and control groups (groups of sorted records with the same value in one or more fields).

Format

```
AT BOTTOM OF { REPORT    }   PRINT summary-element [,...]
             { PAGE      }
             { field-name }
```

Arguments

REPORT

Displays the summary line at the bottom of the last page of the report.

PAGE

Displays the summary line at the bottom of each page of the report.

field-name

Establishes a pattern of control breaks for the entire report, dividing the report into groups of records with a common value for the field. Displays the summary line at the bottom of the control group for which the field name is the sort key.

summary-element

Specifies the value, position, and format of the fields. Table 5-20 summarizes the summary elements that can be used with AT BOTTOM statements.

Table 5-20: Report Writer AT BOTTOM Statement: Summary Elements

Summary Element	Function	Usage Notes
AVERAGE	Displays the average value of the value expressions	Calculated for the detail lines of the report, page, or group specified
COL n	Specifies where the output of the next element begins	Same usage as in the Report Writer PRINT statement
COUNT	Displays the number Calculated for of detail lines the detail lines of the report, page, or group specified	Displays the number Calculated for of detail lines the detail lines of the report, page, or group specified
field-name [modifier]	Prints the common value for a control group	Same usage as in the Report Writer PRINT statement
MAX [value-expression]	Displays the maximum value of the value expressions	Calculated for the detail lines of the report, page, or group specified
MIN [value-expression]	Displays the minimum value of the value expressions	Calculated for the detail lines of the report, page, or group specified
NEW_PAGE	Starts a new page before the output of the next summary element	Same usage as in the Report Writer PRINT statement
NEW_SECTION	Starts a new page and a new sequence of page numbers, beginning with page one	Can be used only in AT TOP and AT BOTTOM statements
SKIP [n]	Prints the next element n lines from the current line	Same usage as in the Report Writer PRINT statement
SPACE [n]	Inserts n spaces between the output of the preceding and following elements	Same usage as in the Report Writer PRINT statement
TAB [n]	Inserts the space of n tab characters before the output of the next element	Same usage as in the Report Writer PRINT statement
TOTAL [value-expression]	Displays the total of values for the expressions	Calculated for the detail lines of the report, page, or group specified
value-expression	Displays the value in a summary line	Same usage as in the Report Writer PRINT statement

Restrictions

- You can use the AT BOTTOM statement only with sort keys. If you specify a field not included in the sort order, the Report Writer displays an error message and does not produce the report.
- Do not specify AT BOTTOM OF PAGE PRINT NEW_PAGE.

Results

- The AT BOTTOM statement calculates and summarizes information contained in the records of the page, group, or report specified in the statement.
- When you specify AT BOTTOM OF field-name PRINT field-name, the Report Writer prints the value in the specified field of the last detail line in the control group.

Usage Notes

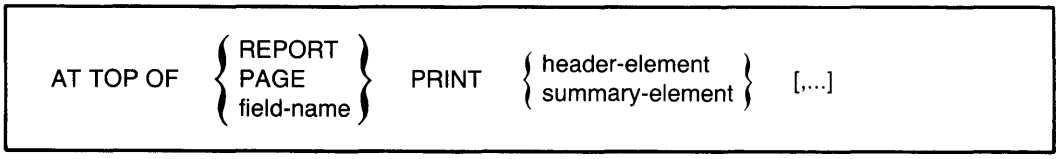
- DATATRIEVE checks the sort order of the collection or record stream you want to report. You can establish a sort order with the FIND, SORT, or REPORT statements.
- If you specify AT BOTTOM OF field-name without having sorted the records, the Report Writer divides the detail lines into control groups anyway. A new control group is formed every time the value in the specified field changes. If no values in that field are repeated in consecutive detail lines, the Report Writer treats each line as a separate control group and prints the header and summary lines above and below each line as indicated in the record specification.
- If the records are already sorted according to an indexed key and there is only one level of control break, the Report Writer divides the control groups at the appropriate places. If you want multiple levels of control groups, you must sort the records in advance with a FIND, SORT or REPORT statement.

5.46.2 AT TOP statement (Report Writer)

Function

Specifies the value, position, and format of the print objects in the header lines and summary lines displayed at the top of reports, report pages, and control groups (groups of sorted records with the same value in one or more fields).

Format



Arguments

REPORT

Displays the header line or summary line above the detail lines on the first page of the report and suppresses the report header on the first page of the report. The report header is displayed on the following pages of the report, and the page numbers begin with Page 1 on the second physical page of the report. To specify a title page for your report, end the print list with a NEW_PAGE or NEW_SECTION statement.

PAGE

Displays the header line or summary line at the top of each page of the report, and replaces the report header on every page.

field-name

Establishes a pattern of control breaks for the entire report, dividing the report into groups of records with a common value for the field. Displays the header line or summary line at the top of the control group for which the field name is the sort key.

header-element
summary-element

Specifies the value, position, and format of the fields. Table 5-21 summarizes the header and summary elements that can be used with AT TOP statements.

Table 5-21: Report Writer AT TOP Statement: Header and Summary Elements

Header or Summary Element	Function	Usage Notes
AVERAGE	Displays the average value of the value expressions in the record stream or the current collection	Does not indicate the average for the detail lines of the report, page, or control group specified
COL n	Specifies where the output of the next header or summary element begins	Same usage as in the Report Writer PRINT statement
COLUMN_HEADER	Displays the column headers defined by the PRINT statement and AT statements	Overrides the suppression of column headers for AT TOP OF REPORT or PAGE
COUNT	Displays the number of records in the record stream or current collection	Does not indicate the number of detail lines of the report, page, or control group specified
field-name [modifier]	Prints the common field value at the top of each control group	Same usage as in the Report Writer PRINT statement

(continued on next page)

Table 5-21: Report Writer AT TOP Statement: Header and Summary Elements (Cont.)

Header or Summary Element	Function	Usage Notes
MAX [value-expression]	Displays the maximum value of all value expressions in the record stream or current collection	Does not indicate the maximum values for the detail lines of the report, page, or control group specified
MIN [value-expression]	Displays the minimum value of all value expressions in the record stream or current collection	Does not indicate the minimum value for the detail lines of the report, page, or control group specified
NEW_PAGE	Starts a new page for the report	Same usage as in the Report Writer PRINT statement
NEW_SECTION	Starts a new page and a new section numbered as page one	Use with AT TOP OF field-name to produce a new section for each group
REPORT_HEADER	Displays the report header, including the report name, date, and page number	Overrides the suppression of a report header in AT TOP OF REPORT or PAGE
SKIP [n]	Prints the next header or summary element n lines from the current line	Same usage as in the Report Writer PRINT statement
SPACE [n]	Inserts n spaces between the output of the preceding and following elements	Same usage as in the Report Writer PRINT statement
TAB [n]	Inserts the space of n tab characters before the output of the next element	Same usage as in the Report Writer PRINT statement
TOTAL [value-expression]	Displays the total for the value expressions in the record stream or current collection	Does not indicate the total for the detail lines of the report, page, or control group specified
value-expression	Displays the value in a header or summary line	Same usage as in the Report Writer PRINT statement

Restrictions

- You can use the AT TOP statement only with sort keys. If you specify a field not included in the sort order, the Report Writer displays an error message and does not produce the report.
- Do not specify AT TOP OF PAGE PRINT NEW_PAGE.

Result

An AT TOP statement summarizes information for the entire group of records in the record stream or in the current collection, not the records of the page, group, or report specified in the statement.

Usage Notes

- The header and summary elements may contain modifiers to specify edit strings or to suppress headers.
- DATATRIEVE checks the sort order of the collection or record stream you want to report. You can establish a sort order with the FIND, SORT, or REPORT statements.
- If you specify AT TOP OF field-name without having sorted the records, the Report Writer displays a warning message and divides the detail lines into control groups anyway. A new control group is formed every time the value in the specified field changes.

5.46.3 END_REPORT Statement (Report Writer)

Function

Ends a report specification.

Format

```
END_REPORT
```

Usage Notes

- The END_REPORT statement must be the last statement in the report specification.
- Following the END_REPORT statement, the Report Writer takes one of the following courses of action:
  - Prompts you for any values you specified with the *.prompt in the record specification and then produces the report.
  - Produces the report and sends it to the device or file you have specified in the REPORT command.

5.46.4 PRINT Statement (Report Writer)

Function

Specifies the following characteristics of the detail lines in a report:

- The content of the detail lines: field values, other values, and text strings
- The format of fields in the detail lines: order, column position, and edit string to be be used for print items
- The column headers for the print items in the detail line

Format

```
PRINT print-list-element [,...]
```

Argument

print-list-element

Specifies values, position, and format of the print items in the detail line. Table 5-22 indicates the parameters of the report controlled by various print list elements and modifiers.

Restrictions

- You can include only one PRINT statement in a report specification. If the report specification contains no AT statements, it must contain a PRINT statement. If your report specification contains an AT statement, then it does not have to contain a PRINT statement.
- Unlike the DATATRIEVE command level PRINT statement, the Report Writer PRINT statement must be followed by at least one print list element. If you do not specify a print list element, the Report Writer prompts for one.

Result

The PRINT statement formats the detail lines of a report according to the print list elements and modifiers specified. Table 5-22 lists and describes the print list elements you can use with the Report Writer PRINT statement. Table 5-23 lists and describes print list modifiers.

Table 5-22: Report Writer Print List Elements

Print List Element	Function
Content of Detail Line:	
field-name [modifier]	Can include elementary, group, list, REDEFINES, or COMPUTED BY fields; to print all fields, specify the top-level field name
value-expression [modifier]	Can be a related value expression derived from field values using arithmetic operators or a literal or variable
Format of a Detail Line:	
COL n	Specifies where the output of the next print list element begins
TAB [n]	Inserts the space of n tab characters before the output of the next print list element
SKIP [n]	Begins printing the next print list element n lines from the current line
SPACE [n]	Inserts n spaces between the output of the preceding and following print list elements
Beginning of New Page:	
NEW_PAGE	Causes the Report Writer to start a new report page

Table 5-23: Report Writer Print List Modifiers

Print List Modifier	Function
Column Headers for Print Items:	
("header-segment"[/...])	Specifies one or two character string literals to be displayed as headers for the preceding field or value expression, overriding the field name or query header from the field definition. The entire modifier must be enclosed in parentheses and must immediately follow its associated field name or value expression.
(–)	Suppresses the printing of the query header or field name associated with the field in its record definition
Format of the Detail Line Item:	
USING edit-string	Imposes the characteristics of the specified edit string on the preceding field or value expression. The edit string must conform to the rules for the EDIT_STRING clause.

Usage Notes

- If the value specified in the COL list element is too small to accommodate all the fields, the Report Writer carries the overflow fields onto the next line. The Report Writer does not split fields between lines, but the column headers of the overflow fields may be lost.
- You can list print list elements and their modifiers in any order you choose.
- When the data you are reporting includes a list, use an inner print list element (ALL [print-list] of rse) in the PRINT statement to specify the value, position, and format for fields in the list. Each item of the list takes at least one physical line of printing.
- If you do not specify positions or edit strings for any of the fields in a detail line, the Report Writer determines the format for those fields using these criteria:
  - The edit string in the field definition, if present, determines the format for the field.
  - If the field definition has no edit string, the PICTURE clause determines the format for the field.
  - If the field definition has neither an edit string nor a PICTURE clause, the Report Writer invents a picture clause to accommodate the data in the field.

To gain full control over the formats of the fields of your detail lines, explicitly define edit strings with the USING edit string modifier.

5.46.5 SET Statement (Report Writer)

Function

Controls the report header and defines the size of report pages and the length of the report. With Report Writer SET statements, you can specify:

- The report header: the report name, the date, the format of data, and page numbering
- The size of report pages: the number of columns and the number of lines per page
- The length of the report: the maximum number of lines and the maximum number of pages

Format

For naming the report:

```
SET    REPORT_NAME =   { "string"[/...] }
                       { *.prompt       }
```

For specifying a date or string at the upper right of each page:

```
SET    DATE = "string"
```

For controlling the printing of a date or page number:

```
SET    { NO DATE   }
       { NUMBER    }   [,...]
       { NO NUMBER }
```

For specifying page width or length, or overall report length:

```
SET    { { COLUMNS_PAGE = }          { n        }
       { { LINES_PAGE =   }          { *.prompt }   [,...]
       { { MAX_LINES =    }
       { { MAX_PAGES =    }
```

Arguments

Table 5-24 summarizes information about each type of Report Writer SET statement:

- The first three columns indicate the argument, the function of the statement, and the Report Writer's default setting if the statement is not used.
- The "Prompt Option" column indicates whether or not you can specify a prompting value expression with a form of the SET statement. If you include a prompt, the Report Writer prompts you for a value when it processes the report specification.
- The "Maximum Value" column indicates the largest value you can specify for a form of the SET statement.

Table 5-24: Report Writer SET Statement Arguments

Argument	Function	Default	Prompt Option	Maximum Value
REPORT_NAME	Specifies a name for the report and centers the name on the first line of each page	No report name	Yes	–
DATE	Specifies a date or string and prints it on the upper right line of each page; NO DATE suppresses the printing of the current date	Current system date	No	–
(NO) NUMBER	Causes the Report Writer to print a page number below the date; NO NUMBER suppresses the printing of page numbers	Current page number	No	–
COLUMNS_PAGE	Specifies the page width in columns	Current terminal width or 80 columns	Yes	255
LINES_PAGE	Specifies the page length in lines	60 lines	Yes	32,767
MAX_LINES	Specifies the maximum lines for the report	No limit	Yes	32,767
MAX_PAGES	Specifies the maximum pages for the report	No limit	Yes	32,767

Examples

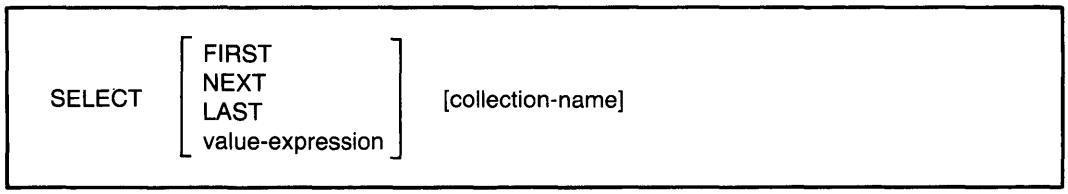
Many examples of report specifications and reports may be found in the *DATATRIEVE-11 Guide to Writing Reports.*

5.47 SELECT Statement

Function

Establishes a selected record for a collection.

Format



Arguments

FIRST

Selects the first record in the target collection.

NEXT

Selects the next record in the target collection. When you omit a position specification, NEXT is the default.

LAST

Selects the last record in the target collection.

value-expression

Evaluates to a positive number; DATATRIEVE uses the integer part of the number to select the record with that position number in the collection.

collection-name

Is the name of the target collection containing the record to be selected. If omitted, the target collection is the CURRENT collection.

Restrictions

- You must establish the target collection with a FIND statement before you can use the SELECT statement.
- The target collection cannot be empty.
- The SELECT statement can move the collection cursor (which points to the selected record in a collection) to the position of a record that has been dropped, but you cannot retrieve any data from the record that occupied that position before you dropped it. You must form a new collection or record stream containing that record to retrieve its data.
- Do not use a SELECT statement in FOR, REPEAT, or WHILE statements.

- If you specify a value expression, it must evaluate to a positive number greater than or equal to one, and less than or equal to the size of the largest possible collection. The size of the largest possible collection is limited by the amount of DATATRIEVE work space available.
- If the integer portion of the evaluated value expression exceeds the number of records in the collection, DATATRIEVE displays the error "Record number out of range for collection."
- You cannot use a SELECT statement within a compound statement.

Results

- The SELECT statement establishes a selected record in the target collection and, thus, establishes a single record context for one record in that collection.
- The record DATATRIEVE selects depends on the arguments you supply, the content of the target collection, and the existence and position of a previously selected record in the collection.
- When you omit the argument that determines the position of the selected record, DATATRIEVE responds as though you had entered SELECT NEXT:
  - If you have not established a selected record in a collection and you enter SELECT NEXT or SELECT with no argument, the first record in the collection becomes the selected record.
  - If the collection cursor (which points to the selected record in a collection) points to the last record in the collection and you specify SELECT NEXT or SELECT with no argument, DATATRIEVE displays an error message on your terminal, and retains the last record in the collection as the selected record.
  - If the selected record in the collection is neither first nor last and you specify SELECT NEXT or SELECT with no argument, the next record in the collection becomes the selected record.
- If the record identified by the SELECT statement has been dropped from the collection with a DROP statement, DATATRIEVE displays a message on your terminal and moves the collection cursor to the position of the dropped record.
- If the target collection is not the CURRENT collection, selecting a record from it does not make it the CURRENT collection.
- When you specify a value expression in a SELECT statement, the positive integer to which the expression evaluates is the position number of the **one** selected record in the collection. The integer does not designate the number of records selected. For example, if you type SELECT 5, you select the fifth record in the target collection, not five records from the collection.
- If you select a record from the CURRENT collection and then select a record from another collection, the CURRENT collection and its selected record remain unchanged.

Usage Notes

- Use the SELECT statement to establish a single record context for a record in a collection. Having a selected record allows you to retrieve and compare values in the fields of a selected record without specifying a target record stream. When referring to fields of records in a single record context, you can frequently use the field names without field name qualifiers, almost as though they were variables.

- When you use a field name by itself in a value expression, its value is retrieved from the "nearest" selected record with a field of that name:

  – DATATRIEVE establishes the "nearness" of selected records based on the order in which you created the collections with the FIND statement. The most recently created collection is the nearest.

  – The order in which you select records has no effect on relative nearness of selected records.

- Use a SELECT statement to establish a target record for the ERASE and MODIFY statements and to establish a target record for PRINT statements in which you include only the print list.

- To show the position of the selected record in the target collection, use the SHOW command. This command prints the name of the collection, the name of the domain from which the collection is derived, the number of records in the collection, the names of fields used to establish the sort order of the collection, and the position number of the selected record in the target collection. It also tells you if the selected record has been dropped from the collection by a previous DROP statement.

- To show the name and attributes of the CURRENT collection, use the SHOW CURRENT command.

- If you use the SELECT statement to establish a selected record for the current collection, you need type only PRINT and press RETURN to display that selected record. If the CURRENT collection has no selected record, DATATRIEVE displays on your terminal the selected record from the most recently established named collection that has a selected record.

- If no existing collection has a selected record and you enter a PRINT statement without a print list, DATATRIEVE displays a message on your terminal and displays the entire current collection.

- To display all fields of the selected record of a named collection that is not the CURRENT collection, you must provide a properly qualified top-level field name in the print list of a PRINT statement:
  - If you created the collections from the same domain, use the collection name as the qualifier. For example, a collection of YACHTS called BIGGIES is not the CURRENT collection, and the CURRENT collection has a selected record. To display the selected record in BIGGIES, type PRINT BIGGIES.BOAT and press RETURN.
  - If you created the collections from different domains, you do not have to qualify the top-level field names unless they are the same. If the top-level field names of the different domains are the same, you can use the domain name to qualify the top-level field names.

- You can refer to the fields of the selected record as though they were variables. DATATRIEVE resolves an unqualified field name to the most recently formed collection with a selected record containing a field with that name. To refer to a field name beyond the most recently formed collection with a selected record containing that same name, you must provide a suitable qualifier to establish the appropriate context. Use the name of the collection containing the target record as the qualifier.
- To distinguish between two or more selected records referred to in one complex DATATRIEVE statement, use context variables. Context variables can be particularly useful if you derived the collections from the same domain and the field names of the collections in question are identical.
- If you want to perform the same set of statements on each record in the CURRENT collection, **do not** use a SELECT statement in a BEGIN-END block inside a REPEAT statement. Use the following method to loop through the CURRENT collection:

```
DTR> SET NO PROMPT(RET)
DTR> FOR CURRENT(RET)
DFN> BEGIN(RET)
          .
          .
          .
DFN> END(RET)
DTR>
```

- See the *DATATRIEVE-11 Interactive User's Guide* for a discussion of name recognition and single record contexts in DATATRIEVE.

Examples

Select the last record in the current collection:

```
DTR> SET NO PROMPT(RET)
DTR> READY YACHTS(RET)
DTR> FIND FIRST 5 YACHTS(RET)
[5 records found]
DTR> PRINT ALL(RET)
```

(continued on next page)

```
                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG       ALL    WEIGHT BEAM  PRICE

 ALBERG       37 MK II    KETCH     37     20,000  12   $36,951
 ALBIN        79          SLOOP     26      4,200  10   $17,900
 ALBIN        BALLAD      SLOOP     30      7,276  10   $27,500
 ALBIN        VEGA        SLOOP     27      5,070  08   $18,600
 AMERICAN     26          SLOOP     26      4,000

DTR> SELECT LAST(RET)
DTR> PRINT(RET)

                                    LENGTH
                                     OVER
MANUFACTURER    MODEL      RIG       ALL    WEIGHT BEAM  PRICE

 AMERICAN     26          SLOOP     26      4,000  08    $9,895

DTR>
```

Select the fifth record in the collection BIG_ONES:

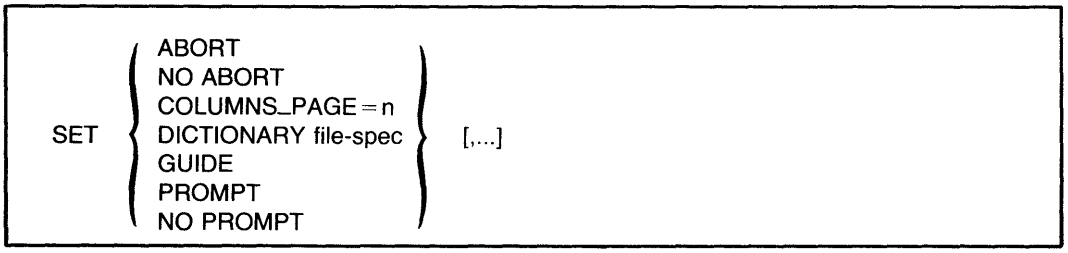
```
DTR> SELECT 5 BIG_ONES(RET)
DTR>
```

5.48 SET Command

Function

- Controls DATATRIEVE's response to ABORT statements
- Sets the maximum number of columns per page for DATATRIEVE output
- Establishes your current dictionary
- Starts DATATRIEVE Guide Mode
- Starts or stops automatic syntax prompting for continued statements and commands

Format



Arguments

ABORT

Causes DATATRIEVE to abort the remainder of a procedure or command file when DATATRIEVE executes an ABORT statement, when you enter a CTRL/Z to a prompt, or when a syntax or logical error occurs during the execution of a command or statement (except the DELETE command).

NO ABORT

Causes DATATRIEVE, when processing a procedure or a command file, to abort only the one statement containing an ABORT statement it executes. SET NO ABORT also causes DATATRIEVE to take the same action when you respond with a CTRL/Z to a prompt in a procedure or command file. DATATRIEVE then executes the next command or statement in the procedure or command file. SET NO ABORT is in effect when you start a DATATRIEVE session.

COLUMNS_PAGE = n

Establishes the number of columns per page for DATATRIEVE output and the default page width for the Report Writer. When you start your session, the default COLUMNS_PAGE setting is 80.

DICTIONARY [file-spec]

Causes DATATRIEVE to set your current dictionary to the file specified in the command. When you start your DATATRIEVE session your current dictionary is your default dictionary. SET DICTIONARY without a file specification returns you to your default dictionary.

The file specification must have this format:

For RSTS/E systems: dev:[PPN]file-name.type

For RSX systems: dev:[UIC]file-name.type;ver

You must specify at least one field in the file specification. DATATRIEVE uses the following defaults for fields not specified:

Field	Default
dev:	SY: (the system device)
[UIC/PPN]	Your default UIC/PPN
file name	QUERY
.type	.DIC
;ver	1 or next higher version (non-RSTS/E systems only)

GUIDE

Starts Guide Mode, the tutorial mode of DATATRIEVE. Refer to the *Introduction to DATATRIEVE-11* manual for a description of Guide Mode.

PROMPT

Causes DATATRIEVE to prompt for elements needed to complete the syntax of the current command or statement. When you press RETURN before completing a command or statement, DATATRIEVE prompts you for the next syntactic element of that statement or command. The prompt takes the form of [Looking for element]. At the start of a DATATRIEVE session, SET PROMPT is in effect.

NO PROMPT

Stops DATATRIEVE from prompting for elements needed to complete the syntax of the current command or statement.

Restrictions

- You must enter SET commands at DATATRIEVE command level, indicated by the DTR> prompt.
- You cannot use SET commands in compound statements (THEN, IF-THEN-ELSE, and BEGIN-END) or in FOR, REPEAT, or WHILE statements.
- In the SET COLUMNS_PAGE command, the argument n must be an unsigned, nonzero integer less than or equal to 255.

Results

- When SET ABORT is in effect and DATATRIEVE is executing a procedure or command file, DATATRIEVE aborts the entire procedure or command file if it executes an ABORT statement or you enter a CTRL/Z in response to a prompt.
- When SET NO ABORT is in effect and DATATRIEVE is executing a procedure or command file, DATATRIEVE aborts only the statement containing the ABORT statement or prompt and executes the next statement in the procedure or command file.
- With the SET COLUMNS_PAGE command, you can affect the output of a PRINT statement that contains no implicit line feeds, that is, no SKIP, no NEW_PAGE, or no COL n with n less than the column reserved for previous print list elements.

  If an element in a print list would extend beyond the right column limit determined by the value of the COLUMNS_PAGE setting, DATATRIEVE shifts that element to the next line. By adjusting the COLUMNS_PAGE setting, you can control the way DATATRIEVE breaks the detail lines of its output.
- If you use a partial file specification in the SET DICTIONARY command, DATATRIEVE uses your default dictionary to supply the missing part of the file specification.
- If a SET DICTIONARY command fails because the specified dictionary file does not exist or because you do not have adequate privileges to it, your current dictionary does not change.
- SET GUIDE on the VT100 family of terminals sets the scrolling attribute of your terminal to smooth scroll. When you leave Guide Mode, DATATRIEVE does not reset the scrolling attribute.

Usage Note

To display the name of your current dictionary, use the SHOW DICTIONARY command.

Example

Set your current dictionary to NEWDIC.DIC, use a command to verify the change, return to your default dictionary with SET DICTIONARY, and verify the current dictionary.

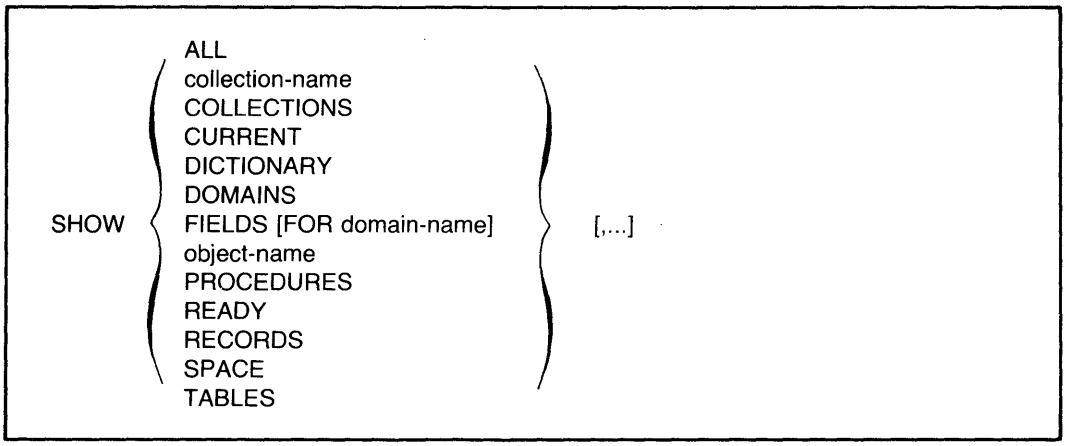
```
DTR> SET DICTIONARY NEWDIC(RET)
DTR> SHOW DICTIONARY(RET)
The current dictionary is DB2:[56,23]NEWDIC.DIC;1
DTR> SET DICTIONARY(RET)
DTR> SHOW DICTIONARY(RET)
The current dictionary is LB:[1,2]QUERY.DIC;1
DTR>
```

5.49 SHOW Command

Function

Displays information about your current data dictionary and its contents.

Format



Arguments

ALL

Displays the names of all the objects catalogued in your current dictionary, the name of your current dictionary, the names of established collections, the readied domains, the loaded tables, and tables in your workspace.

collection-name

Displays the collection name, the name of the domain within which the collection has been established, the number of records in the collection, the status of the selected record within the collection, and the names of the keys on which the collection has been sorted.

COLLECTIONS

Displays the names of the collections in your workspace.

CURRENT

Displays the name of the domain within which the CURRENT collection has been formed, the number of records in the CURRENT collection, the status of the selected record in the CURRENT collection, and the names of the keys on which the collection has been sorted.

DICTIONARY

Displays the full file specification of your current dictionary.

DOMAINS

Displays the names of all domains cataloged in your current dictionary.

FIELDS [FOR domain-name]

Displays the names, data types, and index key information of the fields of all readied domains, or the domain specified in the FOR clause. The SHOW FIELDS command also displays the names and data types of global variables.

object-name

Displays the source text of a domain, record, procedure, or table definition specified by the object name.

PROCEDURES

Displays the names of all procedures cataloged in your current dictionary.

READY

Displays for each readied domain the file type of the associated data file, the access control option, and the access mode.

RECORDS

Displays the names of all record definitions cataloged in your current dictionary.

SPACE

Displays information on DATATRIEVE pool space and current memory usage. See the *DATATRIEVE-11 User's Guide* for information on using the SHOW SPACE command to optimize DATATRIEVE.

TABLES

Displays the names of all loaded tables and dictionary tables cataloged in your current dictionary.

Restrictions

- You must have R (read) access to a dictionary object before you can display it with the SHOW object-name command.
- You can display objects only in your current dictionary, not in other dictionary files.

Results

- DATATRIEVE displays information, in the order requested, on your terminal.
- You need only R (read) access to the objects in your current dictionary to see their names when you enter a SHOW command with one of these options: ALL, DOMAINS, RECORDS, PROCEDURES, TABLES, and DICTIONARY.

- If you do not have the access privileges needed to obtain information for the SHOW object-name command, DATATRIEVE displays an error message on your terminal and returns you to DATATRIEVE command level.
- When you have more than one existing collection and you enter a SHOW COLLECTIONS command, the order in which DATATRIEVE displays the collections reverses the order in which you established them. The CURRENT collection is always at the top of the list, and the "oldest" existing collection is always at the bottom of the list.

  Knowing this order of the collections is useful because you can see the order in which DATATRIEVE searches for a selected record to establish a single record context for the MODIFY, PRINT, ERASE, and DISPLAY statements and resolve field names in value expressions.

Examples

Display the names of the objects in your current dictionary, db2:[56,34]NEWDIC.DIC:

```
DTR> READY YACHTS↵
DTR> READY FAMILIES↵
DTR> READY AUTOMOBILES↵
DTR> SHOW ALL↵
Domains:
        AUTOMOBILES     D1              EDIT_TEST       EMP_REVIEW
        FAMILIES        OWNERS          PERSONNEL       YACHTS
Records:
        AUTOMOBILES_REC D1REC           EDIT_REC        EMP_REV_REC
        FAMILY_REC      OWNER_REC       PART_REC        PERSONNEL_REC
        YACHT
Procedures:
        A               FOO             FOO1            IF1
        NUM1            NUM2            SELL_BOAT
Tables:
The current dictionary is DBA2:[56,34]NEWDIC.DIC;1
No established collections
Ready domains:
        YACHTS: RMS INDEXED, PROTECTED READ
        PERSONNEL: RMS SEQUENTIAL, PROTECTED READ
DTR>
```

Display the record definition OWNER_RECORD:

```
DTR> SHOW OWNER-RECORD↵
RECORD OWNER_RECORD
01 OWNER.
   03 NAME PIC X(10) QUERY_HEADER IS "OWNER"/"NAME"
      EDIT_STRING IS X(5),
   03 BOAT_NAME PIC X(17) QUERY_HEADER IS "BOAT NAME",
   03 TYPE.
      06 BUILDER PIC X(10),
      06 MODEL PIC X(10),
;
DTR>
```

5.50 SHOWP Command

Function

Displays the access control list (ACL) of an object in your current dictionary on your terminal:

Format

Arguments

object-name

Is the name of the dictionary object whose ACL you want to display on your terminal.

(passwd)
(*)

Is the password necessary to gain C (control) access to the domain. The parentheses are required. Use the asterisk to have DATATRIEVE prompt for the password. If you omit this argument, DATATRIEVE uses your UIC/PPN to verify that you have C (control) access privilege.

Restriction

You must have C (control) access privilege to the object before you can display its ACL.

Result

DATATRIEVE displays the entire ACL for the specified dictionary object on your terminal.

Usage Notes

- Use the SHOWP command to verify the sequence number of the entry you want to delete before you use the DELETEP command to remove the entry from an access control list.
- Refer to the *DATATRIEVE-11 User's Guide* for information on protection and access control lists.

Examples

Display the access control list for the YACHTS domain:

```
DTR> SHOWP YACHTS(RET)
  1,UIC, [32,56], "RWMEC"

DTR>
```

Use a password (CEP) to gain C (CONTROL) access to the record YACHT and display its access control list:

```
DTR> SHOWP YACHT (*)(RET)
Enter password for YACHT:       (RET)
DTR> SHOWP YACHT(RET)
     1,UIC, [23,45], "RWMEC"
     2,UIC, [34,64], "RW"

DTR>
```

5.51 SIGN Clause

Function

Specifies the location and representation of a sign (+ or –) in a numeric elementary field in a record definition.

Format

```
SIGN [IS]  { LEADING }  [SEPARATE]
           { TRAILING }
```

Arguments

LEADING
TRAILING

Indicates that the sign is at the left (LEADING) or right (TRAILING) of the field value.

SEPARATE

Indicates that the sign occupies its own character position in the field. If this argument is omitted, the sign shares a character position with the field's leftmost (if LEADING) or rightmost (if TRAILING) digit.

Restrictions

- This clause can be used only with numeric elementary fields.
- A field definition cannot contain both a SIGN and a USAGE clause.

Results

- If you do not include a SIGN clause with a numeric field, the sign shares a character position with the field's rightmost digit. You must include this clause if a program written in COBOL (or other language) uses the record and requires that the sign be a separate character or share the leftmost character position:
- A sign clause does not affect the output format of a field.

Examples

Define the field CURRENT_BALANCE as a six-digit signed field, with the sign sharing the leftmost character position:

```
03 CURRENT_BALANCE
        PIC IS S9999V99
        EDIT_STRING IS $$$$9.99-
        SIGN IS LEADING.
```

Define the field NEW_PRICE as a four-digit signed field. The sign is a separate character in the rightmost character position.

```
03 NEW_PRICE PIC S99V99
       SIGN TRAILING SEPARATE
       EDIT_STRING +++.99.
```

5.52 SORT Statement

Function

Arranges a DATATRIEVE collection according to the order you specify for the contents of one or more fields in the records.

Format

```
SORT [collection-name] [BY] sort-key-1 [,...]
```

Arguments

collection-name

Is the name of the collection to be sorted.

BY

Is an optional language element you can use to clarify syntax.

sort-key

Is a field to be used for the sort. If you specify more than one sort key, use a comma to separate each sort key from the next. The sort key can be preceded or followed by a key word that determines the order in which DATATRIEVE sorts the records in the collection:

ASC[ENDING]
DESC[ENDING]
INCREASING
DECREASING

ASCENDING is the default order.

Restrictions

- You can use the SORT statement only to sort a collection already formed with a FIND statement.
- Do not use a FIND and a SORT statement in the same compound statement.
- Do not use a SORT statement in a FOR, WHILE, or REPEAT statement.

Results

- If you omit the collection name, DATATRIEVE sorts the current collection.
- If you specify ASC[ENDING] or INCREASING in the sort key, DATATRIEVE puts the record with the lowest value in the specified field first in the collection and the one with the highest value last.

- If you specify DESC[ENDING] or DECREASING, DATATRIEVE puts the record with the highest value in the specified field first in the collection and the one with the lowest value last.
- DATATRIEVE sorts the collection according to the order and fields specified:
  - If you specify more than one sort key, DATATRIEVE uses the first field name as the major sort key and each successive field name as an increasingly minor key.
  - If, in the first sort key, you omit a key word specifying the sort order, DATATRIEVE sorts the collection according to the ascending order of the contents of that field.
  - If, in the second or subsequent sort keys, you omit a key word specifying the sort order, DATATRIEVE uses the sort order implied or specified for the preceding sort key.
- When DATATRIEVE executes the SORT statement, any selected record in the collection is released, and the collection cursor does not point to any record in the collection.

Usage Notes

- You can specify any number of sort keys.
- To sort record streams, use the SORTED BY clause of the record selection expression that creates the record stream.
- To sort collections as you form them, use the SORTED BY clause of the RSE in the FIND statement.

Example

Form a collection of the yachts built by Grampian, sort the collection by descending length over all, and then print the collection:

```
DTR> SET NO PROMPT(RET)
DTR> READY YACHTS(RET)
DTR> FIND YACHTS WITH BUILDER = "GRAMPIAN"(RET)
[5 records found]
DTR> SORT CURRENT BY DESC LOA(RET)
DTR> PRINT CURRENT(RET)

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL    WEIGHT BEAM  PRICE

 GRAMPIAN      2-34      SLOOP   34     11,800  10  $29,675
 GRAMPIAN      34        KETCH   33     12,000  10  $29,675
 GRAMPIAN      30        SLOOP   30      8,600  09  $17,775
 GRAMPIAN      28        SLOOP   28      6,900  10  $14,475
 GRAMPIAN      26        SLOOP   26      5,600  08  $11,495

DTR>
```

5.53 STORE Statement

Function

Creates a record in a DATATRIEVE domain and stores values in one or more fields of the record.

Format

```
STORE [context-variable IN] domain-name
        [USING statement-1]
        [VERIFY [USING] statement-2]
```

Arguments

domain-name

Is the given name or alias of the domain to contain the new record. You can prefix a context variable to the domain name by using this format:

[context-variable IN] domain-name

See the *DATATRIEVE-11 User's Guide* and Chapter 3 of this manual for information on context variables.

USING statement-1

Specifies a DATATRIEVE statement that prompts for or contains values for one or more fields in the new record.

VERIFY [USING] statement-2

Specifies a statement DATATRIEVE executes just before storing the new record.

Restrictions

- The domain specified in the STORE statement must be readied for write or extend access before you can store records in it.
- You cannot store a new record in an RMS relative file or a view.
- You cannot store values in the fields of a list with the USING clause. To store values in list fields, omit the USING clause and DATATRIEVE will prompt you for a value for each field in the record. You can also make no reference to the list fields in the USING clause, and use the MODIFY statement later to enter values in the list fields.

- If you include the keyword USING when specifying statement 1, you must put USING on the same input line as the domain name, unless you end the input line with a continuation character (–). If you omit the keyword USING and the continuation character, you must put at least the first element of statement 1 on the same input line as the domain name.
- Unless you end your input line with the continuation character (–), do not press the RETURN key immediately before typing the keyword VERIFY.
- You cannot store a new record in an RMS indexed file if field values of the record duplicate values in either a primary or alternate key field with the NO DUP attribute.
- You cannot store a value in a COMPUTED BY field.
- To use the name of a REDEFINES field when storing a record, you must specify the name in a USING clause. Otherwise, DATATRIEVE prompts you with the name of the elementary field from which the REDEFINES field takes its value.

Results

- If you omit the USING statement 1 clause, DATATRIEVE prompts you for a value for the first elementary field in the record with this message:

```
ENTER field-name:
```

  After you enter a value, DATATRIEVE prompts for a value for the next elementary field in the record, and so on, until you have entered a value for every elementary field in the record.
- After you supply a value to the last elementary field in the record, DATATRIEVE adds the record to the data file.
- If you press only RETURN in response to a prompt, DATATRIEVE repeats the prompt.
- To enter all spaces in an alphabetic or alphanumeric field or to enter all zeros in a numeric field, respond to the STORE statement's prompt with one or more spaces and press RETURN.
- If you enter two or more TABs to a prompt for an alphanumeric field, DATATRIEVE stores the TAB characters in the field. If you enter two or more TABs to a prompt for a numeric field, DATATRIEVE displays an error message on your terminal and prompts you again for valid numeric data.
- If you type CTRL/Z in response to any prompt for a field value, DATATRIEVE aborts the STORE statement and returns you to DATATRIEVE command level. No new record is stored.

- If you enter more characters or digits than the field definition specifies in response to a prompt, DATATRIEVE displays an error message and reprompts for the value.
- If you enter data that conflicts with the conditions specified by a VALID IF clause in the record definition, DATATRIEVE reprompts for valid data.
- If you specify a VERIFY USING clause, no data is stored until DATATRIEVE successfully executes statement 2. If the VERIFY USING clause contains an ABORT statement in an IF-THEN-ELSE statement and the abort conditions are met, DATATRIEVE aborts the STORE statement without storing the record and returns you to DATATRIEVE command level. This abort occurs whether you have SET ABORT or SET NO ABORT in effect.

Usage Notes

- Use the following two forms of the assignment statement in the USING statement-1 clause:

  ```
  field-name = value-expression

  group-field-name-1 = group-field-name-2
  ```

- Each time DATATRIEVE completes the execution of a STORE statement, the new record is stored in the data file. If a STORE statement in a loop creates a record each time DATATRIEVE executes the loop, DATATRIEVE stores each new record in the data file when it executes the STORE statement. If you abort the loop with a CTRL/Z, CTRL/C, or ABORT statement, records already stored in the data file are not affected.
- You must use either the ERASE statement (for indexed files) or the MODIFY statement (for sequential files) to remove records from data files once the STORE statement has executed.
- To store more than one record in a domain, use a REPEAT statement:

  ```
  REPEAT n STORE domain-name
  ```

  The argument n specifies the number of records to be stored. You can avoid counting the new records you want to store by making n larger than the estimated number of new records. When you finish storing records, stop the prompts for data by typing CTRL/Z in response to the prompt. DATATRIEVE then returns you to command level.

- You can use the VERIFY clause to check the validity of data before DATATRIEVE stores a new record by putting an ABORT statement in an IF-THEN-ELSE statement that establishes conditions for the abort:
  - DATATRIEVE first checks the value you supply to a STORE or MODIFY statement prompt against any validation conditions specified for that field in the record definition. If the value passes this validation test, it is then checked against the conditions in the VERIFY clause of the STORE or MODIFY statement.
  - If you always use the same validation conditions for modifying and storing data, put those conditions in VALID IF clauses in the record definition. That way, DATATRIEVE reprompts you for another value for the same field if the value you entered does not pass the validation test. When you use an ABORT statement in the VERIFY clause to validate a value, DATATRIEVE returns you to DATATRIEVE command level if the value does not pass the validation test and you must reenter the STORE or MODIFY statement.
- You can also put a BEGIN-END block containing a series of IF-THEN-ELSE statements in the VERIFY clause to explicitly control the storing of records.
- You can transfer information from one domain to another with the STORE statement by using the following syntax to nest the STORE statement in a FOR loop:

```
FOR domain-1
    STORE domain-2 USING
        group-field-name-2 = group-field-name-1
```

  The group fields do not have to contain identical elementary fields, but DATATRIEVE transfers values only between elementary fields with identical field names. Use this method when a record contains duplicate elementary field names subordinate to different group fields.

  Use this method to transfer data between RMS sequential files and RMS indexed files. Both domains can share the same record definition, but the definition of one domain specifies a sequential file, and the other an indexed file. See the DEFINE FILE command for details on specifying types of file organization.
- You can also transfer elementary field values from one domain to another. Put a STORE statement in a FOR statement whose rse selected the desired source records, then put the necessary assignment statements in the USING clause of the STORE statement.
- Take special care when storing data in primary key fields of RMS indexed files and in other fields with the NO CHANGE attribute. Values in these fields cannot be changed with the MODIFY command, and errors can only be corrected by creating a new record with the correct value and erasing the old record.

- Use a context variable with the domain name in the STORE statement if you want to refer to the values you have entered before DATATRIEVE stores the record. By using context variables, you can use the values for VERIFY USING clauses as in the second example, or you can use the values of one field to calculate the values of other fields to be stored in the record. The following example calculates and stores a yearly total after storing the quarterly quantities:

```
DTR> SET NO PROMPT
DTR> STORE X IN ACCOUNTS USING
CON> BEGIN
CON>      Q1 = *.Q1
CON>      Q2 = *.Q2
CON>      Q3 = *.Q3
CON>      Q4 = *.Q4
CON>      FY = X.Q1 + X.Q2 + X.Q3 + X.Q4
CON> END
Enter Q1:
```

Examples

Store two records in the FAMILIES domain:

```
DTR> SET NO PROMPT
DTR> READY FAMILIES WRITE
DTR> REPEAT 2 STORE FAMILIES
Enter FATHER: RUSSELL
Enter MOTHER: KAREN
Enter NUMBER_KIDS: 2
Enter KID_NAME: JENNIFER
Enter AGE: 12
Enter KID_NAME: LESLIE
Enter AGE: 9
Enter FATHER: WAYNE
Enter MOTHER: SHEEL
Enter NUMBER_KIDS: 2
Enter KID_NAME: BETE
Enter AGE: 8
Enter KID_NAME: ALEX
Enter AGE: 5
DTR> FIND FAMILIES WITH MOTHER = "KAREN", "SHEEL"
[2 records found]
DTR> PRINT ALL

                        NUMBER     KID
  FATHER      MOTHER     KIDS      NAME     AGE

RUSSELL     KAREN          2    JENNIFER    12
                                LESLIE       9
WAYNE       SHEEL          2    BETE         8
                                ALEX         5

DTR>
```

Store a record in the YACHTS domain, using a context variable and a VERIFY clause:

```
DTR> SHOW HINKLEY_STORE↵
PROCEDURE HINKLEY_STORE
STORE A IN YACHTS USING
BEGIN
     BUILDER = "HINKLEY"
     MODEL = "BERMUDA 40"
     RIG = "YAWL"
     LOA = 40
     DISP = 20000
     BEAM = 12
     PRICE = 82000
END VERIFY USING
BEGIN
     PRINT A.BOAT, SKIP
     IF *.CONFIRMATION CONT "N" THEN
     PRINT SKIP THEN ABORT "Bad record"
END
END_PROCEDURE

DTR> READY YACHTS WRITE↵
DTR> :HINKLEY_STORE↵
                                LENGTH
                                 OVER
MANUFACTURER    MODEL      RIG   ALL   WEIGHT BEAM  PRICE

 HINKLEY      BERMUDA 40 YAWL   40    20,000  12  $82,000

Enter CONFIRMATION: N↵

ABORT: Bad record
Execution terminated by "ABORT" statement
DTR>
```

Define a domain for single-digit integers and their squares, and use a WHILE statement to control the number of records stored in the domain:

```
DTR> SET NO PROMPT↵
DTR> DEFINE DOMAIN SQUARES USING↵
DFN> SQUARES_REC ON SQUARES.DAT;↵
DTR> DEFINE RECORD SQUARES_REC USING↵
DFN> 01 SQUARES.↵
DFN>    03 NUMBER PIC 9.↵
DFN>    03 ITS_SQUARE PIC 99.↵
DFN> ;↵
[Record is 3 bytes long.]
DTR> DEFINE FILE FOR SQUARES;↵
DTR> READY SQUARES WRITE↵
DTR> DECLARE N PIC 99.↵
DTR> N = 0↵
DTR> REPEAT 10↵
DTR> STORE SQUARES USING↵
CON>     BEGIN↵
CON>     NUMBER = N↵
CON>     ITS_SQUARE = N * N↵
CON>     N = N + 1↵
CON>     END↵
DTR> FIND SQUARES↵
[10 records found]
DTR> PRINT ALL↵
```

(continued on next page)

```
        ITS
NUMBER SQUARE

  0       00
  1       01
  2       04
  3       09
  4       16
  5       25
  6       36
  7       49
  8       64
  9       81

DTR>
```

Store data from the WORKER domain into an expanded domain, NEW_WORKER. These domains have duplicate elementary field names that are subordinate to different group field names. Use a FOR statement to nest the STORE statement so that DATATRIEVE stores the data from the elementary fields correctly. Here are the record definitions and the STORE statement:

```
DTR> SET NO PROMPT(RET)
DTR> SHOW WORK_REC(RET)
RECORD WORK_REC
 USING
01 WORK.
    03 LOCAL.
         05 CITY    PIC X(10).
         05 STATE   PIC X(2).
    03 REMOTE.
         05 CITY    PIC X(10).
         05 STATE   PIC X(2).
;
DTR> SHOW NEW_WORK_REC(RET)
RECORD NEW_WORK_REC
 USING
01 WORK.
    03 NEW_LOCAL.
         05 CITY    PIC X(10).
         05 STATE   PIC X(2).
    03 NEW_REMOTE.
         05 CITY    PIC X(10).
         05 STATE   PIC X(2).
    03 NAME.
         05 FIRST PIC X(10).
         05 LAST  PIC X(15).
;
DTR> READY WORKER(RET)
DTR> READY NEW_WORKER WRITE(RET)
DTR> FOR WORKER(RET)
CON>     STORE NEW_WORKER USING(RET)
CON>     BEGIN(RET)
CON>             NEW_LOCAL = LOCAL(RET)
CON>             NEW_REMOTE = REMOTE(RET)
DTR      END(RET)
DTR> PRINT WORKER(RET)
```

(continued on next page)

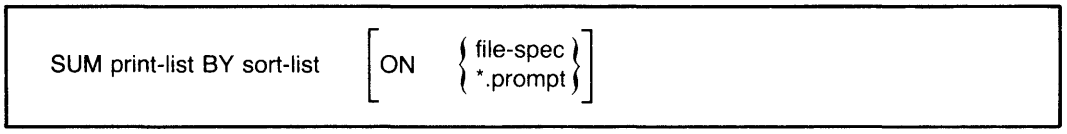
```
   CITY     STATE    CITY     STATE

NASHUA       NH    BOSTON       MA

DTR> PRINT NEW_WORKER(RET)

   CITY     STATE    CITY     STATE   FIRST          LAST

NASHUA       NH    BOSTON       MA
NASHUA       NH    BOSTON       MA

DTR>
```

5.54 SUM Statement

Function

Provides a summary of totals for one or more numeric fields in the current collection. The summary is sorted according to the values in one or more fields of the current collection. The summary includes subtotals for control groups and can be written to a file or an output device.

Format



Arguments

print-list

Is a list of one or more numeric fields, other value expressions, and modifiers. The format of the print list is:

{value-expression [{modifier} [...]] } [,...]

sort-list

Is a list of one or more sort keys that determine the order in which DATATRIEVE presents the summary totals. An item in the sort list consists of the name of a field to be used for the sort, preceded or followed by a key word that determines the order in which DATATRIEVE sorts the records in the collection:

ASC[ENDING]
DESC[ENDING]
INCREASING
DECREASING

ASCENDING is the default order. If you specify more than one sort key, use a comma to separate each sort key from the next.

file-spec

Is the file to which you want to write the output of the statement. The file specification must have this format:

For RSTS/E systems: dev:[PPN]file-name.type

For RSX systems: dev:[UIC]file-name.type;ver

You must specify at least one field in the file specification. On RSTS/E systems, you must specify a file name. DATATRIEVE uses the following defaults for fields not specified:

Field	Default
dev:	SY: (the system device)
[UIC/PPN]	Your default UIC/PPN
file name	No default
.type	.LST
;ver	1 or next higher version (non-RSTS/E systems only)

*.prompt-name

Is a prompting value expression that prompts for the specification of the file to which you want to write the output of the statement.

Restrictions

- You must have established a current collection before you can use the SUM statement.
- You cannot use the SUM statement with COMPUTED BY fields.
- You must include only numeric fields in the print list.

Results

- DATATRIEVE creates a record stream based on the current collection, sorts the records of the record stream according to the specifications in the sort list, and displays the summary on your terminal or writes it to the file specified in the ON clause.
- The summary consists of totals for the fields specified in the print list. For each control group created by the sort list, there is a total for each field in the print list. At the end of the summary is a total for each value expression in the print list of the values for all the records in the record stream.
- The order and format of the data in the report depends on the arguments you select.
- If you omit the ON clause, DATATRIEVE displays the output on your terminal.

Usage Note

Use edit strings to format the output of the totals for items in the print list.

Example

Form a collection of yachts. Use the SUM statement to summarize the prices of yachts in the collection and to display the number of yachts built by each builder. Use edit strings to format the values:

```
DTR> SHOW SUM_YACHTS↵
PROCEDURE SUM_YACHTS
          READY YACHTS
          FIND FIRST 6 YACHTS
          SUM 1 ("NUMBER"/"OF YACHTS") USING 9,
          PRICE USING $$$$,$$$ BY BUILDER
END_PROCEDURE
DTR> :SUM_YACHTS↵

                NUMBER                NUMBER
MANUFACTURER OF YACHTS  PRICE    OF YACHTS  PRICE

 ALBERG           1     $36,951
 ALBIN            3     $64,000
 AMERICAN         2     $28,790
                                       6     $129,741

DTR>
```

5.55 THEN Statement

Function

Joins two or more DATATRIEVE statements into a compound statement.

Format

```
statement-1 {THEN statement-2} [...]
```

Argument

statement

Is a DATATRIEVE statement.

Restrictions

- You must observe all restrictions on the statements included in the compound statement. Restrictions are listed in the descriptions of each statement.
- You cannot use DATATRIEVE commands in a compound statement.
- A procedure invoked in a compound statement cannot contain DATATRIEVE commands.
- Do not include FIND and SELECT in the same compound statement.
- Do not include FIND and SORT in the same compound statement.
- Do not include SELECT and DROP in the same compound statement.

Results

- DATATRIEVE executes statements in compound THEN statement in the order entered.
- If any statement in a compound THEN statement contains a syntax error, DATATRIEVE does not execute any of the statements in the THEN statement.

Usage Notes

- You can use a compound statement anywhere you can use a single statement.
- Use THEN statements rather than BEGIN-END blocks to form short compound statements.
- If you use any statement that creates a context in the compound statement, do not use any other statement in that compound that refers to or depends on that context information.

Example

Use THEN to join a PRINT statement and a MODIFY statement in a FOR loop:

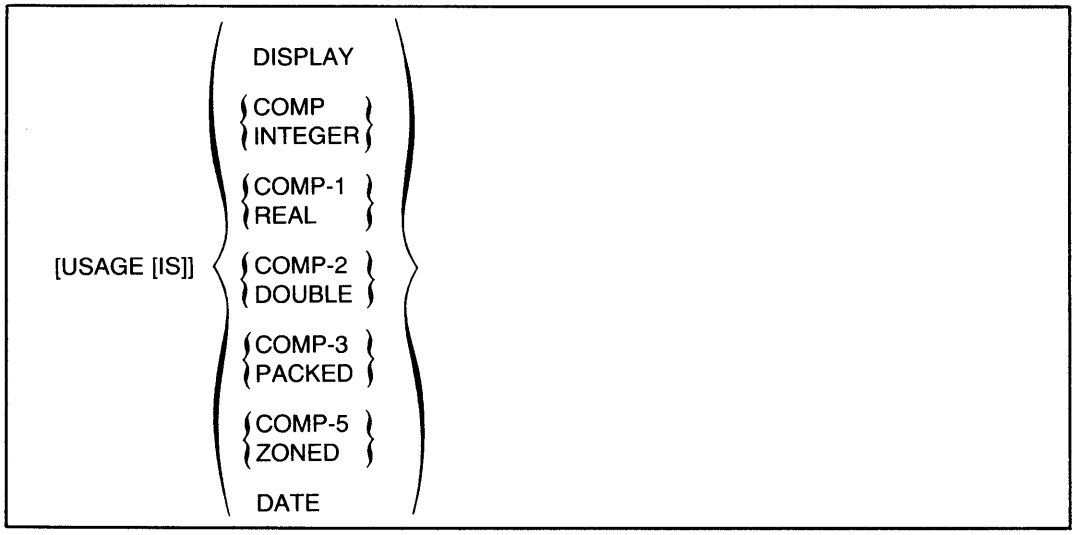
```
DTR> SET NO PROMPT↵
DTR> READY YACHTS MODIFY↵
DTR> FOR YACHTS WITH BUILDER EQ "ALBIN"↵
CON>     PRINT THEN MODIFY↵

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL    WEIGHT BEAM   PRICE

 ALBIN          79        SLOOP  26      4,200  10   $17,900
Enter MANUFACTURER: ^Z
Execution terminated by operator

DTR>
```

5.56 USAGE Clause

Function

Specifies the internal format of a numeric field or specifies a date field.

Format



Arguments

DISPLAY

Indicates that each digit occupies one byte of storage. DISPLAY is the default if you do not include a USAGE clause.

COMPINTEGER

Indicates that the field value is stored in binary format. INTEGER is a synonym for COMP.

COMP-1 REAL

Indicates that the field value is stored in single-precision real format. REAL is a synonym for COMP-1.

COMP-2 DOUBLE

Indicates that the field value is stored in double-precision real format. DOUBLE is a synonym for COMP-2.

COMP-3 PACKED

Indicates that the field value is stored in packed decimal format. PACKED is a synonym for COMP-3.

COMP-5 ZONED

Indicates that the field value is stored in signed decimal format. ZONED is a synonym for COMP-5.

DATE

Indicates that the field is a date field.

Restrictions

- This clause is valid only for elementary numeric or date fields.
- A field definition cannot contain both a USAGE clause and a SIGN clause.

Results

- The USAGE clause determines the internal storage format of a numeric field. If a numeric field definition does not have a USAGE clause, each digit in the field value occupies one character position in the record.
- The USAGE IS DATE clause identifies the field as a DATATRIEVE date field.

Usage Notes

- Use the appropriate form of the USAGE clause when a program written in COBOL, BASIC-PLUS-2, or other language uses the record and requires a different internal format.
- The USAGE clause can cause DATATRIEVE to align fields on hardware storage boundaries. See the section in this chapter on the ALLOCATION clause for information on word boundary alignment.
- A COMP (or INTEGER) field stores values in binary format. The size of a COMP (or INTEGER) field depends on the number of digit positions specified in its PICTURE clause.
- A COMP-1 (or REAL) field stores values in single-precision real (floating point) format. COMP-1 fields are four bytes long.
- A COMP-2 (or DOUBLE) field stores values in double-precision real (floating point) format. COMP-2 fields are eight bytes long.
- A COMP-3 (or PACKED) field stores values in packed decimal format, two digits per byte. The value of a COMP-3 field must contain a sign. The sign occupies the four low-order bits in the rightmost byte. The size of the field depends on the number of digit positions specified by the field's PICTURE clause:

$$\text{size (in bytes)} = \frac{\text{digit-positions} + 1}{2}$$

For example, a field with three digit positions is two bytes long. If the field contains an even number of digits, the size is rounded up. Thus, a six-digit field is stored in four bytes.

- A COMP-5 (or ZONED) field stores values in signed decimal format, one digit per byte. Therefore, the size of a COMP-5 field is the number of digit positions specified in its PICTURE clause.

  The sign of a COMP–5 value shares the rightmost byte with the lowest-valued digit of the value. The lowercase letters p through y represent a negative sign for the values 0 through 9.

- A DATATRIEVE date field stores a date as an eight-byte binary value. Other languages will not interpret the date field correctly. The date is expressed as the number of 100-nanosecond units (clunks) since the base date of 00:00:00 AM on November 17, 1858.

  When you enter a date value, DATATRIEVE translates the input value to clunks before storing it. When you print a date field, DATATRIEVE translates the number of clunks to the format specified in the EDIT_STRING clause, or to the default format if no EDIT_STRING clause is included in the field definition. The default format is DD-MMM-YYYY.

- Use the DATATRIEVE date value expression, "TODAY", to assign the current system date to a DATE variable or field. For example:

```
DTR> DECLARE X USAGE DATE;RET
DTR> X = "TODAY"RET
DTR> PRINT XRET

     X

11-Feb-83

DTR>
```

Examples

Define the field SALE_DATE as a date field, to be printed in the default format for date fields:

```
06 SALE_DATE USAGE IS DATE;
```

Define the field SALE_PRICE as a COMP-1 (REAL) field:

```
05 SALE_PRICE PIC 9(5)
         USAGE REAL
         EDIT_STRING IS $(6);
```

5.57 VALID IF Clause

Function

Validates a value for the field before it is stored in the record.

Format

```
VALID IF boolean-expression
```

Arguments

boolean-expression

Is a DATATRIEVE Boolean expression.

Restriction

A field definition cannot contain both a VALID IF and a COMPUTED BY clause.

Result

When you enter a value for the field in response to a MODIFY or STORE statement prompt, DATATRIEVE evaluates the Boolean expression. If the Boolean expression is true, DATATRIEVE stores the value in the field. If it is false, DATATRIEVE prints an error message and reprompts for the field value.

Example

The YACHT record contains VALID IF clauses for the RIG, LOA, and PRICE fields. DATATRIEVE stores only RIG values of SLOOP, KETCH, MS, and YAWL, LOA values between 15 and 50, and PRICE values greater than 1.3 times the displacement or equal to zero:

```
DTR> SHOW YACHT↵
RECORD YACHT
USING
ALLOCATION IS LEFT_RIGHT
01 BOAT.
  03 TYPE.
     06 MANUFACTURER PIC X(10)
        QUERY_NAME IS BUILDER.
     06 MODEL PIC X(10).
  03 SPECIFICATIONS
     QUERY_NAME SPECS.
     06 RIG PIC X(6)
        VALID IF RIG EQ "SLOOP","KETCH","MS","YAWL".
     06 LENGTH_OVER_ALL PIC XXX
        VALID IF LOA BETWEEN 15 AND 50
        QUERY_NAME IS LOA.
     06 DISPLACEMENT PIC 99999
        QUERY_HEADER IS "WEIGHT"
        EDIT_STRING IS ZZ,ZZ9
        QUERY_NAME IS DISP.
     06 BEAM PIC 99.
     06 PRICE PIC 99999
        VALID IF PRICE>DISP*1.3 OR PRICE EQ 0
        EDIT_STRING IS $$$,$$$.
;
DTR> READY YACHTS MODIFY↵
DTR> FIND FIRST 1 YACHTS WITH BUILDER = "ALBIN"↵
[1 Record found]
DTR> SELECT↵
DTR> MODIFY RIG, LOA, PRICE↵
Enter RIG: CANOE↵
Validation error for RIG
Re-enter RIG: SLOOP↵
Enter LENGTH_OVER_ALL: 62↵
Validation error for LENGTH_OVER_ALL
Re-enter LENGTH_OVER_ALL: 45↵
Enter PRICE: 4200↵
Validation error for PRICE
Re-enter PRICE: ^Z
Execution terminated by operator
DTR>
```

5.58 WHILE Statement

Function

Causes DATATRIEVE to repeat a statement as long as the condition specified in the Boolean expression is true.

Format

```
WHILE boolean-expression statement
```

Arguments

boolean-expression

Is a Boolean expression. Boolean expressions in the WHILE statement are limited to this format:

variable-name boolean-operator value-expression

*.prompt

statement

Is a simple or compound statement you want DATATRIEVE to execute if the Boolean expression evaluates to true.

Restrictions

- You must observe all restrictions for statements in the WHILE statement.
- You cannot use a field name as a value expression on the left side of the Boolean expression of a WHILE statement.

Result

DATATRIEVE repeats the specified statement as long as the Boolean example evaluates to true. If the relationship between the two values in the Boolean expression never changes (1 = 1, for example), the loop repeats infinitely until you end it with a CTRL/C or CTRL/Z.

Usage Notes

- Use the WHILE statement to form and control the execution of loops.
- You can use a prompting value expression or a variable as the first member of the Boolean expression.

Example

Group the boats with LOA less than 35 according to the value of BEAM. Display the TYPE, LOA, and BEAM of the shortest boat from each group of boats with the same value for BEAM:

```
DTR> READY YACHTS↵
DTR> SHOW WHILE_EX↵
PROCEDURE WHILE_EX
BEGIN
          DECLARE X PIC 99,
          X = 0
          FOR YACHTS WITH LOA < 35 AND
                    BEAM NE 0 SORTED BY BEAM, LOA
                    WHILE X < BEAM
                    BEGIN
                              PRINT TYPE, LOA, BEAM, X
                              X = BEAM
                    END
END
END_PROCEDURE
DTR> :WHILE-EX↵

                          LENGTH
                           OVER
MANUFACTURER    MODEL      ALL    BEAM X

 CAPE DORY     TYPHOON      19      06  00
 WINDPOWER     IMPULSE      16      07  06
 ERICSON       23/ SPECIA   23      08  07
 EASTWARD      HO           24      09  08
 ALBIN         79           26      10  09
 BOMBAY        CLIPPER      31      11  10
 IRWIN         25           25      12  11

DTR>
```

DATATRIEVE–11 Keywords A

ABORT
ALIGNED_MAJOR_MINOR
AND
ASC
AVERAGE
BOTTOM
BY
CHARACTERS
COLLECTIONS
COLUMN_HEADER
COMP_1
COMP_5
CONT
DATATYPE
DECLARE
DEFINEP
DEPENDING
DICTIONARY
DISPLAY
DOUBLE
D_FLOATING
ELSE
END_REPORT
EQUAL
EXIT
FIELDS
FIND
FN1
FOR
F_FLOATING
GREATER_THAN
HELP
INCREASING
KEY
LEADING
LESS_EQUAL
LONGWORD
ADT
ALL
ANY
ASCENDING
BEGIN
BT
CHANGE
CLOSE
COLUMN
COMP
COMP_2
COMP_6
CONTAINING
DATE
DECREASING
DELETE
DESC
DIGIT
DOMAIN
DROP
EDIT
END
END_TABLE
ERASE
EXTEND
FILE
FINISH
FN2
FORM
GE
GT
IF
INTEGER
LAST
LEFT
LESS_THAN
LT
ADVANCED
ALLOCATION
AS
AT
BETWEEN
BUT
CHARACTER
COL
COLUMNS_PAGE
COMPUTED
COMP_3
CONNECT
COUNT
DECIMAL
DEFINE
DELETEP
DESCENDING
DIGITS
DOMAINS
DUP
EDIT_STRING
END_PROCEDURE
EQ
EXCLUSIVE
EXTRACT
FILL
FIRST
FN3
FROM
GREATER_EQUAL
GUIDE
IN
IS
LE
LEFT_RIGHT
LINES_PAGE
MAJOR_MINOR

MATCH
MAX
MAX_LINES
MAX_PAGES
MEDIAN
MEMBERS
MIN
MODIFY
NE
NEW_PAGE
NEW_SECTION
NEXT
NO
NOT
NOT_EQUAL
NO_DATE
NO_NUMBER
NUMBER
NUMERIC
OCCURS
OF
ON
OPEN
OR
OVERPUNCHED
OWNER
PACKED
PAGE
PIC
PICTURE
PLOT
PORT
PRINT
PROCEDURE
PROCEDURES
PROMPT
PROTECTED
PW
QUADWORD
QUERY_HEADER
QUERY_NAME
READ
READY
REAL
RECORD
RECORDS
REDEFINES
RELEASE
REPEAT
REPORT
REPORT_HEADER
REPORT_NAME
RIGHT
SCALE
SELECT
SEPARATE
SET
SETS
SHARED
SHOW
SHOWP
SIGN
SIGNED
SIZE
SKIP
SORT
SORTED
SPACE
STORE
STRING
STRUCTURE
SUBSCHEMA
SUM
SUPERCEDE
SUPERSEDE
SYNC
TAB
TABLE
TABLES
TEXT
THE
THEN
TIMES
TO
TOP
TOTAL
TRAILING
UIC
UNSIGNED
USAGE
USING
VALID
VARYING
VERIFY
VIA
WHILE
WITH
WITHIN
WORD
WRITE
ZONED

ASCII Values for Printing Characters B

Table B-1: ASCII Codes

DECIMAL VALUE	CHARACTER	REMARKS
32	SP	Space
33	!	Exclamation point
34	"	Double quotation mark
35	#	Number sign
36	$	Dollar sign
37	%	Percent sign
38	&	Ampersand
39	'	Apostrophe
40	(	Left (open) parenthesis
41	)	Right (close) parenthesis
42	*	Asterisk
43	+	Plus sign
44	,	Comma
45	–	Minus sign (hyphen)
46	.	Period (decimal point)
47	/	Slash (slant)
48	0	Zero
49	1	One
50	2	Two
51	3	Three
52	4	Four
53	5	Five
54	6	Six
55	7	Seven
56	8	Eight
57	9	Nine
58	:	Colon
59	;	Semicolon

Table B-1: ASCII Codes (Cont.)

DECIMAL VALUE	CHARACTER	REMARKS
60	<	Less than (left angle bracket)
61	=	Equal sign
62	>	Greater than (right angle bracket)
63	?	Question mark
64	@	Commercial at sign
65	A	Uppercase A
66	B	Uppercase B
67	C	Uppercase C
68	D	Uppercase D
69	E	Uppercase E
70	F	Uppercase F
71	G	Uppercase G
72	H	Uppercase H
73	I	Uppercase I
74	J	Uppercase J
75	K	Uppercase K
76	L	Uppercase L
77	M	Uppercase M
78	N	Uppercase N
79	O	Uppercase O
80	P	Uppercase P
81	Q	Uppercase Q
82	R	Uppercase R
83	S	Uppercase S
84	T	Uppercase T
85	U	Uppercase U
86	V	Uppercase V
87	W	Uppercase W
88	X	Uppercase X
89	Y	Uppercase Y
90	Z	Uppercase Z
91	[	Left square bracket
92	\	Backslash (reverse slant)
93	]	Right square bracket
94	^	Circumflex (caret)
95	_	Underscore (underline)
96	`	Left single quote (grave accent)
97	a	Lowercase a
98	b	Lowercase b
99	c	Lowercase c
100	d	Lowercase d
101	e	Lowercase e
102	f	Lowercase f
103	g	Lowercase g
104	h	Lowercase h
105	i	Lowercase i
106	j	Lowercase j
107	k	Lowercase k
108	l	Lowercase l
109	m	Lowercase m
110	n	Lowercase n

Table B-1: ASCII Codes (Cont.)

DECIMAL VALUE	CHARACTER	REMARKS
111	o	Lowercase o
112	p	Lowercase p
113	q	Lowercase q
114	r	Lowercase r
115	s	Lowercase s
116	t	Lowercase t
117	u	Lowercase u
118	v	Lowercase v
119	w	Lowercase w
120	x	Lowercase x
121	y	Lowercase y
122	z	Lowercase z
123	{	Left brace
124	|	Vertical line
125	}	Right brace
126	~	Tilde

Syntax Formats for DATATRIEVE–11 C

C.1 DATATRIEVE COMMANDS

ADT Command

ADT

CLOSE Command

CLOSE

DEFINE DICTIONARY Command

DEFINE DICTIONARY file-spec

DEFINE DOMAIN Command

RMS Domain

DEFINE DOMAIN domain-name USING record-name [(passwd) / (*)] ON file-spec

View Domain

```
DEFINE DOMAIN view-name OF domain-name [,...]      USING
        level-number-1 field-name-1      OCCURS FOR rse-1 .
        level-number-2 field-name-2    { OCCURS FOR rse-2 }
                                       { FROM domain-name-2 }  .
              .              .                .
              .              .                .
              .              .                .
;
```

DEFINE FILE Command

Sequential File

```
DEFINE FILE [FOR] domain-name [,]
    [ ALLOCATION = n ]
    [ SUPERSEDE      ]   [,...]
    [ MAX            ]
```

Indexed Sequential File

```
DEFINE FILE [FOR] domain-name [,]
    [ ALLOCATION = n ]
    [ SUPERSEDE      ]   [,...]
    [ MAX            ]
    { KEY = field-name-1  [ ( [NO] CHANGE [,]   [NO] DUP ) ] }   [,...]
```

DEFINE PORT Command

```
DEFINE PORT port-name [USING] record-name ;
```

DEFINE PROCEDURE Command

```
DEFINE PROCEDURE procedure-name
              .
              .
              .
END_PROCEDURE
```

DEFINE RECORD Command

```
DEFINE RECORD record-name [USING]
    [ ALLOCATION IS { MAJOR-MINOR
                    { ALIGNED-MAJOR-MINOR
                    { LEFT-RIGHT                 } ]
      level-number-1 field-name-1 [field-definition-1] .
      [level-number-2 field-name-2 field-definition-2 .]
          .          .          .
          .          .          .
          .          .          .
;
```

DEFINE TABLE Command

```
DEFINE TABLE table-name
    { "code-1" }     { "translation-1" }
    { code-1   }  :  { translation-1   }  ,
  [ { "code-2" }     { "translation-2" }  ]
    { code-2   }  :  { translation-2   }  ,
          .                  .
          .                  .
          .                  .
  [ ELSE  { "translation-n" }  ]
          { translation-n   }
END_TABLE
```

DEFINEP Command

```
DEFINEP object-name  [ (passwd) ]  sequence-number,  { PW, new-passwd }  , priv
                     [ (*)      ]                    { UIC , [m,n]    }
```

DELETE Command

```
DELETE object-name ;
```

DELETE Command (Editor)

```
D[ELETE] [range]
```

DELETEP Command

DELETEP object-name sequence-number

EDIT Command

EDIT object-name [(passwd) / (*)] [ADVANCED]

EXIT Command

{ EXIT / CTRL/Z }

EXIT Command (Editor)

{ EX[IT] }
{ CTRL/Z }

EXTRACT Command

EXTRACT [ON] file-spec object-name [,...]

FINISH Command

FINISH [domain-name] [,...]

HELP Command

HELP [ADVANCED] [topic] [,...]

INSERT Command (Editor)

I[NSERT] [range]

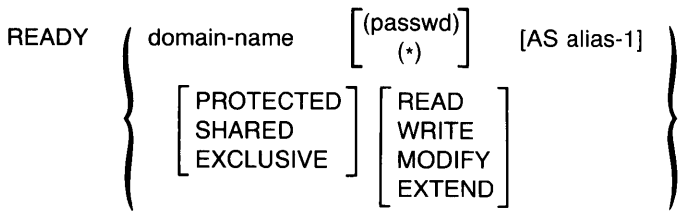
OPEN Command

OPEN file-spec

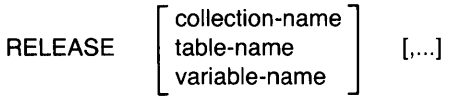
QUIT Command (Editor)

QUIT

READY Command



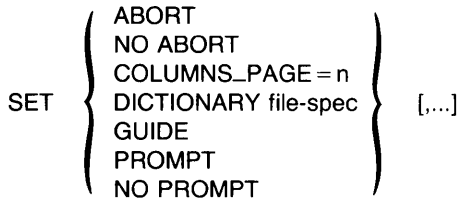
RELEASE Command



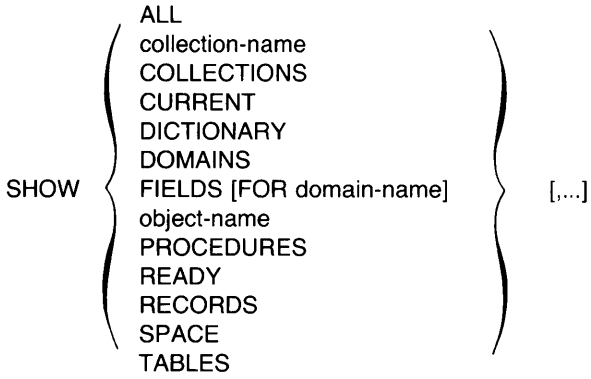
REPLACE Command (Editor)

R[EPLACE] [range]

SET Command



SHOW Command



SHOWP Command

SHOWP object-name [(passwd) | (*)]

SUBSTITUTE Command (Editor)

S /string–1/[string–2][/[range]]

TYPE Command (Editor)

[T[YPE]] [range]

C.2 DATATRIEVE STATEMENTS

ABORT Statement

ABORT [value-expression]

Assignment Statement

Elementary Field

field-name = value-expression

Group Field

group-field-name-1 = group-field-name-2

Variable

variable-name = value-expression

AT BOTTOM Statement (Report Writer)

AT BOTTOM OF { REPORT | PAGE | field-name } PRINT summary-element [,...]

AT TOP Statement (Report Writer)

```
AT TOP OF { REPORT }  PRINT { header-element }  [,...]
          { PAGE   }        { summary-element }
          { field-name }
```

BEGIN-END Statement

```
BEGIN
     statement-1
    [statement-2]
    .
    .
    .
END
```

DECLARE Statement

```
DECLARE variable-name variable-definition.
```

DECLARE PORT Statement

```
DECLARE PORT port-name USING
        level-number-1 field-definition-1.
       [level-number-2 field-definition-2.]
              .              .
              .              .
              .              .
       [level-number-n field definition-n.]
;
```

DISPLAY Statement

```
DISPLAY value-expression
```

DROP Statement

```
DROP [collection-name]
```

END_REPORT Statement (Report Writer)

```
END_REPORT
```

ERASE Statement

```
ERASE [ALL [OF rse]]
```

FIND Statement

```
FIND rse
```

FOR Statement

```
FOR rse statement
```

IF-THEN-ELSE Statement

```
IF boolean-expression [THEN] statement-1 [ELSE statement-2]
```

MODIFY Statement

```
MODIFY [ALL]  [ field-name [,...]
              [ USING statement-1 ]

    [VERIFY [USING] statement-2]

    [OF rse]
```

PRINT Statement

Retrieving from Selected Records and Target Record Streams Formed by FOR Loops

```
PRINT [print-list]  [ ON  { file-spec  } ]
                          { *.prompt   }
```

Retrieving from the Current Collection

```
PRINT ALL [print-list]  [ ON  { file-spec  } ]
                              { *.prompt   }
```

Retrieving From Record Streams Formed by the PRINT Statement

1. One RSE

```
PRINT [print-list OF] rse  [ ON  { file-spec  } ]
                                 { *.prompt   }
```

(continued on next page)

PRINT Statement (Cont.)

2. Two RSEs (Inner print list follows another print list)

```
PRINT print-list, ALL print-list OF rse-1 [,print-list] OF rse-2
      [ ON { file-spec  } ]
           { *.prompt   }
```

3. Two RSEs (Inner print list precedes any other print list)

```
PRINT ALL ALL print-list OF rse-1 [,print-list] OF rse-2
      [ ON { file-spec  } ]
           { *.prompt   }
```

PRINT Statement (Report Writer)

```
PRINT print-list-element [,...]
```

REPEAT Statement

```
REPEAT value-expression statement
```

REPORT Statement

```
REPORT [rse]  [ ON { file-spec  } ]
                   { *.prompt   }
```

SELECT Statement

```
SELECT [ FIRST            ]
       [ NEXT             ]  [collection-name]
       [ LAST             ]
       [ value-expression ]
```

SET Statement (Report Writer)

For naming the report:

```
SET     REPORT_NAME = { "string"[/...] }
                      { *.prompt       }
```

For specifying a date or string at the upper right of each page:

```
SET     DATE = "string"
```

(continued on next page)

SET Statement (Report Writer) (Cont.)

For controlling the printing of a date or page number:

```
SET   { NO DATE   }
      { NUMBER    }   [,...]
      { NO NUMBER }
```

For specifying page width or length, or overall report length:

```
SET   { { COLUMNS_PAGE = }          { n         } }
      { { LINES_PAGE =   }          { *.prompt  } }   [,...]
      { { MAX_LINES =    }                        }
      { { MAX_PAGES =    }                        }
```

SORT Statement

```
SORT [collection-name] [BY] sort-key-1 [,...]
```

STORE Statement

```
STORE [context-variable IN] domain-name
      [USING statement-1]
      [VERIFY [USING] statement-2]
```

SUM Statement

```
SUM print-list BY sort-list   [ ON   { file-spec } ]
                              [      { *.prompt  } ]
```

THEN Statement

```
statement-1 {THEN statement-2} [...]
```

WHILE Statement

```
WHILE boolean-expression statement
```

C.3 Record Definition Clauses

ALLOCATION Clause

ALLOCATION IS { MAJOR_MINOR | ALIGNED_MAJOR_MINOR | LEFT_RIGHT }

COMPUTED BY Clause

COMPUTED BY value-expression

EDIT_STRING Clause

EDIT_STRING [IS] edit-string

OCCURS Clause

Fixed Number of Occurrences

OCCURS n TIMES

Variable Number of Occurrences

OCCURS min TO max TIMES DEPENDING ON field-name

PICTURE Clause

PIC[TURE] [IS] picture-string

QUERY_HEADER Clause

QUERY_HEADER [IS] {"header-segment"} [/...]

QUERY_NAME Clause

QUERY_NAME [IS] query-name

REDEFINES clause

level-no field-name-1 REDEFINES field-name-2

SIGN Clause

```
SIGN [IS]  { LEADING  }  [SEPARATE]
           { TRAILING }
```

SORTED BY Clause

```
SORT [collection-name] [BY] sort-key-1 [,...]
```

USAGE Clause

```
                  DISPLAY
                  { COMP    }
                  { INTEGER }
                  { COMP-1  }
                  { REAL    }
[USAGE [IS]]      { COMP-2  }
                  { DOUBLE  }
                  { COMP-3  }
                  { PACKED  }
                  { COMP-5  }
                  { ZONED   }
                  DATE
```

VALID IF Clause

```
VALID IF boolean-expression
```

C.4 Miscellaneous Syntax

Record Selection Expression

```
[ FIRST n ]                  { domain-name     }
[ ALL     ]  [context-var IN]{ collection-time }
                             { list-name       }

    [WITH boolean-expression]

        [SORTED BY sort-key [,...]]
```

Invoke Procedure

: procedure-name

Invoke Command File

@file-spec

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA and Puerto Rico call **800–258–1710**

In Canada call **800–267–6146**

In New Hampshire, Alaska or Hawaii call **603–884–6660**

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
A&SG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

*Any prepaid order from Puerto Rico must be placed with the Local Digital Subsidiary:
809–754–7575

DATATRIEVE-11
Reference Manual
AA-U049A-TK

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number. _____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

------Do Not Tear - Fold Here and Tape------

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: DISG Documentation ZK02-2/N53
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, N.H. 03062

------Do Not Tear - Fold Here and Tape------

digital
DIGITAL EQUIPMENT CORPORATION

Printed in U.S.A.