

Enform Plus Reference Manual

Abstract

This manual provides detailed information about the syntax of the Enform Plus language.

Product Version

Enform Plus D46

Supported Releases

This manual supports D46.00 and all subsequent D4x releases and G06.00 and all subsequent G-series releases until otherwise indicated in a new edition.

Part Number	Published
-------------	-----------

422684-001	May 1999
------------	----------

Document History

Part Number	Product Version	Published
422684-001	Enform Plus D46	May 1999

Ordering Information

For manual ordering information: domestic U.S. customers, call 1-800-243-6886; international customers, contact your local sales representative.

Document Disclaimer

Information contained in a manual is subject to change without notice. Please check with your authorized representative to make sure you have the most recent information.

Export Statement

Export of the information contained in this manual may require authorization from the U.S. Department of Commerce.

Examples

Examples and sample programs are for illustration only and may not be suited for your particular purpose. The inclusion of examples and sample programs in the documentation does not warrant, guarantee, or make any representations regarding the use or the results of the use of any examples or sample programs in any documentation. You should verify the applicability of any example or sample program before placing the software into productive use.

U.S. Government Customers

FOR U.S. GOVERNMENT CUSTOMERS REGARDING THIS DOCUMENTATION AND THE ASSOCIATED SOFTWARE:

These notices shall be marked on any reproduction of this data, in whole or in part.

NOTICE: Notwithstanding any other lease or license that may pertain to, or accompany the delivery of, this computer software, the rights of the Government regarding its use, reproduction and disclosure are as set forth in Section 52.227-19 of the FARs Computer Software—Restricted Rights clause.

RESTRICTED RIGHTS NOTICE: Use, duplication, or disclosure by the Government is subject to the restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013.

RESTRICTED RIGHTS LEGEND: Use, duplication or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the rights in Technical Data and Computer Software clause in DAR 7-104.9(a). This computer software is submitted with “restricted rights.” Use, duplication or disclosure is subject to the restrictions as set forth in NASA FAR SUP 18-52p227-79 (April 1985) “Commercial Computer Software—Restricted Rights (April 1985).” If the contract contains the Clause at 18-52p227-74 “Rights in Data General” then the “Alternate III” clause applies.

U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

Unpublished — All rights reserved under the Copyright Laws of the United States.

Enform Plus Reference Manual

Glossary

Index

Figures

Tables

[What's New in This Manual](#) iii

[Manual Information](#) iii

[New and Changed Information](#) iii

[About This Manual](#) v

[Your Comments Invited](#) v

[Notation Conventions](#) vii

1. Introduction

[Enform Plus Terminology](#) 1-1

2. Running Enform Plus

[ENFPLUS Macro](#) 2-2

[Interactive Mode](#) 2-2

[Noninteractive Mode](#) 2-3

[The Current Output Listing File](#) 2-3

[Pressing the Terminal Break Key](#) 2-4

[Logical File Assignments](#) 2-4

[Passing Parameters to Compiled Query Files](#) 2-5

[A Server Query Processor](#) 2-6

[The TACL ASSIGN Command](#) 2-7

[The TACL PARAM Command](#) 2-8

[The TACL QP Command](#) 2-9

[Example of Server Query Processor Creation](#) 2-10

[Generic Files](#) 2-10

[Generic Files and a Dedicated Query Processor](#) 2-12

[Generic Files and the Server Query Processor](#) 2-13

[Generic Files and the Current Output Listing File](#) 2-13

3. Enform Plus Language Elements

<u>Reserved Words</u>	3-2
<u>Special Characters</u>	3-4
<u>Comments</u>	3-4
<u>Statements</u>	3-4
<u>Clauses</u>	3-5
<u>Commands</u>	3-5
<u>Rules for Naming User-Defined Elements</u>	3-6
<u>Rules for Referencing Database Elements</u>	3-6
<u>Record Name References</u>	3-6
<u>Field Name References</u>	3-6
<u>Primary Key References</u>	3-7
<u>Subscripts</u>	3-8
<u>Aggregates</u>	3-11
<u>Predefined Aggregates</u>	3-13
<u>User Aggregates</u>	3-13
<u>Target Aggregates</u>	3-15
<u>Qualification Aggregates</u>	3-17
<u>Aggregates and Scale</u>	3-19
<u>Literals</u>	3-20
<u>Numeric Literals</u>	3-20
<u>String Literals</u>	3-20
<u>Arithmetic Expressions</u>	3-21
<u>Evaluation Order of Arithmetic Expressions</u>	3-22
<u>Scale Factor of the Result</u>	3-22
<u>Logical Expressions</u>	3-22
<u>Effect of Parentheses on Compound Logical Expressions</u>	3-24
<u>BEGINS WITH and CONTAINS</u>	3-24
<u>Range of Values in Logical Expressions</u>	3-25
<u>Pattern-Match in Logical Expressions</u>	3-26
<u>IF/THEN/ELSE Expressions</u>	3-26
<u>Parameters</u>	3-27

- [User Variables](#) 3-27
 - [User Variable as a Target-Item](#) 3-27
 - [User Variable in Request-Qualification](#) 3-29
 - [User Tables](#) 3-29
- [Arithmetic Overflow Conditions](#) 3-29

4. Statements

- [AT END Statement](#) 4-3
 - [Specifying a Field Name in an AT END Statement](#) 4-3
 - [Spacing Considerations](#) 4-3
 - [AT END Information for Current Report or All Reports](#) 4-4
 - [Canceling Session-Wide AT END Information](#) 4-4
- [AT START Statement](#) 4-5
 - [Specifying a Field Name in an AT START Statement](#) 4-5
 - [Spacing Considerations](#) 4-5
 - [AT START Information for Current Report or All Reports](#) 4-6
 - [Canceling Session-Wide AT START Information](#) 4-6
- [CLOSE Statement](#) 4-7
 - [The Effect of a CLOSE Statement on the Internal Table](#) 4-7
- [DECLARE Statement](#) 4-9
 - [Declaring a User Aggregate](#) 4-10
 - [Declaring a User Variable or User Table](#) 4-10
- [DELINK Statement](#) 4-12
- [DICTIONARY Statement](#) 4-13
 - [Identifying the Location of the Dictionary](#) 4-13
 - [Clearing the Internal Table](#) 4-14
- [EXIT Statement](#) 4-15
- [FIND Statement](#) 4-16
 - [Output Record Dictionary Description](#) 4-17
 - [Group Definition and Sorting](#) 4-17
 - [Output Fields](#) 4-18
 - [Input Elements](#) 4-19
 - [Request-Qualification](#) 4-21
 - [Summary Records](#) 4-21

<u>Statements and Clauses That Do Not Apply to the FIND Statement</u>	4-22
<u>FOOTING Statement</u>	4-23
<u>Specifying A Field Name Within a FOOTING Statement</u>	4-23
<u>Spacing Considerations</u>	4-23
<u>Footing for Current Report or All Reports</u>	4-24
<u>Canceling Session-Wide Footing</u>	4-24
<u>LINK Statement</u>	4-25
<u>Duration of Link Established by LINK or LINK OPTIONAL Statement</u>	4-26
<u>LINK Statement Considerations</u>	4-26
<u>LINK OPTIONAL Statement Considerations</u>	4-26
<u>LIST Statement</u>	4-30
<u>Input Record Description</u>	4-31
<u>Group Definition and Sorting</u>	4-32
<u>How Values Are Displayed in Report Columns</u>	4-32
<u>Request-Qualification</u>	4-34
<u>Conditional Printing</u>	4-34
<u>Summary Reports</u>	4-34
<u>Optional Clauses</u>	4-36
<u>OPEN Statement</u>	4-38
<u>Using OPEN AS COPY OF</u>	4-38
<u>PARAM Statement</u>	4-39
<u>How Enform Plus Handles Parameters</u>	4-39
<u>SET Statement</u>	4-41
<u>Initializing User-Defined Elements</u>	4-41
<u>Redefining Option Variables</u>	4-42
<u>SUBFOOTING Statement</u>	4-43
<u>Specifying Field Names Within a SUBFOOTING Statement</u>	4-43
<u>Spacing Considerations</u>	4-43
<u>Subfooting for Current Report or All Reports</u>	4-44
<u>Canceling Session-Wide Subfooting</u>	4-44
<u>SUBTITLE Statement</u>	4-45
<u>Specifying a Field Name Within a SUBTITLE Statement</u>	4-45
<u>Spacing Considerations</u>	4-45

[Subtitle for Current Report or All Reports](#) 4-46

[Canceling Session-Wide Subtitle](#) 4-46

[TITLE Statement](#) 4-47

[Specifying a Field Name Within a TITLE Statement](#) 4-47

[Spacing Considerations](#) 4-47

[Title for Current Report or All Reports](#) 4-48

[Canceling Session-Wide Title](#) 4-48

5. Clauses

[AFTER CHANGE Clause](#) 5-4

[Specifying a Field Name Within an AFTER CHANGE Clause](#) 5-4

[Spacing Considerations](#) 5-4

[ASCD and DESC Clauses](#) 5-6

[AS Clause](#) 5-7

[Repeatable Edit Descriptors](#) 5-9

[Nonrepeatable Edit Descriptors](#) 5-13

[Modifiers](#) 5-15

[Decorations](#) 5-18

[AS DATE Clause](#) 5-22

[Default Display Format](#) 5-22

[Examples of Date Display Formats](#) 5-23

[AS TIME Clause](#) 5-24

[Default Time Display Format](#) 5-24

[Examples of the Time Display Format](#) 5-25

[AT END PRINT Clause](#) 5-26

[Specifying a Field Name Within an AT END PRINT Clause](#) 5-26

[Spacing Considerations](#) 5-26

[AT END Information for Current Report or All Reports](#) 5-27

[Overriding Session-Wide AT END Information](#) 5-27

[AT START PRINT Clause](#) 5-28

[Specifying a Field Name in an AT START PRINT Clause](#) 5-28

[Spacing Considerations](#) 5-28

[AT START Information for Current Report or All Reports](#) 5-29

[Overriding Session-Wide AT START Information](#) 5-29

<u>BEFORE CHANGE Clause</u>	5-30
<u>Specifying a Field Name Within a BEFORE CHANGE Clause</u>	5-30
<u>Spacing Considerations</u>	5-30
<u>BY and BY DESC Clauses</u>	5-32
<u>CENTER Clause</u>	5-33
<u>Centering Single Report Items</u>	5-33
<u>Centering All Report Items</u>	5-33
<u>Centering a Print List</u>	5-33
<u>CUM Clause</u>	5-34
<u>CUM With OVER ALL</u>	5-34
<u>CUM With OVER</u>	5-34
<u>CUM Clause Used With User Variable</u>	5-35
<u>Restrictions</u>	5-35
<u>FOOTING Clause</u>	5-36
<u>Specifying a Field Name Within a FOOTING Clause</u>	5-36
<u>Spacing Considerations</u>	5-36
<u>Footing for Current Report or All Reports</u>	5-37
<u>FORM Clause</u>	5-38
<u>FORM Clause With a By-Item</u>	5-38
<u>FORM Clause With a Target-Item</u>	5-38
<u>FORM Clause Within a Print List</u>	5-38
<u>HEADING Clause</u>	5-39
<u>Default Headings</u>	5-39
<u>Multiple-Line Headings</u>	5-39
<u>Printing / in a Column Heading</u>	5-39
<u>Heading for Subscripted Elements</u>	5-40
<u>INTERNAL Clause</u>	5-42
<u>" JULIAN-DATE Conversion Clause</u>	5-43
<u>Conversion to Internal Format</u>	5-43
<u>Display Format</u>	5-44
<u>NOHEAD Clause</u>	5-45
<u>No Headings for Single Report Items</u>	5-45
<u>No Headings for All Report Items</u>	5-45

<u>NOPRINT Clause</u>	5-46
<u>Suppress Single Report Items</u>	5-46
<u>Suppress All Report Items</u>	5-46
<u>Option Variable Clauses</u>	5-48
<u>PCT Clause</u>	5-54
<u>Using PCT OVER ALL</u>	5-54
<u>Using PCT OVER By-Item</u>	5-54
<u>Combining Percentages and Subtotals</u>	5-55
<u>PCT Clause Used With User Variable</u>	5-55
<u>Restrictions</u>	5-55
<u>SKIP Clause</u>	5-56
<u>SKIP Clause With a LIST Target-Item or By-Item</u>	5-56
<u>SKIP Clause With a Print List</u>	5-56
<u>SPACE Clause</u>	5-58
<u>SPACE Clause With a LIST Target-Item or By-Item</u>	5-58
<u>SPACE Clause With a Print List</u>	5-58
<u>SUBFOOTING Clause</u>	5-59
<u>Specifying Field Names in a SUBFOOTING Clause</u>	5-59
<u>Spacing Considerations</u>	5-59
<u>Subfooting for Current Report or All Reports</u>	5-60
<u>SUBTITLE Clause</u>	5-61
<u>Specifying a Field Name in a SUBTITLE Clause</u>	5-61
<u>Spacing Considerations</u>	5-61
<u>Subtitle for Current Report or All Reports</u>	5-62
<u>SUBTOTAL Clause</u>	5-63
<u>SUPPRESS Clause</u>	5-64
<u>System Variable Clauses</u>	5-65
<u>Printing the Current Date or Time</u>	5-65
<u>Printing Line Numbers</u>	5-65
<u>Printing Page Numbers</u>	5-65
<u>TAB Clause</u>	5-66
<u>TAB Clause With a LIST Target-Item or By-Item</u>	5-66
<u>TAB Clause With a Print List</u>	5-66

TIMESTAMP-DATE Clause	5-67
TIMESTAMP-TIME Clause	5-68
TITLE Clause	5-69
Specifying Field Names in a TITLE Clause	5-69
Spacing Considerations	5-69
Title for Current Report or All Reports	5-70
Overriding Session-Wide Title	5-70
TOTAL Clause	5-71
WHERE Clause	5-72
Using the WHERE Clause to Establish a Link	5-72

6. Commands

?ASSIGN Command	6-3
?ATTACH Command	6-6
?COMPILE Command	6-7
?DICTIONARY Command	6-8
Identifying the Dictionary	6-8
Clearing Internal Tables	6-8
?EDIT Command	6-9
?EXECUTE Command	6-10
?EXIT Command	6-11
?HELP Command	6-12
?OUT Command	6-14
?RUN Command	6-15
?SECTION Command	6-16
?SHOW Command	6-17
?SOURCE Command	6-19

A. Enform Plus Syntax Summary

Language Elements	A-1
Statements	A-2
Clauses	A-5
Commands	A-10
Enform Plus Procedures	A-12

[COBOL Procedures](#) A-12

[FORTRAN Functions](#) A-12

[TAL Functions](#) A-13

B. Error Messages

[Enform Plus Initialization Messages](#) B-2

[!!! Error and *** Warning Type Messages](#) B-4

[*** File Error Type Messages](#) B-22

[Enform Plus Trap Messages](#) B-26

[BUILDMK Error Messages](#) B-26

C. Links and the LINK OPTIONAL Statement Rules

[How Enform Plus Defines a LINK](#) C-1

[Links Initiated by a LINK Statement](#) C-1

[Links Initiated by a LINK OPTIONAL Statement](#) C-2

[Links Initiated by a WHERE Clause](#) C-2

[Links Due to the Transitive Property of Links](#) C-5

[Review of Rules for the LINK OPTIONAL Statement](#) C-6

[Examples of Invalid Links](#) C-7

[Examples of Valid Links](#) C-9

[Comparison of the LINK Statement, the LINK OPTIONAL Statement, and the WHERE Clause](#) C-11

Glossary

Index

Figures

[Figure 1-1. A Typical Enform Plus Session](#) 1-2

[Figure 2-1. Server Query Processor, Several Compiler/Report Writer Processes](#) 2-7

[Figure 3-1. Enform Plus Language Elements](#) 3-2

[Figure 3-2. Records With Duplicate Field Names](#) 3-7

[Figure 3-3. Query Outline of Target-Aggregate With OVER ALL Syntax](#) 3-15

[Figure 3-4. Query Outline of Target-Aggregate With OVER Syntax](#) 3-16

[Figure 3-5. Query Outline of Qualification Aggregate With OVER Syntax](#) 3-18

[Figure 3-6. Qualification Aggregate With Embedded WHERE Clause](#) 3-19

Tables

Table 2-1.	Enform Plus Generic Files and Their Uses (page 1 of 2)	2-11
Table 2-2.	Enform Plus Output Files (page 1 of 2)	2-14
Table 3-1.	Enform Plus Reserved Words and Characters	3-3
Table 3-2.	Special Characters	3-4
Table 3-3.	Arithmetic Operators	3-21
Table 3-4.	Conditional Operators	3-22
Table 4-1.	Summary of Statements (page 1 of 2)	4-1
Table 5-1.	Enform Plus Clauses and Their Functions (page 1 of 3)	5-1
Table 5-2.	Permissible Modifiers and Edit Descriptors	5-15
Table 6-1.	Summary of Commands (page 1 of 2)	6-1
Table 6-2.	Information Displayed by the ?SHOW Command (page 1 of 2)	6-17
Table C-1.	Truth Table	C-11

What's New in This Manual

Manual Information

Abstract

This manual provides detailed information about the syntax of the Enform Plus language.

Product Version

Enform Plus D46

Supported Releases

This manual supports D46.00 and all subsequent D4x releases and G06.00 and all subsequent G-series releases until otherwise indicated in a new edition.

Part Number	Published
422684-001	May 1999

Document History

Part Number	Product Version	Published
422684-001	Enform Plus D46	May 1999

New and Changed Information

This is a new manual.

About This Manual

This manual describes Enform Plus, a language/report generator used for information retrieval. It accesses, sorts, and formats information from a database. Enform Plus can use the Data Definition Language (DDL) to define data format.

This manual describes the language syntax of Enform Plus, which is an enhancement of an earlier product, Enform. An earlier manual, the *Enform User's Guide*, which describes the use of Enform in database management, is still valid for use as a companion to this manual.

The principal differences between Enform Plus and Enform are as follows:

- Enform Plus supports format 2 files in addition to format 1 files. Format 2 files, which became available with the G06.00 and D46.00 software releases, provide larger file and partition sizes and have the potential to exceed the two-gigabyte (less one megabyte) maximum size of format 1 files.
- Enform Plus, in date-related activities, uses four-digit year fields to make it year-2000 compliant.

For additional information about Enform Plus and related products, see the following listed manuals:

- *Data Definition Language (DDL) Reference Manual*
- *Enscribe Programmer's Guide*
- *Guardian Procedure Calls Reference Manual*
- *Guardian Programmer's Guide*
- *Guardian User's Guide*
- *Introduction to Enform*
- *TACL Reference Manual*

Your Comments Invited

After using this manual, please take a moment to send us your comments. You can do this by returning a Reader Comment Card or by sending an Internet mail message.

A Reader Comment Card is located at the back of printed manuals and as a separate file on the User Documentation disc. You can either fax or mail the card to us. The fax number and mailing address are provided on the card.

Also provided on the Reader Comment Card is an Internet mail address. When you send an Internet mail message to us, we immediately acknowledge receipt of your message. A detailed response to your message is sent as soon as possible. Be sure to include your name, company name, address, and phone number in your message. If your comments are specific to a particular manual, also include the part number and title of the manual.

Many of the improvements you see in manuals are a result of suggestions from our customers. Please take this opportunity to help us improve future manuals.

Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

This requirement is described under [Backup DAM Volumes and Physical Disk Drives](#) on page 3-2.

General Syntax Notation

The following list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words; enter these items exactly as shown. Items not enclosed in brackets are required. For example:

MAXATTACH

lowercase *italic* letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

file-name

[] Brackets. Brackets enclose optional syntax items. For example:

TERM [\system-name.] \$terminal-name

INT[ERRUPTS]

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list may be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [  num  ]
   [ -num  ]
   [ text  ]
```

K [X | D] address-1

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list may be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                   { $process-name  }
```

ALLOWSU { ON | OFF }

| **Vertical Line.** A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

... **Ellipsis.** An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address-1 [ , new-value ]...
```

```
[ - ] { 0|1|2|3|4|5|6|7|8|9 }...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be entered as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;
```

```
LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must enter as shown. For example:

```
"[ repetition-constant-list ]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In the following example, there are no spaces permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] CONTROLLER
```

```
    [ , attribute-spec ]...
```

1 Introduction

This section introduces the content of this manual and the terminology used by Enform Plus. This manual documents the following information:

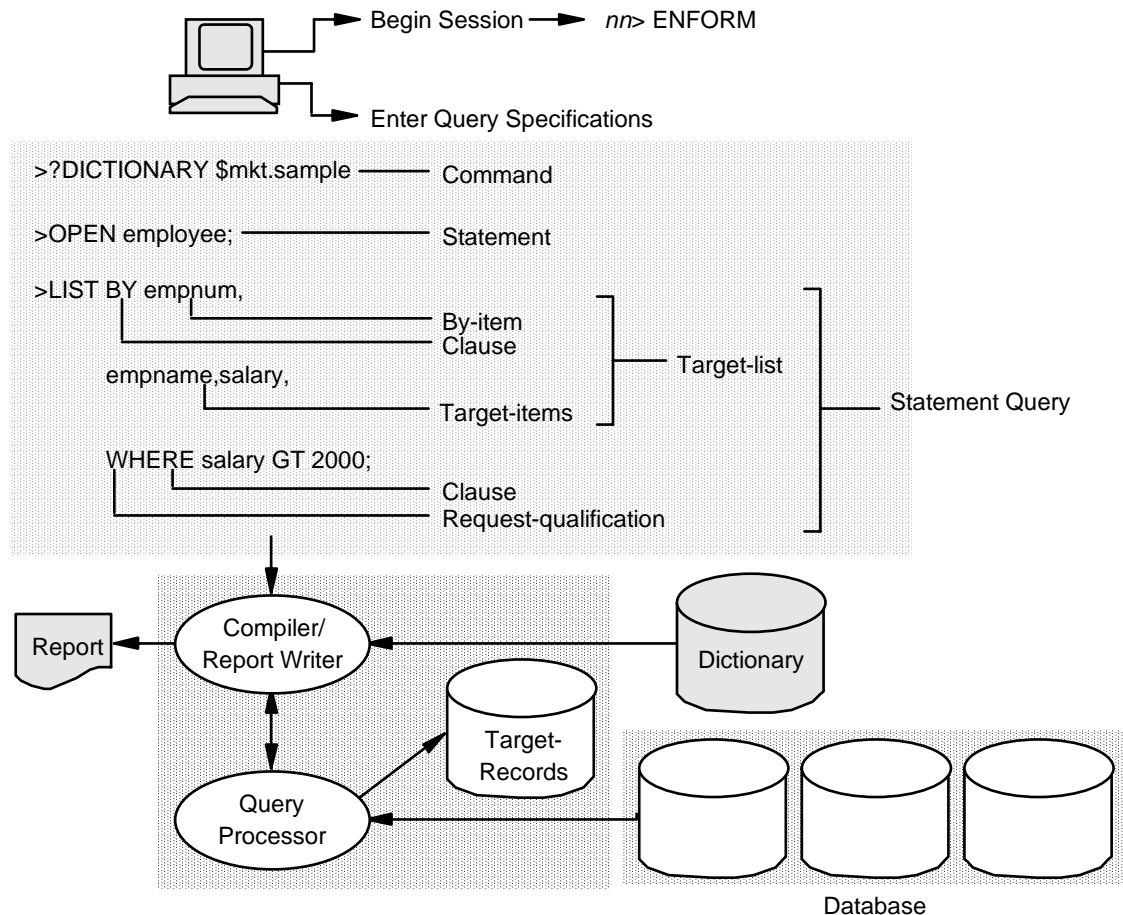
- The syntax of the TACL command ENFORM needed to call the Enform Plus process, the TACL commands needed to create a server query processor, and the names of generic files that can be assigned using either the TACL ASSIGN command or the Enform Plus ?ASSIGN command.
- The syntax of the Enform Plus language elements including statements, clauses, commands, aggregates, expressions, literals, and variables. These elements are alphabetized wherever appropriate.
- The error messages generated by Enform Plus.

The special features of Enform Plus, such as using the host language interface, writing an Enform Plus server, or redefining Enform Plus reserved words and message text are not discussed. For this information and for information about using the Enform Plus language elements, see the *Introduction to Enform* or the *Enform User's Guide*.

The examples in this manual use the database described in the *Enform User's Guide*.

Enform Plus Terminology

This manual uses special terms to describe the various components and features of the Enform Plus language. [Figure 1-1](#) on page 1-2 illustrates these terms by showing a typical Enform Plus session (the period of time that begins when you enter the ENFORM command and ends when you exit from Enform Plus). The following paragraphs describe some of the terms illustrated in [Figure 1-1](#). You should become familiar with these terms, as they are used throughout the manual.

Figure 1-1. A Typical Enform Plus Session

CDT 001CDD

The important terms are:

Query specifications. The language elements (statements, clauses, commands, and so on) that you specify to provide Enform Plus with the information it needs to retrieve data and to establish the query environment.

Query. One complete LIST or FIND statement. Both statements specify the information to be retrieved.

Target-list. A part of the query; a target-list consists of target-items and a special type of target-item called a by-item.

Target-items. Any record names, field names, variable names, aggregate names, literals, or expressions, excluding by-items, whose values you want to appear as output from your query.

By-items. Field names modified by the BY or BY DESC clauses described in [Section 5, Clauses](#). The values of the fields are used to sort and group the query output.

Request-qualification. A condition or conditions that a database element must meet before it is selected to contribute to your query output. A request-qualification begins with a WHERE keyword followed by a logical expression.

Compiler/report writer. The Enform Plus process that compiles your query and writes a report (if one is requested). The compiler/report writer issues error messages for syntax errors. If no errors exist, the compiler/report writer passes your compiled query specifications along with information obtained from the dictionary to the query processor. After the query processor returns the retrieved data (in the form of target-records, described later on this page), the compiler/report writer formats and writes the report.

Dictionary. Physical files that contain information called record descriptions. A record description provides Enform Plus with information about the name of the record being accessed, the name and data type of any fields within the record, the record and field length, the name of any primary or alternate key fields, and the name of the physical file associated with the record description. The Data Definition Language (DDL) compiler creates the dictionary from record descriptions written in DDL. The dictionary must exist before your query specifications are processed. See the *Data Definition Language (DDL) Reference Manual* and the *Enform User's Guide* for more information about the dictionary.

Query processor. The Enform Plus process that uses the information provided by the compiler/report writer to retrieve information from the database. The query processor also performs other functions such as creating a new physical file and transmitting records to a host language program. Creating a new physical file is described with the [FIND Statement](#) on page 4-16 of this manual. The host-language interface is described in the *Enform User's Guide*.

Target-records. The records built by the query processor from which your Enform Plus output is produced. The query processor returns the target-records to the compiler/report writer if the Enform Plus output is to be formatted and written as a report.

Database. The collection of physical files from which the query processor retrieves data. Any physical file from which data is retrieved must be associated with the record description obtained from the dictionary.

2

Running Enform Plus

This section describes the initiation of an Enform Plus session, the scope of such a session, the action of the query processor, TACL commands and macros that control and influence the session, and the files used by the session.

The ENFORM command issued from TACL calls the Enform Plus subsystem. If no parameters are specified, the ENFORM command appears as:

```
nn> ENFORM
```

The *nn* in this example represents the history number that is displayed as part of every TACL prompt. The terminal from which the ENFORM command is entered is called the home terminal. The syntax for the ENFORM command is as follows:

```
ENFORM [ / [ IN input-filename ]  
        [ , [ OUT output-filename ] ] / ]  
        [ dict-subvol-name ] [ , message-table-filename ]
```

IN input-filename

specifies either the name of an Edit-format file containing Enform Plus source code or the name of a compiled query file.

If this option is specified, Enform Plus executes the code in the specified file and returns you to the TACL prompt at the end of execution.

If this option is omitted, the Enform Plus prompt (>) appears and you can enter commands and statements either directly by typing them or indirectly by specifying either the ?RUN or ?SOURCE commands. When this option is omitted, the home terminal becomes the default input file.

OUT output-filename

specifies the name of the physical file to which output is directed. If this option is omitted, Enform Plus directs the output to the current output listing file (described later in this section.)

dict-subvol-name

is the name of the volume and subvolume upon which the dictionary resides. If this parameter is omitted, Enform Plus assumes the dictionary resides on the current volume and subvolume. Either the DICTIONARY statement or the ?DICTIONARY command can be used to supply or change the dictionary name.

message-table-filename

is the name of the key-sequenced file containing a user-defined Enform Plus message table. Enform Plus retrieves error and informational message text, help message text, and/or the list of any redefined reserved words, system variables, option variables, or command names from this file when this parameter is specified. If this parameter is omitted, Enform Plus retrieves message text from a message

table supplied by Compaq. See the *Enform User's Guide* for information on creating a user-defined message table.

The TACL ASSIGN command (described on page [2-7](#)) can be used to make up to 32 logical file assignments before the Enform Plus subsystem is called.

ENFPLUS Macro

The ENFORM command described previously is the same command used to run the older Enform subsystem. To ensure that the Enform Plus executables, rather than those of Enform, are invoked by the ENFORM command (as well as by QP and BUILDMDK), you must first run the TACL macro ENFPLUS, supplied with the Enform Plus product in the ENFPMAC file, before issuing the ENFORM command.

The TACL commands required to execute the ENFPLUS macro are as follows:

```
nn> LOAD / KEEP 1 / $SYSTEM.SYSTEM.ENFPMAC
nn> ENFPLUS
```

If at any time you want to revert to using the older Enform subsystem, you can run another macro, ENFOLD, that is also supplied with Enform Plus. The procedure is similar to that described for ENFPLUS:

```
nn> LOAD / KEEP 1 / $SYSTEM.SYSTEM.ENFPMAC
nn> ENFOLD
```

You must be logged on as the super ID user to be able to run either the ENFPLUS or the ENFOLD macro.

Interactive Mode

Enform Plus functions in interactive mode when the Enform Plus source code is entered from a terminal keyboard. Enform Plus prompts for input by printing the right angle bracket (>). When you enter a carriage return, Enform Plus issues another prompt. For example:

```
22> ENFORM
Enform^Plus - T0295D46 - (15APR98)DATE - TIME : 5/03/1999 - 21:24:58
COPYRIGHT TANDEM COMPUTERS INCORPORATED 1979, 1983, 1991, 1992, 1998
>
```

Enform Plus commands and statements can be entered either directly or indirectly. Commands and statements are entered directly when you type them in response to the Enform Plus prompt. Commands and statements are entered indirectly when you use either the ?RUN or ?SOURCE commands to supply the Enform Plus source code.

When you enter commands and statements directly, the TACL FC command allows you to edit or repeat a line. See the *TACL Reference Manual* for more information about this command.

When you specify the ?EDIT command, the Edit prompt (*) appears and all the functions of the Edit process are available for your use. Either the ?RUN or the

?SOURCE command can be used to execute the source code created in the Edit process. These commands are described in [Section 6, Commands](#).

Exit interactive mode by entering the EXIT statement, the ?EXIT command, or by pressing the Ctrl and Y terminal keys simultaneously.

Noninteractive Mode

Enform Plus functions in noninteractive mode when commands and statements are entered through an input file other than a terminal. The input file can be an Edit-format file or a compiled query file. For example:

```
nn> ENFORM /IN input-filename, OUT output-filename/
```

In this example, Enform Plus reads the commands and statements from the input file.

When Enform Plus functions in noninteractive mode, all commands, statements, and clauses must be part of the input file. When the file specified in the IN option is a compiled query file, values for parameters can be specified by using the PARAM command of TACL prior to the ENFORM command. More information about passing parameters to compiled Enform Plus queries appears later in this section.

Enform Plus terminates when an end-of-file, EXIT statement, or ?EXIT command is encountered on the input file.

The Current Output Listing File

The current output listing file is the file to which Enform Plus directs output. During the course of an Enform Plus session, the current output listing file can change.

At the beginning of an Enform Plus session, the current output listing file is the default output file. Enform Plus determines the default output file by the following process:

- If the OUT option is included in the ENFORM command, the default output file is the file specified in the OUT option.
- If the OUT option is omitted from the ENFORM command, the default output file is the file specified in the IN option of the ENFORM command if that file is a terminal.
- If the file specified for the IN option is not a terminal, the default output file is the home terminal.
- If both the IN option and the OUT option are omitted from the ENFORM command, the default output file is the home terminal.

As the session progresses, the current output listing file might change as follows:

- If a QUERY-COMPILER-LISTING file is specified, that file becomes the current output listing file whenever Enform Plus commands and statements are being processed.
- If either a QUERY-REPORT-LISTING file or an ?OUT command file is specified, that file becomes the current output listing file whenever a report (the output from a LIST statement) is being processed.

The QUERY-COMPILER-LISTING file and the QUERY-REPORT-LISTING file are described under [Generic Files](#) on page 2-10. The ?OUT command file is described on page [6-14](#).

Pressing the Terminal Break Key

Enform Plus acknowledges the terminal Break key whenever the input file is a terminal. The input file is a terminal under the following conditions:

- The IN option is omitted from the ENFORM command making the home terminal the default input file.
- The terminal (whether the home terminal or another terminal) is the file specified in the IN option of the ENFORM command.

Pressing the terminal Break key on any terminal, including the home terminal, has no effect unless the terminal is the input file.

The current activity determines the action taken as follows:

- Pressing the Break key when Enform Plus is processing a query, producing a report, or producing output from the ?SHOW command, returns the terminal to the Enform Plus prompt (>). Query execution and any output terminates. If you press the Break key while Enform Plus is performing a sort operation, Enform Plus does not respond to the Break key until the sort operation finishes.
- Pressing the Break key when commands or statements are being entered (either directly or indirectly), returns the terminal to the TACL prompt (nn>).
- Pressing the Break key when you have entered the Editor process from the Enform Plus prompt has the same effect as when the Editor process is entered from the TACL prompt with one exception: pressing the terminal Break key at the Editor's prompt (*) has no effect.

If the @BREAK-KEY option variable (described on page [5-48](#)) is set to OFF, pressing the Break key returns the terminal to the TACL prompt. It terminates neither output nor query execution. In this case, Enform Plus continues to run and any output directed to the terminal is temporarily interrupted. The TACL PAUSE command can be issued to resume output.

Logical File Assignments

Enform Plus allows logical file assignments to be made either before or after the TACL ENFORM command is entered. When the logical file assignment is made from within the Enform Plus subsystem, use the Enform Plus ?ASSIGN command described on page [6-3](#). When the logical file assignment is made before the ENFORM command is entered, use the TACL ASSIGN command to either associate a physical file with a dictionary record description or to assign some form of Enform Plus output to a generic file.

When Enform Plus is used in noninteractive mode, the TACL ASSIGN command overrides an Enform Plus ?ASSIGN command that is part of the input file. When

Enform Plus is used in interactive mode, the Enform Plus ?ASSIGN command overrides any TACL ASSIGN commands.

The syntax of the TACL ASSIGN command is as follows:

```
ASSIGN [ { record-name } , physical-file-name ]
       [ { generic-file-name }
       [
         [ , exclusion-mode ]
       ]
       ]
```

record-name

is the name of a dictionary record description.

generic-file-name

is the name of one of the following generic files: QUERY-WORK-AREA, QUERY-SORT-AREA, QUERY-QPSTATISTICS, and QUERY-QPSTATUS-MESSAGES. All other names are ignored.

physical-filename

is a fully qualified Guardian file name. See the *Guardian User's Guide* for the exact form of a Guardian file name.

exclusion-mode

is either SHARED, PROTECTED, or EXCLUSIVE. The default is SHARED for an input file. Valid values for generic files are described later in this section.

See the *TACL Reference Manual* for more information about the ASSIGN command.

Passing Parameters to Compiled Query Files

Enform Plus allows you to pass parameters to a compiled query file. The compiled query file must contain a PARAM statement defining the parameter. The PARAM statement is described on page [4-39](#).

Use the TACL PARAM command to specify parameters prior to execution of the compiled query file. The PARAM command overrides any values specified in a SET statement for a parameter. The syntax of the TACL PARAM command is:

```
PARAM [ parameter-name parameter-value ] , ...
```

parameter-name

is the name of a parameter defined in a PARAM statement.

parameter-value

is the value to be assigned to parameter-name. parameter-value can have either of the following forms:

character-string
"*character-string*"

If the first form is used, the string must not contain any embedded commas, and leading and trailing blanks are not included as part of *parameter-value*.

If the second form is used, all the characters, including leading and trailing blanks, between the quotation marks are included as part of *parameter-value*.

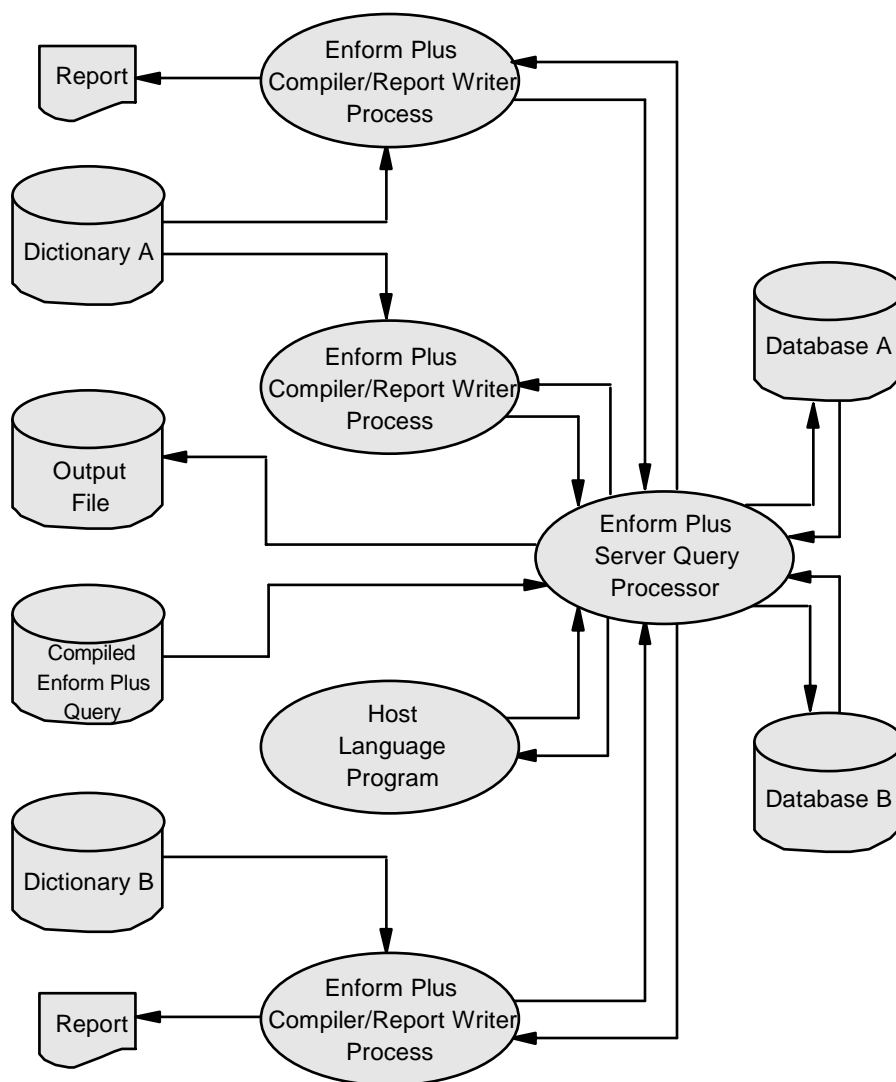
See the *TACL Reference Manual* for more information about the PARAM command.

A Server Query Processor

Unless otherwise specified, each Enform Plus session uses a dedicated query processor. To avoid some overhead, several compiler/report writer processes can be assigned to share a single server query processor and sort process. The assignment is made by an ?ATTACH command. A server query processor processes one query at a time. [Figure 2-1](#) on page 2-7 shows several compiler/report writer processes assigned to one server query processor.

One or more server query processors can be created. Each server query processor runs as a NonStop process pair, handling queries from one compiler/report writer process at a time.

A host-language program (described in the *Enform User's Guide*) can also use a server query processor.

Figure 2-1. Server Query Processor, Several Compiler/Report Writer Processes

CDT 002CDD

Enform Plus server query processors are created by a system manager, using the TACL ASSIGN and PARAM commands followed by the TACL QP command.

The TACL ASSIGN Command

For every physical file an Enform Plus application accesses, the server query processor must do open/close operations that add to the processing time. This processing time can be reduced significantly if Enform Plus applications that frequently use the same physical files are processed by a common server query processor. A server query processor allows for heavily accessed physical files to be kept open.

The TACL ASSIGN command defines the physical file or files kept open. The syntax of this command is as follows:

```
ASSIGN { Fnumber
        { generic-file-name }
        [ , exclusion-mode ]
```

Fnumber

specifies the logical file name. Valid values for logical file names range from F1 through F31.

generic-file-name

is the name of one of the following generic files: QUERY-WORK-AREA, QUERY-SORT-AREA, QUERY-QPSTATISTICS, and QUERY-QPSTATUS-MESSAGES. All other names are ignored.

physical-filename

is a fully qualified Guardian file name. See the *Guardian User's Guide* for the exact form of a Guardian file name.

exclusion-mode

is one of SHARED, PROTECTED, or EXCLUSIVE. The default is SHARED for an input file. Valid values for generic files are described later in this section.

The TACL PARAM Command

To create a server query processor, you must include a PARAM command specifying a REQUESTORS parameter. The REQUESTORS parameter defines the maximum number of requestors the server query processor can accept. The other parameters are optional and can be specified in the same or in a different PARAM statement.

```
PARAM REQUESTORS max-requestors [ , COST max-cost ]
                                     [ , TIMEOUT time-out ]
                                     [ , READS max-reads ]
```

REQUESTORS *max-requestors*

sets the maximum number of requestors a query processor can accept. A requestor can be any Enform Plus application or a host-language program. (Host-language programs are described in the *Enform User's Guide*.)

COST *max-cost*

sets a strategy cost limit for each Enform Plus query using this server query processor. If an Enform Plus query exceeds the limit, it is terminated and an error

message is displayed. See the @COST-TOLERANCE Option Variable in the Option Variable clauses on page [5-48](#) for an explanation of strategy cost limits.

max-cost must be an integer between one and eight. The default is no limit.

TIMEOUT *time-out*

sets the number of minutes a server query processor sits idle before stopping itself. The default is no limit, meaning the query processor continues to run indefinitely.

READS *max-reads*

sets the maximum number of logical database reads per Enform Plus session. If that many reads are performed, the Enform Plus session is terminated and an error message is displayed.

max-reads must be an integer. The default is no limit.

The TACL QP Command

After the parameters are initialized and the physical files to be kept open are specified, the server query processor is created. The syntax is:

```
QP / [ NOWAIT ] , NAME process-name [ , CPU number ]
    [ , PRI priority ] [ , MEM pages ] /
```

NAME *process-name*

is a process name for the server query processor. The name must begin with a dollar sign (\$) followed by an alphabetic character and one to four alphanumeric characters.

CPU *number*

is the number of the CPU where this server query processor resides. The default is the same CPU where TACL resides.

PRI *priority*

is the priority at which this server query processor is to run.

MEM *pages*

is the maximum number of virtual data pages used for this server query processor. It must be an integer from one to 64. The default is 64.

Example of Server Query Processor Creation

Issue the following instructions to create a server query processor named `$qp1` that does the following:

- Keeps open `parts`, `order`, and `odetail`
- Accepts up to 15 requestors
- Sets a limit of 2 on the cost strategy
- Waits up to three minutes before stopping

```
27> ASSIGN F1, $data.database.parts
28> ASSIGN F2, $data.database.order
29> ASSIGN F3, $data.database.odetail
30> PARAM REQUESTORS 15, COST 2, TIMEOUT 3
31> QP / NOWAIT, NAME $qp1 /
```

The server query processor keeps open those files that have been assigned OFO names. In the example, these are F1, F2, and F3. The query itself still refers to these files by their names as described in the data dictionary: `parts`, `order`, and `odetail`.

(Note that `FIND` files cannot be kept open in this manner.)

Generic Files

A generic file is a file that is used to store some form of Enform Plus output. Enform Plus produces many different forms of output, such as statistics and error messages. To enable control over each class of output, generic file names have been defined for each class. These generic file names can be used in `TACL ASSIGN` commands or Enform Plus `?ASSIGN` commands (described on page [6-3](#)) as the record-name to be assigned to a physical-file-name.

When you assign an Enform Plus generic file to a physical file, the physical file must exist at the time Enform Plus attempts to open the file. If you specify an exclusion mode, it is used. An unspecified or meaningless exclusion mode (for example, `protected` for terminals) causes the default (`SHARED` for terminals and `EXCLUSIVE` for other devices) to be used. If you specify an exclusion mode for the generic files `QUERY-WORK-AREA` and `QUERY-SORT-AREA`, it is ignored.

Assigning a generic output file name to a process name causes the process to be treated as if it were the spooler. Enform opens the process, calls `SETMODE`, and writes to the process. It is not possible to assign a `FIND` file to a process because Enform Plus will create an unstructured file and rename the file as the last step in processing the `FIND` statement. [Table 2-1](#) on page 2-11 lists the Enform Plus generic file names and their uses.

Table 2-1. Enform Plus Generic Files and Their Uses (page 1 of 2)

Generic File Name	Description
QUERY-COMPILER-LISTING	<p>All compilation output produced during an Enform Plus session (that is, entering a query either directly or indirectly); compiler errors and warnings; including output, errors, and warnings from other Enform Plus commands (such as ?ASSIGN and ?SHOW).</p> <p>Output is produced in ASCII with a record length of 132Ebytes.</p>
QUERY-REPORT-LISTING	<p>All reports produced as a result of the execution of a LIST statement.</p> <p>Output is produced in ASCII with a record length of 132 bytes.</p>
QUERY-STATISTICS	<p>All statistics produced during an Enform Plus session by the query processor while processing a FIND or LIST statement when @STATS is set to ON. Several sets of statistics can be produced during a session. A set is identified by the requestor's PID and the beginning and ending times of the query execution.</p> <p>Output is produced in ASCII. For each query processed, the output consists of (72 * (the number of record-types in the query + 3)). For display purposes, each output record is 72 characters long.</p>
QUERY-STATUS-MESSAGES	<p>All compiler/report writer error and warning messages produced during an Enform Plus session. All error messages produced during an Enform Plus session by the query processor or SORT while processing a FIND or LIST statement. These messages also appear in the listings.</p> <p>The output is produced in ASCII with a record length of 132 bytes.</p>
QUERY-WORK-AREA	<p>The volume where all temporary files (except SORT work files) are built by the query processor while processing a FIND or LIST statement during an Enform Plus session. The QUERY-WORK-AREA file name should contain only a volume name.</p>

Table 2-1. Enform Plus Generic Files and Their Uses (page 2 of 2)

Generic File Name	Description
QUERY-SORT-AREA	The location where all temporary files are built by the SORT process while processing a FIND or LIST statement during an Enform Plus session. The QUERY-SORT-AREA file name can be a volume name or an explicit file name.
QUERY-QPSTATISTICS	The statistics produced for every FIND or LIST statement that is successfully processed by the query processor during an Enform Plus session (regardless of the setting of @STATS). Each set of statistics is identified by a line containing the requestor's PID and the beginning and ending times of the query execution. If @STATS is set to ON, statistics are also reported to the QUERY-STATISTICS file. The output is produced in ASCII. For each query processed, the output consists of (72 * (the number of record-types in the query + 3)). For display purposes, each output record is 72 characters in length.
QUERY-QPSTATUS-MESSAGES	All error messages produced by the query processor or SORT during the processing of a FIND or LIST statement during an Enform Plus session. To identify the query in error, each message is preceded by the requestor's PID and the time of the error. Errors are also reported to the QUERY-STATUS-MESSAGES file. The output is produced in ASCII with a record length of 132 bytes.

In most cases, a generic output file is an unstructured file. When Enform Plus opens an unstructured file to write records, the records are written starting at the beginning of the file and the existing records are overwritten.

Generic Files and a Dedicated Query Processor

Generic files QUERY-WORK-AREA, QUERY-SORT-AREA, QUERY-QPSTATISTICS, and QUERY-QPSTATUS-MESSAGES can be specified for a dedicated query processor by using either the Enform Plus ?ASSIGN command (see page [6-3](#)) or the TACL ASSIGN command (see [Logical File Assignments](#) on page 2-4). The ASSIGN command is passed through the Enform Plus compiler/report writer to the query processor for each LIST or FIND statement until the ASSIGN is cleared.

The query processor opens both QUERY-QPSTATISTICS and QUERY-QPSTATUS-MESSAGES for every FIND or LIST request. Generally, statistics and errors for multiple requests to a dedicated query processor cannot be collected in a disc file because the results are written over each other. However, by assigning the generic file to a process, the statistics and errors can be collected for multiple requests to a dedicated query processor.

Generic Files and the Server Query Processor

The generic files QUERY-WORK-AREA, QUERY-SORT-AREA, QUERY-QPSTATISTICS, and QUERY-QPSTATUS-MESSAGES can be held open for the lifetime of a server query processor if the TACL ASSIGN command (see [A Server Query Processor](#) on page 2-6) is specified at the time the server query processor is created.

By using the generic file names with a server query processor, you can specify where the server query processor: builds the temporary files, performs sorts, sends statistics, and sends error messages. Because the QUERY-QPSTATISTICS and QUERY-QPSTATUS-MESSAGES files are held open for the lifetime of the associated server query processor, collection of statistics and errors for multiple requests is possible.

The TACL ASSIGN commands specified when the server query processor is started cannot be overridden by either Enform Plus ?ASSIGN commands nor TACL ASSIGN commands specified when the Enform Plus process is initiated. (The server query processor must be stopped and restarted to alter these file assignments.)

If you assign generic files by specifying either the Enform Plus ?ASSIGN command (during Enform Plus processing) or the TACL ASSIGN command (before Enform Plus is initiated) the server query processor directs output to these generic files until you clear the ASSIGN command. Generic files that were not assigned when the server query processor was created are re-opened for each FIND or LIST statement.

Generic Files and the Current Output Listing File

If the Generic files QUERY-REPORT-LISTING and QUERY-COMPILER-LISTING are assigned, these files become the current output listing files under the following conditions:

- QUERY-COMPILER-LISTING file is the current output listing file whenever Enform Plus statements and commands are processed.
- QUERY-REPORT-LISTING file is the current output listing file whenever a report (the output from the LIST statement) is produced unless the ?OUT command specifies an ?OUT file.

Assignment of any of the other generic files does not affect the current output listing file.

[Table 2-2](#) on page 2-14 lists the forms of Enform Plus output (that is, reports, statement and command output, statistics, and error messages) and the corresponding output file when generic files are assigned. The default output file is described earlier in this section.

Table 2-2. Enform Plus Output Files (page 1 of 2)

Enform Plus Output	Output File
Enform Plus banner and trailer Commands Statements Command output (e.g., ?SHOW)	Output is directed to: <ol style="list-style-type: none"> 1. The QUERY-COMPILER-LISTING file, if assigned; otherwise, 2. The default output file.
Report from a LIST statement	Output is directed to: <ol style="list-style-type: none"> 1. The ?OUT file, if specified; otherwise, 2. QUERY-REPORT-LISTING file, if assigned; otherwise, 3. The default output file.
Statistics @STATS on	Output is directed to the QUERY-QPSTATISTICS file, if assigned, in addition to the following: <ol style="list-style-type: none"> 1. QUERY-STATISTICS file, if assigned; otherwise, 2. QUERY-COMPILER-LISTING file, if assigned; otherwise, 3. The default output file.
@STATS off	If @STATS is off, output is sent to: <ol style="list-style-type: none"> 1. QUERY-QPSTATISTICS file, if assigned; otherwise, 2. The output is not written.

The numbers in the Output File column indicate the order in which Enform Plus directs output to the files. If the file with the number 1 exists, Enform Plus directs output to this file only. If this file does not exist, Enform Plus directs output to the file with the number 2, and so on.

Table 2-2. Enform Plus Output Files (page 2 of 2)

Enform Plus Output	Output File
Error messages from the command processor and query compiler	<p>Output is sent to the QUERY-STATUS-MESSAGES file, if assigned, in addition to the following:</p> <ol style="list-style-type: none"> 1. QUERY-COMPILER-LISTING, if assigned; otherwise, 2. The default output file. <p>Note that if the QUERY-STATUS-MESSAGES file is not assigned and the QUERY-COMPILER-LISTING file is not the default output file, the error messages are also written to the default output file.</p>
Error messages from the report writer	<p>Output is sent to the QUERY-STATUS-MESSAGES file, if assigned, in addition to the following:</p> <ol style="list-style-type: none"> 1. ?OUT file, if specified; otherwise, 2. QUERY-REPORT-LISTING, if assigned; otherwise, 3. The default output file. <p>Note that if the QUERY-STATUS-MESSAGES file is not assigned and the list file is not a default output file, the error messages are also written to the default output file.</p>
Error messages from the query processor or SORT	<p>The QUERY-QPSTATUS-MESSAGES file, if assigned, and to the QUERY-STATUS-MESSAGES file, if assigned, in addition to the following:</p> <ol style="list-style-type: none"> 1. QUERY-COMPILER-LISTING, if assigned; otherwise, 2. The default output file. <p>Note that if the QUERY-STATUS-MESSAGES file is not assigned and the current output listing file is not the default output file, the error messages are also written to the default output file.</p>

The numbers in the Output File column indicate the order in which Enform Plus directs output to the files. If the file with the number 1 exists, Enform Plus directs output to this file only. If this file does not exist, Enform Plus directs output to the file with the number 2, and so on.

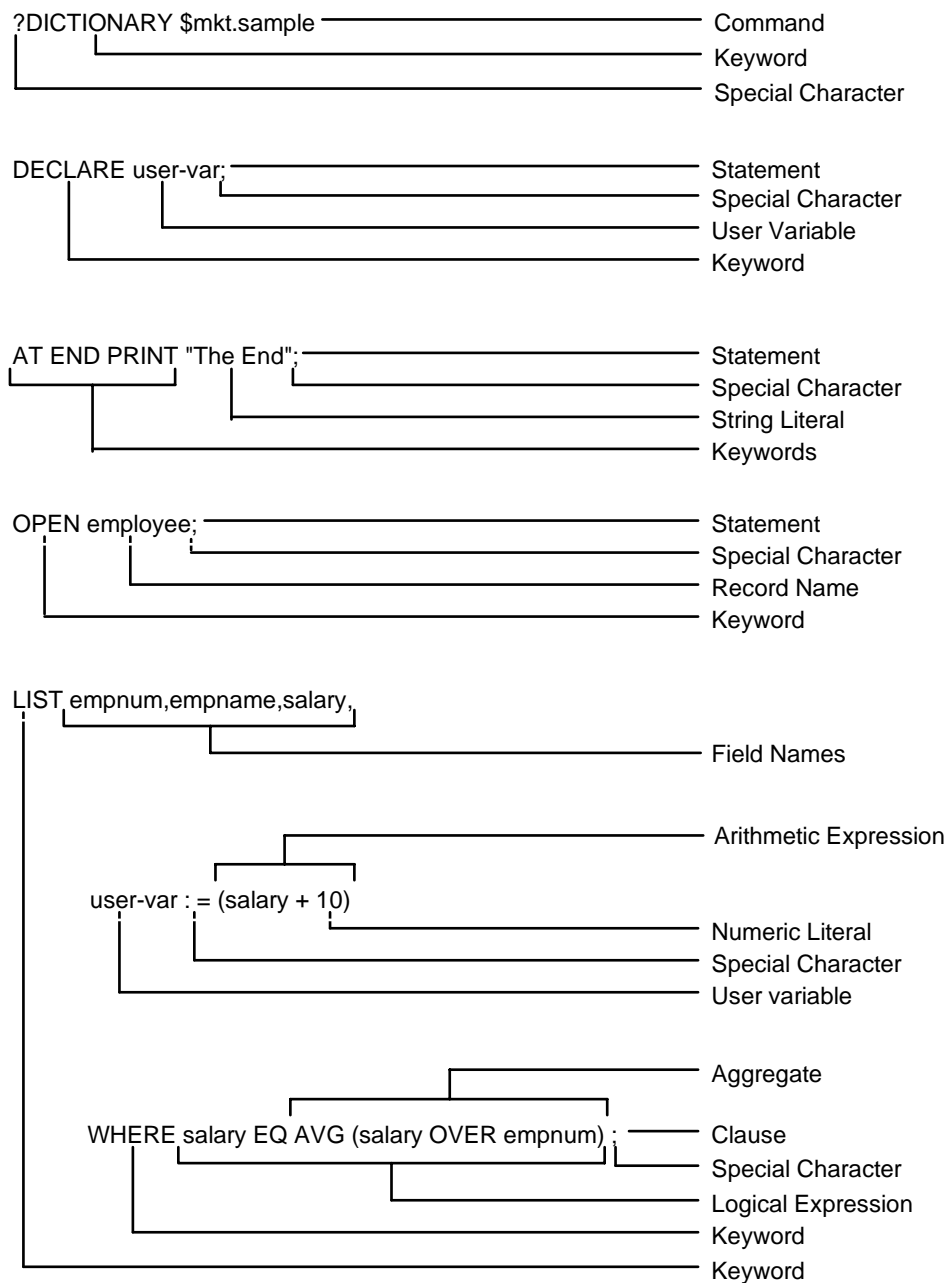
3

Enform Plus Language Elements

An Enform Plus query is built with elements of the Enform Plus language. This section contains the following:

- An explanation of the Enform Plus language elements that are used throughout query specifications. These elements are reserved words, special characters, and comments.
- A brief explanation of the functions of the Enform Plus statements, clauses, and commands. The syntax of these language elements is described in [Section 4, Statements](#), [Section 5, Clauses](#), and [Section 6, Commands](#).
- The syntax and functions of the Enform Plus language elements that can be used either in a target-list or in request-qualification. (Both target-lists and request-qualification are defined in [Section 1, Introduction](#).) These language elements include aggregates, literals, arithmetic expressions, logical expressions, IF/THEN/ELSE expressions, parameters, user variables, and user tables.
- The rules to be used when referencing database records, fields, and primary keys, when including subscripts, and when naming user-defined elements such as user variables, user tables, user aggregates, or parameters.

[Figure 3-1](#) on page 3-2 shows a query specification and some of the language elements described in this section.

Figure 3-1. Enform Plus Language Elements

CDT 003CDD

Reserved Words

Reserved words are words with special meaning to Enform Plus. Both in syntax and elsewhere in this manual, reserved words are shown in uppercase characters. Reserved words must be spelled exactly as shown.

Do not use reserved words to name records, fields, variables, tables, or parameters. Reserved words can be redefined or translated to a language other than English. See the *Enform User's Guide* for instructions on redefining the Enform Plus reserved words. [Table 3-1](#) lists the Enform Plus reserved words.

Table 3-1. Enform Plus Reserved Words and Characters

AFTER	DELINK	LESS	PRINT	VIA
ALL	DESC	LINK	SET	WHERE
AND	DICTIONARY	LIST	SKIP	WITH
AS	ELSE	LT	SPACE	ZERO
ASCD	END	MAX	START	ZEROS
AT	EQ	MIN	SUBFOOTING	'
AVG	EQUAL	NE	SUBTITLE	(
BEFORE	EXIT	NOHEAD	SUBTOTAL)
BEGINS	FIND	NOPRINT	SUM	*
BLANK	FOOTING	NOT	SUPPRESS	+
BLANKS	FORM	NULL	TAB	-
BY	GE	OF	THAN	.
CENTER	GREATER	OFF	THEN	/
CHANGE	GT	ON	THRU	;
CLOSE	HEADING	OPEN	TIME	<
CONTAINS	IF	OPTION	TIMESTAMP-DATE	=
COPY	INTERNAL	OPTIONAL	TIMESTAMP-TIME	>
COUNT	IS	OR	TITLE	@
CUM	JULIAN-DATE	OVER	TO	[
DATE	KEY	PARAM	TOTAL]
DECLARE	LE	PCT	UNIQUE	!

Special Characters

[Table 3-2](#) lists the Enform Plus special characters and describes their functions within a query.

Table 3-2. Special Characters

Character Name	Character	Description
Apostrophe	'	Serves as a delimiter for display formats, some conditional operators.
Assignment syntax	:=	Assigns a value to a target-item.
Blank		Separates keywords and other language elements.
Brackets	[]	Delimits subscripts, modifiers, and decorations. Must appear in balanced pairs.
Comma	,	Separates multiple query specifications in the same statement; always optional.
Parentheses	()	Delimits various language elements. Must appear in balanced pairs.
Question mark	?	Denotes a command when directly followed by a keyword.
Quotation mark	"	Serves as delimiter for various language elements, such as alphanumeric literals, and some display formats.
Semicolon	;	Terminates statements.

Comments

A comment clarifies and documents the purpose of the your query. A comment is denoted by the exclamation character (!). A comment can be the only text on a line, the last text on a line, or text embedded within a line. When a comment is embedded within a line, it must be enclosed with exclamation marks. For example, consider the following comments:

```
!This query produces Finance Report 301
DICTIONARY finance.subvol; ! Information is confidential
OPEN financel, !Only two files are needed! finance2;
```

Statements

Statements contain specifications for selecting and formatting elements from your database. Composed of keywords, clauses, and target-lists, the Enform Plus statements LIST and FIND provide the basic specifications for information selection. Additional statements establish the query environment and provide some report structuring capability.

With the exception of the `LIST` and `FIND` statement, Enform Plus statements remain in effect (unless canceled, reset, or overridden) for the duration of an entire Enform Plus session. Enform Plus requires the `LIST` and `FIND` statements to be terminated with a semicolon. The other Enform Plus statements should be terminated with a semicolon because Enform Plus does not report errors until it encounters either a terminating semicolon or the beginning of a new statement.

See [Section 4, Statements](#) for the syntax of Enform Plus statements.

Clauses

Clauses are optional elements of Enform Plus statements. With the exception of the option variable clauses and the system variable clauses, Enform Plus clauses apply only to the `LIST` or `FIND` statement of which they are a part.

Some of the operations performed by Enform Plus clauses are:

- Sorting and grouping target-records
- Calculating subtotals, totals, percentages, and running totals
- Printing user supplied information within a report
- Formatting a report
- Extracting the current date, time, line number, and page number
- Converting data to internal or display format

See [Section 5, Clauses](#) for the syntax of Enform Plus clauses.

Commands

Commands are compiler directives that tell the compiler/report writer to perform a specific action. For example, commands tell the compiler/report writer to:

- Associate a new physical file with a record description.
- Attach a specific query processor.
- Enter the text editor without leaving Enform Plus.
- Compile a program and save it in a compiled query file.
- Compile and execute an Edit-format file containing source code.
- Execute a compiled query file.
- Include part of an Edit-format file in the input to Enform Plus.
- Display information about the current Enform Plus environment.

See [Section 6, Commands](#) for the syntax of the Enform Plus commands.

Rules for Naming User-Defined Elements

When you name variables, tables, aggregates, or parameters, the name:

- Must be unique
- Must start with either an alphabetic character or a circumflex (^)
- Can contain numbers, hyphens (-), or circumflexes (^)
- Can be from 1 to 31 characters in length
- Must not contain embedded blanks
- Must not end with a hyphen (-)

Rules for Referencing Database Elements

When you reference a database element within your query, you must follow certain rules. The rules used to reference a record name, a field name, and a primary key are described in the following paragraphs.

Record Name References

When you reference a record name within a query, the record name must be unique. If a record name is the same as a field name in an open record description, Enform Plus operates as if the unqualified reference refers to the field name.

Referencing a record name as a target-item is the same as referencing each occurrence of each of the fields individually. A record name cannot be specified as an element in a print list. (A print list is part of the AT END statement and clause, the AT START statement and clause, the FOOTING statement and clause, the SUBFOOTING statement and clause, the SUBTITLE statement and clause, the TITLE statement and clause, and the BEFORE CHANGE and AFTER CHANGE clauses.)

Field Name References

The same field name can exist in more than one database file. If your query involves database files with duplicate field names, the field name must be uniquely qualified.

Field names can be qualified by using two different conventions. The first convention joins the record or group name to the field name with a period:

`record-name.field-name`

or

`group.name.field-name`

The second convention joins the record or group name to the field name with the keyword OF. When the OF syntax is used, the field name or group name is written first followed by OF and the qualifier needed:

field-name OF record-name

or

field-name OF group-name

A field name requires as much qualifying as necessary to uniquely identify the field to Enform Plus. The necessary qualification might be as simple as combining the field name with the record name or group name. It might require combining the field name with both a group name and a record name or with two group names. Consider the record descriptions shown in [Figure 3-2](#).

Figure 3-2. Records With Duplicate Field Names

RECORD stock-items.	RECORD shelf-items.
FILE IS "stock" KEY-SEQUENCED.	FILE is "shelf" KEY-SEQUENCED.
02 depot-num PIC 99.	02 dept-num PIC 99.
02 cont-num PIC 99.	02 dept-name PIC X(10).
02 erasers.	02 cont-num PIC 99.
05 ink PIC 99.	02 pens.
05 gum PIC 99.	05 b-point PIC 99.
05 pink PIC 99.	05 felt-tip PIC 99.
02 ink-pens.	02 erasers.
05 felt-tip PIC 99.	05 ink PIC 99.
05 b-point PIC 99.	05 gum PIC 99.
05 fountain PIC 99.	05 gray PIC 99.

If both the stock-items and shelf-items record descriptions are open, ink must be qualified. To qualify ink within shelf-items, one of the following must be entered:

ink OF erasers OF shelf-items

shelf-items.erasers.ink

Primary Key References

The records in database files can be uniquely identified by the value of a primary key. For database files with key-sequenced file structure, the primary key is part of the record. For database files with relative, unstructured, or entry-sequenced file structure, the primary key is not part of the record. Primary keys can be referenced in two forms:

KEY OF *record-name*

or

record-name.KEY

The form *record-name*.KEY can appear for only one relative, entry-sequenced, or unstructured file per query.

The primary key for files with key-sequenced file structure is a field within the record. For key-sequenced files, referencing the primary key using the form *record-name*.KEY is the same as explicitly naming the field described as the primary key. The

advantage of the form *record-name*.KEY is that you do not need to know the name of the primary key field to reference it. A listing of the primary key values of the parts file can be obtained by the following:

```
OPEN parts;  
LIST parts.KEY;
```

The record with the primary key value of 1403 can be referenced by:

```
WHERE parts.KEY = 1403
```

For files with relative file structure, the primary key is a record number. The record number is the ordinal position of the record relative to the beginning of the physical file. The first record in the physical file has position zero. A listing of the primary keys of a relative file can be obtained by:

```
OPEN rell;  
LIST KEY OF rell;
```

The primary key of the fifth record in the file can be referenced by:

```
WHERE KEY OF rell = 4
```

Remember, the fifth record in a file with relative file structure has position four because the first record is in position zero.

For entry-sequenced and unstructured files, the primary key is a record address—the byte address of the record's block plus the relative record number within the block. A record address is always an even number. For more information about file structures, see the *Enscribe Programmer's Guide*.

A listing of the primary keys of an entry-sequenced file can be obtained by:

```
OPEN entryseq;  
LIST entryseq.KEY;
```

The third record in the file *entryseq* has a primary key whose byte offset is 16. The record can be referenced by:

```
WHERE entryseq.KEY = 16
```

For more information about file structures, see the *Enscribe Programmer's Guide*.

Subscripts

Subscripts, although they are not required, are usually used to reference elements in a user table or database table. (A database table is created when the dictionary description of a database field contains an OCCURS clause.) Subscripts are needed for references to user tables and database tables because all the elements in such tables have the same name. Subscripts can be used in references to database fields and user variables, although they are not necessary.

The syntax for including subscripts is:

```

{ field-name-ref1 } { "[ subscript ] " }
{ user-table-name } { "[ subscript-range ] " }

grp-name { "[ subscript ] " } .field-name-ref2
        { "[ subscript-range ] " } { "[ subscript-range ] " }
```

field-name-ref1

is the qualified name of a database field.

user-table-name

is the name of a user table defined by the DECLARE statement.

subscript

is an integer. The lowest valid value for *subscript* is 1. The highest valid value is the maximum number of elements defined for the user or database table. (See the following description for more information.)

subscript-range

has the form *subscripti* : *subscriptj*

subscripti

is the first element being referenced.

subscriptj

is the last element being referenced.

grp-name

is the name of a group described in the dictionary. A group is defined as a record element whose level number (02, 03, 04,...) is less than that of the next record element.

field-name-ref2

is the name of a subordinate field. A subordinate field is defined as a record element whose level number (05, 06, 07,...) is greater than that of *grp-name*.

When a subscript is included with a reference to a table name, a user variable name, or a field name, Enform Plus determines whether the subscript is a valid subscript value allowed for the table, variable, or field. A valid subscript value for a field or user variable is 1. Valid subscript values for a user table are defined in the DECLARE statement. For example, consider the following user table declaration:

```
DECLARE u-var[ 24 ];
```

The valid subscript values for `u-var` are 1 through 24. Valid subscript values for database tables are defined by the `OCCURS` clause in the dictionary description of the table. For example:

```
02 monthly-sales OCCURS 12 TIMES.
```

The valid subscript values for `monthly-sales` are 1 through 12.

Both user and database tables can be referenced without a subscript. Enform Plus uses a default a subscript of 1. For example:

<code>u-var</code>	Refers to the first element of the database table.
<code>monthly-sales</code>	Refers to the first element of the database table.

Including a subscript with a user or database table reference identifies the individual elements of the table. For example:

<code>u-var [2]</code>	Refers to the second element of the database table.
<code>monthly-sales [4]</code>	Refers to the fourth element of the database table.

A *subscript-range* can be included in user or database table references when the table is the target-item in a `LIST` statement. Subscript-range is illegal if a database table is modified by a `BY`, `BY DESC`, `ASC`, or `DESC` clause. (See [Section 5, Clauses](#) for information about these clauses.) Including *subscript-range* is the same as referencing the table elements individually. For example:

<code>u-var [3:8]</code>	Refers to the third through eighth elements of the database table.
<code>monthly-sales [1:4]</code>	Refers to the first through fourth elements of the database table.

Enform Plus does not allow arithmetic on subscript ranges. For example,

```
LIST (employee [1:5].salary * 1.10)
```

is invalid.

In the dictionary record description, a database table can be defined as a group element with subordinate database field entries. For example:

```
02 sales OCCURS 12 TIMES.
  10 month PIC X(3).
  10 top-dept PIC 9999.
```

Enform Plus allows you to refer to the subordinate database fields as follows:

<code>sales</code>	Refers to month and top-dept within the first element of sales.
<code>sales [2].top-dept</code>	Refers to top-dept within the second element of sales.
<code>sales [2:3].month</code>	Refers to month within the second through third elements of sales.

The subordinate elements of a database table can themselves contain a database table resulting in nested database tables. For example, consider the following record description:

```
02 tot-sales OCCURS 12 times.
10 month PIC X(3).
10 top-dept PIC 999.
10 wkly-sales PIC 99999 OCCURS 4 TIMES.
```

Enform Plus allows you to reference nested database tables. For example:

<code>tot-sales</code>	Refers to month, top-dept, and the first through last elements of wkly-sales within the first element of tot-sales.
<code>tot-sales[4].wkly-sales[2]</code>	Refers to the second element of wkly-sales within the fourth element of tot-sales.
<code>tot-sales[1:5].wkly-sales[3]</code>	Refers to the third element of wkly-sales within the first through fifth elements of tot-sales.
<code>tot-sales[2:3].wkly-sales[1:4]</code>	Refers to the first through fourth elements of wkly-sales within the second through third elements of tot-sales.

Aggregates

An aggregate is the result of a cumulative operation performed for each value that contributes to the aggregate. An aggregate yields a single value for the group of values over which it is processed. Aggregates can be specified only as a target-item or in a request-qualification. Aggregates specified as a target-item are called target aggregates. Aggregates specified in a request-qualification are called qualification aggregates.

Enform Plus provides two types of aggregates: predefined and user-defined.

The syntax used for an aggregate is as follows:

{	{	AVG	}		}
{	{	COUNT	}		}
{	{	MAX	}	({ <i>field-name</i> } [OVER ALL]	}
{	{	MIN	}	({ <i>expression</i> } [OVER <i>over-item</i>]	}
{	{	SUM	}		}
{	{	<i>user-aggregate</i>	}	[WHERE <i>logical-expression</i>]) ,	}
{	{	AVG	}		}
{	{	COUNT	}		}
{	{	MAX	}	([UNIQUE] <i>field-name</i> [OVER ALL]	}
{	{	MIN	}		}
{	{	SUM	}	[WHERE <i>logical-expression</i>]) ,	}
{	{	<i>user-aggregate</i>	}		}

AVG

is a predefined Enform Plus aggregate that computes an average value for a set of numbers or expressions.

COUNT

is a predefined Enform Plus aggregate that tallies the occurrences of a field defined as either numeric or alphanumeric.

MAX

is a predefined Enform Plus aggregate that finds the highest number in a set of numbers or expressions, or finds the alphanumeric string with the highest value based on the ASCII collating sequence.

MIN

is a predefined Enform Plus aggregate that either finds the lowest number in a set of numbers or expressions or finds the alphanumeric string with the lowest value based on the ASCII collating sequence.

SUM

is a predefined Enform Plus aggregate that totals a set of numbers or expressions.

user-aggregate

is the name of an aggregate defined by a DECLARE statement. (See page [4-9](#).)

field-name

is the name of a database field.

expression

is a arithmetic or IF/THEN/ELSE expression (explained on page [3-26](#).)

over-item

is a field used to sort and group the records over which the aggregate is processed. For a target aggregate, *over-item* must be a *by-item* (the name of a field modified by a BY or BY DESC clause). For a qualification aggregate, *over-item* can be any field name.

OVER ALL

defines the range of the aggregate operation as over all the values specified.

OVER

defines a range for the aggregate operations. The aggregate operation takes place only over the specified *over-item*. The operation yields one aggregate value for each unique value of *over-item*.

UNIQUE

excludes duplicate values from contributing to the collecting operation of the aggregate. UNIQUE adds considerable processing overhead and should not be specified unless you know unwanted duplicate values exist. UNIQUE is redundant with MAX or MIN. While the same answer is returned when UNIQUE is specified with these aggregates, processing time increases greatly.

Predefined Aggregates

The five predefined Enform Plus aggregates are: AVG, COUNT, MAX, MIN, and SUM.

AVG finds an average value for a set of numbers. For each record containing a field value to be averaged, the field value is added to the running total of the contributing field values. After all contributing field values are processed, the final total is divided by the number of field values that made up the total. The following example finds the average of the partcost field over the suppnnum field:

```
LIST BY suppnnum,
    AVG(partcost OVER suppnnum);
```

COUNT tallies the instances of an element. In the following example, the number of parts kept in stock are counted by counting the part numbers in parts:

```
OPEN parts;
LIST COUNT(partnum);
```

MIN determines the lowest number in a set of numbers or expressions. MAX determines the highest number in a set of numbers or expressions. The following example finds both the lowest and the highest price of a part:

```
OPEN fromsup;
LIST BY partnum,
    MIN(partcost OVER partnum),
    MAX(partcost OVER partnum);
```

SUM totals a set of numbers or expression values. In the following example, the sum of the sales made by each salesman is obtained:

```
OPEN order,odetail,parts;
LINK order to odetail VIA ordernum;
LINK odetail to parts VIA partnum;
LIST BY salesman,
    SUM (price * quantity) OVER salesman);
```

User Aggregates

When the predefined aggregates do not meet your needs, you can define your own aggregates with a DECLARE statement. A user-defined aggregate can be used anywhere a predefined aggregate can appear with two exceptions:

- A user aggregate cannot be referenced in the end expression of the declaration of another user aggregate.
- A user aggregate declared with an end expression cannot be used as a qualification aggregate over a *by-item*.

The syntax of a user-defined aggregate is shown with the [DECLARE Statement](#) on page 4-9. The syntax is also shown here to clarify this description.

```
user-aggregate-name ( formal-argument ) = ( step-expression
    [ , [ end-expression ] [ , initialize-constant ] ] )
```

user-aggregate-name

is a unique name you give your aggregate. The name must follow the naming rules described under [Rules for Naming User-Defined Elements](#) on page 3-6. This name is also used to obtain the current value of the aggregate in *step-expression* and *end-expression*.

formal-argument

is a unique name used to represent the actual field name or expression in the aggregate definition.

step-expression

is an arithmetic expression to be computed for each value contributing to the aggregate. Normally, *step-expression* includes a reference to the *user-aggregate-name* so that a value is accumulated over the contributing values.

end-expression

is an arithmetic expression to be computed after all qualifying values for the aggregate are processed. *end-expression* can contain a predefined aggregate name. If *end-expression* is omitted and *initialize-constant* is present, an extra comma must precede *initialize-constant*.

initialize-constant

is a numeric literal that is the starting value of the aggregate. Zero is the default. When *end-expression* is omitted and *initialize-constant* is present, an extra comma must precede *initialize-constant*.

Parentheses must appear exactly as shown in this syntax.

Consider the following user aggregate:

```
DECLARE grossavg (x) = (grossavg + 1.10 * x, grossavg/COUNT
    (x) );
```

The user-aggregate name is *grossavg*. The *formal-argument* is *x*. The *step-expression* is *(grossavg + .10 * x)*. The *end-expression* is *grossavg/COUNT (x)*.

This user-aggregate could be used to compare the new gross salaries to the old average salary:

```
LIST AVG (salary), grossavg (salary);
```

Target Aggregates

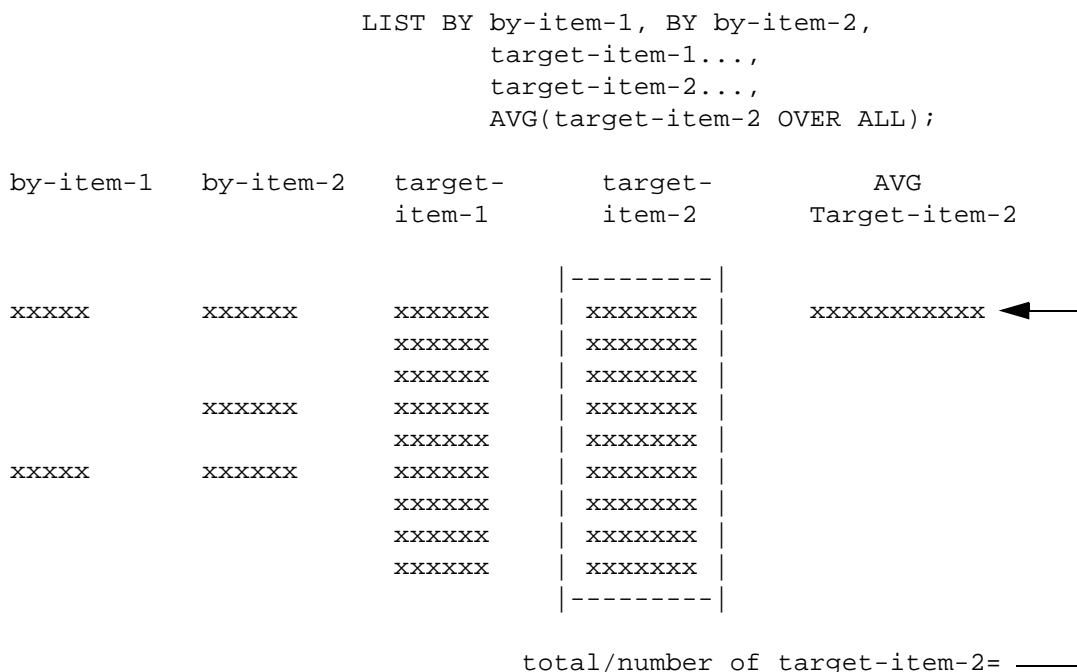
A target aggregate appears as part of the Enform Plus output. When you specify a target aggregate, follow these rules:

- The field names in the target aggregate syntax can come from different database records.
- An *over-item* must be a *by-item*.
- Target aggregates cannot be nested; that is, one aggregate cannot be used as the argument for another aggregate.
- The UNIQUE syntax cannot be used with an aggregate computed over a *by-item*.

Target Aggregate With OVER ALL Syntax

When you specify a target aggregate without specifying either OVER or OVER ALL, Enform Plus uses a default of OVER ALL. [Figure 3-3](#) shows both a query outline using the OVER ALL syntax and an output diagram.

Figure 3-3. Query Outline of Target-Aggregate With OVER ALL Syntax



Specifying the aggregate shown in [Figure 3-3](#) is the same as specifying:

```
AVG(target-item-2);
```

The query groups `target-item-1` and `target-item-2` within `by-item-2`. The aggregate `AVG` is used and the average value of `target-item-2` is found by totaling all of the values of `target-item-2` and dividing that total by the number of `target-item-2` values. Notice that `AVG target-item-2` prints as a separate column on the report with only one nonblank entry. This entry corresponds to the aggregate value of all records in the report.

When a target aggregate is specified in a `FIND` statement with `OVER ALL` either present or used by default, only the first target-record contains the aggregate value. This field is blank in all other target-records generated by the `FIND` statement.

Target Aggregate With OVER Syntax

If you specify a target aggregate in a `LIST` or `FIND` statement with the `OVER` syntax, a single aggregated value is returned for each grouped value. Consider the query outline and output diagram shown in [Figure 3-4](#).

Figure 3-4. Query Outline of Target-Aggregate With OVER Syntax

Query Outline:

```
LIST BY by-item-1,
      BY by-item-2,
      target-item-1
      AVG(target-item-1 OVER by-item-2);
```

Output Diagram:

by-item-1	by-item-2	target-item-1	AVG target-item-1
-----	-----	-----	-----
xxxxxxx	xxxxxxx	xxxxxxx	xxxxxxxxxxxxxxx
		xxxxxxx	→ (total/4)
		xxxxxxx	
		xxxxxxx	

	xxxxxxx	xxxxxxx	xxxxxxxxxxxxxxx
		xxxxxxx	→ (total/2)

xxxxxxx	xxxxxxx	xxxxxxx	xxxxxxxxxxxxxxx
		xxxxxxx	→ (total/3)
		xxxxxxx	

In this example, `AVG target-item-1` prints as a separate report column with a nonblank entry only on the first line of each new `by-item-2` group. This entry represents the average of `target-item-1` over the `by-item-2` group.

When a target aggregate is specified in a `FIND` statement with the `OVER` syntax, the aggregate value is present only in the first target-record for a `by-item`. This field is blank for all other target-records.

Qualification Aggregates

Qualification aggregates must be specified in a WHERE clause. When specifying a qualification aggregate, follow these rules:

- Qualification aggregates cannot be nested; that is, one aggregate cannot be used as the argument for another aggregate.
- A qualification aggregate can contain an embedded WHERE clause.
- The embedded WHERE clause can contain another qualification aggregate; however, that qualification aggregate must be computed OVER ALL.
- All field names used for each individual qualification aggregate expression (the field being aggregated, any field name appearing in an expression being aggregated, any field name appearing in the WHERE clause associated with an aggregate, and any *over-item*) must come from the same database record.
- The UNIQUE syntax cannot be used with an aggregate computed over an *over-item*.

Qualification Aggregate With OVER ALL Syntax

When you specify a qualification aggregate without specifying either OVER or OVER ALL, Enform Plus uses OVER ALL by default. Enform Plus computes the value for the qualification aggregate over all the records in the original database (not over the target-records). For example:

```
LIST odetail,  
    WHERE quantity > AVG (quantity );
```

restricts the records returned from *odetail* to those whose quantity field is greater than the average of all the values of the quantity field.

Qualification Aggregates With OVER Syntax

When a qualification aggregate is used with the OVER syntax, Enform Plus computes one value for each over-item group. When the WHERE clause is evaluated, Enform Plus uses the qualification aggregate value for the particular group to restrict the records selected. [Figure 3-5](#) on page 3-18 shows a query outline and the process that occurs when the OVER syntax is used.

Figure 3-5. Query Outline of Qualification Aggregate With OVER Syntax

```
OPEN employee;
LIST regnum,
    branchnum,
    WHERE salary GT AVG( salary OVER regnum );
```

The qualification aggregate value for each group is:

regnum	AVG salary
1	24666
2	28333
5	38000
99	39500

If employee has the following form when grouped by regnum and branchnum:

regnum	branchnum	salary
1	1	36000
		19000
		25000
		26000
		12000
2	2	30000
	1	37000
		25000
		23000
5	3	38000
99	1	39500

The report produced is:

Region	Branch
-----	-----
2	1
1	1
1	1
1	1
1	2

Qualification Aggregates and Target Records

A qualification aggregate functions as a full subquery. This means that Enform Plus computes the value for the aggregate over all the records in the database file, (not over the target records selected in the rest of the LIST or FIND statement) and uses the result as input to the target list.

For example, the statement:

```
LIST BY job,
    empname,
    AVG (salary)
WHERE salary < AVG (salary);
```


gives:

- All employees whose salaries are less than the average
- The average of all salaries that are less than the average

Qualification Aggregate With an Embedded WHERE Clause

If a qualification aggregate contains a WHERE clause that restricts the records for the aggregate calculation, Enform Plus processes the embedded WHERE clause before the aggregate. If any record for an over-item does not satisfy the restriction specified in the embedded WHERE clause, Enform Plus excludes that record from the aggregate calculation.

For example, consider [Figure 3-6](#), which shows an Enform Plus query containing a qualification aggregate with a WHERE clause. The query prints the part number and the amount in stock of all the parts where the price of the part is greater than the average price of all parts not in stock.

Figure 3-6. Qualification Aggregate With Embedded WHERE Clause

```
OPEN parts;
LIST partnum,
      inventory,
      WHERE price GT AVG (price WHERE inventory LT 0);
```

The report produced is:

Part Number	INVENTORY
-----	-----
212	7
244	3
1403	21
5502	6
5504	-1
5505	0
7102	20

Aggregates and Scale

All the aggregates except COUNT take their display format from the formats of the input values. (COUNT always returns a value with a scale of zero, or no digits to the right of the decimal point.) Consequently, if a field is defined as an integer and the aggregate yields a fractional value, the aggregate returns only the whole portion of the value. If the field is defined as a decimal, the value returned is accurate to the number of decimal places provided by the DDL data definition for the field.

If you want the aggregate to return a value with a scale other than that of the input field, assign the result to a user variable declared with the appropriate scale. For example:

```
DECLARE fixit INTERNAL F6.2;  
LIST fixit := AVG (price OVER partnum);
```

The average of price over part number will now be returned with six total digits, with two digits to the right of the decimal point.

Literals

Literals can be used in both a target-list and a request-qualification. Literals can also be used in many Enform Plus statements and clauses. Literals are used in titles, headings, special text printed within a report's body, and in expressions. The two types of literals are numeric and string.

Literals cannot be continued across lines. The maximum length of a literal is 127 characters.

Numeric Literals

Numeric literals are used in all arithmetic expressions. They can be used in logical expressions when the literal is compared to a database element described in the data dictionary as numeric. Numeric literals:

- Are not enclosed in quotation marks
- Are composed of the digits 0-9
- Can be preceded or followed by a plus or minus sign
- Must be enclosed in parentheses if they are specified outside of a logical expression or a TAB, SPACE, SKIP, or FORM clause

Numeric literals can stand alone as target-items in a LIST or FIND statement. In this case, they must be enclosed in parentheses.

The following are examples of numeric literals:

```
(104) (123.0444) (+267) (.006) (-15)
```

String Literals

String literals can be used in many Enform Plus statements and clauses. String literals can be used in logical expressions if the database element to which the string literal is compared is declared alphabetic or alphanumeric in the data dictionary. String literals:

- Can be composed of any character in the ASCII character set.
- Must be enclosed in quotation marks. If a quotation mark is part of a string literal, the quotation mark must be doubled.

The following are examples of string literals:

```
"This is a string literal"
```

```
"This string literal contains a " quotation mark"
```

```
"1234.99"
```

String literals can stand alone as target-items in a LIST statement. By using a string literal in this manner, you allow printing of one or more constant characters between two columns of data. For example:

```
LIST customer, "....", address;
```

produces the following report:

```

      CUSTOMER                                ADDRESS
-----
CENTRAL UNIVERSITY....UNIVERSITY WAY
BROWN MEDICAL CO   ....100 CALIFORNIA STREET
```

Arithmetic Expressions

Arithmetic expressions are some combination of numeric literals, field values, variables, or aggregates that are added, subtracted, multiplied, or divided to yield a single value. A JULIAN-DATE clause, TIMESTAMP-DATE clause, or a TIMESTAMP-TIME clause can also be used in an arithmetic expression. Arithmetic expressions must be enclosed in parentheses.

[Table 3-3](#) lists the arithmetic operators and their functions.

Table 3-3. Arithmetic Operators

Operator	Function
+	Addition
-	Subtraction
*	Multiplication
/	Division

Spaces are not required before any of the arithmetic operators with the exception of the subtraction sign (-). At least one space must precede a subtraction sign that follows a field or variable name.

Arithmetic expressions can be simple:

```
(price + 10.00)
```

or they can be complex:

```
((price + 10.00)* quantity)
```

Evaluation Order of Arithmetic Expressions

Arithmetic expressions are evaluated in the following order:

1. Nested parenthesized expressions are evaluated first, beginning with the innermost expression.
2. Within a nested parenthesized expression, multiplication and division operations are evaluated next.
3. Within a nested parenthesized expression, addition and subtraction operations are evaluated last.

Scale Factor of the Result

The scale of the result is determined by the number of digits after the decimal point. In an arithmetic expression, the result has the same number of digits after the decimal point as the field or variable in the expression with the greatest precision. This condition could result in loss of significant digits if too great a precision is used for the field or variable.

The resulting scale factor can be controlled by assigning the result of the arithmetic expression to a user variable. (User variables are explained on page [3-27](#).) The precision of the user variable can be specified by an `INTERNAL` clause within the `DECLARE` statement that defines the user variable. The maximum number of digits allowed is 18. All calculations with Enform Plus are performed with `QUAD` arithmetic.

Logical Expressions

Logical expressions evaluate to a truth value—either true or false based on a condition specified within the expression. Both the conditions that can be specified and the conditional operators are listed in [Table 3-4](#).

Table 3-4. Conditional Operators

Condition	Keyword	Abbreviation	Symbol
Equal	EQUAL IS	EQ	=
Not equal		NE	<>
Greater than	GREATER [THAN]	GT	>
Greater than or equal to		GE	>=
Less than	LESS [THAN]	LT	<
Less than or equal to		LE	<=

Enform Plus provides two other conditional operators: `BEGINS WITH` and `CONTAINS`. The three symbols '`]]]`' are synonymous with `BEGINS WITH` and the three symbols '`> > >`' are synonymous with `CONTAINS`.

The syntax of a logical expression is:

```
[NOT] condition [ { AND | OR } ] [NOT] condition ] ...
```

condition

has one of the following forms:

<i>field-name</i>	[not]	{ BEGINS WITH ']' CONTAINS '>' conditional operator EQUAL EQ IS [NOT] = NE <> }	<i>string-literal</i>
{ variable field-name expression }	[NOT]	conditional-operator	{ variable field-name expression }

conditional-operator

is one of the conditional operators shown in [Table 3-4](#) on page 3-22.

expression

is an IF/THEN/ELSE or arithmetic expression.

pattern-match

is a pattern of numbers or characters, enclosed within both quotation marks and brackets. The syntax for a pattern-match is:

{ " [" { n m,n - } }	<i>string-literal</i>	{ n m,n - } }
------------------------------------------------------	-----------------------	----------------------------------------

n

is an integer indicating that exactly *n* number of characters must precede or follow *string-literal* when it is found in a field value.

string-literal

is the pattern of characters or numbers to which the field is being compared. The *string-literal* must be enclosed in quotation marks.

m, n

is two integers, separated by comma, indicating that at least *m* characters but not more than *n* characters must precede or follow *string-literal* when it is found in a field value.

–

is a dash indicating any number of characters (0 through 255) can precede or follow *string-literal* when it is found as a field value. Specification of a dash indicates you do not care about the contents of this part of the field.

value-range

is a range of values with the form *value-1* THRU *value-2*

A logical expression can be simple or compound. A simple logical expression consists of one condition. A compound logical expression uses the Boolean operators AND, OR, and NOT to operate over two or more logical expressions.

By using the Boolean operators AND, OR, and NOT, you have the following effect on the evaluation of the logical expression:

- When you precede a condition with the Boolean operator NOT, the result of the expression is evaluated as true if the condition is evaluated as false.
- When you join two or more conditions with the Boolean operator AND, the result of the expression is evaluated as true only if all the conditions are evaluated as true.
- When you join two or more conditions with the Boolean operator OR, the result of the expression is evaluated as true if any of the conditions are evaluated as true.

Effect of Parentheses on Compound Logical Expressions

Compound logical expressions are evaluated in this sequence:

1. Conditions within parentheses are evaluated first.
2. Conditions preceded by the Boolean operator NOT are evaluated second.
3. Conditions joined with the Boolean operator AND are evaluated third.
4. Conditions joined with the Boolean operator OR are evaluated last.

BEGINS WITH and CONTAINS

BEGINS WITH and CONTAINS are special conditional operators that can be used to yield a true or false value if a field either begins with or contains a specific alphanumeric string.

The BEGINS WITH operator determines if a field starts with a specified alphanumeric string. For example, the following can be used to limit the data retrieved from the supplier field to only those records whose suppname field begins with COMPAQ:

```
LIST supplier WHERE suppname BEGINS WITH "COMPAQ",
```

In this example, the logical expression is evaluated as true only if the field begins with a value in which COMPAQ is in uppercase characters.

The **BEGINS WITH** operator can be used only with fields specified as alphanumeric in the data dictionary. A field is alphanumeric when its corresponding data description entry is specified as **PIC X**.

The **CONTAINS** operator determines if a field contains a specified alphanumeric string. For example, the following determines whether the **empname** field contains the value **GEORGE**:

```
empname CONTAINS "GEORGE"
```

Enform Plus does not support variable-length parameters. The following query, in which **xyz** has been defined as a five-character field, will not work if you pass **xyz** a three-character value:

```
PARAM xyz INTERNAL A5;
OPEN order;
LIST order WHERE KEY BEGINS WITH "xyz";
```

Enform Plus looks for a value of "xyz" plus two blanks, and the query fails.

Range of Values in Logical Expressions

A field can be compared to a range of values in a logical expression. In the range, **value-1** must be less than **value-2**. Values used in a range can be either numeric literals or string literals.

Specifying field **EQ value1 THRU value2** is equivalent to specifying:

```
field-name GE value-1 AND field-name LE value-2
```

A range can contain a numeric literal if the field being examined is defined as numeric in the data dictionary. The **inventory** field of the parts file is defined as **PIC 999**. The following logical expression is evaluated as true if the value of the **inventory** field falls within the range of 5 to 15, including the values 5 and 15:

```
inventory EQ 5 THRU 15,
```

A range expression can also use a string literal if the field being examined is defined as alphanumeric in the data dictionary. The **partname** field is defined as **PIC X(18)**. The following logical expression is evaluated as true if **partname** field contains a value that falls within the range of A to L:

```
partname EQ "A" THRU "LZZZZZZZZZZZZZZZZZZZZ"
```

Note that the second string literal is 18 characters long, the length of the **partname** field. Specifying the literal in this manner ensures that all of the part names beginning with L are included.

Pattern-Match in Logical Expressions

In a logical expression, a field described as alphanumeric in the data dictionary can be compared to a pattern-match. A pattern-match is actually a comparison template that the field value is compared to. In the following logical expression, the `partname` field is compared to a pattern-match:

```
partname = [ -"DISK" - "MB" - ],
OR partname = [ -"DISK" - "Mb" - ],
OR partname = [ -"Disk" - "MB" - ],
OR partname = [ -"Disk" - "Mb" - ],
```

In the following logical expression, two numbers precede the *string-literal*, indicating that at least one character but no more than two characters must precede *string-literal* in the field value:

```
partname EQ [1,2 "T" - ],
```

IF/THEN/ELSE Expressions

IF/THEN/ELSE expressions yield a value determined by the result of a logical expression. IF/THEN/ELSE expressions can be used wherever an arithmetic expression can be used. IF/THEN/ELSE expressions can be nested. The syntax of an IF/THEN/ELSE expression is as follows:

```
( IF logical-expression THEN value-1 ELSE value-2 )
```

logical-expression

is a logical expression that evaluates as true or false.

value-1 or *value-2*

is a field name, an arithmetic expression, or IF/THEN/ELSE expression, or one of the following value keywords: NULL, BLANK, BLANKS, ZERO, ZEROS.

If the logical expression is evaluated as true, *value-1* is used. If the logical expression is evaluated as false, *value-2* is used. *Value-1* and *value-2* must be the same data type, either both numeric or both alphanumeric.

The value keywords NULL, BLANK, and BLANKS print blanks on reports. The value keywords ZERO and ZEROS print zeros on reports.

Consider the following IF/THEN/ELSE expression:

```
IF partnum = 2001 THEN ZEROS ELSE partnum,
```

This expression specifies that if `partnum` is equal to 2001, then zeros are to be printed on the report. If `partnum` has any other value except 2001, the value of `partnum` is printed.

Parameters

You can use a parameter either for request qualification or as a target-item. You can pass a parameter to a stored compiled query file. Define parameters by issuing the `PARAM` statement, described on page [4-39](#).

Enform Plus handles parameters syntactically as if they were literals. Enform Plus handles any parameter declared with an alphanumeric internal format as a string literal; it handles all other parameters as numeric literals. You must enclose a parameter with parentheses wherever you would have to enclose a numeric literal with parentheses.

User Variables

A user variable can be used to store numeric or string literals, save a field value, or hold the result of a calculation for later printing.

Before the user variable is specified in a query, the `DECLARE` statement (see page [4-9](#) for the syntax of the `DECLARE` statement) must be entered. This statement defines the variable name and optionally defines the internal storage format (the default internal storage format is a 64-bit signed integer), a default display format, and a default heading. The name given to the user variable must conform to the naming conventions described under [Rules for Naming User-Defined Elements](#) on page 3-6.

The default value of a user-declared variable is zero. An initial value for the user variable can be defined with the `SET` statement.

User Variable as a Target-Item

When a user variable is specified as a target-item, Enform Plus uses the default value or the initial value, whichever is appropriate. When a user variable is a target-item in a `LIST` statement, assignment syntax can be used to specify a new value for the variable. The value of a user variable changes as target list elements are evaluated, so that at any time, the value of the user variable depends upon the value most recently assigned.

Assignment syntax is as follows:

<i>user-variable-name-1</i>	<code>:=</code>	<div> <div>{</div> <div><i>aggregate</i></div> <div>}</div> <div>{</div> <div><i>expression</i></div> <div>}</div> <div>{</div> <div><i>field-name</i></div> <div>}</div> <div>{</div> <div><i>literal</i></div> <div>}</div> <div>{</div> <div><i>user-table-element</i></div> <div>}</div> <div>{</div> <div><i>user-variable-name-2</i></div> <div>}</div> </div>
-----------------------------	-----------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

user-variable-name-1

is the name of the user variable being defined.

aggregate

is the value of a predefined aggregate or user aggregate.

expression

is an arithmetic expression or an IF/THEN/ELSE expression.

field-name

is the name of a database field.

literal

is a numeric literal or string literal that agrees in type with the user variable.

user-table-element

is the subscripted name of a user table element. You cannot assign a subscript range to a user variable.

user-variable-2

is the name of a previously defined user variable.

When assignment syntax is used, Enform Plus reassigns a value to the user variable for each target-record; therefore, the value of the user variable might be different for each target-record. For example:

```
LIST u-var, u-var := salesman;
```

Enform Plus uses the default or initial value for the first occurrence of *u-var* in every target-record. For the second occurrence of *u-var* Enform Plus uses the value of the *salesman* field. This value changes for every target-record.

A user variable can be assigned the value of an expression that contains the user variable. For example:

```
DECLARE u-var;
SET u-var TO 10;
OPEN parts;
LIST partname,
    u-var := (u-var + 10);
```

Enform Plus uses the initial or default value of the user variable to determine the value of the expression. In the example, the value of the expression is 20. Enform Plus then assigns this value to the user variable. Within the same LIST statement, assignment syntax can subsequently be used to assign the user variable to another expression containing the user variable. For example:

```
DECLARE u-var;
SET u-var TO 10;
OPEN parts;
LIST partname,
    u-var := (u-var +10),
    u-var := (u-var +20);
```

Enform Plus uses 20, the value assigned in the first assignment syntax (`u-var + 10`) for the value of `u-var` in the second expression. After determining the value of the expression $((u-var + 10) + 20)$, Enform Plus assigns the value of the expression, 40, to the user variable.

Enform Plus performs this process for every target-record. Enform Plus continues to reevaluate the value of a user variable until it encounters the end of the target-list.

User Variable in Request-Qualification

A user variable can be specified in a request-qualification. When a user variable is used for request qualification, Enform Plus always uses either the default or initial value, whichever is appropriate. Consider the following:

```
SET u-var to 10;
LIST ordernum,
    u-var := (ordernum + u-var)
WHERE u-var > 10;
```

The preceding query always returns zero target-records, even though in every target-record the value of `u-var` (`ordernum + u-var`) is greater than 10. No target-records are returned because Enform Plus uses the initial value of the user variable to evaluate the `WHERE` clause. Because `u-var` is set to 10, no target-records are selected for a `WHERE` clause with a logical expression `u-var > 10`.

User Tables

User tables are special kinds of user variables that can store more than one value. Currently a user table can have a maximum of 64 elements called occurrences. Enform Plus issues an error message if you attempt to define a table with more than 64 occurrences.

The individual elements of a user table can be referenced by using subscripts. See the description of [Subscripts](#) on page 3-8 for more information.

Like user variables, user tables can be initialized by the `SET` statement. The default value of all elements in a user table is zero. When a user table is specified as a target-item, Enform Plus determines its value in the same way it determines the value of a user variable. Assignment syntax can be used to assign values to single elements of a table.

Arithmetic Overflow Conditions

An arithmetic overflow condition can occur when Enform Plus computes a value for:

- An arithmetic expression
- A predefined aggregate (such as `SUM` or `AVG`) or a user-defined aggregate

An arithmetic overflow condition does not cause Enform Plus to issue either an error or a warning message. Instead, Enform Plus truncates the result of the computation. If an arithmetic overflow condition occurs when Enform Plus is computing an intermediate result, Enform Plus uses the truncated result to complete the computation.

4 Statements

This section describes the syntax of the Enform Plus statements. The statements are arranged in alphabetic order to provide ease of access.

The Enform Plus statements, with the exception of the LIST and FIND statements, have a session-wide affect unless canceled or overridden. The LIST and FIND statements effect only the queries of which they are a part.

The AT END, AT START, FOOTING, SUBFOOTING, SUBTITLE, and TITLE statements apply only to queries containing a LIST statement. These statements supply information to be printed in a report.

The LIST and FIND statements must be terminated with a semicolon. The other Enform Plus statements should be terminated with a semicolon. Enform Plus neither executes the statement nor reports any syntax errors until it encounters either a terminating semicolon or the keyword indicating the start of the next statement; therefore, if you enter a ?SHOW command after a statement without a terminating semicolon, the effect of the statement is not shown in the output produced by the ?SHOW command.

[Table 4-1](#) lists the Enform Plus statements and their functions.

Table 4-1. Summary of Statements (page 1 of 2)

Information Selection

Statement	Function
LIST	Specifies the information selected for a report and prints the report.
FIND	Specifies the information retrieved from the database and either writes the information to a physical file or transmits the information to a host-language program.

Query Environment

Statement	Function
CLOSE	Deletes a user variable, aggregate, or table, a parameter, or a record description from the internal table.
DECLARE	Defines a user variable, user aggregate, or user table.
DELINK	Clears a connecting relationship between record descriptions.
DICTIONARY	Identifies the subvolume containing a dictionary. It also clears the internal table and reclaims table space.
EXIT	Terminates the current Enform Plus session.
LINK	Specifies a connecting relationship between record descriptions.
OPEN	Accesses a record description.
PARAM	Names and defines a parameter that can receive a value from a TACL PARAM command.

Table 4-1. Summary of Statements (page 2 of 2)

Statement	Function
SET	Initializes a user variable, user table, or a parameter and resets option variables.
Report Information Formatting	
Statement	Function
AT END	Prints information at the end of all subsequent reports in the current session. See also the AT END PRINT Clause on page 5-26.
AT START	Prints information just before the first set of column headings for all subsequent reports in the current session. See also the AT START PRINT Clause on page 5-28.
FOOTING	Prints a footing at the bottom of each report page for all subsequent reports in the current session. See also the FOOTING Clause on page 5-36.
SUBFOOTING	Prints a subfooting at the bottom of each report page for all subsequent reports in the current session. See also the SUBFOOTING Clause on page 5-59.
SUBTITLE	Prints a subtitle at the top of each page immediately following the title for all subsequent reports in the current session. See also the SUBTITLE Clause on page 5-61.
TITLE	Prints a title at the top of each page for all subsequent reports in the current session. See also the TITLE Clause on page 5-69.

AT END Statement

The AT END statement allows you to specify information that is printed at the end of all subsequent reports in the current session unless canceled or reset by another AT END statement or overridden by an AT END clause. (See the [AT END PRINT Clause](#) on page 5-26.) The syntax of the AT END statement is:

```
AT END [ PRINT print-list [ CENTER ] ] [ ; ]
```

print-list

can contain any combination of literals, FORM, SKIP, SPACE, or TAB clauses. The *print-list* can also contain the following elements that can be modified by AS, AS DATE or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user tables, user variables, or parameter names.

The clauses that can be used in a *print-list* are described in [Section 5, Clauses](#). The other elements are described in [Section 3, Enform Plus Language Elements](#).

Specifying a Field Name in an AT END Statement

If you specify a field name within the *print-list* of an AT END statement, Enform Plus prints the same field value as in the last row of the report. A field name appearing within the *print-list* of an AT END statement need not be explicitly included within the following LIST statements. If the field name is not included, Enform Plus in effect adds the field name with a NOPRINT clause.

Spacing Considerations

By default the information you specify in the *print-list* of an AT END statement begins printing in the same column position as the leftmost column of the report. By using either the SPACE or TAB clause as the first element in the *print-list*, you can override the default. The SPACE or TAB clause can appear anywhere within the *print-list*. For example, the SPACE clause in the following AT END statement causes the two literals to be separated by 15 spaces:

```
AT END PRINT "Report" SPACE 15 "Total Sales";
```

```
Report                Total Sales
```

If you specify either a SKIP clause or the symbol / (slash) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the AT END *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause in the AT END statement causes Enform Plus to print two lines:

```
AT END PRINT "End of Report for" SKIP "Region " regnum;
```

```
End of Report for  
Region 1
```

By using a FORM clause within an AT END statement, you can cause Enform Plus to print the remainder of the AT END *print-list* on a new page and to increment the page number.

By using the CENTER clause following the *print-list* of an AT END statement, you can center the information on the page.

The CENTER, Option Variable, SKIP, SPACE, and TAB clauses are described in [Section 5, Clauses](#).

AT END Information for Current Report or All Reports

An AT END statement prints information at the end of all subsequent reports in the current session. The current AT END statement can be reset by specifying a new AT END statement with a different *print-list*. By using an AT END PRINT clause, you can temporarily override an AT END statement. An AT END PRINT clause prints information for the current report only.

Canceling Session-Wide AT END Information

You can cancel the AT END statement by:

- Using the AT END statement without the *print-list* parameter
- Using either the DICTIONARY statement or the ?DICTIONARY command (which removes the AT END statement information from the internal table)

AT START Statement

The AT START statement allows you to specify information that is printed just before the first set of column headings for all subsequent reports in the current session unless canceled or reset by another AT START statement or overridden by an AT START clause. (See the [AT START PRINT Clause](#) on page 5-28.) The syntax of the AT START statement is:

```
AT START [ PRINT print-list [ CENTER ] ] [ ; ]
```

print-list

can contain any combination of literals, FORM, SKIP, SPACE, or TAB clauses. The *print-list* can also contain the following elements that can be modified by AS, AS DATE, AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE clauses, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, or parameter names.

The clauses that can be used in a *print-list* are described in [Section 5, Clauses](#). The other elements are described in [Section 3, Enform Plus Language Elements](#).

If you specify both an AT START statement and a TITLE or SUBTITLE statement, the AT START information is printed after the title or subtitle. The AT START statement differs from the TITLE statement in that the AT START information is printed only on the first page of a report while the title or subtitle is printed on every page of a report.

Specifying a Field Name in an AT START Statement

If you specify a field name within the *print-list* of an AT START statement, Enform Plus prints the same field value as in the first row of the report. A field name appearing within the *print-list* of an AT START statement need not be explicitly included within the following LIST statements. If the field name is not included, Enform Plus effectively adds it with a NOPRINT clause.

Spacing Considerations

By default, the AT START information begins printing in the same column position as the leftmost report column. By using SPACE or TAB clauses as the first element of the *print-list*, you can override this default. SPACE or TAB clauses can also appear anywhere within the *print-list*. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
AT START PRINT "Report" SPACE 15 "Total Sales";
```

```
Report                Total Sales
```

If you specify either a SKIP clause or the symbol / (slash) within a *print-list*, the printer advances one or more lines before printing the rest of the AT START *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the

option variable @VSPACE. In the following example, the SKIP clause of the AT START statement causes two lines to be printed:

```
AT START PRINT "Report For" SKIP "Region " regnum;
```

```
Report For  
Region 2
```

By using the FORM clause within an AT START statement, you can cause the remainder of the AT START *print-list* to be printed on a new page and increments the page number.

By using the CENTER clause following the *print-list* of an AT START statement, you can center the information on the page.

The CENTER, Option Variable, SKIP, SPACE, and TAB clauses are described in [Section 5, Clauses](#).

AT START Information for Current Report or All Reports

An AT START statement prints information just before the first set of column headings for all subsequent reports in the current session. The current AT START statement can be reset by specifying a new AT START statement with a different *print-list*. By using the AT START PRINT clause within a LIST statement, you can temporarily override the AT START statement for the report generated by the LIST statement.

Canceling Session-Wide AT START Information

You can cancel the AT START statement by:

- Using the AT START statement without the *print-list* parameter
- Using either the DICTIONARY statement or the ?DICTIONARY command (which clears the AT START information from the internal table)

CLOSE Statement

The CLOSE statement allows you to delete a user variable, user aggregate, user table, a parameter, or a record description from the internal table. The syntax of the CLOSE statement is:

CLOSE	$\left\{ \begin{array}{l} \text{record-name} \\ \text{user-variable-name} \\ \text{user-aggregate-name} \\ \text{user-table-name} \\ \text{param-name} \end{array} \right\}, \dots [;]$
-------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

record-name

is the name of a dictionary record description previously accessed by an OPEN statement.

user-variable-name

is the name of a user variable previously defined by a DECLARE statement.

user-aggregate-name

is the name of a user aggregate previously defined by a DECLARE statement.

user-table-name

is the name of a user table previously defined by a DECLARE statement.

param-name

name of a parameter previously defined by a PARAM statement.

The Effect of a CLOSE Statement on the Internal Table

As a session progresses, Enform Plus maintains an internal table for opened record descriptions, links of record descriptions, and definitions of user variables, user aggregates, user tables, and parameters. This internal table grows with each new OPEN, LINK, DECLARE, or PARAM statement entered.

Enform Plus has no way of knowing when a table entry is no longer required for a subsequent query within the current session. For this reason, Enform Plus allows you to clear unwanted table entries from the internal table with a CLOSE statement. The CLOSE statement does not reclaim space from the internal table, but it does eliminate the effect that unwanted entries might have on subsequent queries. Furthermore, regular use of CLOSE statements reduces the need to qualify fields that are present in more than one record description.

Closing a dictionary record description also clears all links for that particular record description.

An alternative to the CLOSE statement is the DICTONARY statement or ?DICTONARY command. Both completely clear the Enform Plus internal table and reclaim table space.

DECLARE Statement

The DECLARE statement allows you to define a user variable, user aggregate, or user table. The syntax of the DECLARE statement is:

```

DECLARE { { user-variable-name } }
        { { user-table-name "[" max-subscript "]" } }
        { { user-aggregate-name ( formal-argument ) } }
          { { = ( step-expression [ , [ end-expression ] } }
            { { [ , initialize-constant ] ] ) } }
          { { [ INTERNAL internal-format ] } }
          { { [ AS display-format ] } } , ... [ ; ]
          { { [ HEADING heading-string ] } }

```

user-aggregate-name, *user-variable-name*, or *user-table-name*

is the name of the declared element. Names of user-defined elements should conform to the rules described under [Rules for Naming User-Defined Elements](#) on page 3-6.

`"[" max-subscript "]"`

is the number of occurrences for the user table; the maximum number allowed is 64. The *max-subscript* must be enclosed within brackets []. See [Subscripts](#) on page 3-8 for a description of subscripts.

formal-argument

is the name used to represent the actual argument of the user aggregate.

step-expression

is the operation to be performed for each record contributing to the user aggregate.

end-expression

is the operation to be performed after all qualifying records for the user aggregate are processed by the *step-expression*.

initialize-constant

is the numeric literal that will be the starting value for the user aggregate.

internal-format

is the internal format for storing the user variable, aggregate or table.

display-format

is the default display format for printing the declared item.

heading-string

is a string literal that is the default heading for the declared element. Remember string literals must be enclosed in quotation marks (“ ”).

Specify the DECLARE statement before you reference a user variable, user aggregate, or user table. Enform Plus stores information about each user-defined element in the internal table. This information remains in the internal table until you issue a subsequent CLOSE statement for the user-defined element, issue a DICTIONARY statement or a ?DICTIONARY command, or end the Enform Plus session.

Declaring a User Aggregate

User aggregates are processed just like the predefined aggregates. The *step-expression* of a user aggregate can contain:

- Arithmetic expressions
- IF/THEN/ELSE expressions

The optional *end-expression* is the final operation to be performed after all of the qualifying records are processed by the *step-expression*. The *end-expression* can contain any of the following:

- Predefined aggregates
- Arithmetic expressions
- IF/THEN/ELSE expressions

By default, the *starting-value* for a user aggregate is zero unless the user aggregate is defined with alphanumeric internal format. In this case, the default value is blanks. An initial value is supplied when you specify *initialize-constant*. If you omit end-expression but specify initialize-constant, precede *initialize-constant* with two commas.

The following example uses the DECLARE statement to define a user aggregate:

```
DECLARE project (x) = (IF x < 2000 THEN project + x + .05 * x
    ELSE project + x + .04 * x);
```

More information about user aggregates can be found under [Aggregates](#) on page 3-11.

Declaring a User Variable or User Table

A user variable or table declaration remains in effect until the end of the current Enform Plus session unless you override the user variable or table by declaring a new variable or table with the same name.

By default, both user variables and elements in user tables are stored as 64-bit signed integers. To change this default, specify the optional INTERNAL clause described under [INTERNAL Clause](#) on page 5-42. In the following example, the user variable

no^orders is defined. The INTERNAL clause indicates that no^orders is to be stored as alphanumeric with a length of 9 bytes:

```
DECLARE no^orders INTERNAL A9;
```

The default display format for either a user variable or an element in a user table is a fourteen character integer. To change this default, specify the optional AS clause. The display format specified in the AS clause formats the user variable or table element unless you provide an explicit AS clause in the LIST statement. The AS clause is described under [AS Clause](#) on page 5-7.

Specify the HEADING clause to provide a default heading for either a user variable or a user table. Enform Plus uses the default heading for the user variable or table whenever an explicit HEADING clause is not specified in the LIST statement. The HEADING clause is described under [HEADING Clause](#) on page 5-39. In the following example, the default display heading “Quarterly Totals” is supplied for the table qtr^totals:

```
DECLARE qtr^totals [4] HEADING "Quarterly Totals";
```

Initialize both user variables and user tables by using the SET statement described on page [4-41](#).

[Section 3, Enform Plus Language Elements](#) provides more information about user variables and user tables.

DELINK Statement

The DELINK statement allows you to clear a connecting relationship between dictionary record descriptions. The syntax of the DELINK statement is:

DELINK	{	<i>record-name1</i> [TO [OPTIONAL]]	}	
	{	<i>record-name2</i> VIA <i>field-name</i>	}	
	{	<i>qualified-field-name1</i> [TO [OPTIONAL]]	}	, ... [;]
	{	<i>qualified-field-name2</i>	}	

record-name1 and *record-name2*

are the names of dictionary record descriptions.

field-name

is the name of a field common to both of the record descriptions.

qualified-field-name1 and *qualified-field-name2*

are the names of fields uniquely identified as components of *record-name1* and *record-name2*, respectively.

Both forms of the DELINK statement are equivalent. The field names must be specified in the DELINK statement in the same order as the corresponding LINK statement. For example, if the LINK statement is:

```
LINK parts TO odetail VIA partnum;
```

the corresponding DELINK statement can be:

```
DELINK parts.partnum TO odetail.partnum;
```

Enform Plus stores all links of record descriptions in the current Enform Plus internal table. All links apply to all subsequent LIST or FIND statements of the current Enform Plus session until you issue a DELINK, CLOSE, DICTIONARY statement, or ?DICTIONARY command. Because unnecessary links can produce undesirable results, delete relationships that do not apply to the current query from the internal table. Use a DELINK statement to clear a linking relationship between two record descriptions without affecting other links.

If you want to clear all links, use a CLOSE statement, a DICTIONARY statement, or a ?DICTIONARY command. A CLOSE statement for a record description deletes all links referencing that record description from the internal table. A DICTIONARY statement or ?DICTIONARY command clears the entire internal table.

DICTIONARY Statement

The `DICTIONARY` statement allows you to identify the subvolume containing your dictionary. It also allows you to clear the internal table. The `DICTIONARY` statement has the same effect as the `?DICTIONARY` command. The syntax of the `DICTIONARY` statement is:

```
DICTIONARY [ dict-subvol-name ]
```

dict-subvol-name

is the name of the subvolume where your dictionary files reside. See the *Guardian Programmer's Guide* for information on specifying Guardian file names.

The dictionary identified in a `DICTIONARY` statement must be created by the Data Definition Language compiler. Enform Plus issues an error message if you attempt to use a dictionary compiled with a version of DDL that is not current. You must then recompile the dictionary with a current version. See the *Data Definition Language (DDL) Reference Manual* for more information about creating and compiling a dictionary.

Identifying the Location of the Dictionary

You can identify the location of the dictionary by:

- Specifying the volume and subvolume where the dictionary resides as a part of the `TACL ENFORM` run command. If you do not specify a volume and subvolume, Enform Plus uses your default volume and subvolume as the dictionary location.
- Specifying either the `DICTIONARY` statement or `?DICTIONARY` command to identify where the dictionary resides. Use of either the `DICTIONARY` statement or the `?DICTIONARY` command overrides the dictionary identified at the time of the `TACL ENFORM` run command. When a new dictionary is specified, the internal table associated with the old dictionary is cleared.

In the following example, the `DICTIONARY` statement is used to identify the volume that the dictionary resides on as `$data` and the subvolume as `database`:

```
DICTIONARY $data.database;
```

Clearing the Internal Table

Entering the `DICTIONARY` statement without a volume and subvolume name is a simple means of clearing the entire internal table and reclaiming table space without changing the dictionary. To clear only certain elements of the internal table, see the [CLOSE Statement](#) on page 4-7 and the [DELINK Statement](#) on page 4-12. The elements cleared by the `DICTIONARY` statement are:

- All dictionary record descriptions from previous `OPEN` statements
- All previous links
- All user variable, user aggregate, and user table definitions
- All parameter definitions
- All information from `AT END`, `AT START`, `FOOTING`, `SUBFOOTING`, `SUBTITLE`, and `TITLE` statements

The `DICTIONARY` statement does not change the value set for any option variable (such as `@HEADING` or `@STATS`).

EXIT Statement

The `EXIT` statement terminates the current Enform Plus session. The syntax of the `EXIT` statement is:

<code>EXIT [;]</code>

The `EXIT` statement returns control to the invoking process, usually TACL. The `EXIT` statement is the same as the `?EXIT` command. Pressing the Ctrl and Y terminal keys simultaneously is an alternative way to exit Enform Plus.

FIND Statement

The FIND statement allows you to specify the input fields and records that contribute to the *target-record* and either write output records to a physical file or transmit output records to a host-language program. The FIND statement must end with a semicolon. The syntax of the FIND statement is:

```
FIND [ UNIQUE ] output-record-name

    {
      {
        ( { [ output-field-name := ] { target-item } } , ... )
        {
          { BY by-item } }
          { BY DESC by-item } }
          { ASCD target-item } }
          { DESC target-item } }
        }
      }
    [ WHERE logical-expression ] ;
```

UNIQUE

is an option that prevents duplicate output records. UNIQUE adds processing overhead and should be avoided unless you know unwanted duplicate records exist.

output-record-name

is the name of the dictionary record description of the output record.

output-field-name

is the name of a field in the dictionary record description of the output record. Enform Plus allows you to qualify *output-field-name*. If you do not qualify the name, Enform Plus qualifies *output-field-name* with *output-record-name*. In either case, *output-field-name* must be sufficiently qualified to avoid ambiguity between it and any other name specified in the query.

by-item

is an input field name. An input field name must be sufficiently qualified to avoid ambiguity between it and any other name specified in the query. You cannot specify more than 63 *by-items* for your query.

target-item

is an input field. Valid values for an input field are: a field name, a string literal enclosed in parentheses, a predefined aggregate, a user aggregate, an arithmetic expression, an IF/THEN/ELSE expression, a user table name, a user variable, or a system variable. An input field name must be sufficiently qualified to avoid ambiguity between it and any other name specified in the query.

logical-expression

is an expression returning a true or false value.

Output Record Dictionary Description

Before executing the `FIND` statement, you must include a description of the output record in the dictionary. When the `FIND` statement is executed, Enform Plus either writes the output records to the physical file specified in the dictionary or transmits the output records to a host-language program. In either case, the Data Definition Language (DDL) record description must describe the file type as unstructured.

The following example shows the DDL source code used to produce a dictionary record description for the output record `findfil`:

```
RECORD findfil.
02 custname          PIC X(18).
02 custnum           PIC 9(4).
02 partcost          PIC 9(6)V99.
end
```

Enform Plus allows you to specify database tables (see the description of subscripts under [Subscripts](#) on page 3-8 for information about database tables) in the record description of the output record. The following record description contains a database table named `child`:

```
RECORD newemp.
02 name              PIC X(18).
02 child              PIC X(18)    OCCURS 4 times.
end
```

You can specify that output records are to be written to a particular physical file either by including the `DDL FILE IS` clause in the dictionary record description or by specifying the `?ASSIGN` command. You can also use the `?ASSIGN` command to cause Enform Plus to write the records to a file other than the one described in the dictionary. The physical file generated by a `FIND` statement is always an unstructured file.

In the record description, an output field can differ in data format (picture size, scale, BINARY vs. ASCII numeric string, and so on) from the description of the input field as long as the output field has the same data type (numeric or alphanumeric) as the input field. Enform Plus performs the data format conversion automatically, including: truncation or padding with blanks for alphanumeric input and output fields of different lengths, binary to ASCII string conversion (or ASCII to binary), and scaling conversion for numeric input and output fields.

Group Definition and Sorting

The `BY` and `BY DESC` clauses group and sort records. The appearance of a `BY` or `BY DESC` clause in a `FIND` statement causes every occurrence of a grouped *by-item* value to be written to the output record (unlike the `LIST` statement where only the first occurrence of a grouped value is written to the output record). The `BY` and `BY DESC` clauses are described under [BY and BY DESC Clauses](#) on page 5-32.

The `ASCD` and `DESC` clauses sort records in ascending or descending order. They do not identify a group. The `ASCD` and `DESC` clauses are also described under [ASCD and DESC Clauses](#) on page 5-6.

When a `FIND` statement contains more than one `BY`, `BY DESC`, `ASCD`, or `DESC` clause, Enform Plus determines a major to minor sort precedence by the order in which the `BY`, `BY DESC`, `ASCD`, or `DESC` clauses appear in the `FIND` statement. The first clause has highest priority and is sorted first, the next one second priority, down to the last clause.

Output Fields

When the `FIND` statement is executed, Enform Plus takes the values from the input elements and stores them in the output field. Values are stored in the output field by name (when the output field has the same name as an input field) or by assignment syntax (`:=`). Enform Plus allows you to qualify *output-field-name*.

Enform Plus allows you to use subscripted output field names in a `FIND` statement when the record description of the output record contains a database table; however, a subscript range is not valid anywhere in a `FIND` statement. (See the description of subscripts under [Subscripts](#) on page 3-8 for the syntax used to include subscripts.) For example, the following `FIND` statement is valid:

```
FIND newemp (name      := emp.name,
              child [ 1 ] := emp.child1,
              child [ 2 ] := emp.child2,
              child [ 3 ] := " ",
              child [ 4 ] := " "           );
```

The following rules apply to *output-field-name*:

- The *output-field-name* can be qualified or subscripted.
- The *output-field-name* cannot contain a subscript range.
- If you omit *output-field-name*, Enform Plus operates as if it is the same as the input field name (including any subscripts or qualifications that apply to the input field name). Consider, for example, the following record descriptions and `FIND` statements:

RECORD OLD.	RECORD new.
FILE IS old KEY-SEQUENCED.	FILE IS new UNSTRUCTURED.
02 A PIC XX.	02 A PIC XX.
02 B PIC 99.	02 B PIC 99.
02 C OCCURS 2 times	02 D1 PIC AA.
04 DD PIC AA.	02 D2 PIC AA.
04 EE PIC XX.	02 E1 PIC XX.
KEY 0 IS A.	02 E2 PIC XX.
END	END


```
OPEN old, new;
FIND new (A, B, C);
```

The preceding `FIND` statement omits the output field names. Therefore, Enform Plus treats the names of the output fields as if they were `new.A`, `new.B`, and `new.C`. As the new record description does not contain a field named `C`, the query fails and Enform Plus issues an error message.

- The *output-field-name* can be a group name; however, Enform Plus might not store the fields within the group in the manner you expect. To understand the way that Enform Plus stores group elements, you must first understand what a group is. A group is defined in DDL as any field whose level number (03, 04, and so on) is less than that of the next field in the record. Consider, for example, the following DDL record description:

```
RECORD findfl.
FILE IS $mkt.sample.findfl key-sequenced.
02 account-num.
    05 type          PIC 9(4).
    05 num           PIC BINARY 16.
02 custnum          PIC 9(4).
```

Within this record, *account-num* is a group. The data type of a group is always alphanumeric. When a group name is specified as *output-field-name* within a `FIND` statement, Enform Plus stores each element within the group as alphanumeric data. If one of the fields within the group is defined as binary, using a group name as *output-field-name* results in an output record that might contain undesirable data. (The undesirable results occur because the binary representation of some numbers does not correspond to the ASCII representation of these same numbers.)

Before you execute the `FIND` statement, the dictionary must include a record description that defines the structure of the input record. This record description also defines the file type (key-sequenced, relative, entry-sequenced, or unstructured) of any data file that can be associated with the record description. The actual file type of the data file must be the same as the file type specified in the record description; otherwise, your query might return incorrect results.

Input Elements

The output fields receive values from any combination of the following input elements:

- The value of a field from the input record. The assignment to an *output-field-name* is optional. When you do not specify the *output-field-name*, Enform Plus operates as if it is the same as the field name from the input record. When the *output-field-name* is omitted, the input field name must be fully qualified.

```
FIND ...
( parts.partnum,
  ... ) ;
```

- The value of a numeric literal. Remember to enclose numeric literals in parentheses.

```
FIND ...
( region := (5),
  ... );
```

- The value of a string literal. A string literal must be enclosed in parentheses.

```
FIND ...
  ( jobtitle := ("manager"),
  ... ) ;
```

- The value of an arithmetic expression. Enclose an arithmetic expression within parentheses. Use any combination of numeric fields, numeric literals, numeric user variables, predefined aggregates, or user aggregates for the arithmetic expression.

```
FIND ...
  ( sales := (price * quantity),
  ... ) ;
```

- The value of an IF/THEN/ELSE expression. Enclose the IF/THEN/ELSE expression within parentheses.

```
FIND ...
  (stock := (IF inventory GT 100 THEN inventory
             ELSE ZERO),
  ...);
```

- The value of a user variable. Define the user variable by a DECLARE statement before referencing it in a FIND statement. The SET statement should be used to initialize the user variable. If the user variable is not initialized, the value is zero.

```
DECLARE region;
SET region TO 5;
FIND ...
  ( reg := region,
  ... ) ;
```

- The value of either a user aggregate or a predefined aggregate.

```
FIND ...
  ( BY employee.job,
    employee.salary,
    rate := AVG (salary OVER job),
    ... ) ;
```

- A group name. Enform Plus allows you to specify a group name as an input element. Avoid this specification, however, unless the data type of the fields receiving the data is exactly the same as the data type of the input group.

Although Enform Plus allows you to specify an output field name as an input element, this specification is not recommended and might lead to unexpected results.

The following rules apply to input elements:

- An input element can be qualified or subscripted.
- An input element cannot contain a subscript range.

- An input element must be sufficiently qualified to avoid any ambiguity between it and any other element specified in the query.

Request-Qualification

Use the WHERE clause to limit the records written to the physical output file or transmitted to the host-language program. See the [WHERE Clause](#) on page 5-72.

Summary Records

A FIND statement can be specified that either creates a new file containing only summary records or transmits only summary records to a host-language program. Such summary records contain only the first *target-record* from each group (created by a BY or BY DESC clause) down to some level. Summary records can be generated only by a query that contains an aggregate.

The two methods of obtaining summary records are:

- Explicitly request summary records by setting the @SUMMARY-ONLY option variable to ON before issuing the only statement.
- Implicitly request summary records by specifying only *by-items* and aggregates over *by-items* in the query.

When you explicitly request summary records you get *target-records* summarized down to the lowest level where an aggregate is calculated over that level. For example:

```
SET @SUMMARY-ONLY TO ON;
FIND findfil
( BY employee.dept,
  BY employee.job,
  BY employee.empname,
  employee.salary,
  COUNT(employee.empname OVER employee.job);
```

returns one *target-record* for each job in each department. Only the first employee name (empname) for each job is returned.

When you implicitly request summary records, you get *target-records* summarized down to the lowest level where an aggregate is computed over that level. (A query requesting only *by-items* and aggregates over ALL is not an implicit request for summary records). For example:

```
SET @SUMMARY-ONLY TO OFF;
FIND findfil
(BY employee.dept,
 BY employee.job,
 BY employee.empname,
 numemp:= COUNT (employee.empname OVER employee.job));
```

returns one *target-record* for each job in each department. Only the first employee name (empname) for each job is returned.

If you want summary records that consist of only *by-items* and aggregates to include those where the last value of a *by-item* has not changed but a subordinate *by-item* value has changed, then you must include a *target-item* in the *target-list* that has not appeared as a *by-item* or aggregate. Of course, you must also set @SUMMARY-ONLY to OFF. For example:

```
SET @SUMMARY-ONLY TO OFF;
FIND findfil2
    (BY employee.dept,
     BY employee.job,
     BY employee.empname,
     employee.salary,
     numemp:= COUNT (empname OVER job));
```

Statements and Clauses That Do Not Apply to the FIND Statement

Enform Plus queries with a FIND statement produce records only. There are no report features such as headings, titles, summary information, and special formatting. The following statements and clauses apply only to queries using the LIST statement and cannot be used with the FIND statement.

- AFTER CHANGE clause
- AS clause
- AS DATE clause
- AS TIME clause
- AT END statement and AT END PRINT clause
- AT START statement and AT START PRINT clause
- BEFORE CHANGE clause
- CENTER clause
- CUM clause
- FOOTING statement and clause
- FORM clause
- HEADING clause
- NOHEAD clause
- NOPRINT clause
- PCT clause
- SKIP clause
- SPACE clause
- SUBFOOTING statement and clause
- SUBTITLE statement and clause
- SUBTOTAL clause
- System Variable clauses
- TAB clause
- TITLE statement and clause
- TOTAL clause

FOOTING Statement

The FOOTING statement allows you to specify a footing to be printed at the bottom of each report page for all reports in the current session unless overridden or reset by another FOOTING statement or temporarily overridden by a FOOTING clause. (See the [FOOTING Clause](#) on page 5-36.) The syntax of the FOOTING statement is:

```
FOOTING [ print-list [ CENTER ] ] [ ; ]
```

print-list

can be any combination of literals, FORM, SKIP, SPACE or TAB clauses. The *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, or parameter names.

The clauses that can be used in a *print-list* are described in [Section 5, Clauses](#). The other elements are described in [Section 3, Enform Plus Language Elements](#).

Specifying A Field Name Within a FOOTING Statement

If you specify a field name within the *print-list* of a FOOTING statement, Enform Plus prints the same field value as in the last row of data on the current page. A field name appearing within the FOOTING statement need not be explicitly included within the following LIST statements. If the field name is not included, Enform Plus effectively adds it to the LIST statement with a NOPRINT clause.

Spacing Considerations

By default, the footing begins printing in the same column position as the leftmost report column. Using SPACE or TAB clauses as the first element of the *print-list* overrides this default. SPACE or TAB clauses can also appear anywhere within the *print-list*. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
FOOTING "Inventory" SPACE 15 "Parts in Stock";
```

The following footing appears at the bottom of the next report that is generated without a LIST statement FOOTING clause.

```
Inventory           Parts in Stock
```

If you specify either a SKIP clause or the symbol slash (/) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the FOOTING *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause in the FOOTING statement causes two lines to be printed:

```
FOOTING "Report 2-A" SKIP "Total Sales";
```

The following footing appears at the bottom of the next report that is generated without a LIST statement FOOTING clause:

```
Report 2-A  
Total Sales
```

By using a FORM clause within a FOOTING statement, you can force a new page. Enform Plus continues with the remainder of the FOOTING *print-list*. The page number remains the same. A single logical page can span multiple physical pages such that a TITLE can appear on one page, the data on the next, and a FOOTING on the next. The same page number applies to all physical pages in a logical page.

By using the CENTER clause within the FOOTING statement, you can center the footing on the page.

The CENTER, Option Variable, SKIP, SPACE, and TAB clauses are described in [Section 5, Clauses](#).

Footings for Current Report or All Reports

A FOOTING statement prints a footing at the bottom of each page for all subsequent reports in the current session. Specifying a new FOOTING statement with a different *print-list* resets the current FOOTING statement. Using the FOOTING clause within a LIST statement temporarily overrides the FOOTING statement. A FOOTING clause within a LIST statement prints a footing for the current report only.

Canceling Session-Wide Footing

You can cancel the FOOTING statement by:

- Using a FOOTING statement without the *print-list* parameter.
- Using either a DICTIONARY statement or a ?DICTIONARY command (which clears the FOOTING information from the internal table).

LINK Statement

The LINK statement allows you to specify a connecting relationship between dictionary record descriptions. The syntax of the LINK statement is:

	{	<i>record-name1</i>	[TO]	[OPTIONAL]	<i>record-name2</i>	}	
		VIA	<i>field-name</i>			}	
LINK	{					}	, ... [;]
	{	<i>qualified-field-name1</i>	[TO]	[OPTIONAL]		}	
		<i>qualified-field-name2</i>				}	

record-name1 and *record-name2*

are the names of dictionary record descriptions containing a common field.

field-name

is the name of a field common to both of the record descriptions.

qualified-field-name1 and *qualified-field-name2*

are the names of fields uniquely identified as components of the record descriptions being linked.

OPTIONAL

indicates that Enform Plus is to build logical records for the link by including all of the records from the data file whose record description is specified on the left side of the LINK OPTIONAL statement. Enform Plus includes each record from this data file even when no matching record is found in the data file whose record description is specified on the right side of the LINK OPTIONAL statement.

Both the LINK statement and the LINK OPTIONAL statement establish a connecting relationship, called a link. (You can also establish a link with a WHERE clause.) A link allows you to view your data in a new manner. When you specify either statement, Enform Plus establishes a new logical record description that consists of fields from both linked record descriptions. Enform Plus then builds a set of logical records for the new logical record description.

Enform Plus builds the set of logical records by selecting records from each of the data files associated with the linked record descriptions. When you use either statement, you must identify a field (called a linking field) from each record description. Enform Plus uses the data value of these linking fields to determine which records from each data file become part of the set of logical records. Enform Plus then uses the set of logical records to build the target records from which your Enform Plus output is produced. If you include a nonlinking WHERE clause (see the [WHERE Clause](#) on page 5-72 for a definition of nonlinking WHERE clauses) in a query, such as:

```
WHERE empnum GT 100
```

the target records include information only from those logical records that satisfy the restriction specified in the WHERE clause.

Duration of Link Established by LINK or LINK OPTIONAL Statement

When you use either the LINK statement or the LINK OPTIONAL statement, you establish a link that exists for the duration of the Enform Plus session. You can clear such a link by entering one of the following:

- A CLOSE statement for one of the record descriptions
- A DELINK statement for the link
- A DICTIONARY statement or a ?DICTIONARY command that clears the entire internal table; when you clear the internal table, you also clear all links

Enform Plus allows you to specify up to 32 LINK or LINK OPTIONAL statements. If you include the LINK OPTIONAL statement in your query specifications, be sure to follow the rules described later in this section under [LINK OPTIONAL Statement Considerations](#) on page 4-26.

LINK Statement Considerations

When you use a single LINK statement, Enform Plus builds a set of logical records by searching through the data files associated with the linked record descriptions. A record from one data file becomes part of the set of logical records only if the content of its linking field matches the content of the linking field in a record in the other data file.

When you use a LINK statement, you indicate that the values of the linking fields must be equal. (If you want to indicate inequality for the values of the linking fields, you must use a WHERE clause.) A single LINK statement is equivalent to the following form of the WHERE clause:

```
WHERE qualified-field-name1 EQ qualified-field-name2,
```

When you use a LINK statement, the order in which you enter the record description names is unimportant. For example:

```
LINK employee.empnum TO branch.manager;
```

is the same as:

```
LINK branch.manager TO employee.empnum;
```

LINK OPTIONAL Statement Considerations

When you establish a link by specifying the LINK OPTIONAL statement, you preserve all of the information from the data file whose record description is specified on the left side of the statement. You preserve this information because Enform Plus builds the logical records for such a link in a different manner than it does for a link established by a LINK statement. For example, suppose that you entered the following LINK OPTIONAL statement:

```
LINK employee TO OPTIONAL branch VIA branchnum;
```

This statement causes Enform Plus to build:

- A logical record description that consists of fields from both employee and branch.
- One logical record for each matching employee and branch record. (Each logical record contains field values from both records.)
- One logical record for each employee record for which there is not a corresponding branch record. Enform Plus builds this logical record by supplying blanks for the fields in the logical record that correspond to the branch record description.

Because of the way that Enform Plus builds the logical records for this type of link, the order in which you enter the record description names is important. For example:

```
LINK employee TO OPTIONAL branch VIA branchnum;
```

is not the same as:

```
LINK branch TO OPTIONAL employee VIA branchnum;
```

The first LINK OPTIONAL statement preserves the information in the employee records; the second preserves the information in the branch records.

Note. If you specify a LINK OPTIONAL statement for record descriptions that are associated with data files containing variable-length records, the linking field must be present in all of the records from both data files. If the linking field is not present, a sort operation might fail.

Rules for Using LINK OPTIONAL Statements

When you include LINK OPTIONAL statements in your query specifications, Enform Plus requires you to adhere to certain rules for all of the links that affect your query. These rules exist to eliminate the possibility of ambiguous links and to eliminate links that are not supported by the current query processor.

Enform Plus uses these rules to determine whether your query specifications contain legal links. (Unless otherwise stated, the word link applies to a link established by a WHERE clause, a LINK statement, or a LINK OPTIONAL statement.) These rules are:

- A record description on the right side of a LINK OPTIONAL statement must not be linked back to the record description on the left side of the same LINK OPTIONAL statement.
- A given record description must not appear on the right side of more than one LINK OPTIONAL statement. A record description appears on the right side of more than one LINK OPTIONAL statement when you:
 - Specify the same record description on the right side of two or more LINK OPTIONAL statements.
 - Specify a link (either with a LINK statement or a WHERE clause) between two record descriptions, both of which appear on the right side of a LINK OPTIONAL statement.

Note. When checking your query for violations of these rules, Enform Plus ignores any links (initiated by either a LINK statement or a WHERE clause) that reference the same record descriptions as referenced in a LINK OPTIONAL statement. In effect, Enform Plus “cancels” any other link between the record descriptions when determining the legality of the links. Enform Plus ignores these links only when checking for violations of the LINK OPTIONAL rules. After Enform Plus determines that all the links in your query are legal, Enform Plus uses each LINK statement and WHERE clause to evaluate your query.

See [Appendix C, Links and the LINK OPTIONAL Statement Rules](#) for more detailed information about links and the LINK OPTIONAL statement.

How Enform Plus Handles “Noncontributing” Record Descriptions

When your query specifications include LINK OPTIONAL statements, Enform Plus produces target records from the logical records built for the link in a normal manner with certain exceptions. These exceptions involve “noncontributing” record descriptions. For a given logical record, a record description is “noncontributing” if the logical record does not contain any data values from the data file associated with the record description. For example, consider the following query specifications:

```
OPEN employee, branch;
LINK employee TO OPTIONAL branch VIA branchnum;
LIST empname, branchname;
```

Each logical record built for the preceding query will contain data values from the employee records; therefore, the employee record description is said to “contribute” to each logical record. However, some of the logical records might not contain data values from the branch records. For these logical records, the branch record description is said to be “noncontributing.”

Any record description that is linked to a “noncontributing” record description is also “noncontributing.” For example, consider the following query specifications:

```
OPEN employee, region, order;
LINK order.salesman TO OPTIONAL employee.empnum;
LIST ordernum, empname, regname
WHERE employee.regnum = region.regnum;
```

If the employee record description is “noncontributing” for a given logical record, the region record description is also “noncontributing” for that logical record.

Enform Plus handles “noncontributing” records as follows:

- If your query specifications contain a WHERE clause, Enform Plus evaluates each logical record to determine whether it satisfies the terms of the WHERE clause. For a given logical record, Enform Plus ignores any term in a WHERE clause that references a “noncontributing” record description. If several record descriptions are “noncontributing” for a given logical record, Enform Plus ignores all terms in the WHERE clause that reference the “noncontributing” record descriptions. If all of the

record descriptions referenced in the WHERE clause contribute to the logical record, Enform Plus uses all the terms in the WHERE clause.

- For a given logical record, Enform Plus supplies null values for any fields that correspond to a “noncontributing” record description. If your query specifications include expressions or aggregates that reference these null field values, Enform Plus computes the expression or aggregate value by using blank values. In some cases, this use of blanks might cause undesirable results. For example:
 - If you specify that the aggregates COUNT, MAX, or MIN (or a similar user aggregate) are to be performed for a field containing null values, Enform Plus will count or compare the null values just like any other value.
 - If you specify that the aggregates SUM or AVG (or a similar user aggregate) are to be performed for a binary numeric field containing null values, Enform Plus will include the null values as ASCII blanks in the resulting computations.
 - If you include the UNIQUE specification with an aggregate, the null value will be considered a UNIQUE value. (This will occur if null values are not stored in the field for which the aggregate is being computed.)
 - If a binary numeric field that contains null values is included in an arithmetic expression, Enform Plus will include the null values as ASCII blanks in the specified computations. Because null values correspond to a very large binary number, this could result in a very large, and probably, meaningless, result.

If your query involves aggregates, you can eliminate these invalid results by including the WHERE option with the aggregate. For example, suppose that you want to use the aggregate COUNT to count the occurrences of no-num, a two-character field that is declared alphanumeric (PIC XX or TYPE CHARACTER 2) in the record description. If you specify the aggregate as follows:

```
COUNT (no-num WHERE no-num <> BLANKS)
```

Enform Plus does not count any occurrences of no-num that contain null values because Enform Plus treats null values as blanks in alphanumeric fields.

If no-num was declared numeric (PIC 99) in the record description, you could specify the aggregate as follows:

```
COUNT (no-num WHERE no-num <> 0)
```

Enform Plus does not count any occurrences of no-num that contain null values because Enform Plus treats null values in numeric fields as zeros.

If no-num was declared to be a two-byte binary numeric field (PIC 99 USAGE IS COMP or TYPE BINARY 16) in the record description, you could eliminate an invalid result by specifying:

```
COUNT (no-num WHERE no-num <> 8224)
```

Enform Plus does not count any occurrences of no-num that contain null values because Enform Plus treats null values in two-character binary fields as decimal 8224.

LIST Statement

The LIST statement allows you to select the information printed in a report and prints the report. The LIST statement must end with a semicolon. The syntax of the LIST statement is:

```

LIST [ UNIQUE ] {
    { BY by-item
      BY DESC by-item
        target-item
      ASCD target-item
      DESC target-item
      user-var-name := target-item }
    [ CUM [ OVER ALL ] ]
    [ CUM OVER by-item ]
    [ PCT [ OVER ALL ] ]
    [ PCT OVER by-item ]
    [ TOTAL ]
    [ SUBTOTAL ]
    [ SUBTOTAL OVER by-item ] , ... , ...
    [ NOHEAD ]
    [ NOPRINT ]
    [ CENTER ]
    [ HEADING string-literal ]
    [ AS display-format ]
    [ AS DATE display-format ]
    [ AS TIME display-format ]

    [ FORM [ n ] ]
    [ SKIP [ n ] ] , ...
    [ SPACE [ n ] ]
    [ TAB [ n ] ]

    [ WHERE logical-expression ]

    [ NOHEAD ALL ]
    [ NOPRINT ALL ]
    [ CENTER ALL ]

    [ SUPPRESS [ WHERE ] logical-expression ]

    [ BEFORE CHANGE [ON] by-item PRINT print-list [ CENTER ] ]
    [ AFTER CHANGE [ON] by-item PRINT print-list [ CENTER ] ]
    [ AT START PRINT print-list [ CENTER ] ]
    [ AT END PRINT print-list [ CENTER ] ]
    [ TITLE print-list [ CENTER ] ]
    [ SUBTITLE print-list [ CENTER ] ]
    [ FOOTING print-list [ CENTER ] ]
    [ SUBFOOTING print-list [ CENTER ] ] ;

```

UNIQUE

prevents identical records from contributing to the report. UNIQUE adds processing overhead and should not be used unless undesirable duplicate records are known to exist.

by-item

is the name of a field modified by a BY or a BY DESC clause. Enform Plus sorts and groups the report according to the value of this field. You cannot specify more than 63 *by-items* in your query.

target-item

is a record name, a field name, a literal, a predefined aggregate, a user aggregate, an arithmetic expression, an IF/THEN/ELSE expression, a user table name, a user variable, or a System Variable clause.

user-var-name

is the name of a user variable.

string-literal

is one or more alphanumeric characters enclosed within quotation marks.

display-format

is the format in which you want an element displayed.

/

is a symbol that is equivalent to the SKIP clause. You can specify as many of these symbols as you want to indicate the number of lines to advance.

logical-expression

is an expression that returns a true or false value.

print-list

contains any combination of literals, FORM, SKIP, SPACE or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, and parameter names.

Input Record Description

When you use a LIST statement, Enform Plus opens the data files associated with any open record descriptions. (You open record descriptions by using OPEN statement.) Enform Plus obtains information about the structure of the data file from the record description. The record description describes the file type (key-sequenced, relative,

entry-sequenced, unstructured) of the data file. The actual file type of a data file must be the same as the file type described in the record description; otherwise, your query might return incorrect results.

Group Definition and Sorting

BY and BY DESC clauses group and sort records. Enform Plus prints the by-item value for a group only for the first record of all the records that have the same value for the *by-item*. A *by-item* can be referred to by CUM OVER, PCT OVER, SUBTOTAL OVER, AFTER CHANGE, and BEFORE CHANGE clauses. These clauses are described in [Section 5, Clauses](#).

ASCD and DESC clauses sort records in ascending or descending order. They do not identify a group. The ASCD and DESC clauses are described in [Section 5, Clauses](#).

When a LIST statement contains more than one BY, BY DESC, ASCD, or DESC clause, Enform Plus determines a major to minor sorting precedence. Enform Plus determines the sorting precedence by the order in which the clauses appear in the LIST statement. The first clause has highest priority and is sorted first, the next one second priority, down to the last specified clause.

How Values Are Displayed in Report Columns

Enform Plus displays *target-items* and *by-items* in report columns, one column per item. If a record name is a *target-item* in a LIST statement, Enform Plus expands the record to as many *target-items* as the number of elementary fields in the record. If a field within the record is described with an OCCURS clause, Enform Plus prints each occurrence of the field in a separate column of the report. Do not specify a record name as a *target-item* if it is the same as a field name in a record. See [Section 3, Enform Plus Language Elements](#) for information about field names. A *target-item* in a LIST statement can be any of the following:

- A record name

```
LIST ...,
  parts;
```

- The name of a field

```
LIST ...,
  partnum;
```

- A numeric literal. Enclose numeric literals within parentheses.

```
LIST ...,
  (2001);
```

- A string literal. Enclose string literals within quotation marks.

```
LIST ...,
  "manager";
```

- An arithmetic expression enclosed in parentheses. The expression can use any combination of numeric fields, numeric literals, numeric user variables, predefined aggregates, user aggregates, JULIAN-DATE clauses, or System Variable clauses.

```
LIST ...,
    (price * quantity);
```

- An IF/THEN/ELSE expression. Enclose IF/THEN/ELSE expressions in parentheses.

```
LIST ...,
    (IF inventory GT 0 THEN inventory ELSE ZERO);
```

- A user variable. Define the user variable with a DECLARE statement. The variable should be either initialized with a SET statement or assigned a value with assignment syntax. If the variable value is neither initialized nor assigned, the default value is zero. A user variable cannot be assigned the values of a subscript range.

```
DECLARE new-var;
SET new-var TO 3;
LIST ... ,
    new-var;
```

- A System Variable clause.

```
LIST ... ,
    @LINENO AS I5;
```

- A user aggregate or predefined aggregate.

```
LIST ...,
    COUNT (branchnum OVER regnum);
```

- A field or user table with a subscript range.

```
LIST ... ,
    months [ 1:6 ];
```

- A parameter name. Parameters that have not been declared with an alphanumeric internal format must be enclosed in parentheses. Parameters declared with an alphanumeric internal format must not be enclosed within parentheses.

```
PARAM num;
...
LIST ...,
    (num);
```

Enform Plus allows you to specify a group name as a target-item within a LIST statement; however, Enform Plus might not display the fields within the group in the manner you expect. To understand the way that Enform Plus displays group elements,

you must first understand what a group is. A group is defined in DDL as any field whose level number (03, 04, 05, and so on) is less than that of the next field. For example, consider the following DDL record description:

```
RECORD test.
FILE IS "$mkt.sample.test" key-sequenced.
02 account-num.
    05 type          PIC 9(4).
    05 num           TYPE BINARY 16.
02 custnum          PIC 9(4).
end
```

Within this record, `account-num` is a group. The data type of a group is always alphanumeric. When a group name is specified as a *target-item* within a `LIST` statement, Enform Plus displays each field within the group as alphanumeric data. If one of the fields within a group contains binary data, using a group name as a *target-item* causes undesirable results. For example, specifying:

```
LIST account-num;
```

causes Enform Plus to display `account-num` without first converting it to a readable form.

Request-Qualification

Use the `WHERE` clause to limit the records that contribute to the report. The `WHERE` clause is described under [WHERE Clause](#) on page 5-72.

Conditional Printing

Use the `SUPPRESS` clause to define a condition or conditions that prevent specific records from printing throughout a report. Enform Plus still includes the suppressed records in `AFTER CHANGE` and `BEGIN CHANGE` clauses, subtotals, totals, and other calculations specified for the report. Note that the value of the first record of an `AFTER CHANGE` clause or the last record of a `BEFORE CHANGE` clause is used for the *print-list* regardless of whether that record is printed. The `SUPPRESS` clause is described under [SUPPRESS Clause](#) on page 5-64.

Summary Reports

Summary reports contain only the first target-record from each group (created by a `BY` or `BY DESC` clause) down to some level. Summary reports can be generated only by a query that contains an aggregate.

The two methods of obtaining a summary report are:

- Explicitly request a summary report by setting the `@SUMMARY-ONLY` option variable to `ON` before issuing the `LIST` statement.
- Implicitly request a summary report by specifying only *by-items* and aggregates over *by-items* in the query.

When you explicitly request a summary report, you get a report summarized down to the lowest level where an aggregate is calculated over that level. For example:

```
SET @SUMMARY-ONLY TO ON;
LIST BY dept, BY job, BY empname, salary,
COUNT(empname OVER job);
```

returns one record for each job in each department. Only the first employee name (empname) for each job is returned, as follows:

DEPT	JOB	EMPNAME	SALARY	COUNT EMPNAME
----	---	-----	-----	-----
0101	MANAGER	JACK RAYMOND	36000	2
	SALESMAN	JIM HERMAN	19000	2
	SECRETARY	KATHRYN DAY	12000	1
	SYS.-ANAL	LARRY CLARK	25000	1
0102	MANAGER	TONY CRAFT	37000	1
	SALESMAN	GEORGE FORSTER	30000	1
0201	MANAGER	DAVID STRAND	39000	2
	PROGRAMMER	SUSAN CHAPMAN	17000	1

When you implicitly request a summary report, you also get a report summarized down to the lowest level where an aggregate is computed over that level. (A query requesting only by-items and aggregates over ALL is not an implicit request for a summary report.) For example:

```
SET @SUMMARY-ONLY TO OFF;
LIST BY dept, BY job, BY empname,
COUNT (empname OVER job);
```

returns one record for each job in each department. Again, only the first employee name (empname) for each job is returned:

DEPT	JOB	EMPNAME	COUNT EMPNAME
----	---	-----	-----
0101	MANAGER	JACK RAYMOND	2
	SALESMAN	JIM HERMAN	2
	SECRETARY	KATHRYN DAY	1
	SYS.-ANAL	LARRY CLARK	1
0102	MANAGER	TONY CRAFT	1
	SALESMAN	GEORGE FORSTER	1
0201	MANAGER	DAVID STRAND	2
	PROGRAMMER	SUSAN CHAPMAN	1

If you want a report that consists of only by-items and aggregates to contain all report lines (including those where the last value of a *by-item* has not changed but a subordinate *by-item* value has changed), then you must include a *target-item* in the target-list that has not appeared as a *by-item* or aggregate. This *target-item*

can be modified by the NOPRINT clause if desired. Of course, you must also set @SUMMARY-ONLY to OFF. For example:

```
SET @SUMMARY-ONLY TO OFF;
LIST BY dept, BY job, BY empname, salary NOPRINT,
COUNT (empname OVER job);
```

returns more than one record for each job in each department. All the employee names (empname) for each job are returned, as follows:

DEPT	JOB	EMPNAME	COUNT EMPNAME
----	---	-----	-----
0101	MANAGER	JACK RAYMOND	2
		TIM WALKER	
	SALESMAN	JIM HERMAN	2
		TOM HALL	
	SECRETARY	KATHRYN DAY	1
	SYS.-ANAL	LARRY CLARK	1
0102	MANAGER	TONY CRAFT	1
	SALESMAN	GEORGE FORSTER	1
0201	MANAGER	DAVID STRAND	2
		JERRY HOWARD	
	PROGRAMMER	SUSAN CHAPMAN	1

Optional Clauses

These clauses are optional and are described in [Section 5, Clauses](#). If you use one of these clauses to modify a *target-item* that contains a subscript range, the clause modifies each element in the range. Briefly, the clauses do the following:

- AFTER CHANGE clause prints information preceding the records for each by-item within a printed report.
- AS clause specifies the display format for a *target-item* or a *by-item*.
- AS DATE clause specifies the date format for printing a date.
- AS TIME clause specifies the time format for printing a time.
- AT END PRINT clause prints information at the end of a report.
- AT START PRINT clause prints information at the beginning of a report.
- BEFORE CHANGE clause prints information following the records for each *by-item* within a printed report.
- CENTER clause centers an object within its context.
- CUM clause prints a running total for a *target-item* or *by-item*.
- FOOTING clause prints a footing for a report.
- HEADING clause specifies a column heading for a report.

- NOHEAD clause suppresses the printing of the column heading for a *target-item* or *by-item*.
- NOPRINT clause suppresses the printing of a *target-item* or *by-item*.
- PCT clause prints a percentage for a *target-item* or *by-item*.
- SUBFOOTING clause prints a subfooting for a report.
- SUPPRESS clause defines a condition that prevents specific records from printing in a report.
- SUBTOTAL clause prints a subtotal for a *target-item* within each *by-item*.
- SUBTITLE clause prints a subtitle for a report.
- TITLE clause prints a title for a report.
- TOTAL clause prints a total for a *target-item* or *by-item*.
- WHERE clause limits the records that contribute to the report.

OPEN Statement

The OPEN statement accesses the dictionary record description. The syntax of the OPEN statement is:

```
OPEN { record-name
      { record-name2 [ AS ] COPY [ OF ] record-name1 }, ... [ ; ]
```

record-name and *record-name1*

are the names of dictionary record descriptions.

record-name2

is a name that you supply for the copy of *record-name1*. *record-name2* can be the name of a dictionary record description.

Specifying an OPEN statement does not actually open a physical file; rather, the OPEN statement causes Enform Plus to access the dictionary record description of the file. One OPEN statement can open several record descriptions. Enform Plus stores the record description in the internal table. The record description remains open until one of the following occurs:

- A CLOSE statement clears the record description.
- Either a DICTIONARY statement or ?DICTIONARY command clears the entire internal table.
- The current Enform Plus session ends.

Using OPEN AS COPY OF

The OPEN AS COPY OF statement is useful when a record description is designed so that records within a file relate to other records within the same file. Because LINK statements associate record descriptions only with different record names, the OPEN AS COPY OF statement refers to the same record description by a different record name.

Note that although the OPEN AS COPY OF statement opens *record-name2*, you must open *record-name1* before specifying this statement. Note also that *record-name1* cannot be specified as *record-name2* in another OPEN AS COPY OF statement.

If you want to assign *record-name2* to a file other than the file specified in the dictionary for *record-name1*, you must use an Enform Plus ?ASSIGN command or a TACL ASSIGN command to assign *record-name2*. An assignment of *record-name1* does not apply to *record-name2*.

PARAM Statement

The PARAM statement allows you to name and define a parameter that can receive a value from a TACL PARAM command. The syntax of the PARAM statement is:

```
PARAM { param-name [ INTERNAL internal-format ] } , ... [ ; ]
```

param-name

is the name of the parameter being defined. The name must conform to the naming conventions described under [Rules for Naming User-Defined Elements](#) on page 3-6.

internal-format

is the internal format for storing the parameter. Valid values for internal-format are:

An alphanumeric, where *n* is the length.

In integer, where *n* is the length.

Fw.d fixed, where *w* is the number of digits, and *d* is the number of decimal places.

The default is a 64-bit signed integer.

Up to 32 parameters can be defined per Enform Plus session. The SET statement can initialize the value of the parameter. The following PARAM statement defines a parameter named regno:

```
PARAM regno INTERNAL I2;
```

Values for parameters can be passed only to a running compiled query file. (A compiled query file is created by the ?COMPILE command, as described under [?COMPILE Command](#) on page 6-7.) Before executing the stored compiled query file, a TACL PARAM command can be used to pass a value for the parameter. The value specified by the PARAM command supersedes the parameter value specified by a SET statement. The PARAM command must precede the TACL ENFORM run command. See the *Guardian User's Guide* for more information about the TACL PARAM command.

Parameter values can also be passed through the host-language interface. See the *Enform User's Guide* for information about the host-language interface.

How Enform Plus Handles Parameters

Enform Plus handles a parameter syntactically as if it were a literal. Enform Plus handles parameters declared with an alphanumeric internal format as string literals. Enform Plus handles all other parameters as numeric literals. When you specify a parameter as a *target-item* or as an item in a *print-list*, you must enclose the parameter in parentheses just as you would an actual numeric literal. For example, suppose rept, a compiled query file, contains the following Enform Plus statements:

```
PARAM reptnum 13;
TITLE "REPORT ", (reptnum);
OPEN parts;
LIST parts;
```

Notice that `reptnum` is enclosed in parentheses when specified in the `TITLE` statement. To provide a value for `reptnum`, enter the following `TACL PARAM` command:

```
nn> PARAM reptnum 333
```

If you then specify the `ENFORM` command:

```
nn> ENFORM /IN rept,OUT $s/
```

the resulting report is:

```
REPORT 333
```

Part Number	PARTNAME	INVENTORY	LOCATION	PRICE
212	SYSTEM 192KB CORE	7	J87	92000.00
244	SYSTEM 192KB SEMI	3	B78	87000.00
1403	PROC 96KB SEMI	21	A21	22000.00
...

You can use a parameter wherever a literal is allowed. In certain cases, Enform Plus allows you to use a parameter but treats the parameter strictly as a numeric literal. Therefore, you cannot change the value of the parameter at execution time. Enform Plus treats a parameter strictly as a numeric literal when you use the parameter as:

- A subscript
- The “*max-subscript*” in the declaration of a user table
- The integer in a `FORM`, `SKIP`, `TAB`, or `SPACE` clause
- The integer in a pattern-match string of a logical expression

Enform Plus issues a warning message when you have specified a parameter in this manner.

SET Statement

The SET statement allows you to initialize or reset a user variable, user table, or a parameter. The SET statement also allows you to reset option variables. The syntax of the SET statement is:

SET	{	{	<i>user-variable-name</i>	}	[TO]	{	<i>string-literal</i>	}
			<i>user-table-name</i>					
			"[" <i>subscript</i> "]"				<i>number-literal</i>	
			<i>param-name</i>					
							{	
							ON	
							OFF	
			<i>option-variable-name</i>		[TO]		<i>unsigned-digit</i>	
							<i>string-literal</i>	
							<i>display-format</i>	

user-variable-name or *user-table-name* "["*subscript*"]"

is the name of an element defined by a DECLARE statement.

param-name

is the name of a parameter defined by a PARAM statement.

string-literal

is one or more alphanumeric characters enclosed within quotation marks.

unsigned-digit

is an unsigned numeric literal.

option-variable-name

is the name of an option variable.

numeric-literal

is an integer.

display-format

is the default display format enclosed in quotation marks for printing dates or times when AS DATE * and AS TIME * clauses are used.

Initializing User-Defined Elements

When initializing a user variable, user table, or a parameter, specify values that are consistent with the internal format type. If the internal format type is alphanumeric, specify a string literal; if the internal format type is numeric, specify a numeric literal. In

the following example, the user variable `u-var` is defined as numeric by the `DECLARE` statement and set to the value of 99 by the `SET` statement:

```
DECLARE u-var INTERNAL I2;  
SET u-var TO 99;
```

The `SET` statement can initialize a user table. An unsigned numeric integer subscript must be used. In the following example, the third element of the `months` table is set to `MARCH`:

```
DECLARE months [12] INTERNAL A10;  
SET months [3] TO "MARCH";
```

Redefining Option Variables

Use the `SET` statement to redefine option variables. Several option variables exist, each with default values assigned by Enform Plus. See [Option Variable Clauses](#) on page 5-48 for information about valid values for the option variables.

SUBFOOTING Statement

The SUBFOOTING statement allows you to specify a subfooting to be printed at the bottom of each report page for all reports in the current session unless overridden or reset by another SUBFOOTING statement or temporarily overridden by a SUBFOOTING clause. See the [SUBFOOTING Clause](#) on page 5-59. The syntax of the SUBFOOTING statement is:

```
SUBFOOTING [ print-list [ CENTER ] ] [ ; ]
```

print-list

can be any combination of literals, FORM, SKIP, SPACE, or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, and parameter names.

The clauses that can be used in a *print-list* are described in [Section 5, Clauses](#). The other elements are described in [Section 3, Enform Plus Language Elements](#).

Specifying Field Names Within a SUBFOOTING Statement

If you specify a field name within a print-list of a SUBFOOTING statement, the field value printed is the same value as in the last row of data on the current page. A field appearing within the SUBFOOTING statement need not be explicitly included within the following LIST statements. If a SUBFOOTING statement field is not included, Enform Plus effectively adds it to the LIST statement with a NOPRINT clause.

Spacing Considerations

By default, the subfooting begins in the same column position as the leftmost report column. Using either SPACE or TAB clauses as the first element of the print-list overrides this default. The SPACE or TAB clauses can also appear anywhere within the print-list. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
SUBFOOTING "Inventory" SPACE 15 "Parts in Stock";
```

The following subfooting will appear at the bottom of the next report that is generated without a LIST statement SUBFOOTING clause:

```
Inventory           Parts in Stock
```

If you specify either a SKIP clause or the symbol / (slash) within a *print-list*, the printer advances one or more lines before printing the rest of the SUBFOOTING print-list. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option

variable @VSPACE. In the following example, the SKIP clause of the SUBFOOTING statement causes the subfooting to print on two lines:

```
SUBFOOTING "Report 2-A" SKIP "Total Sales";
```

The following subfooting will appear at the bottom of the next report that is generated without a LIST statement SUBFOOTING clause.

```
Report 2-A  
Total Sales
```

By using the FORM clause within a SUBFOOTING statement, you can force a new page. The remainder of the SUBFOOTING print-list is printed on the new page. The page number remains the same. A single logical page can span multiple physical pages such that a TITLE can appear on one page, the data on the next, and a SUBFOOTING on the next. The same page number applies to all physical pages in a logical page.

By using the CENTER clause within the SUBFOOTING statement, you can center the subfooting on the page.

The CENTER, Option Variable, SKIP, SPACE, and TAB clauses are described in [Section 5, Clauses](#).

Subfooting for Current Report or All Reports

A SUBFOOTING statement prints a subfooting at the bottom of each page for all subsequent reports in the current session. Specifying a new SUBFOOTING statement with a new print-list resets the current SUBFOOTING statement. A SUBFOOTING statement can be temporarily overridden by a SUBFOOTING clause within a LIST statement. A SUBFOOTING clause prints a subfooting for the current report only.

Canceling Session-Wide Subfooting

You can cancel a SUBFOOTING statement by:

- Using a SUBFOOTING statement without the print-list parameter
- Using either a DICTIONARY statement or a ?DICTIONARY command (which clears the SUBFOOTING information from the internal table)

SUBTITLE Statement

The SUBTITLE statement allows you to specify a subtitle for all subsequent reports in the current session. The subtitle is printed at the top of each page immediately following the title. The SUBTITLE statement can be overridden or reset by another SUBTITLE statement or temporarily overridden by a SUBTITLE clause. The syntax of the SUBTITLE statement is:

```
SUBTITLE [ print-list [ CENTER ] ] [ ; ]
```

print-list

can be any combination of literals, FORM, SKIP, SPACE or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, and parameter names.

The clauses that can be used in a *print-list* are described in [Section 5, Clauses](#). The other elements are described in [Section 3, Enform Plus Language Elements](#).

Specifying a Field Name Within a SUBTITLE Statement

If you specify a field name within the print-list of a SUBTITLE statement, the field name has the same value as in the first row of the page. A field name appearing within the SUBTITLE statement need not be explicitly included within the following LIST statements. If the field name is not included, Enform Plus effectively adds it with a NOPRINT clause.

Spacing Considerations

By default the subtitle begins printing in the same column position as the leftmost report column. Using SPACE or TAB clauses as the first element of the print-list overrides this default. SPACE or TAB clauses can also appear anywhere within the print-list. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
SUBTITLE "Inventory" SPACE 15 "Parts in Stock";
```

```
Inventory           Parts in Stock
```

If you specify either a SKIP clause or the slash symbol (/) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the SUBTITLE *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause of the SUBTITLE statement causes two lines to be printed:

```
SUBTITLE "Report 2-A" SKIP "Total Sales";
```

```
Report 2-A  
Total Sales
```

By using the `FORM` clause within a `SUBTITLE` statement, you can force a new page. Enform Plus prints the remainder of the `SUBTITLE` *print-list*, starting at the top of the next physical page. The page number remains the same. A single logical page can span multiple physical pages such that a `SUBTITLE` can appear on one page, the data on the next, and a `FOOTING` on the next. The same page number applies to all physical pages in a logical page.

By using the `CENTER` clause within a `SUBTITLE` statement, you can center the subtitle on the page.

The `CENTER`, `Option Variable`, `SKIP`, `SPACE`, and `TAB` clauses are described in [Section 5, Clauses](#).

Subtitle for Current Report or All Reports

A `SUBTITLE` statement prints a subtitle at the top of each page immediately following the title for all subsequent reports in the current session. The `SUBTITLE` statement can be reset by specifying a new `SUBTITLE` statement with a different `print-list`. A `SUBTITLE` clause within a `LIST` statement temporarily overrides the `SUBTITLE` statement. A `SUBTITLE` clause within a `LIST` statement prints a subtitle for the current report only.

Canceling Session-Wide Subtitle

You can cancel a `SUBTITLE` statement by:

- Using a `SUBTITLE` statement without the `print-list` parameter
- Using either a `DICTIONARY` statement or a `?DICTIONARY` command (which clears the `SUBTITLE` information from the internal table)

TITLE Statement

The TITLE statement allows you to specify a title to be printed the top of each page for all subsequent reports in the current session unless canceled or reset by another TITLE statement or temporarily overridden by a TITLE clause. (See the [TITLE Clause](#) on page 5-69.) The syntax of the TITLE statement is:

```
TITLE [ print-list [ CENTER ] ] [ ; ]
```

print-list

can be any combination of literals, FORM, SKIP, SPACE or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, and parameter names.

The clauses that can be used in a *print-list* are described in [Section 5, Clauses](#). The other elements are described in [Section 3, Enform Plus Language Elements](#).

Specifying a Field Name Within a TITLE Statement

If you specify a field name within the *print-list* of a TITLE statement, the field has the same value as in the first row of the page. A field appearing within the TITLE statement need not be explicitly included within the following LIST statements. If it is not included, Enform Plus effectively adds it with a NOPRINT clause.

Spacing Considerations

By default, the title begins printing in the same column position as the leftmost report column. By using SPACE or TAB clauses as the first element of the *print-list*, you can override this default. SPACE or TAB clauses can also appear anywhere within the *print-list*. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
TITLE "Inventory" SPACE 15 "Parts in Stock";
```

```
Inventory           Parts in Stock
```

If you specify either a SKIP clause or the symbol / (slash) within a *print-list*, the printer advances one or more lines before printing the rest of the TITLE *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause of the TITLE statement causes two lines to be printed:

```
TITLE "Report 2-A" SKIP "Total Sales";
```

```
Report 2-A  
Total Sales
```

By using the `FORM` clause within a `TITLE` statement, you can force a new page. The remainder of the `TITLE print-list` is printed, starting at the top of the next physical page. The page number remains the same. A single logical page can span multiple physical pages such that a `TITLE` can appear on one page, the data on the next, and a `FOOTING` on the next. The same page number applies to all physical pages in a logical page.

By using the `CENTER` clause within a `TITLE` statement, you can center the title on the page.

The `CENTER`, `Option Variable`, `SKIP`, `SPACE`, and `TAB` clauses are described in [Section 5, Clauses](#).

Title for Current Report or All Reports

A `TITLE` statement prints a title at the top of each page for all subsequent reports in the current session. Specifying a new `TITLE` statement resets the `TITLE` statement to the value of the new `print-list`. A `TITLE` statement can be temporarily overridden by a `TITLE` clause within a `LIST` statement. A `TITLE` clause within a `LIST` statement prints a title for the current report only.

Canceling Session-Wide Title

You can cancel a `TITLE` statement by:

- Using the `TITLE` statement without the `print-list` parameter
- Using the `DICTIONARY` statement or `?DICTIONARY` command (which clears the `TITLE` information from the internal table)

5 Clauses

This section describes the syntax of the Enform Plus clauses. The clauses are specified in alphabetic order.

Enform Plus clauses are components of statements. They provide additional specifications of the Enform Plus program to be performed. Most of the Enform Plus clauses remain in effect only for the query associated with the LIST or FIND statement of which they are a part.

An Enform Plus program with a FIND statement produces records only; it has no report features such as headings, titles, summary information, and special formatting. The following clauses are applicable only to Enform Plus programs with a LIST statement and cannot be used in programs with a FIND statement:

AFTER CHANGE clause	NOPRINT clause
AS clause	PCT clause
AS DATE clause	SKIP clause
AS TIME clause	SPACE clause
AT END PRINT clause	SUBFOOTING clause
AT START PRINT clause	SUBTITLE clause
BEFORE CHANGE clause	SUBTOTAL clause
CENTER clause	SUPPRESS clause
CUM clause	System Variable clauses
FORM clause	TAB clause
FOOTING clause	TITLE clause
HEADING clause	TOTAL clause
NOHEAD clause	WHERE clause
Several of the Option Variable clauses	

[Table 5-1](#) lists the Enform Plus clauses and their functions.

Table 5-1. Enform Plus Clauses and Their Functions (page 1 of 3)

Field Selection, Grouping, and Sorting

Clause	Function
ASCD and DESC	Sort <i>target-records</i> in ascending or descending order, respectively, according to the value of a specified field.
BY and BY DESC	Group and sort <i>target-records</i> according to the value of a specified field.
SUPPRESS	Prevents certain records from being printed in the report, but does not prevent the records from contributing to the report calculations.
WHERE	Selects which records will contribute to the output.

Table 5-1. Enform Plus Clauses and Their Functions (page 2 of 3)**Calculating Running Total, Total, Subtotal, and Percentage**

Clause	Function
CUM	Prints a running total for a numeric <i>target-item</i> . The CUM OVER clause prints the running group total for a numeric <i>target-item</i> .
PCT	Prints the value of the percentage of the grand total for a numeric <i>target-item</i> . The PCT OVER clause prints the percentage that each grouped value represents of the total of all values in the group.
SUBTOTAL	Prints the value of the <i>target-item</i> subtotals for each group.
TOTAL	Prints the value of the grand total for a <i>target-item</i> .

Extracting Current Values for Date, Time, Line Number, and Page

Clause	Function
System Variable	Return the current value for the current date, time, line number, and page number.
TIMESTAMP-DATE	Extracts the date portion of a timestamp field that was created by the Guardian procedure TIMESTAMP.
TIMESTAMP-TIME	Extracts the time portion of a timestamp field that was created by the Guardian procedure TIMESTAMP.

Printing User-Supplied Information on a Report

Clause	Function
AFTER CHANGE	Prints information for the current report preceding the records for each grouped field value.
AT END PRINT	Prints information at the end of the current report. See also the AT END Statement on page 4-3.
AT START PRINT	Prints information just before the first set of column headings for the current report. See also the AT START Statement on page 4-5.
BEFORE CHANGE	Prints information for the current report following the records for each grouped field value.
FOOTING	Prints a footing at the bottom of each page for the current report. See also the FOOTING Statement on page 4-23.
SUBFOOTING	Prints a subfooting at the bottom of each page immediately preceding the footing for the current report. See also the SUBFOOTING Statement on page 4-43.
SUBTITLE	Prints a subtitle at the top of each page immediately following the title for the current report. See also the SUBTITLE Statement on page 4-45.
TITLE	Prints a title at the top of each page for the current report. See also the TITLE Statement on page 4-47.

Table 5-1. Enform Plus Clauses and Their Functions (page 3 of 3)

Converting Data to Internal or Display Format

Clause	Function
AS	Specifies the display format for printing a <i>target-item</i> or <i>by-item</i> .
AS DATE	Specifies the display format for printing a date.
AS TIME	Specifies the display format for printing a time.
INTERNAL	Specifies the storage format for a user-defined element.
JULIAN-DATE	Translates date information to internal format.

Formatting a Report

Clause	Function
CENTER	Centers a <i>target-item</i> within its context.
FORM	Controls when to skip to a new page.
HEADING	Overrides the default column title for a <i>target-item</i> or <i>by-item</i> in a report.
NOHEAD	Suppresses the printing of the column heading for a <i>target-item</i> or <i>by-item</i>
NOPRINT	Suppresses the printing of a <i>target-item</i> or <i>by-item</i> and its column heading.
SKIP	Specifies how many lines to skip.
SPACE	Specifies horizontal spacing.
TAB	Specifies in which report column a <i>target-item</i> or <i>by-item</i> is to be printed.

Supplying Operational Variables

Clause	Function
Option Variable	Redefines the default values for several operational variables.

AFTER CHANGE Clause

The AFTER CHANGE clause prints information preceding the records for each group for the current report. The AFTER CHANGE clause is an optional part of a LIST statement. The syntax of the AFTER CHANGE clause is:

```
AFTER CHANGE [ ON ] by-item PRINT print-list [ CENTER ]
```

by-item

is the name of a field that has been grouped by a BY or BY DESC clause.

print-list

can contain any combination of literals, FORM, SKIP, SPACE, or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, or parameter names.

The clauses that can be part of a *print-list* are described in this section. The other elements are described in [Section 3, Enform Plus Language Elements](#)

The AFTER keyword of the AFTER CHANGE clause refers to the values printed, not to the location of the printed information. For the exact location of where the AFTER CHANGE information is printed within a report, see the *Enform User's Guide*.

When more than one AFTER CHANGE clause is specified, the specified information prints in the order in which the AFTER CHANGE clauses are entered.

Specifying a Field Name Within an AFTER CHANGE Clause

If you specify a field name within the *print-list* of an AFTER CHANGE clause, Enform Plus uses the same field value as in the first row of the next group. A field name appearing within the *print-list* of an AFTER CHANGE clause need not be explicitly included within the LIST statement. If the field name is not included, Enform Plus effectively adds it with a NOPRINT clause.

Spacing Considerations

By default the AFTER CHANGE clause information begins printing in the same column position as the leftmost report column. Using SPACE or TAB clauses as the first element of the *print-list* overrides this default. SPACE or TAB clauses can also appear anywhere within the print list. In the following example, the SPACE clause causes the two literals to be separated by five spaces:

```
AFTER CHANGE ON ordernum
  PRINT "*****" SPACE 5 "Orders for " ordernum,

*****      Orders for 122
```


If you specify either a SKIP clause or the slash symbol (/) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the AFTER CHANGE *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause within the AFTER CHANGE clause causes two lines to be printed:

```
AFTER CHANGE ON regnum PRINT "Beginning of Report for"  
    SKIP "Region " regnum,
```

```
Beginning of Report for  
Region 1
```

By using the CENTER clause following the *print-list* of an AFTER CHANGE clause, you can center the information on the page.

The CENTER, Option Variables, SKIP, SPACE, and TAB clauses are described in this section.

ASCD and DESC Clauses

The ASCD and DESC clauses sort target-records, in ascending or descending order respectively, according to the value of the specified field. The syntax of the ASCD and DESC clauses is:

```
{ ASCD } target-item  
{ DESC }
```

target-item

is the name of a field from an input record that serves as a sort key for the target records.

The ASCD and DESC clauses do not group field values. If you want field values both grouped and sorted, use the BY or BY DESC clauses; see [BY and BY DESC Clauses](#) on page 5-32.

When more than one ASCD, BY, BY DESC, or DESC clause is specified in a LIST or FIND statement, Enform Plus uses a major-to-minor sorting precedence. Enform Plus determines the sorting precedence by the order in which the ASCD or DESC clauses appear in the LIST or FIND statement. The first sort clause has the highest priority, the next one second priority, down to the last specified clause.

The following ASCD clause specifies that the target records are to be sorted on the value of `partnum`:

```
ASCD partnum,
```

AS Clause

The AS clause specifies a display format for printing a *target-item* or *by-item*. The syntax of the AS clause is:

```
{ report-item AS [ nonrepeatable-edit-descriptors ]
  repeatable-edit-descriptors
}
{ report-item AS " "[" [ decorations, ... ]
  [ modifiers, ... ] "]"
  repeatable-edit-descriptors "
}
{ report-item AS " "[" [ decorations, ... ]
  [ modifiers, ... ] "]"
  ( nonrepeatable-edit-descriptors
    repeatable-edit-descriptors ) "
}
```

report-item

is either a *by-item* or a *target-item*.

nonrepeatable-edit-descriptors

specify some general ways *report-items* are to be printed. *nonrepeatable-edit-descriptors* should not be specified without a *repeatable-edit-descriptor*. Valid values for *nonrepeatable-edit-descriptors* are:

<i>nP</i>	multiplies value by 10** <i>n</i> ; <i>n</i> is an integer.
S, SP, SS	control printing of the plus sign (+).

repeatable-edit-descriptors

specify data conversion to the Guardian Formatter for printing the *report-item* values. Valid values for *repeatable-edit-descriptors* are:

A [<i>w</i>]	for alphanumeric values
Iw [<i>.m</i>]	for integer values
Fw.d [<i>.m</i>]	for fixed-point values
M <i>mask</i>	for a template to combine literals and values

where

- w* specifies the width of the report-item.
- m* specifies the number of digits that appear to the left of the decimal for fixed-point values and the minimum number of digits for integer values
- d* specifies the number of digits to the right of the decimal.
- mask* is a combination of the characters 9, Z, V, . (period) and literals. The combination must be enclosed within apostrophes (' ') or greater-than and less-than symbols (< >).

" ["*decorations*"] "

specify character strings that can be added to a *report-item* depending on a condition. The syntax is as follows:

conditions location char-string

where

conditions

are one or more of the following:

- M add *char-string* if value is negative.
- N add *char-string* if value is null.
- P add *char-string* if value is positive.
- Z add *char-string* if value is zero.
- O add *char-string* if overflow condition occurs.

location

specifies where the character string is to be printed:

- An indicates *char-string* is to be printed at absolute position *n*.
- F indicates *char-string* is to be inserted after the value is formatted. If *condition* is satisfied, *char-string* is printed immediately to the left of the item value.
- P indicates *char-string* is inserted before the value is formatted. If *condition* is satisfied, *char-string* is printed to the right of the value.

char-string

is one or more alphanumeric characters enclosed within apostrophes (' ').

" [*modifiers*] "

alter the effect of the edit descriptors as follows:

BN, BZ	prints blanks for null or zero values, respectively.
FL <i>char</i>	specifies a substitute fill character.
OC <i>char</i>	respecifies the overflow character.
LJ, RJ	specifies right or left justification.
SS <i>pr-of-symbols</i>	allows substitution of symbols.

where

<i>char</i>	is an ASCII character enclosed in apostrophes.
<i>pr-of-symbols</i>	is a special mask symbol (see <i>repeatable-edit-descriptors</i>) and a substitution character.

The AS clause identifies how an *target-item* or *by-item* is printed in the current report.

Default Display Format.

When no AS clause is specified, Enform Plus looks for either a DDL display format (for fields) or a DECLARE statement (for variables). If neither of these is present, it uses the data type of the field. (The data type is derived from the DDL PIC clause or DECLARE INTERNAL clause, if present. For example, PIC 999999V99 is converted to F9.2—fixed, nine characters long with two decimal places.) If there is no PIC or INTERNAL clause, a 64-bit signed integer (or 20-character display format) is used.

The AS clause allows you to use some of the capabilities of the Guardian Formatter. When you specify a display format, Enform Plus passes your format specifications to the Formatter. See the *Guardian Programmer's Guide* for more information about the Formatter.

Repeatable Edit Descriptors

Repeatable edit descriptors specify the conversion of data values by the Formatter for the *target-item* or *by-item* being printed. The word *repeatable* has no real meaning in the context of Enform Plus; the word is used to simplify cross references to the Formatter terminology. Currently, Enform Plus makes only single *target-items* or *by-items* available for modification by edit descriptors.

Alphanumeric Edit Descriptor

The alphanumeric edit descriptor specifies the *target-item* or *by-item* is to be printed using alphanumeric display format. The syntax is:

A[w]

w

is an unsigned integer that specifies the width in characters of the value of the *target-item* or *by-item* to be printed. The maximum value for *w* is 255 characters.

If you specify *w*, Enform Plus prints the number of characters specified for the value of the *target-item* or *by-item*. If you omit *w*, Enform Plus prints the actual number of characters in the value of the *target-item* or *by-item*. In either case, Enform Plus prints the value of the *target-item* or *by-item* left-justified with blank fill.

If *w* is too small for the value of the *target-item* or *by-item*, an overflow condition occurs and Enform Plus truncates the value.

The following examples show the affect of the A edit descriptor:

Format	Item Value	Printed Item
-----	-----	-----
A	WORD	WORD
A4	WORD	WORD
A3	WORD	WOR
" [LJ] A3 "	WORD	WOR
" [RJ] A3 "	WORD	ORD

Integer Edit Descriptor

The integer edit descriptor specifies an integer display format. The syntax is:

Iw[.m]

w

is an unsigned integer that specifies the width of the output.

m

is an unsigned integer that specifies the number of digits that must be printed. The value of *m* must not exceed the value of *w*.

When you modify a *target-item* or *by-item* with an integer edit descriptor, Enform Plus prints the value right-justified with blank fill. Enform Plus always prints at least one digit for the value of such a *target-item* or *by-item*. If you specify *m*, Enform Plus prints the number of digits specified, using leading zeros if necessary.

Note that if the value to be printed is zero and you specify `Iw . 0`, Enform Plus prints blanks.

The following examples show the effect of the integer edit descriptor:

Format	Item Value	Printed Item
-----	-----	-----
I7	100	100
I7.2	-1	-01
I7.6	100	000100
I7.6	-1	-000001

Fixed Format Edit Descriptor

The fixed format edit descriptor specifies a fixed-point display format. The syntax is:

`Fw.d[.m]`

w

is an unsigned integer that defines the total width of the *target-item* or *by-item* value.

d

is an unsigned integer that defines the number of digits that are to appear to the right of the decimal point.

m

is an unsigned integer that defines the number of digits that are to appear to the left of the decimal point.

When you specify the fixed format edit descriptor, Enform Plus prints the value of the *target-item* or *by-item* right-justified with leading blanks if necessary. If the value is negative, Enform Plus prints a minus sign before the first digit. Both the minus sign and the decimal point occupy one position of the format; therefore, *w* must be wide enough to accommodate the total size of the output field including the minus sign and the decimal point. If *w* is not wide enough, an overflow condition occurs.

If you specify *m*, Enform Plus prints that number of digits, using leading zeros if necessary.

The following examples show the effect of the fixed format edit descriptor:

Format	Item Value	Printed Item
-----	-----	-----
F10.4	123.4567	123.4567
F10.4	0.000123	0.0001
F10.4.3	-4.56789	-004.5679
"[FL'*.'] F10.2"	123.4567	****123.46

Mask Edit Descriptor

The mask edit descriptor specifies a display format according to a template. The syntax is as follows:

M mask

mask

is a set of symbols or characters enclosed within quotation marks (“ ”), apostrophes (‘ ’), or angle brackets (< >). The symbols in a mask that serve a special function are:

- Z** is a digit selector. Z specifies that if no digit exists, zeros are suppressed. Z can be used with alphanumeric or numeric data. Note: a capital Z must be used for the special symbol.
- 9** is a digit selector. If no digit exists, zeros are printed; used for numeric data only.
- V** indicates decimal alignment for the display format. When V is specified, the decimal point is not printed. Note: a capital V must be used for the special symbol.
- .** indicates decimal alignment for the display format. When the symbol . (period) is used, the decimal point is printed.

Numeric Values

If the mask specified for a numeric value is too small, an overflow condition occurs.

The special symbols Z and 9 describe numeric values. If the digit selector is a 9, Enform Plus prints the corresponding data digit. If the digit selector is a Z, Enform Plus prints the corresponding data digit unless it is a leading or trailing zero. In this case, Enform Plus blank fills the position held by the zero.

Decimal point location can be indicated for numeric values by using either the symbol . or the symbol V. When the symbol . is used, a decimal point prints. When the symbol V is used, it indicates only decimal alignment and no decimal point prints. If neither the symbol . nor the symbol V is specified, Enform Plus places the decimal point as the rightmost character of the entire mask.

Alphabetic Values

Alphabetic values must be specified by the symbol Z in a mask edit descriptor.

Display Format and Scale

The output mask used to display a value has no effect on the scale of the value. For example, if the value of the user variable `cost` is an integer, and the variable appears in the query as:

```
LIST (cost) AS M<99.99>, ...
```


zeros are displayed to the right of the decimal point (the mask does not change the integer format). If `cost` were declared as `INTERNAL F4.2`, and appeared in the query as:

```
LIST (cost) AS M<99>, ...
```

the value of `cost` would be displayed without the digits to the right of the decimal point.

More information on scale appears under [Aggregates](#) on page 3-11 and [Arithmetic Expressions](#) on page 3-21.

The following examples show the affect of the mask edit descriptor:

Format	Item Value	Printed Item
-----	-----	-----
M"99/99/99"	103179	10/31/79
M'Z,ZZ9.99'	32.009	32.01
M<Z,ZZZ>	666	666
M<9,999>	666	0,666
M<9,999>	66666	*****
M<\$ZZZ,ZZ9.99>	92000.00	\$ 92,000.00

A query with these AS clauses:

```
amount AS M<$ZZ,ZZ9>,
date AS M<Z9/Z9/99>,
district AS A8,
telephone AS M<(999) 999-9999>
```

for these *target-item* values:

```
Amount      := 9758  21573  15532
Date        := 031777 091779 090579
District    := West  Midwest  South
Telephone   := 2135296800 2162296270 4047298400
```

produces this report:

```
$ 9,758    3/17/77    West      (213) 529-6800
$21,573    9/17/77    Midwest   (216) 229-6270
$15,532    9/ 5/79    South     (404) 729-8400
```

Nonrepeatable Edit Descriptors

Nonrepeatable edit descriptors indicate some ways *target-items* or *by-items* are to be printed. Values described by the nonrepeatable edit descriptors do not require data conversion by the Formatter, but the Formatter does process these values. When nonrepeatable edit descriptors are specified with modifiers or decorations, both the nonrepeatable edit descriptor and the repeatable edit descriptor must be enclosed in parentheses. The types of nonrepeatable edit descriptors are:

- Scale factor edit descriptor
- Optional plus sign edit descriptor

Scale Factor Edit Descriptor

The scale factor edit descriptor specifies a scale of 10^{**n} for a fixed-point (F) number. The value printed equals the internally represented number multiplied by 10^{**n} . The syntax is:

nP

n

is the exponent for the scale factor (10^{**n}).

P

is the implied decimal point of the number.

The following examples show the effect of the scale factor edit descriptor:

Format	Item Value	Printed Item
-----	-----	-----
"2P F10.2"	100.00	10000.00
"-2P F10.2"	100.00	1.00

Optional Plus Sign Edit Descriptor

The optional plus sign edit descriptors are used to control the printing of a plus (+) sign. The syntax is:

$\left\{ \begin{array}{l} S \\ SS \\ SP \end{array} \right\}$

S or SS

specifies that no plus sign is to be printed.

SP

specifies that a plus sign is to be printed.

When an *object-item* or *by-item* with a positive value is printed, Enform Plus does not normally precede the value with a plus (+) sign. Specify SP when you want Enform Plus to precede a value with an optional plus sign.

The following examples show the effect of the optional plus edit descriptor:

Format	Item Value	Printed Item
-----	-----	-----
"SP F10.2"	123.00	+123.00
F10.2	123.00	123.00
"SP F10.2"	-123.00	-123.00
"SP F10.2"	000.00	.00

Modifiers

Modifiers alter the normal effect of edit descriptors. A modifier must immediately precede the edit descriptor it modifies. Modifiers are enclosed within brackets ([]) and separated by commas.

[Table 5-2](#) indicates which modifiers can be used with which edit descriptors. An X indicates the combination is permitted.

Table 5-2. Permissible Modifiers and Edit Descriptors

Modifiers	Edit Descriptors			
	A	F	I	M
BZ, BN	X	X	X	X
LJ, RJ	X			
OC		X	X	X
FL	X	X	X	X
SS		X		X

Field Blanking Modifiers

Field blanking modifiers indicate under what circumstances to print blanks. The syntax is:

```
{ BN }
{ BZ }
```

BN

prints a blank field if the value is null.

BZ

prints a blank field if the value is zero.

Although most edit descriptors cause a minimum number of characters to be printed regardless of the value of the field, a field blanking modifier causes the entire field to be filled with blanks if the specified condition is met.

The following examples show the effect of the field blanking modifiers:

Format	Item Value	Printed Item
-----	-----	-----
"[BZ] F10.2"	.00	
"[BN] F10.2"	null value	
"[BN] F10.2"	100.00	100.00

Fill Character Modifier

The fill character modifier specifies the fill character that is used in the current display format. The default fill character is a blank. The syntax is:

`FL char`

char

is a single ASCII character enclosed within apostrophes (' ').

A fill character prints in each appropriate character position when one of the following occurs: alphanumeric data contains fewer characters than the field specified by the alphanumeric edit descriptor, leading zero suppression is performed, or embedded text in an mask edit descriptor is not printed because its neighboring digits are not printed.

The following examples show the affect of the fill character modifier:

Format	Item Value	Printed Item
-----	-----	-----
"[FL'.'] A10"	THEN	THEN.....
"[RJ,FL'>'] A10"	HERE	>>>>>HERE
"[FL'*'] M<\$ZZ,ZZ9.99>"	127.39	\$***127.39

Overflow Character Modifier

The overflow character modifier temporarily overrides the global default overflow indicator for the current display format. The overflow indicator is printed when a value exceeds the width specified in the display format. The syntax is:

`OC char`

char

is a single ASCII character enclosed within apostrophes (' ').

The overflow condition occurs if there are more characters to be printed than the display format specifies. When the overflow condition occurs, Enform Plus prints the overflow character. The default overflow character is an asterisk (*). The overflow character modifier allows you to temporarily substitute another ASCII character for the asterisk. The overflow character modifier applies only to the display format where it is specified. The overflow modifier does not apply to the alphanumeric (A) edit descriptor.

If you want to change the default overflow character for all display formats in the current session, use the @OVERFLOW option variable described under [Option Variable Clauses](#) on page 5-48.

The following examples show the effect of the overflow character modifier:

Format	Item Value	Printed Item
-----	-----	-----
"[OC'!'] I2"	100	!!
"[OC'!'] F5.2"	100000.00	!!!!

Justification Modifiers

The justification modifiers specify right or left justification for the values of an alphanumeric *target-item* or *by-item*. The syntax is:

```
{ LJ }
{ RJ }
```

LJ

specifies that an item is to be printed justified to the left of the specified display format width.

RJ

specifies that an item is to be printed justified to the right of the specified display format width.

The justification modifiers apply only to alphanumeric (A) edit descriptors. Alphanumeric *target-items* or *by-items* are normally left-justified. Padding of the item width automatically takes place. If the item value is wider than the width specified, truncation occurs.

The following examples show the effect of the justification modifiers:

Format	Item Value	Printed Item
-----	-----	-----
"[RJ] A12"	HELLO	HELLO
A12	HELLO	HELLO
"[RJ] A2"	HELLO	LO

Symbol Substitution Modifier

The symbol substitution modifier allows you to change the standard symbols ("9," "V," "Z," ".", and ",") used in edit descriptors. The syntax is:

```
SS 'pair-of-symbols'
```

pair-of-symbols

are the standard and substitute symbols enclosed within apostrophes (' '). The standard symbol must appear first, followed by the substitute symbol.

The symbol substitution modifier allows you to replace the standard symbols used in edit descriptors. You can change the symbols for 9, Z, V, ., and , in the mask (M) edit descriptor or change the symbol for decimal point (.) in the fixed-point (F) edit descriptor.

The symbol substitution modifier is useful when you need to specify a character string, that is also a standard mask symbol. An example would be 9 in a mask for a date. See the last of the following examples.

The following examples show the effect of the symbol substitution modifier:

Format	Item Value	Printed Item
-----	-----	-----
"[SS'.:'] F6.2"	12.45	12:45
"[SS'.:'] M<ZZZ.99>"	12.45	12:45
"[SS',,'] F10.2"	12345.67	12345,67
"[SS'9X'] M<XX/XX/XXXX>"	10311979	10/31/1999

Decorations

Decorations specify character strings that can be printed along with the value of a *target-item* or *by-item*. Decorations also specify the conditions under which the character string is added, the location at which the character string is added, and whether the character string is added before normal formatting is done or after it is completed. The syntax of a decoration is:

condition location char-string

condition

is one or more of the following:

- M Add *char-string* if value is negative.
- N Add *char-string* if value is null.
- P Add *char-string* if value is positive.
- Z Add *char-string* if value is zero.
- O Add *char-string* if overflow condition occurs.

location

is one of the following:

An Absolute position *n*

F Floating

P Prior

char-string

is one or more alphanumeric characters enclosed within apostrophes (‘ ’).

The following rules apply to decorations:

- Separate multiple decorations by commas.
- Enclose decorations in brackets along with any modifiers.
- Specify decorations immediately before the edit descriptor they modify.
- Enclose decorations and the edit descriptors they modify in quotation marks (“ ”).

Enform Plus evaluates decorations from left to right.

You are responsible for ensuring that the display format width is large enough to contain both the *target-item* or *by-item* value and the inserted character string.

Conditions

The condition specifiers (negative, positive, zero, null, or overflow) indicate that a character string is printed only when the specified condition occurs. A null condition takes precedence over negative, positive, and zero conditions. The overflow condition test is done after the other conditions are tested. Conditions specified for alphanumeric *target-items* or *by-items* can be positive or null only.

More than one condition specifier can exist for a decoration. Conditions are coded without separators.

Location

The location specifier indicates where the character string prints in relation to the value of the *target-item* or *by-item*. The uses of the location specifiers are as follows:

- The *A* location specifier indicates the character string is to be printed starting in the absolute position *n*.
- The *F* specifier indicates the character string is to occupy the position or positions immediately to the left (for right-justified values) of the leftmost data character. If the value is left-justified, the *F* specifier indicates the character string occupies the position or positions immediately to the right of the rightmost data character.
- The *P* location specifier indicates that prior to normal formatting, the string is to be inserted in either the rightmost position (for right-justified values) or the leftmost

position for left-justified values. The data values are shifted an appropriate number of positions.

Processing Order

Enform Plus processes decorations in the following order:

1. The data is tested to determine if it has a null value.
2. The data is tested to determine if it has a positive, negative, or zero value.
3. If the P location decoration is specified, the character string is added to the item value.
4. Normal formatting according to the edit descriptor (alphanumeric (A), integer (I), or fixed point (F)) is performed.
5. Decorations for alphanumeric and fixed point edit descriptors are performed.
6. The overflow condition is tested.

Default Decorations

When decorations are not specified, Enform Plus prints a negative value with a preceding negative (–) sign; that is, [MF' – '] is used by default.

If a decoration is specified that tests for a positive value, such as [PF' + '], the default [MF' – '] no longer automatically applies. In this case, the negative condition must be explicitly indicated if you want Enform Plus to print the negative sign.

When an overflow condition occurs, Enform Plus replaces the value by enough asterisks (*) to fill the field; that is, [OA1' ***** . . . ****'] is used by default. The OC *char* modifier temporarily overrides the default asterisk overflow character.

Possible decorations and their meanings are:

MA <i>n char-string</i>	if value is negative, print <i>char-string</i> in position <i>n</i> .
MF <i>char-string</i>	if value is negative, print <i>char-string</i> immediately to the left of right-justified value, immediately to the right of left-justified value.
MP <i>char-string</i>	if value is negative, print <i>char-string</i> immediately to the right of value.
NA <i>n char-string</i>	if value is null, print <i>char-string</i> in position <i>n</i> .
NF <i>char-string</i>	if value is null, print <i>char-string</i> immediately to the left of right-justified value; immediately to the right of left-justified value.
NP <i>char-string</i>	if value is null, print <i>char-string</i> immediately to the right of value.
PA <i>n char-string</i>	if value is positive, print <i>char-string</i> in position <i>n</i> .
PF <i>char-string</i>	if value is positive, print <i>char-string</i> immediately to the left of value.

PP <i>char-string</i>	if value is positive, print <i>char-string</i> immediately to the right of value.
ZAn <i>char-string</i>	if value is zero, print <i>char-string</i> in position <i>n</i> .
ZF <i>char-string</i>	if value is zero, print <i>char-string</i> immediately to the left of right-justified value, immediately to the right of left-justified value.
ZP <i>char-string</i>	if value is zero, print <i>char-string</i> immediately to the right of value.
OAn <i>char-string</i>	if overflow condition occurs, print <i>char-string</i> in position <i>n</i> .

The following are examples of decorations:

Format	Item Value	Printed Item
-----	-----	-----
"[MF'<',MP'>',ZPP'b'] F12.2"	1000.00	1,000.00
"[MF'<',MP'>',ZPP'b'] F12.2"	-1000.00	<1,000.00>
"[MA1'CR',MPF'\$'] F12.2"	1000.00	\$1,000.00
"[MA1'CR',MPF'\$'] F12.2"	-100.00	CR \$100.00
"[MA1'CR',MPF'\$'] F12.2"	0.00	0.00
"[OA1'**overflow**'] F12.2"	1000000.00	1000000000
"[OA1'**overflow**'] F12.2"	1000000000.00	**overflow**
"[ZPA2'+'] I8"	-10	10
"[ZPA2'+'] I8"	100	+ 100
"[ZPA2'+'] I8"	0	+ 0

AS DATE Clause

The AS DATE clause allows you to specify the display format for printing a date. The syntax of the AS DATE clause is:

```
date-in-internal-format AS DATE { *
                                { display-format } }
```

date-in-internal-format

is the name of a variable or field that contains a date in internal format. The option variable @DATE can be specified to give the current date in internal format.

*

specifies the default display format. The default display format specifies the date in the form *month/day/year*. It is comparable to the display-format "M2/D2/Y2".

display-format

is the format for printing a date. Display-format must be specified within quotation marks (" "). The *display-format* can include date keywords, and other characters such as blanks, commas, hyphens, or slashes. The date keywords are:

- M Specifies a month.
- D Specifies a day.
- Y Specifies a year.
- A Abbreviates or completely spells out the month or day. If *n* is specified with the keyword A, only *n* letters are displayed.
- B Suppresses leading zeros.
- O Abbreviates or completely spells out the number corresponding to the day. If *n* is specified with the keyword O, only *n* letters are displayed.
- n* Is an integer that specifies the number of characters (1-3) or numbers (2-4) to be printed.

When you want a date printed, the date must be in internal format. For instructions on how to convert a date to internal format, see the ["JULIAN-DATE Conversion Clause"](#) on page 5-43 or the ["TIMESTAMP-DATE Clause"](#) on page 5-67. The date can be a *target-item* within a LIST statement or any element that can be modified by an AS DATE clause within a print list.

Default Display Format

The Enform Plus default date *display-format*, "M2/D2/Y4", might handle most of the date formatting required. Change the default format by redefining the @DATE-FORMAT option variable described under [Option Variable Clauses](#) on page 5-48.

Examples of Date Display Formats

These date keywords produce the following:

MA	JANUARY, FEBRUARY, . . . , DECEMBER
MA3	JAN, FEB, . . . , DEC
M2	01, 02, . . . , 12
M	1, 2, . . . , 12
MB2	1, 2, . . . , 12
DA	MONDAY, TUESDAY, . . . , SUNDAY
DA3	MON, TUE, . . . , SUN
D2	01, 02, . . . , 31
DB2	1, 2, . . . , 31
D3	001, 002, . . . , 366
DB3	1, 2, . . . , 366
DOB2	1ST, 2ND, . . . 31ST
DAO	FIRST, SECOND, . . . , THIRTY-FIRST
Y2	00, 01, . . . , 96, 97, 98 . . .
YB2	0, 1, . . . , 96, 97, 98 . . .
Y4	1900, 1901, . . . , 1998, 1999, 2000

AS TIME Clause

The AS TIME clause allows you to specify the display format for printing a time. The syntax of the AS TIME clause is:

```
time-in-internal-format AS TIME { *
                                { display-format } }
```

time-in-internal-format

is the name of a variable or field that contains a time in internal format.

*

specifies the default display format, which specifies the hour, minute, and second as HP2:M2:S2 (see the following description of *display-format*).

display-format

is the format for printing the time. The *display-format* must be specified within quotation marks (“ ”). The *display-format* can be specified by using time keywords and other characters such as blanks, commas, hyphens, or slashes. In addition, alphanumeric characters enclosed in apostrophes (‘ ’) can be embedded within *display-format*. The time keywords are:

- H Specifies an hour.
- M Specifies a minute.
- S Specifies a second.
- P Expresses the hour as modulo 12 with AM or PM.
- B Suppresses leading zeros.
- n* Is an integer digit specifying the number of digits (1 or 2) to be printed.

If you want Enform Plus to print the time, it must be in internal format. For information on how to convert a time to internal format, see the [TIMESTAMP-TIME Clause](#) on page 5-68. The time can be a *target-item* within a LIST statement or any element that can be modified by an AS TIME clause within a print list.

Default Time Display Format

The Enform Plus default time format, HP2:M2:S2, can handle most of the time formatting required. Change the default format by redefining the @TIME-FORMAT option variable described under [Option Variable Clauses](#) on page 5-48.

Examples of the Time Display Format

These time keywords produce the following:

HB2	1, 2, , 24
HP2	01, 02, , 12 AM or PM
HPB2	1, 2, , 12 AM or PM
M2	00, 01, , 59
MB2	0, 1, , 59
S2	00, 01, , 59
SB2	0, 1, , 59

AT END PRINT Clause

The AT END PRINT clause prints information at the end of the current report. This clause is an optional part of the LIST statement. The syntax of the AT END PRINT clause is:

```
AT END PRINT print-list [ CENTER ]
```

print-list

can contain any combination of literals, FORM, SKIP, SPACE, or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, or parameter names.

The clauses that can be part of a *print-list* are described in this section. The other elements are described in [Section 3, Enform Plus Language Elements](#)

Specifying a Field Name Within an AT END PRINT Clause

If you specify a field name within the *print-list* of an AT END PRINT clause, Enform Plus prints the same value as in the last row of the report. A field name appearing within the *print-list* of an AT END PRINT clause need not be explicitly included within the associated LIST statement. If the field name is not included, Enform Plus in effect adds the field to the LIST statement with a NOPRINT clause.

Spacing Considerations

By default, the information in the *print-list* of an AT END PRINT clause begins printing in the same position as the leftmost column of the report. Using the SPACE or TAB clause as the first element of the *print-list* overrides the default. SPACE or TAB clauses can be used anywhere within the *print-list*. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
AT END PRINT "Report" SPACE 15 "Total Sales",
```

```
Report                Total Sales
```

If you specify a SKIP clause or the slash symbol (/) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the AT END *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause in the *print-list* causes Enform Plus to print two lines:

```
AT END PRINT "End of Report for" SKIP "Region " regnum,
```

```
End of Report for
Region 1
```

By using the FORM clause within the *print-list*, you can cause Enform Plus to start a new page, increment the page number, and continue with the rest of the *print-list*.

By using the CENTER clause following the *print-list*, you can center the information on the page.

The CENTER, Option Variable, SKIP, SPACE, and TAB clauses are described in this section.

AT END Information for Current Report or All Reports

The optional AT END PRINT clause prints information only for the current report. This clause temporarily overrides the session-wide AT END statement. If you want to print information at the end of all subsequent reports in the current session, use the AT END statement.

Overriding Session-Wide AT END Information

Temporarily override session-wide AT END information by specifying the AT END PRINT clause with “ ” for the *print-list* parameter.

AT START PRINT Clause

The AT START PRINT clause allows you to specify information to be printed just before the first set of column headings for the current report. This clause is an optional part of the LIST statement. The syntax of the AT START PRINT clause is:

```
AT START PRINT print-list [ CENTER ]
```

print-list

can contain any combination of literals, FORM, SKIP, SPACE, or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, or parameter names.

The clauses that can be part of a *print-list* are described in this section. The other elements are described in [Section 3, Enform Plus Language Elements](#).

The AT START PRINT clause differs from both the TITLE and SUBTITLE statements and clauses in that a title or subtitle is printed on every page of a report while the AT START PRINT information is printed only on the first page of a report. The information supplied in an AT START PRINT clause prints after either a title or a subtitle.

Specifying a Field Name in an AT START PRINT Clause

If you specify a field name within a *print-list* of an AT START PRINT clause, Enform Plus prints the same field value as in the first row of the report. A field name appearing within the print list of a AT START PRINT clause need not be explicitly included within the associated LIST statement. If the field name is not included, Enform Plus in effect adds it with a NOPRINT clause.

Spacing Considerations

By default the AT START PRINT information begins printing in the same column position as the leftmost report column. Using SPACE or TAB clauses as the first element of the *print-list* overrides this default. SPACE or TAB clauses can also appear anywhere within the *print-list*. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
AT START PRINT "Report" SPACE 15 "Total Sales",
```

```
Report                Total Sales
```

If you specify either a SKIP clause or a slash (/) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any)

following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause causes two lines to be printed:

```
AT START PRINT "End of Report for" SKIP "Region " regnum,  
End of Report for  
Region 1
```

By using the FORM clause within a *print-list*, you can cause Enform Plus to start a new page, increment the page number, and continue with the rest of the *print-list*.

By using the CENTER clause following the *print-list* of the AT START PRINT clause, you can center the information on the page.

The CENTER, Option Variables, SKIP, SPACE, and TAB clauses are described in this section.

AT START Information for Current Report or All Reports

An AT START PRINT clause prints information only for the report generated by the associated LIST statement. It temporarily overrides the session-wide AT START statement. If you want to print the same information just before the first set of column headings for all subsequent reports in the current session, use the AT START statement.

Overriding Session-Wide AT START Information

Temporarily override the session-wide AT START statement by using the AT START PRINT clause with “ ” for the *print-list* parameter.

BEFORE CHANGE Clause

The BEFORE CHANGE clause allows you to specify information to be printed following the records for each group for the current report. The BEFORE CHANGE clause is an optional part of the LIST statement. The syntax of the BEFORE CHANGE clause is:

```
BEFORE CHANGE [ ON ] by-item PRINT print-list [ CENTER ]
```

by-item

is the name of a field grouped by a BY or BY DESC clause.

print-list

can contain any combination of literals, FORM, SKIP, SPACE, or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, or parameter names.

The clauses that can be part of a *print-list* are described in this section. The other elements are described in [Section 3, Enform Plus Language Elements](#)

The keyword BEFORE in the BEFORE CHANGE clause refers to the values printed, not to the location of the printed information. For the exact location of where the BEFORE CHANGE information is printed within a report, see the *Enform User's Guide*.

When more than one BEFORE CHANGE clause is specified, Enform Plus prints the BEFORE CHANGE information in the order in which the BEFORE CHANGE clauses are entered.

Specifying a Field Name Within a BEFORE CHANGE Clause

If you specify a field name within a *print-list* of a BEFORE CHANGE clause, Enform Plus prints the same field value in the last row of the previous group; that is, before the value changes. A field name appearing within the *print-list* of a BEFORE CHANGE clause need not be explicitly included within the LIST statement. If the field name is not included, Enform Plus in effect adds it with a NOPRINT clause.

Spacing Considerations

By default, the BEFORE CHANGE clause information begins printing in the same column position as the leftmost report column. Using SPACE or TAB clauses as the first element of the print list overrides this default. SPACE or TAB clauses can also appear anywhere within the *print-list*. In the following example, the SPACE clause causes the two literals to be separated by five spaces:

```
BEFORE CHANGE ON ordernum  
PRINT "*****" SPACE 5 "Orders for " ordernum,  
  
*****      Orders for 122
```

If you specify either a SKIP clause or the symbol / (slash) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the BEFORE CHANGE *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause causes two lines to be printed:

```
BEFORE CHANGE ON regnum PRINT "Begin of Report for"  
    SKIP "Region " regnum,  
  
Begin of Report for  
Region 1
```

By using the CENTER clause following the *print-list* of the BEFORE CHANGE clause, you can center the information on the page.

The CENTER, Option Variables, SKIP, SPACE, and TAB clauses are described in this section.

BY and BY DESC Clauses

The BY and BY DESC clauses group and sort target-records according to the value of a specified field. The syntax of the BY and BY DESC clauses is:

```
{ BY      } by-item
{ BY DESC }
```

by-item

is the name of a field from an input record whose values are to be used to group and sort *target-records*.

BY and BY DESC clauses group and sort records according to the field values. The BY clause sorts the records in ascending order according to the value of the specified field; the BY DESC clause sorts in descending order. The following example groups and sorts the *partnum* field:

BY *partnum*,

When more than one sort is specified, Enform Plus uses a major-to-minor sorting precedence. Enform Plus determines the sorting precedence by the order in which a BY or BY DESC clause appears in a LIST or FIND statement. The first BY or BY DESC clause has the highest priority, the next one second priority, down to the last BY or BY DESC clause.

When a BY or BY DESC clause is used with a LIST statement, Enform Plus prints only the first instance of a grouped value in a report. When a BY or BY DESC clause is used with a FIND statement, Enform Plus writes all of the values of a grouped value to the physical output file or transmits all of the values to the host language program.

Several clauses require that a field be grouped by a preceding BY or BY DESC clause. See the CUM, PCT, SUBTOTAL, AFTER CHANGE and BEFORE CHANGE clauses in this section and Target Aggregates in [Section 3, Enform Plus Language Elements](#).

CENTER Clause

The CENTER clause centers an object within its context. The syntax of the CENTER clause is:

```
{ { target-item } CENTER , }
  { by-item }
  { CENTER ALL, }
```

target-item

is a record name, a field name, a string literal, a predefined aggregate, a user aggregate, an arithmetic expression, an IF/THEN/ELSE expression, a user variable, or a System Variable clause.

by-item

is the name of a field grouped by a BY or BY DESC clause.

Centering Single Report Items

The CENTER clause causes a *target-item* or *by-item* to be centered under its column heading. Numeric data is normally right-justified; alphabetic data is normally left-justified. *Target-items* and *by-items* are centered based on the width for which they are formatted, not their actual values. If the space divides unevenly, Enform Plus places the extra space to the right of the item.

Centering All Report Items

The CENTER ALL clause causes all elements in a LIST statement to be centered under their headings. It must follow the WHERE clause in the LIST statement. See the syntax of the LIST statement under [LIST Statement](#) on page 4-30 for the relative locations of the clauses within a LIST statement.

Centering a Print List

The CENTER clause following the print list of an AT END statement or clause, an AT START statement or clause, an AFTER CHANGE clause, a BEFORE CHANGE clause, A FOOTING statement or clause, a SUBTITLE statement or clause, a SUBFOOTING statement or clause, or a TITLE statement or clause, centers the *print-list* information on the page.

CUM Clause

The CUM clause allows you to specify printing of a running total for a numeric *target-item* either for all the instances of the *target-item* or for the instances of the *target-item* grouped within the each value of a *by-item*. The syntax of the CUM clause is:

```
target-item CUM [ OVER ALL      ]
                  [ OVER by-item ]
```

target-item

is a record name, a field name, a numeric literal, a predefined aggregate, a user aggregate, an arithmetic expression, an IF/THEN/ELSE expression, a user variable, or a System Variable clause. The data type of a *target-item* used in a CUM clause must be numeric.

by-item

is the name of a field grouped by a BY or BY DESC clause.

CUM With OVER ALL

When you specify the CUM OVER ALL clause, Enform Plus prints a running total in place of each value of the numeric *target-item*. When you specify only CUM, Enform Plus assumes CUM OVER ALL.

CUM With OVER

When you specify the CUM OVER *by-item* clause, Enform Plus prints a running total for the instances of the numeric *target-item* within the *by-item* in place of the value of the numeric *target-item*. The *by-item* must be previously defined by a BY or BY DESC clause. The running total begins anew each time the *by-item* value changes. The following example prints the running total of all parts for each location:

```
LIST BY location,
      partnum,
      inventory,
      inventory CUM OVER location;
```

LOCATION	Part Number	INVENTORY	CUM INVENTORY
-----	-----	-----	-----
L98	5103	8	8
	5502	6	14
V66	6603	40	40
...

CUM Clause Used With User Variable

When a numeric *target-item* with a CUM clause is assigned to a user variable, Enform Plus assigns the value to the user variable first, before the running total is calculated. When the user variable is referenced as a *target-item* element in a LIST statement or as an element within a LIST *target-item*, Enform Plus uses the value of the user variable. When the user variable is referenced in a print list, Enform Plus uses the value of the running total.

Restrictions

You cannot combine the CUM clause with the PCT clause, the TOTAL clause, or the SUBTOTAL clause.

FOOTING Clause

The FOOTING clause allows you to specify information for printing at the bottom of each page for the current report. This clause is an optional part of the LIST statement. The syntax of the FOOTING clause is:

```
FOOTING  print-list [ CENTER ]
```

print-list

can be any combination of literals, FORM, SKIP, SPACE, or TAB clauses. A *print-list* can contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variable names, or parameter names.

The clauses that can be part of a *print-list* are described in this section. The other elements are described in [Section 3, Enform Plus Language Elements](#)

Specifying a Field Name Within a FOOTING Clause

If you specify a field name within a *print-list* of a FOOTING clause, Enform Plus prints the same field value as in the last row of data on the current report page. A field name appearing within the FOOTING clause need not be explicitly included within the LIST statement. If the field name is not included, Enform Plus effectively adds it to the LIST statement with a NOPRINT clause.

Spacing Considerations

By default the footing begins printing in the same column position as the leftmost report column. By using SPACE or TAB clauses as the first element of the *print-list*, you can override this default. SPACE or TAB clauses can also appear anywhere within the *print-list*. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
FOOTING "Inventory" SPACE 15 "Parts in Stock",
```

The following footing is printed on the next report:

```
Inventory           Parts in Stock
```

If you specify either a SKIP clause or the slash symbol (/) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the FOOTING *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause causes two lines to be printed:

```
FOOTING "Report 2-A" SKIP "Total Sales",
```


The following footing is printed on the next report:

```
Report 2-A  
Total Sales
```

To force a new page, use a `FORM` clause within the `FOOTING` statement. Printing continues with the remainder of the `FOOTING print-list`, starting at the top of the next physical page. The page number remains the same. A single logical page can span multiple physical pages, such that a `TITLE` can appear on one page, the data on the next, and a `FOOTING` on the next. The same page number applies to all physical pages in the logical page.

Using the `CENTER` clause centers the footing on the page.

The `CENTER`, `Option Variables`, `SKIP`, `SPACE`, and `TAB` clauses are described in this section.

Footing for Current Report or All Reports

A `FOOTING` clause within a `LIST` statement prints a footing only for the current report. A `FOOTING` clause temporarily overrides a session-wide `FOOTING` statement. A `FOOTING` statement prints a footing at the bottom of each page for all subsequent reports in the current session.

FORM Clause

The FORM clause allows you to control when to skip to a new page. The syntax of the FORM clause is:

FORM [<i>number</i>]

number

is an unsigned integer.

FORM Clause With a By-Item

When the FORM clause follows a field name that has been grouped with a BY or BY DESC clause, Enform Plus starts a new page whenever the field value changes. If *number* is specified, Enform Plus starts a new page only if there are fewer than that number of lines remaining on the current page. The following example shows the use of the FORM clause with a *by-item*:

BY regnum FORM,

FORM Clause With a Target-Item

When the FORM clause precedes a *target-item*, Enform Plus starts a new page each time the *target-item* is printed. When the FORM clause follows a *target-item*, Enform Plus prints the *target-item* before starting a new page. The *number* parameter is not allowed when the FORM clause modifies a *target-item*.

FORM Clause Within a Print List

The FORM clause can be part of a print list for an AFTER CHANGE, AT START PRINT, AT END PRINT, BEFORE CHANGE, FOOTING, SUBFOOTING, SUBTITLE, or TITLE clause or statement. Enform Plus starts printing on a new page every time the FORM clause is processed. When used with a print list, the FORM clause does not use the *number* parameter.

HEADING Clause

The HEADING clause allows you to override the default column title for a *target-item* or *by-item* in a report. The HEADING clause also allows you to define a column title for *target-items*, such as arithmetic expressions, that do not have a default column title. The syntax of the HEADING clause is:

```
{ by-item
  { target-item } } HEADING "heading-string"
```

by-item

is the name of a field that has been grouped by a BY or BY DESC clause

target-item

is a record name, a field name, a numeric literal, a predefined aggregate, a user aggregate, an arithmetic expression, an IF/THEN/ELSE expression, a user variable, a System Variable clause, or a JULIAN-DATE clause.

heading-string

is a string literal. Remember string literals must be enclosed within quotation marks (“ ”).

Default Headings

When the HEADING clause is not specified explicitly, Enform Plus obtains a heading from one of these sources:

- The HEADING clause specified either in the dictionary or in a DECLARE statement
- The field name specified in the dictionary or name of the user variable, aggregate, or table specified in a DECLARE statement

If you want to print more than one *target-item* or *by-item* under the same column heading, use a NOHEAD clause to prevent Enform Plus from printing the unwanted heading. The NOHEAD clause is described later in this section on page [5-45](#).

Multiple-Line Headings

Multiple-line headings are created by using a slash (/) within the heading string.

Printing / in a Column Heading

Sometimes, the / character needs to appear within a column heading. In this case, use the SET statement, described under [SET Statement](#) on page 4-41, to change the @NEWLINE option variable from a / to a different character. After the new line character is redefined, the new special character can be used within the heading string instead of the / character. In the following example, the new line character is changed to

a number sign (#), causing Region and Number to print on two lines, and the symbol / inside the print list is printed between Branch and Number:

```
SET @NEWLINE TO "#";
LIST regnum HEADING "Region#Number",
branchnum HEADING "Branch/Number"/;
```

Region Number	Branch/Number
-----	-----
1	1
1	2

Notice that when the / symbol appears outside of the heading clause, it causes Enform Plus to advance one line before printing the next *target-item*. Changing the @NEWLINE character does not affect this use of the / symbol.

Heading for Subscripted Elements

When a single subscripted element is modified by a HEADING clause, Enform Plus prints the specified heading. For example:

```
LIST month [ 3 ], HEADING "MARCH";
```

causes Enform Plus to print:

```
MARCH
-----
...
```

When an element including a subscript range is modified by a HEADING clause, Enform Plus includes the subscript in the specified heading. For example:

```
LIST month [ 1:3 ], HEADING "FIRST QUARTER";
```

causes Enform Plus to print:

FIRST QUARTER	FIRST QUARTER	FIRST QUARTER
[1]	[2]	[3]
-----	-----	-----
...

If a HEADING clause is not included, Enform Plus includes the subscript with the default heading. For example:

```
LIST month [ 3 ];
```

causes Enform Plus to print:

```
MONTH
[ 3 ]
-----
...
```

Remember the default heading for a field name is either the heading declared in the dictionary or, if no heading is so declared, the field name. The default heading for a user variable is either the heading defined in the `DECLARE` statement, or if no heading is defined, the user variable name.

INTERNAL Clause

The INTERNAL clause allows you to specify the storage format for a user-defined element. The syntax of the INTERNAL clause is:

`INTERNAL internal-format`

internal-format

is the format for storing the user-defined element. *Internal-format* can be:

`An` Alphanumeric; *n* is the length

`In` Integer; *n* is the length

`Fw.d` Fixed; *w* is the number of digits and *d* is the number of decimal places

The optional INTERNAL clause can appear in a PARAM or DECLARE statement. When the INTERNAL clause is not used, Enform Plus stores user-defined elements (variables, aggregates, tables, and parameters) as 64-bit signed integers. The following INTERNAL clause specifies *internal-format* as fixed. The element has a total length of eight digits with two digits following the decimal point:

`INTERNAL F8.2`

"JULIAN-DATE Conversion Clause

The JULIAN-DATE Conversion clause allows you to specify translation of a date *target-item* into internal format. The syntax for the JULIAN-DATE Conversion clause is:

```
JULIAN-DATE ( year , month , day )
```

year

is the year in 4 digits.

month

is the month in 2 digits, in the range 1-12.

day

is the day in 2 digits, in the range 1-31.

Dates are a common part of database records. From an information standpoint, dates need to be printed on a report in a form recognizable as a date, such as 05/15/1982, Apr 1, 1999 or 06-01-2000. From an analytical standpoint, dates need to be stored in a form for use in calculations or expressions, such as 1999 11 27. Dates used in calculations or expressions must be in an internal format. If the date is not in this internal format, it can be converted by using the JULIAN-DATE Conversion clause. The internal format represents a date as the number of days that have elapsed from an arbitrary date in the past.

Conversion to Internal Format

To change a date to internal format, use the JULIAN-DATE Conversion clause. For example, suppose the data description entry of the day and year portion of a data base date is as follows:

```
05  date.
    10  yy          PIC "9999".
    10  mm          PIC "99".
    10  dd          PIC "99".
```

The month (1-12), day (1-31), and year (a 4-digit number) can be passed to the JULIAN-DATE Conversion clause as follows:

```
JULIAN-DATE ( yyyy , mm , dd )
```

Gregorian dates must be converted to an internal date format when used for purposes other than printing on a report. When the date is only to be printed, convert it with the JULIAN-DATE clause and then use an AS DATE Conversion clause. For example:

```
JULIAN-DATE ( yyyy , mm , dd ) AS DATE *
```

Alternatively, convert the date to an integer that can be formatted with an AS clause. For example:

```
date AS M<99-99-9999>
```

To convert a date stored as a yearly Julian date, where the day of the year is relative to January 1 of that year, define each part of the date:

```
05  yearly-julian-date.  
    10  day-of-year      PIC "999".  
    10  current-year     PIC "9999".
```

Then add day-of-year to the internal date for December 31 of the previous year:

```
(JULIAN-DATE ((current-year - 1)), 12, 31) + day-of-year)
```

Display Format

To print a date in internal format on a report, convert the date to a display format with an AS DATE clause. See the [AS DATE Clause](#) on page 5-22.

NOHEAD Clause

The NOHEAD clause allows you to specify suppression of the printing of the column heading of a *target-item* or *by-item*. The syntax of the NOHEAD clause is:

```
{ { target-item } NOHEAD }
  { by-item      }
{ NOHEAD ALL }
```

target-item

is a record name, a field name, a numeric literal, a predefined aggregate, a user aggregate, an arithmetic expression, an IF/THEN/ELSE expression, a user table name, a user variable, a System Variable clause, or a JULIAN-DATE clause.

by-item

is the name of a field grouped by a BY or BY DESC clause.

No Headings for Single Report Items

The NOHEAD clause following a LIST *target-item* suppresses the column heading for the *target-item*.

Sometimes it is undesirable to print a column heading for an item, such as when you want to print more than one *target-item* under the same column heading. In the following example, the address, city, and state are all printed in the same column:

```
LIST suppnun,
    address / TAB 10,
    city NOHEAD / TAB 10,
    state NOHEAD;
```

SUPPNUM	ADDRESS
-----	-----
1	19333 VALLCO PARKWAY CUPERTINO CALIFORNIA
2	2000 BAKER STREET IRVINE CALIFORNIA

No Headings for All Report Items

The NOHEAD ALL clause suppresses column headings for all the *target-items* specified in a LIST statement. It must follow the WHERE clause in the LIST statement. See the syntax of the LIST statement under [LIST Statement](#) on page 4-30 for the relative locations of the clauses within that statement.

NOPRINT Clause

The NOPRINT clause allows you to specify suppression of the printing of a *target-item* or *by-item* and its associated column heading. The syntax of the NOPRINT clause is:

```
{ { target-item } NOPRINT }
  { by-item      }
{ NOPRINT ALL }
```

target-item

is a record name, a field name, a numeric literal, a predefined aggregate, a user aggregate, an arithmetic expression, an IF/THEN/ELSE expression, a user table name, a user variable, a System Variable clause or a JULIAN-DATE clause.

by-item

is the name of a field grouped by a BY or BY DESC clause.

Suppress Single Report Items

The NOPRINT clause following a LIST *target-item* or *by-item* suppresses the printing of the *target-item* or *by-item* and its heading. The *target-item* or *by-item* can still be used for sorting, grouping, or calculations elsewhere in the report.

You can use the NOPRINT clause to suppress printing of a *target-item* or *by-item* within the body of a report. In the following example, the records are grouped by region, but the region number is not printed as a *by-item* in the report. Instead the region number is printed at the end of the group using a BEFORE CHANGE clause:

```
LIST BY regnum NOPRINT,
      branchnum,
      manager,
      BEFORE CHANGE ON regnum PRINT "Region " regnum;
```

BRANCHNUM	MANAGER
-----	-----
1	75
2	129

Region 1

Suppress All Report Items

The NOPRINT ALL clause suppresses the printing of all columns and column headings. It must follow the WHERE clause in the LIST statement. See the syntax of the LIST statement under [LIST Statement](#) on page 4-30 for the relative locations of the clauses

within that statement. Title and summary information, using AT END, AT START, FOOTING, SUBFOOTING, SUBTITLE, TITLE statements and clauses, and AFTER CHANGE and BEFORE CHANGE clauses still appear in the report. This strategy is useful for reports where only summary information is desired.

Option Variable Clauses

The Option Variable clauses allow you to redefine the default values for several operational variables. See the [SET Statement](#) on page 4-41. The Option Variables and their valid values are:

```

{ @BLANK-WHEN-ZERO }
{ @BREAK-KEY       }
{ @CENTER-PAGE     }
{ @HEADING         } TO BBC { ON   }
{ @STATS           }      { OFF  }
{ @SUMMARY-ONLY    }
{ @WARN            }

{ @COPIES          }
{ @COST-TOLERANCE  }
{ @DISPLAY-COUNT   }
{ @LINES           }
{ @MARGIN          }
{ @PAGES           }
{ @PRIMARY-EXTENT-SIZE } TO number
{ @SECONDARY-EXTENT-SIZE }
{ @READS           }
{ @SPACE           }
{ @TARGET-RECORDS  }
{ @VSPACE          }
{ @WIDTH           }

{ @DECIMAL         }
{ @NEWLINE         }
{ @NONPRINT-REPLACE } TO "character"
{ @OVERFLOW        }
{ @UNDERLINE       }

@SUBTOTAL-LABEL TO string-literal

{ @DATE-FORMAT } TO display-format
{ @TIME-FORMAT }
```

number

is an integer.

"*character*"

is a single ASCII character enclosed within quotation marks.

string-literal

is a string literal; remember that string literals must be enclosed in quotation marks ("").

display-format

specifies the default format for printing dates or times when AS DATE * and AS TIME * clauses are used; it must be enclosed in quotation marks.

@BLANK-WHEN-ZERO

When set ON, Enform Plus suppresses the printing of *target-item* or *by-item* values of zero in reports. When set OFF, zeros are printed. The default is OFF.

@BREAK-KEY

When set ON, pressing the Break key while a query is running either terminates output (if output is being produced) or terminates the current query and returns you to the Enform Plus prompt. See [Section 2, Running Enform Plus](#) for more information. When set OFF, pressing the Break key temporarily suspends output and returns you to the Command Interpreter prompt but it does not terminate processing. Output resumes when you enter the TACL PAUSE command. If the file specified in the IN option of the ENFORM command is not a terminal, Enform Plus ignores the Break key regardless of the setting of @BREAK-KEY.

Enform Plus does not store the setting of @BREAK-KEY in a compiled query.

The default is ON.

@CENTER-PAGE

When set ON, Enform Plus centers the report body on the page according to the smaller of:

- The listing device's maximum page width
- The value of the @WIDTH Option Variable

If @MARGIN was specified, that margin value is added to the report body before the report is centered on the page.

TITLE, SUBTITLE, AT START, AT END, FOOTING, and SUBFOOTING information is centered on the page only if a CENTER clause was included. Otherwise, this information appears at the left margin of the report body.

When not set or set OFF, the report is not centered. The default is OFF.

@COPIES

specifies how many copies to print. The default is 1.

@COST-TOLERANCE

can be set to a number from 1 through 8. Setting @COST-TOLERANCE prevents Enform Plus from executing a strategy that is more expensive than the user wants. When a larger number is specified, the query processor performs more work than when a smaller number is specified.

This condition does not mean an Enform Plus session that meets criteria for level 2 will always take less time to run than another Enform Plus session that requires level 7. The execution time also depends on the amount of data that must be processed.

When not set or set to 0 (zero), Enform Plus proceeds with whatever strategy it chooses.

If the specified strategy is exceeded, an error message is received and the Enform Plus program stops.

The desired cost tolerance level can be defined as follows:

- 0 Allows the Enform Plus session to proceed with no cost limit.
- 1 Keyed access is used on all files.
- 2 One file might require one full-file read.
- 3 More than one full-file read is required.
- 4 One file might be sorted once for a link.
- 5 Two files might be sorted once for a link.
- 6 One file might be sorted more than once for a link.
- 7 Two files might be sorted more than once for a link.
- 8 Two or more files can require full-file read at least once, with no convenient strategy for doing the link available.

@DATE-FORMAT

specifies a default format for printing dates when AS DATE * is used. For date format specifications, see the [AS DATE Clause](#) on page 5-22. The default is "M2/D2/Y4".

@DECIMAL

When set to a single ASCII character, Enform Plus prints that character for a decimal point. Enform Plus also accepts the specified character in place of a decimal point in a numeric literal. The default value is the period character (.).

@DISPLAY-COUNT

determines how many lines to display on output device at one time. When *number* lines have been displayed, Enform Plus pauses. To continue the display, press the Return key. Execution of the Enform Plus program, while at the pause, can be aborted by any of the following:

- Entering two slashes (/ /)
- Pressing the Ctrl and Y keys simultaneously
- Pressing the Break key

To reset the display count so that the entire Enform Plus program is printed, enter zero for *number*. Specifying @DISPLAY-COUNT when using an Enform Plus server (described in the *Enform Plus User's Guide*) is inadvisable. The default value is zero.

@HEADING

When set ON, Enform Plus prints column headings in the report. When set OFF, Enform Plus does not print headings. OFF is equivalent to NOHEAD ALL for every report. The default is ON.

@LINES

specifies how many lines are to be printed per page. The default value is 60.

@MARGIN

specifies the left margin size in columns for reports. The default value is 0 (zero).

@NEWLINE

The new line character can appear within a heading string, inside the quotation marks. It is used to specify multiple line column headings. The @NEWLINE Option Variable clause can be reset to any single ASCII character except circumflex (^) or hyphen (-). The default NEWLINE character is / (slash).

@NONPRINT-REPLACE

Enform Plus prints the replacement character for values that do not have a printable character. When this clause is set to a character, that character is printed for values that do not have a printable character. The default value is OFF.

@OVERFLOW

When a value does not fit within its formatted width, Enform Plus replaces the value with overflow characters. When this clause is set to a single ASCII character, Enform Plus prints that character when overflow occurs. The default overflow character is an asterisk (*).

If you set @OVERFLOW to ASCII blank, Enform Plus truncates the value in a field whenever an overflow occurs instead of replacing the value with blanks.

@PAGES

specifies a maximum number of pages to print per report. The default value is 32,767 pages.

@PRIMARY-EXTENT-SIZE

specifies the primary extent size of the output file. If necessary, the file will be created with an extent size larger than the one you specified.

When this clause is not set, Enform Plus calculates an estimate of the primary and secondary extent sizes needed for each output file. The estimate could be too large,

creating a file system error 43 unable to obtain disc space for file extent; or the estimate could be too small, producing more data than the file can hold. Each time an output file is full, Enform Plus must create a new output file, with a larger extent size, and copy the data from the full file into the new file. This strategy results in very inefficient processing.

@READS

specifies the maximum number of records to be returned from the database per Enform Plus program. This option is a useful tool when debugging. For an idea of how many reads are required by a given Enform Plus program, see the @STATS Option Variable clause under [Option Variable Clauses](#) on page 5-48. To remove a limit, set the @READS Option Variable to 0 (zero). The default value is 0, meaning an unlimited number of records can be returned.

@SECONDARY-EXTENT-SIZE

specifies the extent size of the output files. If necessary, the file will be created with an extent size larger than the one you specified.

When this clause is not set, Enform Plus uses the primary extent size.

@SPACE

specifies how many blanks appear between report columns. The default value is two blanks.

@STATS

When set ON, Enform Plus prints the statistics regarding the records after the Enform Plus program is completed. Enform Plus prints the following statistics:

FILENAME	Physical file name of a set of records.
LEVEL READ	Order in which this file was read in relation to all physical files read. The first file read is indicated by the number 1.
RECORDS READ	Total number of logical records read from this physical file.
POSITIONS	Total number of positioning operations performed to read this physical file.
STRATEGY COST	Number, 1 through 8, used to represent the cost of the strategy Enform Plus developed to handle the Enform Plus program. For a definition of the cost numbers, see the @COST-TOLERANCE option variable under Option Variable Clauses on page 5-48.

The default value is OFF.

@SUBTOTAL-LABEL

specifies a character string that is to appear on the same line as and to the left of the subtotal. Enform Plus prints the character string in the column of the group item over which the subtotal is computed. If the item is defined with the NOPRINT clause, Enform Plus does not print the character string. If the character string does

not fit in the column, Enform Plus truncates it on the right. The character string can consist of 1 to 15 ASCII characters, enclosed within quotation marks. The default value is an asterisk (*).

@SUMMARY-ONLY

causes a summary report to be produced. See the explanation of summary reports found with the LIST and FIND statements in [Section 4, Statements](#).

The default value is OFF.

@TARGET-RECORDS

specifies the maximum number of records to be selected per Enform Plus program. The default value is zero, meaning no limit. Setting @TARGET-RECORDS to *number* when your query contains BY, BY DESC, ASCD, or DESC clauses returns the first target-records produced by the query processor, which are not necessarily the first records that appear in a full report.

@TIME-FORMAT

specifies a default format enclosed within quotation marks for printing times when AS TIME * is used. For time format specifications, see the [AS TIME Clause](#) on page 5-24. The default value is "HP2:M2:S2".

@UNDERLINE

specifies a single ASCII character used to underline headings and totals. To specify no underlining, set the @UNDERLINE Option Variable clause to blank, "". The default character is "_".

@VSPACE

specifies how many lines are skipped before a report line when the SKIP clause is used. The default number is one.

@WARN

specifies when warning messages are to appear on the terminal. When set OFF, warning messages do not appear. The default is ON.

@WIDTH

specifies the maximum width of the output. The maximum value is 132 characters.

Verify the device's width before using this clause. This clause is convenient for setting a report page width smaller than the device's default width. The default width is the width of the output device being used. The default value for unstructured disk files is 132.

PCT Clause

The PCT clause prints the percentage of the grand total for a numeric *target-item*, based either on a total figure for all instances of the *target-item* or a total for the instances of the *target-item* grouped over a *by-item*. The syntax of the PCT clause is:

```
target-item PCT { OVER ALL  
                  { OVER by-item }
```

target-item

is a field name, a numeric literal, the predefined aggregates SUM and COUNT, an arithmetic expression, an IF/THEN/ELSE expression, a user table name, or a System Variable clause. The data type of the *target-item* used in a PCT clause must be numeric.

If *target-item* is an expression, that expression cannot contain an aggregate except a simple SUM or COUNT. If *target-item* is a user variable, that variable must not have been assigned an aggregate value unless the value is the result of a simple SUM or COUNT operation.

by-item

is the name of a numeric field grouped by a BY or BY DESC clause.

Using PCT OVER ALL

When the PCT OVER ALL clause is used, Enform Plus prints a percentage of the grand total for the numeric *target-item* in place of each *target-item* value. When only PCT is specified, Enform Plus assumes PCT OVER ALL.

Using PCT OVER By-Item

When the PCT OVER *by-item* clause is used, Enform Plus prints the percentage of the instances of the numeric *target-item* within the group in place of each *target-item* value. The group must have been defined earlier by a BY or BY DESC clause. In the following example, Enform Plus prints the percentage value of the cost of one part over the total value of all the parts at each location. For example, consider the following query and report:

```
LIST BY location,  
      partnum,  
      price,  
      price PCT OVER location;
```

LOCATION	Part Number	PRICE	PCT PRICE
-----	-----	-----	-----
H76	3102	4800.00	66.67
	7301	2400.06	33.33
H57	2402	7500.00	41.67
	3103	10500.00	58.33
...

Combining Percentages and Subtotals

The SUBTOTAL OVER clause can be combined with the PCT OVER clause, causing Enform Plus to print the percentage the subtotal represents of the total value of the *target-item*. Enform Plus does not print the value of each *target-item*. The percentage values do not always exactly total 100 percent because of truncation during division. If you execute the following query, for example, the total percentage values do not equal 100 percent:

```
OPEN employee
LIST BY regnum, empname, salary PCT, SUBTOTAL OVER regnum;
```

PCT Clause Used With User Variable

When a numeric *target-item* with a PCT clause is assigned to a user variable, Enform Plus makes the assignment to the user variable first, before the percentage is calculated. When the user variable is referenced as a *target-item* in a LIST statement or as an element within a *target-item* of a LIST statement, Enform Plus uses the value of the user variable. When the user variable is referenced in a print list, Enform Plus uses the value of the percentage for the user variable.

Restrictions

You cannot combine the PCT clause with the CUM clause.

SKIP Clause

The SKIP clause allows you to indicate the number of lines Enform Plus should move forward before continuing printing. The syntax of the SKIP clause is:

```
SKIP [ number ]
```

number

is an integer representing the number of lines to skip.

Sometimes it is desirable to print more than one *target-item* under the same column heading. If you want to do this, you can specify a SKIP clause after the first *target-item*, then use the TAB clause to position the printing of the second *target-item* under the first *target-item*. The second *target-item* should be modified by a NOHEAD clause to prevent Enform Plus from printing the unwanted column heading. Enform Plus prints the *target-items* under a suppressed column heading, one *target-item* per line.

The option variable @VSPACE affects the number of lines Enform Plus advances when the SKIP clause is specified. See the [Option Variable Clauses](#) on page 5-48 for more information.

Specifying the symbol / (slash) is equivalent to SKIP or SKIP 1.

SKIP Clause With a LIST Target-Item or By-Item

The SKIP clause can precede or follow a *target-item* or *by-item* within a LIST statement. If *number* is specified, Enform Plus moves forward the specified number of lines before printing continues. If *number* is not specified, Enform Plus moves forward to the next line.

In the following example, address and city are printed in the same column on separate lines:

```
LIST address,SKIP,
      city NOHEAD SKIP 2;
```

```
ADDRESS
```

```
-----
```

```
UNIVERSITY WAY
PHILADELPHIA
```

```
100 CALIFORNIA STREET
SAN FRANCISCO
```

SKIP Clause With a Print List

The SKIP clause can be part of the *print-list* of an AT END statement or clause, an AT START statement or clause, a FOOTING statement or clause, a SUBFOOTING statement or clause, a SUBTITLE statement or clause, or an AFTER CHANGE or

BEFORE CHANGE clause. When *number* is specified, Enform Plus moves forward the specified number of lines every time the SKIP clause is processed. When *number* is not specified, Enform Plus moves forward to the next line when the SKIP clause is processed.

SPACE Clause

The SPACE clause allows you to specify horizontal spacing. The syntax of the SPACE clause is:

SPACE [<i>number</i>]

number

is an integer.

SPACE Clause With a LIST Target-Item or By-Item

The SPACE clause can precede either a *target-item* or a *by-item* within a LIST statement. When *number* is specified, Enform Plus inserts the specified number of spaces every time the SPACE clause is processed. When *number* is not specified, one space is inserted.

The default spacing between columns on a report is initially set to two spaces. It can be temporarily overridden by using the SPACE clause with the *number* parameter. To delete all spaces between columns, use SPACE 0.

SPACE Clause With a Print List

The SPACE clause can be part of a *print-list* for an AFTER CHANGE, BEFORE CHANGE, AT START PRINT, AT END PRINT, FOOTING, SUBFOOTING, SUBTITLE, or TITLE clause or statement. When *number* is specified, Enform Plus inserts the specified number of spaces every time the SPACE clause is processed. When *number* is not specified, one space is inserted.

SUBFOOTING Clause

The SUBFOOTING clause allows you to specify printing of information at the bottom of each page preceding the footing for the current report. This clause is an optional part of the LIST statement. The syntax of the SUBFOOTING clause is:

```
SUBFOOTING print-list [ CENTER ]
```

print-list

contains any combination of literals, FORM, SKIP, SPACE, or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, or parameter names.

The clauses that can be part of a *print-list* are described in this section. The other elements are described in [Section 3, Enform Plus Language Elements](#).

Specifying Field Names in a SUBFOOTING Clause

If you specify a field name within a *print-list* of a SUBFOOTING clause, Enform Plus prints the same field value as the last row of data on the current page. A field name appearing within the SUBFOOTING clause need not be explicitly included within the associated LIST statement. If the field name is not included, Enform Plus effectively adds the field to the LIST statement with a NOPRINT clause.

Spacing Considerations

By default, the footing begins in the same position as the leftmost report column. Using SPACE or TAB clauses as the first element of the *print-list* overrides this default. SPACE or TAB clauses can also appear anywhere within the *print-list*. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
SUBFOOTING "Inventory" SPACE 15 "Parts in Stock",
```

```
Inventory           Parts in Stock
```

If you specify either a SKIP clause or the slash symbol (/) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the SUBFOOTING *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause causes two lines to be printed:

```
SUBFOOTING "Report 2-A" SKIP "Total Sales",
```

The following subfooting prints on the current report:

Report 2-A
Total Sales

Using the FORM clause within a SUBFOOTING *print-list* forces a new page. Enform Plus continues printing the remainder of the SUBFOOTING *print-list*, starting at the top of the next physical page. The page number remains the same. A single logical page can span multiple physical pages, so that a TITLE can appear on one page, the data on the next, and a SUBFOOTING on the next. The same page number applies to all physical pages in the logical page.

Using the CENTER clause centers the subfooting on the page.

The CENTER, Option Variables, SKIP, SPACE, and TAB clauses are described in this section.

Subfooting for Current Report or All Reports

A SUBFOOTING clause within a LIST statement prints a subfooting only for the current report. A SUBFOOTING clause temporarily overrides a session-wide SUBFOOTING statement. A SUBFOOTING statement prints a subfooting at the bottom of each page for all subsequent reports in the current session.

SUBTITLE Clause

The SUBTITLE clause allows you to specify printing of information at the top of each page immediately following the title for the current report. This clause is an optional part of the LIST statement. See also the [TITLE Clause](#) on page 5-69 and the [SUBTITLE Statement](#) on page 4-45. The syntax of the SUBTITLE clause is:

```
SUBTITLE print-list [ CENTER ]
```

print-list

contains any combination of literals, FORM, SKIP, SPACE, or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, or parameter names.

The clauses that can be part of a *print-list* are described in this section. The other elements are described in [Section 3, Enform Plus Language Elements](#)

Specifying a Field Name in a SUBTITLE Clause

If you specify a field name within a *print-list* of a SUBTITLE clause, Enform Plus prints the same field value as in the first row of the report. A field name appearing within the SUBTITLE clause need not be explicitly included within the associated LIST statement. If the field name is not included, Enform Plus in effect adds it with a NOPRINT clause.

Spacing Considerations

By default the subtitle begins in the same position as the leftmost report column. Using SPACE or TAB clauses as the first element of the *print-list* overrides this default. SPACE or TAB clauses can also appear anywhere within the *print-list*. In the following example, the SPACE clause causes the two literal to be separated by 15 spaces:

```
SUBTITLE "Inventory" SPACE 15 "Parts in Stock",
```

```
Inventory           Parts in Stock
```

If you specify either a SKIP clause or the slash symbol (/) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the SUBTITLE *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SKIP clause causes two lines to be printed:

```
SUBTITLE "Report 2-A" SKIP "Total Sales",  
  
Report 2-A  
Total Sales
```

Using the `FORM` clause within the *print-list* causes Enform Plus to start a new page. Enform Plus continues with the remainder of the *print-list* starting at the top of the next physical page. The page number remains the same.

Using the `CENTER` clause centers the subtitle on the page.

The `CENTER`, `Option Variable`, `SKIP`, `SPACE`, and `TAB` clauses are described in this section.

Subtitle for Current Report or All Reports

A `SUBTITLE` clause within a `LIST` statement prints a subtitle only for the current report. It temporarily overrides the session-wide `SUBTITLE` statement. A `SUBTITLE` statement prints a subtitle at the top of each page immediately following the title for all subsequent reports in the current session.

SUBTOTAL Clause

The SUBTOTAL clause allows you to specify the printing of a subtotal for a numeric *target-item*. This clause is an optional part of the LIST statement. The syntax for the SUBTOTAL clause is:

<i>target-item</i> SUBTOTAL [OVER <i>by-item</i>]

target-item

is a record name, a field name, a numeric literal, a predefined aggregate, a user aggregate, an arithmetic expression, an IF/THEN/ELSE expression, a user table name, a user variable, or a System Variable clause. The data type of the *target-item* being subtotaled must be numeric.

by-item

is the name of a field grouped by a BY or BY DESC clause.

When the SUBTOTAL clause is specified, Enform Plus prints the subtotals for each *target-item* within a *by-item* value. Enform Plus prints the subtotal in the column of the *target-item* being subtotaled and precedes the subtotal by a row of underline characters specified by the @UNDERLINE option variable. Enform Plus marks the subtotal with a subtotal string specified by the @SUBTOTAL-LABEL option variable. See the [Option Variable Clauses](#) on page 5-48.

When a SUBTOTAL OVER clause is used, Enform Plus subtotals the *target-item* each time the value of the specified *by-item* changes. When more than one SUBTOTAL OVER clause is specified, Enform Plus prints the subtotals in the order that the clauses are entered in the LIST statement.

When a SUBTOTAL clause is used without OVER, Enform Plus prints a subtotal in the specified *target-item* column each time the value of any *by-item* appearing to the left of the *target-item* changes. Enform Plus prints the subtotals using a minor-to-major order precedence; that is, Enform Plus prints the subtotal for the first *by-item* to the left of the *target-item*, followed by the subtotal for the second *by-item* to the left of the *target-item*, and so on until the subtotal for the last *by-item* in the report is printed.

SUBTOTAL does not generate a grand total at the end of the report. If a grand total is desired, use the TOTAL clause. The TOTAL clause is described on page [5-71](#).

If the width of the value of a subtotal exceeds the width of the format specified for a *target-item*, an overflow condition occurs causing asterisks to be printed in place of the value. To prevent this, enlarge the *target-item* display format by adding an AS clause to the *target-item* being subtotaled. The AS clause is described on page [5-7](#).

SUPPRESS Clause

The SUPPRESS clause allows you to eliminate certain records from being printed in the report. The records still contribute to the report calculations. This clause is an optional part of the part of the LIST statement. This clause cannot be used with the FIND statement. The syntax of the SUPPRESS clause is:

```
SUPPRESS [ WHERE ] logical-expression
```

logical-expression

is an expression returning a true or false value. See [Logical Expressions](#) on page 3-22 for more information.

The SUPPRESS clause defines a condition or conditions that prevents specific records from printing throughout a report. For example, in the following query, the SUPPRESS clause causes Enform Plus to print only the inventory, part number, and part name where the inventory is greater than zero:

```
LIST ASCD inventory,partnum,partname,
    SUPPRESS WHERE inventory GT 0;
```

This query generates the following report:

INVENTORY	Part Number	PARTNAME
-----	-----	-----
-100	2001	DECIMAL ARITH
-32	6402	TERM CRT PAGE
-16	6201	SYNC CONTROLLER
-1	5504	LP 900 LPM
0	5505	LP 1500 LPM

Enform Plus still includes the suppressed records in AFTER CHANGE and BEGIN CHANGE clauses, subtotals, totals, and other calculations specified for the report. Note that the value of the first record of an AFTER CHANGE clause or the last record of a BEFORE CHANGE clause is used for the print list whether or not that record is printed.

Aggregates cannot be used in a SUPPRESS clause; you can, however, reference a user variable that has been assigned the aggregate value.

System Variable Clauses

The System Variable clauses allow you to obtain the current value for the current date, time, line number, and page number. The syntax for the System Variable clauses is:

```
@DATE  
@TIME  
@LINENO  
@PAGENO
```

Printing the Current Date or Time

The @DATE and @TIME System Variable clauses return the current date and time in internal format. When used in an expression, Enform Plus treats them as numeric literals.

When the @DATE or @TIME system variables are to be printed on a report, convert them to a display format using the AS DATE or AS TIME clauses. Frequently the default format is satisfactory and can be specified by:

```
@DATE AS DATE *
```

For more information on date and time display formats, see the [AS DATE Clause](#) on page 5-22 and the [AS TIME Clause](#) on page 5-24.

Printing Line Numbers

The @LINENO System Variable clause prints the current line number within a page of the report. @LINENO can be a *target-item* in a LIST statement, part of an AFTER CHANGE, BEFORE CHANGE, AT START, AT END, FOOTING, TITLE, SUBFOOTING, and SUBTITLE clauses or statements.

The @LINENO option variable specifies how many lines are printed on a page. It is initially set to 60 lines. To change the number of lines to be printed per page, reset the @LINENO Option Variable clause. See the [SET Statement](#) on page 4-41 and the [Option Variable Clauses](#) on page 5-48.

Printing Page Numbers

The @PAGENO System Variable prints the page number. The following example prints the page number at the top of a page:

```
TITLE TAB 100 "Page " @PAGENO;
```

Page 10

TAB Clause

The TAB clause allows you to specify in which column in the report a *target-item* or *by-item* is to begin. This clause is an optional part of the LIST statement and must not be specified in the FIND statement. The syntax of the TAB clause is:

TAB [<i>number</i>]

number

is an integer.

The TAB clause specifies the column to which to tab before printing an element on a report. Care must be taken not to overlap elements. If overlap occurs, the last numbers or characters specified appear on the printed report. Note that TAB never causes Enform Plus to advance to the next line, so you can tab backwards on the current line.

TAB Clause With a LIST Target-Item or By-Item

The TAB clause can precede a *target-item* or *by-item* within a LIST statement. If *number* is specified, Enform Plus tabs to that position before the next *target-item* or *by-item* is printed; if *number* is not specified, Enform Plus begins printing in column one.

TAB Clause With a Print List

The TAB clause can be part of a *print-list* for an AT END statement or clause, an AT START statement or clause, a FOOTING statement or clause, a SUBFOOTING statement or clause, a SUBTITLE statement or clause, and the AFTER CHANGE and BEFORE CHANGE clauses. When *number* is specified, Enform Plus tabs to that position before the printing next element on the current line. If *number* is not specified, Enform Plus uses column one as the default.

TIMESTAMP-DATE Clause

The `TIMESTAMP-DATE` clause extracts the date portion of a timestamp field that has been created by the Guardian procedure `TIMESTAMP`. The syntax of the `TIMESTAMP-DATE` clause is:

<code>TIMESTAMP-DATE (<i>field-name</i>)</code>

field-name

is the name of a field to which the `TIMESTAMP-DATE` clause returns a date value. The field must be described in the dictionary.

You must define the field that receives the date value from the `TIMESTAMP-DATE` clause in your data dictionary. You must define the field as a six-character alphanumeric field, such as:

```
05  TIME-STAMP      TYPE CHARACTER 6.
```

The value returned to this field is a quantity of days in internal format.

See the [TIMESTAMP-TIME Clause](#) on page 5-68 for extracting the time portion of a timestamp field.

TIMESTAMP-TIME Clause

The `TIMESTAMP-TIME` clause extracts the time portion of a timestamp field that has been created by the Guardian procedure `TIMESTAMP`. The syntax of the `TIMESTAMP-TIME` clause is:

<code>TIMESTAMP-TIME (<i>field-name</i>)</code>

field-name

is the name of a field to which a time value is returned. The field must be described in your data dictionary.

Your data dictionary must contain a definition of the field to which the `TIMESTAMP-TIME` clause returns a time value. You must define the field as a six-character alphanumeric field, such as:

```
05  TIME-STAMP      TYPE CHARACTER 6.
```

The value returned to this field is a quantity in hundredths of a second in internal time format. You can obtain this value in seconds by using the `AS` clause with an integer edit descriptor or by using the `AS TIME` clause. If you need to obtain this value in hundredths of a second, you must write your own conversion routine.

See the [TIMESTAMP-DATE Clause](#) on page 5-67 for extracting the date portion of a timestamp field.

TITLE Clause

The TITLE clause allows you to specify printing of information at the top of each page for the current report. See also the [SUBTITLE Clause](#) on page 5-61 and the [TITLE Statement](#) on page 4-47. The syntax of the TITLE clause is:

```
TITLE print-list [ CENTER ]
```

print-list

contains any combination of literals, FORM, SKIP, SPACE, or TAB clauses. A *print-list* can also contain the following elements that can be modified by AS, AS DATE, or AS TIME clauses: field names, arithmetic expressions, IF/THEN/ELSE expressions, System Variable clauses, JULIAN-DATE clauses, TIMESTAMP-DATE clauses, TIMESTAMP-TIME clauses, user variables, or parameter names.

The clauses that can be part of a *print-list* are described in this section. The other elements are described in [Section 3, Enform Plus Language Elements](#)

Specifying Field Names in a TITLE Clause

If you specify a field name within a *print-list* of a TITLE clause, Enform Plus prints the same field value as in the first row of the page. A field name appearing within the TITLE clause need not be explicitly included within the associated LIST statement. If field name is not included, Enform Plus in effect adds it with a NOPRINT clause.

Spacing Considerations

By default, the title begins in the same position as the leftmost report column. Using SPACE or TAB clauses as the first element of the *print-list* overrides this default. SPACE or TAB clauses can also appear anywhere within the *print-list*. In the following example, the SPACE clause causes the two literals to be separated by 15 spaces:

```
TITLE "Inventory" SPACE 15 "Parts in Stock",
```

```
Inventory           Parts in Stock
```

If you specify either a SKIP clause or the slash symbol (/) within a *print-list*, Enform Plus advances one or more lines before printing the rest of the AFTER CHANGE *print-list*. The number of lines advanced can be affected by one or more of the following: the digit (if any) following the keyword SKIP, the number of slashes specified, or the option variable @VSPACE. In the following example, the SPACE clause causes two lines to be printed:

```
TITLE "Report 2-A" SKIP "Total Sales",
```

```
Report 2-A
Total Sales
```

Using the `FORM` clause within a *print-list* causes Enform Plus to start a new page and continue with the rest of the *print-list*. The page number remains the same. A single logical page can span multiple physical pages such that a `TITLE` appears on one page, the data on the next, and a `FOOTING` on the next. The same page number applies to all physical pages in a logical page.

Using the `CENTER` clause centers the title on the page.

The `CENTER`, `Option Variables`, `SKIP`, `SPACE`, and `TAB` clauses are described in this section.

Title for Current Report or All Reports

A `TITLE` clause within a `LIST` statement prints a title only for the current report. It temporarily overrides the session-wide `TITLE` statement. A `TITLE` statement prints a title at the top of each page for all subsequent reports in the current session.

Overriding Session-Wide Title

You can temporarily override a session-wide `TITLE` statement by using the `TITLE` clause with “” for the *print-list* parameter.

TOTAL Clause

The TOTAL clause prints the grand total for a numeric *target-item*. This clause is an optional part of the LIST statement and cannot be specified in a FIND statement. The syntax of the TOTAL clause is:

```
{ target-item } TOTAL  
{ by-item }
```

target-item

is a record name, a field name, a numeric literal, a predefined aggregate, a user aggregate, an arithmetic expression, an IF/THEN/ELSE expression, a user table name, a user variable, or a System Variable clause. The data type of the *target-item* being totaled must be numeric.

by-item

is a field whose values are grouped by a BY or BY DESC clause.

When the TOTAL clause is specified, Enform Plus prints the grand total for a *target-item* or *by-item* after printing the last value of the *target-item* or *by-item*. Enform Plus precedes the total by two rows of underline characters, as specified by the @UNDERLINE option variable. See the [Option Variable Clauses](#) on page 5-48.

If the width of the value of a total exceeds the width of the *target-item* or *by-item* display format, an overflow condition occurs causing asterisks to be printed in place of the total. To prevent this, enlarge the display format by adding an AS clause to the *target-item* being totaled. See the [AS Clause](#) on page 5-7.

WHERE Clause

The WHERE clause allows you to restrict the records that contribute to the target file. The syntax of the WHERE clause is:

```
WHERE logical-expression
```

logical-expression

is a condition that returns a true or false value. See [Logical Expressions](#) on page 3-22 for more information.

The logical expression in a WHERE clause defines which records are restricted from contributing to the target file. For example:

```
WHERE employee; BEGINS WITH "BROWN";
```

Using the WHERE Clause to Establish a Link

Using the WHERE clause to specify a link causes only those records that satisfy the condition specified in the *logical-expression* to be included in the report. The link created by the WHERE clause remains in effect only for the associated LIST or FIND statement. For a more complete description of linking, see the [LINK Statement](#) on page 4-25. The following example shows how the WHERE clause is used to specify a link:

```
WHERE parts.partnum EQ odetail.partnum,
```

6 Commands

This section contains a complete description of the syntax of the Enform Plus commands. The commands are arranged in alphabetic order.

Enform Plus commands are compiler directives that instruct the Enform Plus compiler/report writer to perform a specific operation. The compiler/report writer recognizes a command by the presence of a question mark in column 1. You can either enter commands at the level of the Enform Plus prompt (>) or place commands in an Edit-format file. Enform Plus executes commands in an Edit-format file as if they were entered interactively with one exception: Enform Plus ignores a ?RUN command in an Edit-format file.

The following commands are not saved in a compiled query file (created by the ?COMPILE command): ?ATTACH, ?COMPILE, ?EDIT, ?EXECUTE, ?EXIT, ?HELP, ?OUT, ?RUN, ?SECTION, ?SHOW, ?SOURCE. For this reason, these commands cannot be used:

- Through the host-language interface (described in the *Enform User's Guide*). The host-language interface requires a compiled query file.
- When you specify the ?EXECUTE command. The ?EXECUTE command can be used only with a compiled query file.
- If the file name specified in the IN option of the ENFORM command is a compiled query file.

[Table 6-1](#) lists the Enform Plus commands and their functions.

Table 6-1. Summary of Commands (page 1 of 2)

Command	Function
?ASSIGN	Associates a new physical file with a record description.
?ATTACH	Specifies a query processor (QP) to use during an Enform Plus session.
?COMPILE	Compiles an Enform Plus query and saves it in a compiled query file.
?DICTIONARY	Names a subvolume that contains a dictionary. It also clears the internal table and reclaims table space.
?EDIT	Accesses the Text Editor without leaving Enform Plus.
?EXECUTE	Executes a compiled query file.
?EXIT	Terminates the current Enform Plus session.
?HELP	Displays information about the syntax of the Enform Plus language. This command can also display user-defined information.
?OUT	Specifies the output device for a report.

Table 6-1. Summary of Commands (page 2 of 2)

Command	Function
?RUN	Compiles and executes Enform Plus source queries in an Edit-format file.
?SECTION	Identifies a section of Enform Plus commands and statements within an Edit-format file.
?SHOW	Displays information about the environment of the current Enform Plus session.
?SOURCE	Reads an Edit-format file or a collection of commands and statements within an Edit-format file.

?ASSIGN Command

The ?ASSIGN command associates a physical file with a dictionary record description or a generic file. This command is functionally equivalent to the TACL ASSIGN command. The syntax of the ?ASSIGN command is:

```
?ASSIGN [ { record-name } [ , ] physical-file-name
         [ { generic-file-name } [ TO ] ]

        [ , create-open-spec ... ]

?ASSIGN record-name , { , create-open-spec } ...
```

record-name

is the name of a record description from a DDL dictionary.

generic-filename

is the name of one of the Enform Plus generic files described under [Generic Files](#) on page 2-10. A generic file is used to store some class of Enform Plus output. The following names are allowed:

```
QUERY-COMPILER-LISTING
QUERY-REPORT-LISTING
QUERY-STATISTICS
QUERY-STATUS-MESSAGES
QUERY-WORK-AREA
QUERY-SORT-AREA
QUERY-QPSTATISTICS
QUERY-QPSTATUS-MESSAGES.
```

physical-filename

For *record-name*, *physical-filename* is the name of a physical file being associated with the record description. TO must be specified if the file name is TO. A partial file name is expanded by using the default system, volume, and subvolume names.

For *generic-file-name*, *physical-filename* is the name of the physical device to which you want Enform Plus to direct output.

create-open-spec

is one file creation or open attribute. Only *exclusion-spec* is used by Enform Plus. *create-open-spec* is of the form:

```

{ extent-spec          }
{ CODE file-code      }
{ exclusion-spec      }
{ access-spec         }
{ REC record-size     }
{ BLOCK block-size    }

```

extent-spec

is one of the following:

```

{ EXT [ ( ] pri-extent-size [ ) ] }
{ EXT ( [ pri-extent-size ] , sec-extent-size ) }

```

exclusion-spec

is one of the following:

```

{ EXCLUSIVE }
{ SHARED    }
{ PROTECTED }

```

access-spec

is one of the following:

```

{ I-O }
{ INPUT }
{ OUTPUT }

```

The ?ASSIGN command can be entered at any time prior to running a query. Each time an ?ASSIGN command is entered, an entry of the assignment is made in the internal table. The internal table holds up to 32 assignments. When the internal table is full, the ?ASSIGN command can be entered without any parameters to clear all of the assignments.

Each DDL dictionary RECORD statement describes records in a physical file, named by a DDL FILE IS clause. Enform Plus uses the physical file name in the DDL FILE IS clause to locate the records and retrieve information. The actual records desired are not always in the physical file named in the dictionary. An example would be when test records are used in place of actual records. The ?ASSIGN command temporarily overrides the physical file named by the DDL FILE IS clause. In the following example, the employee record description is assigned to the physical file \$data.database.sample:

```
?ASSIGN employee TO $data.database.sample
```


The ?ASSIGN command can be used to assign the generic Enform Plus files to a physical device. In the following example, the generic file QUERY-COMPILER-LISTING is assigned to a printing device named \$ep:

```
?ASSIGN QUERY-COMPILER-LISTING TO $s.#ep
```

The ?ASSIGN command can be used to change the exclusion specification for a file; the default is SHARED. If the DDL physical file name is not being overridden, only the record name and the exclusion specification need be specified. For example:

```
?ASSIGN employee, ,SHARED
```

When Enform Plus is run interactively, an ?ASSIGN command overrides a TACL ASSIGN command entered before the ENFORM command. When Enform Plus is run with a command file providing the input, however, a TACL ASSIGN command entered before the ENFORM command overrides an ?ASSIGN command in the command file.

?ATTACH Command

The ?ATTACH command allows you to specify a query processor to use during an Enform Plus session. The syntax of the ?ATTACH command is:

```
?ATTACH [ process-name ]
```

process-name

is the name of a query processor.

When one or more server query processors exist on a Compaq system, you can use the ?ATTACH command to specify the query processor you want to use. If you do not specify the ?ATTACH command, Enform Plus defaults to a dedicated query processor. A dedicated query processor is an Enform Plus process that is created for and provides services to an individual compiler/report writer process. The dedicated query processor runs for the duration of an Enform Plus session until either an error occurs or an ?ATTACH command is issued specifying a new query processor. When you enter the ?ATTACH command without a process name, Enform Plus starts a new dedicated query processor for your use.

An ?ATTACH failure is not evident until the first FIND or LIST statement is executed. The possible ?ATTACH failures and the actions you can take are as follows:

- The server query processor already has the maximum number of users it can handle. In this case, you can wait and then try again specifying the same query processor; name a different server query processor; or select a dedicated query processor.
- The server query processor does not exist. In this case, you can start the server query processor, name a server query processor that does exist, or select a dedicated query processor.
- The attached server query processor reached one of two limits set for that particular server query processor. Either the query reached a database read limit, or the cost strategy specified by Enform Plus exceeded its limit. In this case, you can name a server query processor with greater limits or without built-in limits; change the query; or select a dedicated query processor. The @READS and @COST-TOLERANCE Option Variable clauses are described under [Option Variable Clauses](#) on page 5-48.

?COMPILE Command

The ?COMPILE command compiles a query and stores it in a physical file. The syntax of the ?COMPILE command is:

```
?COMPILE edit-filename [ ( section-name , ... ) ]  
      TO compiled-physical-filename
```

edit-filename

is the name of the Edit-format file containing the Enform Plus query.

(*section-name* , ...)

is a list of the names of the section(s) within an Edit-format file. The list must be enclosed within parentheses.

compiled-physical-filename

is the name of the physical file to contain the compiled query.

The ?COMPILE command compiles a query. The Enform Plus internal table becomes part of the compiled query file. If *physical-filename* existed before compilation, the old version is purged and a new physical file is created. If errors are produced during a compilation attempt, the original compiled query file is not affected at all.

The ?COMPILE command compiles only one LIST or FIND statement. If your Edit-format file contains more than one LIST or FIND statement, Enform Plus compiles the first LIST or FIND statement to the compiled query file. Enform Plus does not compile any statements or commands following the first LIST or FIND statement; instead Enform Plus continues processing, executing the statements and commands as they are encountered in the Edit-format file.

When the LIST or FIND statement of the query is written to the compiled query file, most of the commands, options, and other statements that provide environmental information are stored in the internal table. The internal table is then saved in the compiled query file.

Enform Plus assigns a file code of 888 to the compiled query file. Enform Plus creates this code to identify the physical file as a stored query produced by Enform Plus.

A query called by a host-language program must be compiled first. Other queries do not need to be compiled before run time; however, there are some advantages to creating a compiled query file. The compiled query file protects the query from being inadvertently changed. Less processing time is required when the compiled query file is saved and reused for subsequent runs.

?DICTIONARY Command

The ?DICTIONARY command names a subvolume that contains your dictionary. It also clears the internal table and reclaims table space. The ?DICTIONARY command is the same as the DICTIONARY statement.

```
?DICTIONARY [ dict-subvol-name ]
```

dict-subvol-name

is the name of the subvolume where your dictionary files reside. See the *Guardian Programmer's Guide* for information on specifying Guardian file names.

The dictionary identified in a DICTIONARY command must be created by the Data Definition Language compiler. Enform Plus issues an error message if you attempt to use a dictionary compiled with a version of DDL that is not current. You must then recompile the dictionary with a current version. See the *Data Definition Language (DDL) Reference Manual* for more information on creating and compiling a dictionary.

Identifying the Dictionary

There are two ways to identify the location of the dictionary:

- The volume and subvolume where the dictionary resides can be specified as a part of the TACL ENFORM command. If none is specified, it is assumed that the dictionary resides on the default volume and subvolume.
- The DICTIONARY statement or ?DICTIONARY command can specify where the dictionary resides. Either overrides the dictionary identified at the time of the TACL ENFORM command. When a new dictionary is specified, the old internal table is cleared.

Clearing Internal Tables

Entering the ?DICTIONARY command without a volume and subvolume name is a simple means of clearing the entire internal table, without changing the dictionary. To clear only certain items of the internal table, see the [CLOSE Statement](#) on page 4-7 and the [DELINK Statement](#) on page 4-12. The items cleared by the ?DICTIONARY command are:

- All record descriptions from previous OPEN statements
- All previous linking relationships
- All user variable, user aggregate, and user table definitions
- All parameter definitions.
- All information from AT END, AT START, FOOTING, SUBFOOTING, SUBTITLE, and TITLE statements.

The ?DICTIONARY command does not change the value set for any option variable (such as @HEADING or @STATS).

?EDIT Command

The ?EDIT command allows you to access the Editor without leaving Enform Plus. The syntax of the ?EDIT command is:

```
?EDIT [ edit-filename ]
```

edit-filename

is the name of an Edit-format file.

A query can be stored in an Edit-format file. The Edit-format file can be created or changed without leaving Enform Plus. An ?EDIT command invokes the Editor and operates just as if the TACL EDIT command was used. If a new file name is specified while using the Editor, that name becomes the default *edit-filename* for use in a subsequent ?EDIT or ?RUN command. To exit from the Editor, enter EXIT at the Editor's prompt (*) and control returns to Enform Plus.

?EXECUTE Command

The ?EXECUTE command executes a compiled query file. The syntax of the ?EXECUTE command is:

<code>?EXECUTE <i>compiled-physical-filename</i></code>

compiled-physical-filename

is the name of the physical file containing the stored compiled query.

The ?EXECUTE command resets the internal table to the same state that existed at time of compilation. The ?EXECUTE command changes the dictionary being used for the duration of the execution. Internal table information from the compiled query file is used until execution terminates, then Enform Plus uses the dictionary specified before you entered the ?EXECUTE command.

If the stored compiled query requires a parameter value passed to it during execution, the TACL PARAM command can be issued prior to executing the query. For information on the PARAM command, see the *Guardian User's Guide*.

The ?EXECUTE command accepts only a physical file with a file code of 888. Enform Plus assigns a file code of 888 when it creates a compiled query file.

?EXIT Command

The ?EXIT command terminates the current Enform Plus session. The syntax of the ?EXIT command is:

?EXIT

The ?EXIT command returns control to TACL. It is the same as the EXIT statement. An alternative way of exiting from Enform Plus is to press the Ctrl and Y keys simultaneously.

?HELP Command

The ?HELP command displays information about the syntax of the Enform Plus language. The command can also display information about user-defined topics if your system manager has provided this information. The syntax of the ?HELP command is:

```
?HELP [ help-element ]
```

help-element

is a topic for which help is available. Among these topics is the syntax of the Enform Plus statements, clauses, commands, and language elements. To obtain a current list of the *help-elements*, enter the ?HELP command without *help-element*. Enform Plus responds by displaying a list of all the topics for which help is available.

When specifying the ?HELP command, enter *help-element* in either upper or lower case characters. Enform Plus accepts unambiguous abbreviations for *help-element*. If you use an ambiguous abbreviation, Enform Plus displays help text for all *help-elements* beginning with those characters.

A *help-element* might be an Enform Plus keyword (for example, LIST). If *help-element* is an Enform Plus keyword, Enform Plus displays the syntax of the statement, clause, command, or language element associated with that keyword. Alternatively, a *help-element* might be a category of Enform Plus keyword (for example, report format). If *help-element* is a category, Enform Plus displays the syntax for all Enform Plus keywords in that category.

Enform Plus retrieves the text of help messages from the Enform Plus message table. By modifying the message table, you can modify the help text. See the *Enform User's Guide* for information about modifying the message table.

The following example shows a list of topics that might be displayed when you enter the ?HELP command without *help-element*:

```
>?HELP Help is available for the following topics:
```

```
STATEMENTS:  AT END      AT START  CLOSE    DECLARE  DELINK
              DICTIONARY EXIT      FIND     FOOTING  LINK
              LIST      OPEN      PARAM    SET      SUBFOOTING
              SUBTITLE   TITLE
```

```
CLAUSES:    global modifier      modifier positional control
            report format         target-item
```

```
OTHER:      aggregate          arithmetic expression
            logical expression  option variable
            print list          system variable
```

```
COMMANDS:   ?ASSIGN  ?ATTACH  ?COMPILE  ?DICTIONARY
            ?EDIT    ?EXECUTE  ?HELP     ?OUT
            ?RUN     ?SECTION  ?SHOW     ?SOURCE
```


In the following example, Enform Plus displays information about a target item when you enter ?HELP target-item:

```
>?HELP target-item
```

A target-item is a record name, a field name, a literal, a user variable name, a parameter name, a system variable clause, an aggregate, or an arithmetic expression.

Help is also available for SYSTEM VARIABLE, AGGREGATE, and ARITHMETIC EXPRESSION.

?OUT Command

The ?OUT command specifies the output device for a report. The syntax of the ?OUT command is:

```
?OUT [ physical-filename ]
```

physical-filename

is the name of the output device to which any subsequent report is directed.

The ?OUT command is used to name the physical device on which a report is printed. The physical device can be a printing device such as \$m1p. The physical device can also be a file name. The ?OUT command remains in effect until a new ?OUT command is issued or the session is terminated.

Issuing the ?OUT command without specifying a *physical-filename* causes the report to be printed on the listing file specified by ENFORM's run-time OUT option.

The ?OUT command affects only where the report is sent. The source listing goes either to the default listing file specified by the OUT option of the ENFORM command or to the QUERY-REPORT-LISTING file.

The ?OUT command can be part of the Edit-format file or can precede an ?EXECUTE or ?RUN command. The ?OUT command is not saved in a compiled query file.

?RUN Command

The ?RUN command executes one or more queries stored in an Edit-format file. The syntax of the ?RUN command is:

```
?RUN [ edit-filename [ ( section-name , ... ) ]
```

edit-filename

is the name of the Edit-format file containing the Enform Plus source query.

(*section-name* , ...)

is a list of the names of one or more sections of the Edit-format file. A *section-name* must be enclosed within parentheses.

If both *edit-filename* and *section-name* are specified, Enform Plus compiles and executes that specific collection of commands statements identified by the section name. Multiple sections of an Edit-format file can be combined to create complete queries.

If only *edit-filename* is specified, Enform Plus compiles and executes the entire Edit-format file. If neither *edit-filename* nor *section-name* is specified, Enform Plus compiles and executes the Edit-format file specified by the most recent ?EDIT or ?RUN command.

Note that Enform Plus issues a warning message and ignores a ?RUN command if:

- The ?RUN command appears in the file specified as the IN option of the ENFORM command.
- The ?RUN command appears in an Edit-format file compiled and executed by a ?SOURCE command.
- The ?RUN command appears in an Edit-format file compiled and executed by another ?RUN command.

?SECTION Command

The ?SECTION command names a collection of Enform Plus commands and statements within an Edit-format file. The syntax of the ?SECTION command is:

?SECTION <i>section-name</i>

section-name

is the name to be used for a collection of Enform Plus commands and/or statements within an Edit-format file.

The ?SECTION command names a collection of Enform Plus commands and/or statements within an Edit-format file. The ?COMPILE, ?RUN, and ?SOURCE commands can be used to read a section or sections of an Edit-format file.

The names for a section must follow these rules:

- Must be unique
- Must start with an alphabetic character or circumflex (^)
- Can contain from 1 to 31 characters (names longer than 31 characters are truncated and a warning message is issued)
- Can include numbers, hyphen (–) and/or circumflex (^)
- Cannot contain embedded blanks
- Cannot end with a hyphen (–)

?SHOW Command

The ?SHOW command displays information about the environment of the current Enform Plus session. The syntax of the ?SHOW command is:

	[OPEN]
	[LINK]
	[CONTROL]
?SHOW	[LIMITS]
	[ASSIGN [<i>record-name</i>]]
	[<i>user-variable-name</i>]
	[<i>record-name</i>]
	[<i>param-name</i>]

user-variable-name

is the name of a user variable.

record-name

is the name of an opened dictionary record description.

param-name

is the name of a parameter.

Note that if you do not terminate statement preceding a ?SHOW command with a semicolon, the information displayed by the ?SHOW command will not reflect the result of the statement.

[Table 6-2](#) shows the environmental information displayed by the ?SHOW command:

Table 6-2. Information Displayed by the ?SHOW Command (page 1 of 2)

Command	Display Message
?SHOW	Lists the various items that can be displayed with the ?SHOW command.
?SHOW OPEN	Lists the open record descriptions. If none have been opened, nothing is displayed. An OPEN statement does not actually open a physical file, but accesses the record description in the dictionary.
?SHOW LINK	Lists links that are in effect. If no record descriptions have been linked, nothing is displayed. A LINK statement does not actually link physical files, but accesses their record descriptions from the internal table.

Table 6-2. Information Displayed by the ?SHOW Command (page 2 of 2)

Command	Display Message
?SHOW CONTROL	Displays the current values of all of the option variables. Option variables can be changed by setting the Option Variable clauses. See the SET Statement on page 4-41 and the Option Variable Clauses on page 5-48.
?SHOW LIMITS	Displays the current space available for the symbol table, literals and formats, LINK table, PRINT table, PARAM table, and OVER clause table.
?SHOW ASSIGN	Displays all of the opened record descriptions and physical file name pairs specified by an ?ASSIGN command or TACL ASSIGN command. If none were assigned, nothing is displayed.
?SHOW ASSIGN <i>record-name</i>	Displays all ASSIGN table information related to the specified record name.
?SHOW <i>user-var-name</i>	Displays the current value(s) of the user variable or table.
?SHOW <i>record-name</i>	Displays the following for the specified opened record description. <ul style="list-style-type: none"> ● The file name that exists in the data dictionary for the opened record description ● Each field's name, data type, offset, length, number of occurrences, length of each occurrence, and whether a field is a key field.
?SHOW <i>param-name</i>	Displays the current value of the specified parameter.

?SOURCE Command

The ?SOURCE command reads an Edit-format file or a collection of commands and statements within an Edit-format file. The syntax of the ?SOURCE command is:

<pre>?SOURCE <i>edit-filename</i> [(<i>section-name</i> , ...)]</pre>

edit-filename

is the name of the Edit-format file containing an Enform Plus query.

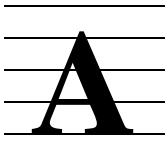
(*section-name* , ...)

is a list of the names of the sections of the Edit-format file; *section-name* must be enclosed with parentheses.

The ?SOURCE command can be used to read in an Edit-format file or a collection of commands and/or statements within an Edit-format file. When a ?SOURCE command is entered, the specified commands or statements are read in just as if they had been entered a line at a time. ?SOURCE commands can be nested up to a depth of four.

If both the *edit-filename* and *section-name* are specified, that specific collection of commands and statements of the Edit-format file is read in. If only the *edit-filename* is specified, the entire Edit-format file is read in.

Issuing the ?SOURCE command does not set the default Edit-format file name for subsequent ?RUN or ?EDIT commands.



Enform Plus Syntax Summary

This appendix summarizes the Enform Plus syntax. For specific details of the syntax, see [Section 3, Enform Plus Language Elements](#); [Section 4, Statements](#); [Section 5, Clauses](#); and [Section 6, Commands](#).

Language Elements

Aggregates:

```
{ { AVG } }
{ { COUNT } }
{ { MAX } ( { field-name } [ OVER ALL ] }
{ { MIN } ( { expression } [ OVER over-item ] }
{ { SUM } }
{ { user-aggregate } [ WHERE logical-expression ] ) , }

{ { AVG } }
{ { COUNT } }
{ { MAX } ( [ UNIQUE ] field-name [ OVER ALL ] }
{ { MIN } }
{ { SUM } [ WHERE logical-expression ] ) , }
{ { user-aggregate } }
```

Arithmetic operators:

```
+
-
*
/
```

IF/THEN/ELSE expression:

```
( IF logical-expression THEN value-1 ELSE value-2 )
```

Logical Expression:

[NOT] *condition* [{ AND | OR }] [NOT] *condition*] ...

where *condition* has one of the following forms:

<i>field-name</i>	{	[not]	{	BEGINS WITH	}	<i>string-literal</i>
			{	' '	}	
			{	CONTAINS	}	
			{	'>'	}	
			{	conditional operator	}	
	{	[NOT]	{	EQUAL	}	
			{	EQ	}	
			{	IS	}	
			{	=	{ value-range }	
			{	NE	{ "[pattern-match]" }	
{	<>	}				

{	variable	}	[NOT] conditional-operator	{	variable	}
{	field-name	}		{	field-name	}
{	expression	}		{	expression	}

Statements

AT END [PRINT *print-list* [CENTER]] [;]

AT START [PRINT *print-list* [CENTER]] [;]

CLOSE { *record-name*
user-variable-name
user-aggregate-name
user-table-name
param-name } , ... [;]

DECLARE { { *user-variable-name*
user-table-name "[" *max-subscript* "]" }
user-aggregate-name (*formal-argument*)
= (*step-expression* [, [*end-expression*]])
[, initialize-constant]) }
{ [INTERNAL *internal-format*]
[AS *display-format*]
[HEADING *heading-string*] } , ... [;]

DELINK { *record-name1* [TO [OPTIONAL]]
record-name2 VIA *field-name* }
{ *qualified-field-name1* [TO [OPTIONAL]]
qualified-field-name2 } , ... [;]

DICTIONARY [*dict-subvol-name*]

EXIT [;]

FIND [UNIQUE] *output-record-name*

({ [*output-field-name* :=] { { BY *by-item*
BY DESC *by-item*
target-item
ASCENDING *target-item*
DESCENDING *target-item* } } } , ...)

[WHERE *logical-expression*] ;

FOOTING [*print-list* [CENTER]] [;]

LINK { *record-name1* [TO] [OPTIONAL] *record-name2*
VIA *field-name* } , ... [;]
{ *qualified-field-name1* [TO] [OPTIONAL]
qualified-field-name2 }

```

LIST [ UNIQUE ] {
  {
    BY by-item
    BY DESC by-item
    target-item
    ASCD target-item
    DESC target-item
    user-var-name := target-item
  }
  [ CUM [ OVER ALL ] ]
  [ CUM OVER by-item ]
  [ PCT [ OVER ALL ] ]
  [ PCT OVER by-item ]
  [ TOTAL ]
  [ SUBTOTAL ]
  [ SUBTOTAL OVER by-item ] , ... , ...
  [ NOHEAD ]
  [ NOPRINT ]
  [ CENTER ]
  [ HEADING string-literal ]
  [ AS display-format ]
  [ AS DATE display-format ]
  [ AS TIME display-format ]

  [ FORM [ n ] ]
  [ SKIP [ n ] ] , ...
  [ SPACE [ n ] ]
  [ TAB [ n ] ]
}

[ WHERE logical-expression ]

[ NOHEAD ALL ]
[ NOPRINT ALL ]
[ CENTER ALL ]

[ SUPPRESS [ WHERE ] logical-expression ]

[ BEFORE CHANGE [ON] by-item PRINT print-list [ CENTER ] ]
[ AFTER CHANGE [ON] by-item PRINT print-list [ CENTER ] ]
[ AT START PRINT print-list [ CENTER ] ]
[ AT END PRINT print-list [ CENTER ] ]
[ TITLE print-list [ CENTER ] ]
[ SUBTITLE print-list [ CENTER ] ]
[ FOOTING print-list [ CENTER ] ]
[ SUBFOOTING print-list [ CENTER ] ] ;

```

```
OPEN { record-name
      { record-name2 [ AS ] COPY [ OF ] record-name1 }, ... [ ; ]
```

```
PARAM { param-name [ INTERNAL internal-format ] }, ... [ ; ]
```

```
SET { { user-variable-name } [ string-literal ]
      { user-table-name } [TO] [ ]
      { ["subscript"] } [ number-literal ]
      { param-name }
      {
        { ON
          { OFF
            { unsigned-digit
              { string-literal
                { display-format
              }
            }
          }
        }
        { option-variable-name [TO]
          { unsigned-digit
            { string-literal
              { display-format
            }
          }
        }
      }
```

```
SUBFOOTING [ print-list [ CENTER ] ] [ ; ]
```

```
SUBTITLE [ print-list [ CENTER ] ] [ ; ]
```

```
TITLE [ print-list [ CENTER ] ] [ ; ]
```

Clauses

```
AFTER CHANGE [ ON ] by-item PRINT print-list [ CENTER ]
```

```
{ ASCD } target-item
{ DESC }
```

```
{ report-item AS [ nonrepeatable-edit-descriptors ]
  { repeatable-edit-descriptors
  { report-item AS " [" [ decorations, ... ]
    { modifiers, ... ] "
    { repeatable-edit-descriptors "
  { report-item AS " [" [ decorations, ... ]
    { modifiers, ... ] "
    ( nonrepeatable-edit-descriptors
      { repeatable-edit-descriptors ) "
  }
```

report-item

is either a *by-item* or a *target-item*.

nonrepeatable-edit-descriptors

specify some general ways *report-items* are to be printed.
nonrepeatable-edit-descriptors should not be specified without a
repeatable-edit-descriptor. Valid values for *nonrepeatable-*
edit-descriptors are:

P multiplies value by 10^{**n} ; *n* is an integer.

S, SP, SS control plus (+) sign printing.

repeatable-edit-descriptors

specify data conversion to the Guardian Formatter for printing the *report-*
item values. Valid values for *repeatable-edit-descriptors* are:

A [*w*] for alphanumeric values

I*w* [.*m*] for integer values

F*w.d* [.*m*] for fixed-point values

M *mask* for a template to combine literals and values

where

w specifies the width of the report-item.

m specifies the number of digits that appear to the left of the
 decimal for fixed-point values and the minimum number of digits
 for integer values

d specifies the number of digits to the right of the decimal.

mask is a combination of the characters 9, Z, V, . (period) and literals.
 The combination must be enclosed within apostrophes (') or
 greater-than and less-than symbols (< >).

" ["*decorations*"] "

specify character strings that can be added to a *report-item* depending on a
 condition. The syntax is as follows:

conditions location char-string

where

conditions

are one or more of the following:

M add *char-string* if value is negative.

N add *char-string* if value is null.

- P add char-string if value is positive.
- Z add char-string if value is zero.
- O add char-string if overflow condition occurs.

location

is where the character string is to be printed:

- An* indicates *char-string* is to be printed at absolute position *n*.
- F* indicates *char-string* is to be inserted after the value is formatted. If *condition* is satisfied, *char-string* is printed immediately to the left of the item value.
- P* indicates *char-string* is inserted before the value is formatted. If *condition* is satisfied, *char-string* is printed to the right of the value.

char-string

is one or more alphanumeric characters enclosed within apostrophes (').

" [*modifiers*] "

alter the effect of the edit descriptors as follows:

- | | |
|-------------------------|------------------------------------------------------|
| BN, BZ | prints blanks for null or zero values, respectively. |
| FL <i>char</i> | specifies a substitute fill character. |
| OC <i>char</i> | respecifies the overflow character. |
| LJ, RJ | specifies right or left justification. |
| SS <i>pr-of-symbols</i> | allows substitution of symbols. |

where

- char* is an ASCII character enclosed in apostrophes.
- pr-of-symbols* is a special mask symbol (see *repeatable-edit-descriptors*) and a substitution character.

```
date-in-internal-format AS DATE { *
                                   { display-format }
```

```
time-in-internal-format AS TIME { *
                                   { display-format }
```

```
AT END PRINT print-list [ CENTER ]
```

```
AT START PRINT print-list [ CENTER ]
```

```
BEFORE CHANGE [ ON ] by-item PRINT print-list [ CENTER ]
```

```
{ BY
  { BY DESC } } by-item
```

```
{ { target-item } CENTER , }
  { by-item }
  {
    { CENTER ALL,
  }
```

```
target-item CUM [ OVER ALL
                  [ OVER by-item ]
```

```
FOOTING print-list [ CENTER ]
```

```
FORM [ number ]
```

```
{ by-item
  { target-item } } HEADING "heading-string"
```

```
INTERNAL internal-format
```

```
JULIAN-DATE ( year , month , day )
```

```
{ { target-item } NOHEAD }
  { by-item }
  {
    { NOHEAD ALL
  }
```



```
{ { target-item } NOPRINT }
{ { by-item } }
{ NOPRINT ALL }
```

The Option Variables and their valid values are as follows:

```
{ @BLANK-WHEN-ZERO }
{ @BREAK-KEY }
{ @CENTER-PAGE }
{ @HEADING } TO BBC { ON }
{ @STATS } { OFF }
{ @SUMMARY-ONLY }
{ @WARN }
```

```
{ @COPIES }
{ @COST-TOLERANCE }
{ @DISPLAY-COUNT }
{ @LINES }
{ @MARGIN }
{ @PAGES }
{ @PRIMARY-EXTENT-SIZE } TO number
{ @SECONDARY-EXTENT-SIZE }
{ @READS }
{ @SPACE }
{ @TARGET-RECORDS }
{ @VSPACE }
{ @WIDTH }
```

```
{ @DECIMAL }
{ @NEWLINE }
{ @NONPRINT-REPLACE } TO "character"
{ @OVERFLOW }
{ @UNDERLINE }
```

@SUBTOTAL-LABEL TO *string-literal*

```
{ @DATE-FORMAT } TO display-format
{ @TIME-FORMAT }
```

```
target-item PCT { OVER ALL }
{ OVER by-item }
```

SKIP [*number*]

SPACE [*number*]

SUBFOOTING *print-list* [CENTER]

SUBTITLE *print-list* [CENTER]

target-item SUBTOTAL [OVER *by-item*]

SUPPRESS [WHERE] *logical-expression*

The System Variables are as follows:

@DATE

@TIME

@LINENO

@PAGENO

TAB [*number*]

TIMESTAMP-DATE (*field-name*)

TIMESTAMP-TIME (*field-name*)

TITLE *print-list* [CENTER]

```
{ target-item } TOTAL
{ by-item }
```

WHERE *logical-expression*

Commands

```
?ASSIGN [ { record-name } [ , ] physical-file-name
          [ { generic-file-name } [ TO ]
```

```
          [ , create-open-spec ... ]
```

```
?ASSIGN record-name , { , create-open-spec } ...
```

```
?ATTACH [ process-name ]
```

?COMPILE *edit-filename* [(*section-name* , ...)]
TO *compiled-physical-filename*

?DICTIONARY [*dict-subvol-name*]

?EDIT [*edit-filename*]

?EXECUTE *compiled-physical-filename*

?EXIT

?HELP [*help-element*]

?OUT [*physical-filename*]

?RUN [*edit-filename* [(*section-name* , ...)]

?SECTION *section-name*

[OPEN]
[LINK]
[CONTROL]
?SHOW [LIMITS]
[ASSIGN [*record-name*]]
[*user-variable-name*]
[*record-name*]
[*param-name*]

?SOURCE *edit-filename* [(*section-name* , ...)]

Enform Plus Procedures

COBOL Procedures

```

ENTER ENFORMSTART USING  ctlblock                                !INT:ref
                        , compiled-physical-filename            !INT:ref
                        , buffer-length                        !INT:value
                        , error-number                        !INT:ref
                        [ , restart-flag ]                    !INT:value
                        [ , param-list ]                      !INT:ref
                        [ , assign-list ]                     !INT:ref
                        [ , process-name ]                    !INT:ref
                        [ , cpu ]                              !INT:value
                        [ , priority ]                        !INT:value
                        [ , timeout ]                          !INT32:value
                        [ , reserved-for-expansion ]          !INT:ref

ENTER ENFORMRECEIVE USING ctlblock, buffer [ GIVING count ]
                        !INT:ref    INT:ref                INT:function!

ENTER ENFORMFINISH USING ( ctlblock )                            !INT:ref

```

FORTRAN Functions

```

CALL ENFORMSTART ( ctlblock                                !INT:ref
                  , compiled-physical-filename            !INT:ref
                  , \ buffer-length \                    !INT:value
                  , error-number                        !INT:ref
                  [ , \ restart-flag \ ]                  !INT:value
                  [ , param-list ]                      !INT:ref
                  [ , assign-list ]                     !INT:ref
                  [ , process-name ]                    !INT:ref
                  [ , \ cpu \ ]                          !INT:value
                  [ , \ priority \ ]                      !INT:value
                  [ , \ timeout \ ]                      !INT32:value
                  [ , reserved-for-expansion ]          !INT:ref
                  , \ maskword \                          !INT:value

count ENFORMRECEIVE ( ctlblock, buffer )
!INT:function          INT:ref    INT:ref    !

CALL ENFORMFINISH ( ctlblock )                                !INT:ref

```

TAL Functions

```
CALL ENFORMSTART ( ctlblock                !INT:ref
                  , compiled-physical-filename !INT:ref
                  , buffer-length            !INT:value
                  , error-number            !INT:ref
                  [ , restart-flag ]         !INT:value
                  [ , param-list ]          !INT:ref
                  [ , assign-list ]         !INT:ref
                  [ , process-name ]        !INT:ref
                  [ , cpu ]                  !INT:value
                  [ , priority ]             !INT:value
                  [ , timeout ]             !INT32:value
                  [ , reserved-for-expansion ] !INT:ref
```

```
[ count := ] ENFORMRECEIVE ( ctlblock , buffer )
!INT:function                INT:ref    INT:ref !
```

```
ENFORMFINISH ( ctlblock )                !INT:ref
```


B Error Messages

This appendix documents the following types of messages:

- **!!! ERROR** *error-number* types: these indicate a serious error has occurred. Statement execution terminates. If this type of error occurs for a LIST or FIND statement, the query terminates.
- ***** WARNING** *warning-number* types: these point out an error that could change the expected results. The error does not abort the query although it could lead to more serious error conditions.
- ***** FILE ERROR ...** types: these indicate a serious error has occurred within the file system. If there is a file error with the run-time IN input file, the dictionary file, or the vocabulary file, the entire Enform Plus session is terminated.
- ***** ...** types: these occur during Enform Plus initialization. If this type of error occurs, Enform Plus terminates abnormally.
- **Enform Plus [QP] TRAP**: this means that either a hardware failure or an unexpected software error has occurred. Please save the information produced by this message and report the error to Tandem.
- ***** ERROR** types: these indicate that an error has occurred during execution of the BUILD MK utility. BUILD MK terminates abnormally. Correct the problem and rerun BUILD MK or you cannot use the key-sequenced version of the message table with Enform Plus.

Error messages are listed in the following order within this appendix.

1. Enform Plus initialization errors are listed in alphabetic order.
2. **!!! ERROR** and ***** WARNING** type errors are listed together in numeric order (where applicable) with the error message text and additional comments.
3. ***** FILE ERROR** type errors are described in alphabetic order.
4. Enform Plus TRAP messages are listed in alphabetic order.
5. BUILD MK error messages are listed. These messages consist of the following types of messages: **!!!ERROR** and **FILE ERROR** messages.

Enform Plus Initialization Messages

Current reserved word cannot be used to redefine another reserved word

A reserved word redefinition in the ?VOCABULARY section of the message table contains an old reserved word where a new word is expected. (The key-sequenced message table file was not built by the BUILDMDK utility or the file has been modified since it was built.)

Invalid DICTIONARY file name

The dictionary file name specified on the ENFORM command line is not a valid Guardian file name.

Invalid MESSAGE TABLE file name

The message table file name specified on the ENFORM command line is not a valid Guardian file name.

Message table does not contain a version number record.
Rebuild key-sequenced file

Either the key-sequenced message table file was not built by the BUILDMDK utility or the file has been modified since it was built.

Message table must be a disk file

Self-explanatory. Use the BUILDMDK utility to build the key-sequenced message table file.

Message table must be a key-sequenced file

Self-explanatory. Use the BUILDMDK utility to build the key-sequenced message table file.

Message table must contain both ?MESSAGES and ?HELP sections

Self-explanatory. Use the BUILDMDK utility to build the key-sequenced message table file.

Message table version number is not correct

The version number in the message table does not match the version number expected by Enform Plus. Rebuild the key-sequenced message table file using the appropriate version of BUILDMDK.

Primary key for message table file must be offset at 0 and have length 34

Self-explanatory. Use the BUILDMDK utility to build the key-sequenced message table file.

Sorry, you're not allowed to run ENFORM on this processor

This processor does not have the required Enform Plus microcode.

!!! Error and *** Warning Type Messages

!!! ERROR [26]

```
Invalid use of range item
```

A subscript range may be used as a *target-item* (but cannot be used when modified by a BY, BY DESC, ASCD, or DESC clause) only in a LIST or FIND statement. Its use is invalid in all other circumstances.

!!! ERROR [27]

```
Unknown ENFORM directive or syntactically incorrect
```

A command name has been misspelled or an attempt was made to execute a command from a different subsystem.

!!! ERROR [28]

```
The Boolean operators AND and OR cannot be used in a TITLE or  
PRINT statement expression
```

Only a simple logical expression may be used in an IF/THEN/ELSE expression within an AFTER CHANGE, AT END, AT START, BEFORE CHANGE, FOOTING, SUBFOOTING, SUBTITLE, or TITLE statement or clause.

!!! ERROR [29]

```
Reference has been attempted to an undefined or illegal item  
in PRINT statement
```

An item has been used in a Print List clause that does not appear in the LIST statement.

*** WARNING [30]

```
Name too long. Truncated to 31 characters
```

The variable or aggregate name must be less than 31 characters in length.

!!! ERROR [31]

```
Invalid file name
```

The file name printed with this error message is not a valid Guardian physical file name.

!!! ERROR [32]

```
Invalid SET-variable specification
```

The “@” symbol is not followed by a valid string for an option variable name.

!!! ERROR [33]

```
Name not found
```

A field name has been misspelled or the wrong dictionary is being used.

!!! ERROR [34]

```
Name not sufficiently qualified to avoid ambiguity
```

Field name appears in more than one opened record description. Use more qualification.

!!! ERROR [35]

```
Record description not found in dictionary
```

The record name has been misspelled or the wrong dictionary is being used.

!!! ERROR [36]

```
Symbol table overflow
```

The maximum available space for file descriptions, user defined variables, and so on, has been exceeded. All tables are cleared.

!!! ERROR [37]

```
Overflow encountered on input conversion of numeric-literal
```

Numeric literal exceeds 32767.

!!! ERROR [38]

```
?SOURCE file nesting > 4
```

You cannot have more than four levels of nested ?SOURCE commands.

!!! ERROR [39]

Too many references to user aggregates

The total number of references to user aggregates exceeds 32.

!!! ERROR [40]

Multiply defined name

The name already exists as a record name, user-defined item, or parameter name.

!!! ERROR [41]

The maximum target length of 2000 bytes was exceeded. Unable to process query

The maximum length of a LIST or FIND statement requiring sorting exceeded 2000 bytes, or the maximum length of a LIST or FIND statement without sorting requirements exceeded 4095 bytes.

!!! ERROR [42]

Expression too large to process

Expression must contain fewer than 512 items.

!!! ERROR [43]

The specified relation is invalid in the above context

CONTAINS, BEGINS WITH, and pattern match conditions require string arguments. The pattern match operation allows only EQ and NE operators.

!!! ERROR [44]

Too many actual file assignments

The table of assignments exceeds eight entries. Clear the table by entering ?ASSIGN without a physical file name.

!!! ERROR [45]

An integer literal is required in the above context

A number with decimal places is not allowed. It must be an integer.

!!! ERROR [46]

```
Too many LINKs
```

The number of links exceeds 32. Clear some links by using a DELINK statement.

***** WARNING [47]**

```
Source line was truncated
```

The line must be 255 characters or fewer.

!!! ERROR [48]

```
Only field names may appear in a qualification
```

The item name in the WHERE or SUPPRESS clause has not been defined or is misspelled. This condition is usually due to an internal error.

!!! ERROR [49]

```
User variable assignments are illegal in the scope of a FIND statement
```

A user-defined variable or table is not an acceptable output field name in a FIND statement.

!!! ERROR [50]

```
Insufficient memory available for data buffer (SERVER-related failure on name)
```

An Enform Plus server (process file) cannot be opened because there is no space for a message buffer.

***** WARNING [51]**

```
Null target list
```

The LIST or FIND statement is not processed.

!!! ERROR [52]

```
Not currently supported
```

The indicated feature or operation can not be used.

!!! ERROR [53]

Invalid subscript range specification

The subscript range specified ($[x:y]$) for an item is wrong. Subscripts must be numeric literals. The first number (x) must be smaller than the second number (y).

!!! ERROR [54]

Invalid AS format description

The display format for AS, AS DATE, or AS TIME is invalid or the INTERNAL format is invalid.

!!! ERROR [55]

A user aggregate may not be used in a user aggregate end-expression

Self-explanatory.

!!! ERROR [56]

An aggregate may not be used as the argument to another aggregate

Self-explanatory.

!!! ERROR [57]

Item type incompatible with use

Expecting a field or user-defined variable to subscript or invalid use of a condition in an Arithmetic Expression clause. Similar to a type mismatch.

!!! ERROR [58]

Illegal use of KEY item

Record-name.KEY or KEY OF *record-name* is not allowed in a LINK statement or in an Aggregate clause.

!!! ERROR [58]

Illegal use of KEY item (SERVER-related failure on name)

Record-name.KEY or KEY OF *record-name* is not allowed when the data for record-name is from an Enform Plus server (process file).

!!! ERROR [59]

Maximum read count exceeded

Enform Plus has read the limit number of records specified by the @READS Option Variable clause.

!!! ERROR [60]

A user aggregate declaration may not reference the value of another user aggregate

Self-explanatory.

!!! ERROR [61]

Initialization expression must be numeric

You attempted to initialize a user-defined aggregate with something other than a number or used a JULIAN-DATE clause within a SET statement that does not evaluate to a literal.

!!! ERROR [62]

Too many target items

The number of items or output fields within a LIST or FIND statement exceeds 400.

!!! ERROR [63]

Too many PRINT statements

The number of items in Print List clauses exceeds 172. Processing of the Enform Plus program is stopped and the contents of the internal table are reset to the values held at the start of processing the statement that produced the error.

!!! ERROR [64]

By-item not found

Either the grouped item was not defined in a BY or BY DESC clause or was misspelled.

!!! ERROR [65]

An aggregate may not be used in a print-list clause

Self-explanatory.

!!! ERROR [66]

Only one OPTIONAL LINK request allowed per LIST or FIND statement

Only one link (regardless of whether it is OPTIONAL) can be used when OPTIONAL is specified.

!!! ERROR [67]

Field type incompatibility

Data types being compared must both be numeric or alphabetic.

!!! ERROR [68]

Illegal LINK field

One of the following has occurred:

- You misspelled a qualified field name or attempted to use a subscripted field where one is not allowed.
- You attempted to use an Enform Plus server (process file) improperly in a LINK OPTIONAL statement. For example, in LINK A TO OPTIONAL B ..., A cannot be an Enform Plus server because of an implementation restriction.

!!! ERROR [69]

Invalid range

You incorrectly defined a range for a comparison pattern or THRU within a Logical Expression clause.

!!! ERROR [70]

Nonnumeric item in arithmetic expression

You used alphanumeric item in an Arithmetic Expression clause.

!!! ERROR [71]

The table containing literals, AS formats and headings has overflowed

The literal table overflowed its maximum size of 5,915 words.

!!! ERROR [72]

Invalid occurrence number

You attempted to subscript past the end of a table.

!!! ERROR [73]

Too many or too few parameters

You entered the wrong number of parameters for JULIAN-DATE, TIMESTAMP-TIME, or TIMESTAMP-DATE clauses.

!!! ERROR [74]

Too many PARAM declarations

The number of parameters for the current Enform Plus session exceeds 32. Clear some with the CLOSE statement.

!!! ERROR [75]

Invalid occurrence specification. Not in range [1,64]

A user-defined table cannot contain more than 64 elements.

!!! ERROR [76]

Variable subscript illegal in this context

The subscript used must be an explicit numeric literal, not a field name.

!!! ERROR [77]

A destination name must be specified

An item in a FIND statement must be assigned to an output field name, because the “name-correspondence” rules are insufficient here.

!!! ERROR [78]

The attribute UNIQUE may not be used with an OVER clause

UNIQUE can not be used with aggregates computed OVER a *grouped-item*.

!!! ERROR [79]

TAB 0, SKIP 0 or FORM 0 not defined

The number must be greater than zero.

***** WARNING [80]**

Section name not found

A section name was misspelled or there is no such section in the Edit-format file.

!!! ERROR [81]

The preceding text contains a syntactically incorrect element

Check the preceding line. If it is correct, check the next few preceding lines.

***** WARNING [82]**

Value is being truncated to one character

The value for the Option Variable must be a single ASCII character.

!!! ERROR [83]

The type of the argument in the SET clause is invalid

You have attempted to assign a string to a numeric entity, or a numeric value to a string entity, or to assign a noninteger numeric literal.

!!! ERROR [84]

Too many OVER clauses

The number of AFTER CHANGE, BEFORE CHANGE, TOTAL, SUBTOTAL, CUM, or PCT clauses in the LIST statement exceeds 64.

!!! ERROR [85]

More than one PCT or CUM modifies list item

Only one PCT or CUM clause is allowed per item.

!!! ERROR [86]

Server QP process has failed repeatedly

Either the primary or the backup process for the Enform Plus query processor has failed more than 10 times. The QP terminates abnormally when this condition occurs and must be restarted (preferably in another CPU).

***** WARNING [87]**

No RUN file has been named

Specify the name of an Edit-format file. No Edit-format file name has been specified in this session by a previous ?RUN or ?EDIT command.

!!! ERROR [88]

Illegal CHECKPOINT parameter

The primary process for the Enform Plus query processor executed a bad checkpoint call; probably an internal error. The QP terminates abnormally when this condition occurs and must be restarted.

!!! ERROR [89]

Too many expressions in target list

A LIST or FIND statement contains too many Arithmetic Expression or Logical Expression clauses.

!!! ERROR [90]

All field names referenced in a qualification aggregate must belong to the same record

All fields in the expression being aggregated, the *over-item*, and the embedded WHERE clause must belong to the same record.

***** WARNING [91]**

No report will be listed. The target list is composed of literals only

An Enform Plus report will not print alphanumeric or numeric literals only. Include at least one field name from an opened file description.

!!! ERROR [92]

At least one record has no LINK or a qualification relating it to any other record

You have entered query specifications that reference two or more record descriptions. At least one of these record descriptions has no relationship (link) to any other record description in the query. Enform Plus will not execute your query because this could produce an undesirable cross-product. Check your query specifications and add the necessary LINK statements or a WHERE clause.

!!! ERROR [93]

A user aggregate having an end-expression may not be used in this context

A user aggregate declared with an *end-expression* cannot be used as a qualification aggregate OVER a *grouped-item*.

!!! ERROR [94]

Your dictionary is bad

See the *Data Definition Language (DDL) Reference Manual*.

!!! ERROR [95]

Missing dictionary

Wrong subvolume or the specified dictionary does not exist.

!!! ERROR [96]

Invalid dictionary subvolume name

An internal error has occurred.

!!! ERROR [98]

Insufficient memory available to OPEN record description

An internal error has occurred.

!!! ERROR [99]

Multiply defined SECTION name

A section name can appear only once in a ?COMPILE, ?RUN, or ?SOURCE command.

!!! ERROR [100]

Undefined SET variable

A user-defined item or parameter used in a SET statement has not been defined yet.

***** WARNING [101]**

The param table would overflow if updated to the SET value

Parameter table is full and the last value has not been added. Use the CLOSE statement to clear parameter values not needed.

!!! ERROR [102]

Field referenced in TITLE statement not found in target list

Usually an internal error with some unsupported item within a AFTER CHANGE, AT END, AT START, BEFORE CHANGE, FOOTING, SUBFOOTING, SUBTITLE, or TITLE statement or clause.

!!! ERROR [103]

Invalid ENFORM version. Recompile to execute

The compiled physical file was compiled by a version of Enform or Enform Plus that is not compatible with the current Enform Plus version. Compile it again.

!!! ERROR [104]

```
Output line would exceed buffer space
```

Divide the output line in the LIST statement using SKIP or FORM clauses.

!!! ERROR [105]

```
SUBTOTAL, TOTAL, CUM, and PCT only modify numeric items
```

Cannot use alphanumeric string items here. Numeric strings are allowed.

!!! ERROR [106]

```
Field or expression must be numeric
```

Self-explanatory.

!!! ERROR [107]

```
Insufficient memory to build query processor representation  
of your query
```

Reduce the size of the requested Enform Plus query.

!!! ERROR [108]

```
An aggregate may not be used in a SUPPRESS clause
```

Self-explanatory.

!!! ERROR [109]

```
An aggregate may not be used as a parameter to a function
```

Self-explanatory.

!!! ERROR [110]

```
Insufficient memory available to produce the report
```

Try running Enform Plus with a MEM greater than 52 (see the *Guardian Programmer's Guide*).

!!! ERROR [112]

```
Illegal dictionary description (SERVER-related failure on name)
```

The dictionary description for an Enform Plus server (process file) must specify a file type of UNSTRUCTURED or no file type at all.

!!! ERROR [113]

```
An aggregate may not be used in this context with PCT
```

Only the aggregates SUM and COUNT can be used with PCT and they must be used alone (not in an expression).

!!! ERROR [114]

```
Incorrect reply length (SERVER-related failure on name)
```

The Enform Plus server (process file) returned a reply with an unexpected length to the query processor. One way to get this error is to specify an odd data record byte size.

!!! ERROR [115]

```
Dictionary is outdated.  Recompile with the current DDL
```

Current dictionary has an old version number; recompile it with a new version of DDL.

!!! ERROR [123]

```
Physical file type does not match DDL
```

The file type (key-sequenced, relative, entry-sequenced, or unstructured) given in the DDL record description does not match the type of the physical file read by the query processor.

!!! ERROR [133]

```
Invalid dictionary specification
```

The dictionary name specified in either a DICTONARY statement or a ?DICTIONARY command is not a valid file name.

!!! ERROR [136]

Page count exceeded

Your report has produced more pages than the number specified for the @PAGES option variable.

!!! ERROR [137]

Invalid date specified

The data being formatted with an AS DATE clause is not a valid date.

!!! ERROR [138]

Invalid time specified

The data being formatted with an AS TIME clause is not a valid time.

!!! ERROR [143]

Data type not supported

A field within a DDL record description is described with a data type that is not supported by Enform Plus. (For example, Enform Plus does not support COMPLEX, REAL, or LOGICAL data types.)

!!! ERROR [166]

String literal must be terminated with a quotation mark

The closing quotation mark is missing. Remember that a string literal cannot be continued from one source line to the next.

!!! ERROR [167]

String literal cannot contain more than 127 characters

Self-explanatory.

!!! ERROR [168]

TOTAL may not be specified OVER a BY item

TOTAL can be specified only as OVER ALL. If you want to compute a total over a BY item, specify SUBTOTAL instead.

!!! ERROR [169]

```
?RUN command is ignored unless entered interactively
```

The ?RUN command must be typed in at the terminal.

!!! ERROR [170]

```
Illegal value for this option variable
```

Check the syntax of the [Option Variable Clauses](#) on page 5-48 for the values allowed.

!!! ERROR [172]

```
Item on left side of assign operator must be a field in the  
FIND record
```

The *output-field-name* in a FIND statement cannot be a field from an input record or the name of the FIND record itself.

!!! ERROR [173]

```
Value must be a single ASCII character, not "^" or "-"
```

This is a restriction on the value for the Option Variable @NEWLINE.

!!! ERROR [174]

```
Value is being truncated to 15 characters
```

The value for this Option Variable must be a string literal containing 15 characters or fewer.

!!! ERROR [175]

```
A subscript range cannot be used in a field in a FIND  
statement
```

Specify each item in the range individually.

!!! ERROR [176]

```
Help item phrase must be less than 32 characters long
```

The phrase following the ?HELP keyword must be less than 32 characters long, including embedded blanks and the initial question mark (if present).

!!! ERROR [177]

```
Parameter is treated like a literal here.  Its value cannot  
be changed
```

In certain cases, Enform Plus treats a parameter exactly like a numeric literal. This means that you cannot change the value of the parameter at execution time, either with a TACL PARAM command or an Enform Plus SET statement. See the [PARAM Statement](#) on page 4-39 for more details.

!!! ERROR [178]

```
Record on right side of link optional is linked back to  
record on left
```

Within your query specifications, a link exists that illegally links the record description specified on the right side of a LINK OPTIONAL statement back to the record description specified on the left side of the LINK OPTIONAL statement. See [Section 4, Statements](#) and [Appendix C, Links and the LINK OPTIONAL Statement Rules](#) for more information about links and the LINK OPTIONAL statement.

!!! ERROR [179]

```
Record appears on the right side of more than one link  
optional
```

Your query specifications contain one or more invalid links that violate rule 2 for the LINK OPTIONAL statement. (Rule 2 states that a record description can appear only once on the right side of a LINK OPTIONAL statement.) See [Section 4, Statements](#) and [Appendix C, Links and the LINK OPTIONAL Statement Rules](#) for more information about links and the LINK OPTIONAL statement.

!!! ERROR

```
(Attempt to divide by zero)
```

An error occurred because an attempt was made to divide by a field containing a data value of zero.

!!! ERROR

```
(Cost tolerance exceeded) : required cost = n
```

The strategy that the query processor will need to use to execute the query is greater than the value (*n*) you specified for the @COST-TOLERANCE option variable.

!!! ERROR

(Messages not in expected order)

The query processor received an unexpected message from the query compiler/report writer. If the query processor is running as a server in NonStop mode, this message might indicate failure of the primary process. Otherwise, this message indicates an error in the Enform Plus software.

!!! ERROR

(SORT failure) *sort-error-number*
[*** FILE ERROR #*file-error-number*] [on *name*]

An error occurred in the SORT process. For an explanation of *sort-error-number*, see the *FastSort Manual*.

!!! ERROR

Invalid response from Query Processor

The query compiler/report writer process has received an invalid response from the query processor. This message usually indicates an error in the Enform Plus software. Inform your system manager.

*** File Error Type Messages

Enform Plus reports file management errors with *** FILE ERROR . . . messages. In the following messages, *#file-error-number* is a Guardian file management error number; *name* is the physical file name.

*** FILE ERROR

(Abnormal termination of Query Processor)

Self-explanatory.

*** FILE ERROR

(Communication with Query Processor failed)
#file-error-number on *name*

Enform Plus lost communication with the query processor.

*** FILE ERROR

(CONTROL failure) *#file-error-number* on *name*

A control failure occurred on the physical file named.

*** FILE ERROR

(CREATE failure) *#file-error-number* on *name*

There was a problem creating the physical file.

*** FILE ERROR

(Dictionary file access failure) *#file-error-number* on *name*

See the *Data Definition Language (DDL) Reference Manual*.

*** FILE ERROR

(Illegal ENFORM execution file) on *name*

The file must be a compiled query file created with the ?COMPILE command.

***** FILE ERROR**

(Illegal list device) on *name*

The listing device name is misspelled or the device does not exist.

***** FILE ERROR**

(Illegal input device) on *name*

The input file name or Edit-format file name is misspelled or the file does not exist.

***** FILE ERROR**

(Not an Edit file) on *name*

The file named is not an Edit-format file.

***** FILE ERROR**

(OPEN failure) #*file-error-number* on *name*

There was a problem opening the physical file.

***** FILE ERROR**

(POSITION failure) #*file-error-number* on *name*

A position failure occurred on the physical file named.

***** FILE ERROR**

(Process nonexistent, insufficient system resources or full queue) #*file-error-number* on *name*

The server query processor named in the ?ATTACH command does not exist or cannot accept more users.

***** FILE ERROR**

(PURGE failure) on *name*

There was a problem purging the physical file.

***** FILE ERROR**

```
(READ failure) #file-error-number on name
```

Enform Plus could not read the physical file named.

***** FILE ERROR**

```
(RENAME failure) on name
```

There was a problem renaming the physical file.

***** FILE ERROR**

```
(SERVER-related failure) # number on name
```

There was a failure related to the use of an Enform Plus server (process file). The number and name are optional values supplied by the server instead of a standard Guardian file error and file name.

***** FILE ERROR**

```
(SETMODE failure) #file-error-number on name
```

A SETMODE error occurred on the physical file named.

***** FILE ERROR**

```
(Specified ENFORM compile file exists as edit or TAL object  
file) on name
```

The file must be a compiled query file created with an Enform Plus ?COMPILE command.

***** FILE ERROR**

```
(Unable to open ENFORM message table) #file-error-number on  
name
```

Self-explanatory. Enform Plus terminates abnormally. Correct the problem with the message table and restart the session.

***** FILE ERROR**

```
(Unable to position ENFORM message table) #file-error-number
on name
```

Enform Plus is unable to use the message table file. The session continues, but all messages contain “???” instead of text.

***** FILE ERROR**

```
(Unable to read ENFORM message table) #file-error-number on
name
```

Enform Plus is unable to use the message table file. The session continues, but all messages contain “???” instead of text.

***** FILE ERROR**

```
(WRITE failure) #file-error-number on name
```

A write error occurred on the physical file named. If error number 45 occurs on the target file (*name* appears as #*nnnn*), the target file has overflowed at least twice. See Section 5 in the *Enform User's Guide* for information about controlling the size of the target file.

Enform Plus Trap Messages

```
ENFORM TRAP: nnn S: xxxxxx P: xxxxxx E: xxxxxx L: xxxxxx
```

The Enform Plus Compiler/Report Writer process has failed. *nnn* is the trap number as described in the *Guardian Programmer's Guide*. *xxxxxx* are values in the hardware registers.

```
ENFORM QP TRAP: nnn S:xxxxxx P: xxxxxx E: xxxxxx L: xxxxxx
```

The Enform Plus Query Processor process has failed. *nnn* is the trap number as described in the *Guardian Programmer's Guide*. *xxxxxx* are values in the hardware registers.

BUILDMK Error Messages

*** ERROR

```
Key-sequenced file must be empty
```

The key-sequenced file that is to contain the message table must be empty before you run BUILDMK.

*** ERROR

```
Second parameter in command line must be a key-sequenced file  
name
```

The second parameter of the BUILDMK command must be the name of a key-sequenced file.

*** ERROR

```
Primary key for key-sequenced file must be at offset 0 and  
have length 34
```

Self-explanatory.

**** ERROR**

```
Edit file contains ?HELP section but no ?MESSAGES section
```

The Edit-format file version of the message table must contain a ?MESSAGES section if a ?HELP section is included.

**** ERROR**

```
Edit file contains ?MESSAGES section but no ?HELP section
```

The Edit-format file version of the message table must contain a ?HELP section if a ?MESSAGES section is included.

***** ERROR**

```
Identifier contains an illegal character
```

An identifier specified in the Edit-format file version of the message table contains an invalid character.

***** ERROR**

```
Identifier must begin with an alphabetic character or ^
```

An identifier specified in the Edit-format file version of the message table must begin with either an alphabetic character or a circumflex.

***** ERROR**

```
Identifier must not end with a hyphen
```

An identifier specified in the Edit-format file version of the message table must not end with a hyphen.

***** ERROR**

```
?HELP subsection must contain at least one subsection
```

Self-explanatory.

***** ERROR**

```
?MESSAGES section must contain at least one line of text
```

Self-explanatory.

***** ERROR**

?VOCABULARY section must redefine at least one reserved word

Self-explanatory.

***** ERROR**

Identifier must contain less than 32 characters

Self-explanatory.

***** ERROR**

In a reserved word redefinition, the old reserved word is missing

Supply the old reserved word.

***** FILE ERROR**

(EDITREAD read error) on *name*

The indicated error occurred on the specified file during the execution of BUILDMK.

***** FILE ERROR**

(EDITREAD sequence error) on *name*

The indicated error occurred on the specified file during the execution of BUILDMK.

***** FILE ERROR**

(EDITREAD text file format error) on *name*

The indicated error occurred on the specified file during the execution of BUILDMK.

***** FILE ERROR**

(EDITREADINIT I/O error) on *name*

The indicated error occurred on the specified file during the execution of BUILDMK.

***** FILE ERROR**

(OPEN failure) #*file-error-number* on *name*

The indicated file error occurred on *name* during the execution of BUILDMK.

***** FILE ERROR**

(POSITION failure) #*file-error-number* on *name*

The indicated file error occurred on *name* during the execution of BUILDMK.

***** FILE ERROR**

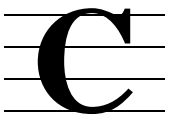
(READ failure) #*file-error-number* on *name*

The indicated file error occurred on *name* during the execution of BUILDMK.

***** FILE ERROR**

(WRITE failure) #*file-error-number* on *name*

The indicated file error occurred on *name* during the execution of BUILDMK.



Links and the LINK OPTIONAL Statement Rules

To fully understand the rules for the LINK OPTIONAL statement, you must first understand how Enform Plus defines links. This appendix describes how Enform Plus defines a link, reviews the rules for the LINK OPTIONAL statement, and provides some examples of invalid links.

How Enform Plus Defines a LINK

Enform Plus recognizes links that are established in these ways:

- by a LINK statement that references both record descriptions
- by a LINK OPTIONAL statement that references both record descriptions
- by a WHERE clause containing a term that references both record descriptions
- through the transitive property of links.

The following paragraphs discuss the characteristics of these links.

Links Initiated by a LINK Statement


A link initiated by a LINK statement is a two-directional link. Therefore, a two-way dependency condition exists between the record occurrences associated with the linked record descriptions. For example, if you specify:

```
LINK apple TO orange VIA seeds;
```

the following is true of the logical records built for the link:

- A given apple record occurrence appears in the logical records only if its linking field value matches the linking field value in an orange record occurrence. Thus, an apple record occurrence depends upon an orange record occurrence.
- A given orange record occurrence appears in the logical records only if its linking field value matches the linking field value in an apple record occurrence. Thus, an orange record occurrence depends upon an apple record occurrence.

A two-directional link established for a LINK statement can be sketched as follows:

```
record-description-1  record-description-2
```

where the two-headed arrow indicates the two-way dependency condition.

Links Initiated by a LINK OPTIONAL Statement

A link initiated by a LINK OPTIONAL statement is a one-directional link. Therefore, a one-way dependency condition exists where the record occurrences associated with the record description on the right of the statement depend upon the record occurrences associated with the record description on the left of the statement. For example, if you specify:

```
LINK apple TO OPTIONAL orange VIA seeds;
```

the following is true of the logical records built for the link:

- A given apple record occurrence appears in the logical records even if its linking field value does not match a linking field value in an orange record occurrence. Thus, an apple record occurrence does not depend on an orange record occurrence.
- A given orange record occurrence appears in the logical records only if its linking field value matches the linking field value of an apple record occurrence. Thus, an orange record occurrence depends upon an apple record occurrence.

A one-directional link established by the LINK OPTIONAL statement can be sketched as follows:

```
record-description-1 —○→ record-description-2
```

where the arrowhead points at the dependent record description.

Links Initiated by a WHERE Clause

A link initiated by a WHERE clause is a two-directional link. Before you can fully understand when a WHERE clause establishes a link, you must first understand what Enform Plus does when a WHERE clause appears in a query.

How Enform Plus Handles a WHERE Clause

When you use a WHERE clause, Enform Plus converts the WHERE clause into conjunctive normal form. This means that Enform Plus takes the logical expression in the WHERE clause and converts it into one or more “terms.” Each term consists of a logical expression that is connected to another logical expression by the Boolean operator AND. For example, consider the WHERE clause in the following query:

```
OPEN fruit, peach;
LIST BY fruit.color, peach.variety;
  WHERE fruit.fuzz = peach.fuzz;
```

The logical expression in this WHERE clause is already in conjunctive normal form. The single term in this WHERE clause is:

```
fruit.fuzz = peach.fuzz —————▶ one term
```

Each term in a WHERE clause can contain subterms connected by the Boolean operator OR. For example, consider the WHERE clause in the following query:

```
OPEN fruit, peach;
LIST ...
  WHERE fruit.fuzz = peach.fuzz
         OR fruit.color = "ORANGE";
```

The logical expression in this WHERE clause is also in conjunctive normal form. The single term in this WHERE clause is:

```
fruit.fuzz = peach.fuzz OR fruit.color = "ORANGE"  —one term
```

Note that what appears as a term in your WHERE clause might not be a term in a converted WHERE clause. Consider the WHERE clause in the following query:

```
OPEN fruit, orange, peach;
LIST fruit.color, orange.variety, peach.variety
  WHERE (fruit.season = "summer" AND fruit.color
         = orange.color) OR (fruit.season <> "spring" AND
         fruit.fuzz = peach.fuzz);
```

Enform Plus converts this WHERE clause into conjunctive normal form as follows:

```
fruit.season = "summer" OR fruit.season <> "spring"
AND
fruit.season = "summer" OR fruit.fuzz = peach.fuzz
AND
fruit.color = orange.color OR fruit.season <> "spring"
AND
fruit.color = orange.color OR fruit.fuzz = peach.fuzz
```

When a WHERE Clause Establishes a Link

A WHERE clause establishes a link when an AND term in the converted WHERE clause references two or more record descriptions. For example, consider the WHERE clause in the following query:

```
OPEN apple, fruit;
LIST ...
  WHERE fruit.color = apple.color;
```

The converted form of this WHERE clause contains a single term:

```
fruit.color = apple.color;
```

Because this term references two record descriptions (fruit and apple), the preceding WHERE clause establishes a two-directional link between fruit and apple.

The WHERE clause in the following query does not establish a link:

```
OPEN apple, banana;
LIST ...
  WHERE apple.color = "RED"
        AND banana.color = "YELLOW";
```

The converted form of this WHERE clause contains two terms:

```
apple.color = "RED" —————> first term
AND banana.color = "YELLOW" —————> second term
```

Because neither term references two record descriptions, the WHERE clause does not establish a link.

Comparison of the WHERE Clause and the LINK Statement

A WHERE clause differs from a LINK statement as follows:

- A WHERE clause can link more than two record descriptions. A single LINK statement can link only two record descriptions.
- A WHERE clause can indicate that the record descriptions are to be linked where the values of linking fields are not equal. A LINK statement indicates that the values of the linking fields must be equal.
- You cannot specify OPTIONAL in a WHERE clause.

Consider the WHERE clause in the following query:

```
OPEN apple, orange, banana;
LIST ...
  WHERE apple.seeds = orange.seeds
        OR orange.skin <> banana.skin;
```

The converted WHERE clause contains one term:

```
apple.seeds = orange.seeds OR orange.skin <> banana.skin
```

Because this term references three record descriptions (apple, orange, and banana), the preceding WHERE clause establishes three two-directional links between apple, orange, and banana.

Like a LINK statement, a WHERE clause establishes a multiple dependency condition between the linked record descriptions. A two-directional link established by a WHERE clause can be sketched as:

```
record-description-1 <-----> record-description-2
```

where the arrowheads point at the dependent record descriptions.

Links Due to the Transitive Property of Links

The transitive property of links applies to both two-directional and one-directional links. For two-directional links, the transitive property is defined as:

If A is linked to X and B is linked to X, then A is linked to B and B is linked to A.

For one-directional links, A is linked to B if:

A is optionally linked to X and X is linked to B.

A is linked to X and X is optionally linked to B.

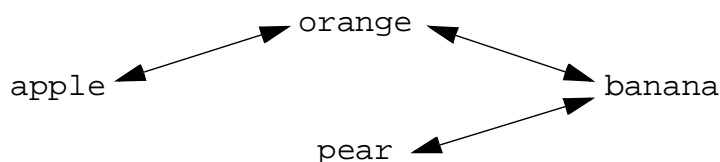
A is optionally linked to X and X is optionally linked to B.

B is not, however, linked to A.

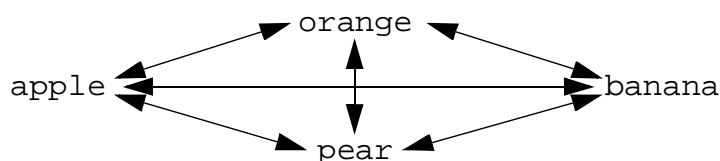
Consider the two-directional links in the following query specifications:

```
OPEN apple, orange, banana, pear;
LINK apple TO orange VIA seeds;
LINK orange TO banana VIA skin;
LINK banana TO pear VIA skin;
LIST ...
WHERE banana.color = pear.color;
```

A sketch of the explicit links in these specifications appears as follows:



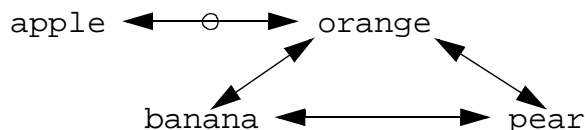
The following sketch shows the preceding explicit links and the implicit links that exist due to the transitive property of links:



The following query specifications contain both one-directional links and two-directional links:

```
OPEN apple, orange, banana, pear;
LINK apple TO OPTIONAL orange VIA seeds;
LINK orange TO banana VIA skins;
LIST ...
WHERE banana.color = pear.color;
```

Under the transitive property of links, apple is linked optionally to orange, banana, and pear. The orange, banana, and pear are linked to each other. These links could be symbolically represented as:



Note that apple is linked to orange, banana, and pear; but orange, banana, and pear are not linked to apple.

Review of Rules for the LINK OPTIONAL Statement

Enform Plus uses the following rules to determine the validity of links when your query specifications contain LINK OPTIONAL statements.

1. A record description on the right side of a LINK OPTIONAL statement must not be linked back to the record description on the left side of the same LINK OPTIONAL statement.

Enform Plus requires you to adhere to this rule because a LINK OPTIONAL statement is a one-directional link. When you adhere to this rule, Enform Plus has a starting place from which to begin building the logical record occurrences. If Enform Plus did not require adherence to this rule, the starting place could be ambiguous due to the transitive property of links.

2. A given record description must not appear on the right side of more than one LINK OPTIONAL statement. A record description appears on the right side of more than one LINK OPTIONAL statement when you:
 - Specify the same record description on the right side of two or more LINK OPTIONAL statements.
 - Specify a two-directional link between record descriptions that appear on the right sides of two separate LINK OPTIONAL statements. (Note that a two-directional link caused by the transitive property of links is also invalid.)

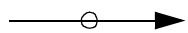
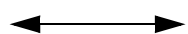
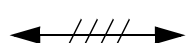
Enform Plus requires adherence to this rule because the query processor cannot support links that violate this rule.

When checking your queries for violations of these rules, Enform Plus ignores any two-directional links that reference the same two record descriptions referenced by a LINK OPTIONAL statement. However, after Enform Plus determines that all links are valid, it uses all links when evaluating the rest of your query.

The following paragraphs describe links that violate these rules. These paragraphs also describe some valid links that at first glance appear to violate these rules.

Examples of Invalid Links

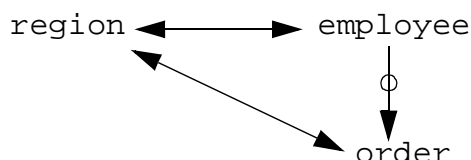
The following examples involve query specifications that use the sample data base shown in the *Enform User's Guide*. The purpose of these examples is to show invalid links, not to show useful queries. Each example includes a sketch of its links. The following symbols are used for the sketches:

-  Represents a one-directional link initiated by a LINK OPTIONAL statement.
-  Represents a two-directional link initiated by either a LINK statement or a WHERE clause.
-  Represents a two-directional link canceled by a LINK OPTIONAL statement.

The links in the following query specifications are invalid because they violate rule 1:

```
LINK region TO employee VIA regnum;
LINK employee.empnum TO OPTIONAL order.salesman;
LINK order TO region VIA regnum;
LIST regname, empname, ordernum;
```

A sketch of these links appears as follows:

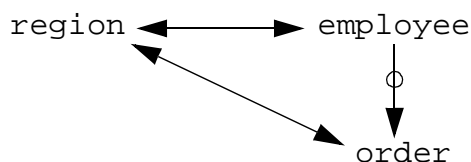


In the preceding example, order (the record description on the right side of the LINK OPTIONAL statement) is linked back to employee by the transitive property of links.

The query specifications in the following example violate rule 1. These query specifications illustrate the fact that Enform Plus evaluates a link established by a WHERE clause in the same manner that it evaluates a link established by a LINK statement:

```
LINK region TO employee VIA regnum;
LINK employee.empnum TO OPTIONAL order.salesman;
LIST regname, empname, ordernum
  WHERE order.regnum = region.regnum;
```

A sketch of these links appears as follows:

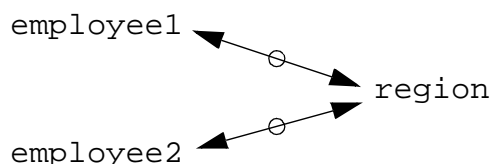


The preceding query specifications are effectively the same as the first query specifications. These specifications contain invalid links because the WHERE clause links order back to employee due to the transitive property of links.

The following query specifications violate rule 2:

```
OPEN employee, region;
OPEN employee1 AS COPY OF employee;
OPEN employee2 AS COPY OF employee;
LINK employee1 TO OPTIONAL region VIA regnum;
LINK employee2 TO OPTIONAL region VIA regnum;
LIST employee1.empname, employee.empname, regname;
```

A sketch of these links appears as follows:

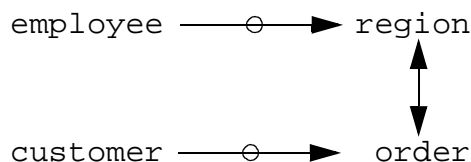


The preceding query specifications violate rule 2 because region appears on the right side of more than one LINK OPTIONAL statement.

The following query specifications violate rule 2 because both order and region appear on the right side of two LINK OPTIONAL statements due to the transitive properties of links:

```
OPEN employee, region, customer, order;
LINK employee TO OPTIONAL region VIA regnum;
LINK customer TO OPTIONAL order VIA custnum;
LINK order.salesman TO region.manager;
LIST empname, regname, custname;
```

Symbolically, these links appear as follows:



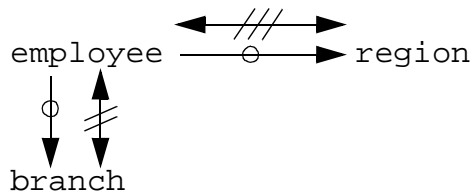
The links in the preceding query are invalid because the LINK statement specifies a two-directional link between two record descriptions that appear on the right side of LINK OPTIONAL statements.

The following query specifications, which attempt to generate a “double” exception report, violate rule 2:

```
OPEN employee, region, branch;
LINK employee TO OPTIONAL region VIA regnum;
LINK employee TO OPTIONAL branch VIA branchnum;
LIST empname, employee.regnum, employee.branchnum
```

```
WHERE employee.regnum <> region.regnum OR
       employee.branchnum <> branch.branchnum;
```

A sketch of these links appears as follows:

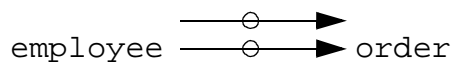


The links in this query are invalid because both `region` and `branch` appear on the right side of more than one LINK OPTIONAL statement. The WHERE clause establishes three two-directional links between `employee`, `region`, and `branch`. Although the LINK OPTIONAL statements cancel two of the three two-directional links (between `region` and `employee` and between `branch` and `employee`), the link between `region` and `branch` still exists. This link causes both `region` and `branch` to appear more than once on the right side of a LINK OPTIONAL statement.

The following query specifications violate rule 2:

```
LINK employee TO OPTIONAL order VIA regnum;
LINK employee.empnum TO OPTIONAL order.salesman;
LIST empname, ordernum;
```

A sketch of these links appears as follows:



These specifications attempt to optionally link `employee` to `order` through two different linking fields. Even though different linking fields are specified, `order` still appears on the right side of two LINK OPTIONAL statements.

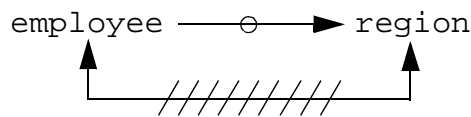
Examples of Valid Links

The following examples contain valid links that might appear to be invalid. These examples also include a sketch of their links. The symbols used for the sketches are the same as those used for the invalid link examples.

The following query specifications contain a LINK OPTIONAL statement and a WHERE clause:

```
LINK employee TO OPTIONAL region VIA regnum;
LIST empname, employee.regnum
  WHERE employee.regnum <> region.regnum;
```

A sketch of these links appears as follows:



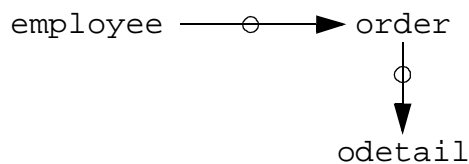
The link initiated by the WHERE clause might appear to violate rule 1. Remember, however, that when Enform Plus checks the validity of a link, a LINK OPTIONAL statement “cancels” any two-directional link between the same record descriptions. Enform Plus does use this two-directional link when evaluating the query.

The following query specifications contain two LINK OPTIONAL statements:

```

OPEN employee, order, odetail;
LINK employee.empnum TO OPTIONAL order.salesman;
LINK order TO OPTIONAL odetail VIA ordernum;
LIST empname, order.ordernum, partnum, quantity;
  
```

A sketch of these links appears as follows:



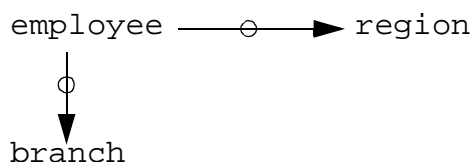
Initially, you might assume that these links violate rule 2 because odetail appears on the right side of the first as well as the second LINK OPTIONAL statement, but this is not the case. Rule 2 states that a record description appears on the right side of more than one LINK OPTIONAL statement only if a two-directional link exists between it and another record description on the right side of a LINK OPTIONAL statement. Because the one-directional LINK OPTIONAL statement links order and odetail, odetail appears only on the right side of the second LINK OPTIONAL statement.

The following query specifications show a valid method of generating a “double” exception report:

```

OPEN employee, region, jobs;
LINK employee TO OPTIONAL region VIA regnum;
LINK employee TO OPTIONAL branch VIA branchnum;
LIST empname, employee.regnum, employee.branchnum
    SUPPRESS WHERE employee.regnum = region.regnum OR
                    employee.branchnum = branch.branchnum;
  
```

A sketch of these links appears as follows:



The LINK OPTIONAL statements establish the only links for this query. Although the SUPPRESS clause contains a WHERE phrase, this WHERE phrase does not establish a link. (If the WHERE phrase did establish a link, the links in this query would be invalid.)

Comparison of the LINK Statement, the LINK OPTIONAL Statement, and the WHERE Clause

[Table C-1](#) is a truth table that compares the LINK statement, the LINK OPTIONAL statement, and the WHERE clause.

Table C-1. Truth Table

	LINK Statement	LINK OPTIONAL Statement	WHERE Clause
Establishes a session-wide link.	True	True	False
Establishes a two-directional link.	True	False	True
Can link more than two record descriptions.	False	False	True
The order in which you specify record description names is unimportant.	False	True	True sometimes (the order is not important if you link record descriptions by specifying that their linking field values must be equal)
Builds logical records from matching records in the data files associated with all linked record descriptions.	True	False	True
Builds logical records by preserving data from the data file associated with the record description name on the left side.	False	True	False

Glossary

aggregate. A cumulative operation on set(s) of numbers, producing a single value per set. See also [predefined aggregate](#). and [user aggregate](#).

by-item. The field name used to group and sort Enform Plus output; always associated with a BY or BY DESC clause. A by-item is a special kind of target-item.

clause. Component of an Enform Plus statement.

command. A directive to the Enform Plus compiler.

compiler/report writer. The Enform Plus process that both compiles Enform Plus queries and formats and writes Enform Plus reports.

compiled query file. The physical file containing a saved query that has been compiled by the ?COMPILE command.

current output listing file. The file to which Enform Plus directs output; this file can change during an Enform Plus session.

database. A set of related files defined in a dictionary.

Data Definition Language (DDL). The language used to describe the record and file structure of a database.

dictionary. A database of file descriptions and record types created by the Data Definition Language (DDL); also called a data dictionary.

default output file. The file to which Enform Plus directs output at the beginning of an Enform Plus session. See also [current output listing file](#).

default input file. The file from which the Enform Plus source code is entered when the IN option of the Enform Plus command is omitted; usually the home terminal.

elementary field. The smallest named unit of a record.

field. Either an elementary field or group field.

field name. Name given to a field in a DDL RECORD statement.

field value. Value of a specific field within a specific stored record.

file. A collection of similarly structured records.

file name. The name of a physical file.

file type. Identifies the organization of the physical file, such as a key-sequenced, entry-sequenced, or relative file.

format 1 file. A file created under software releases earlier than G06.00/D46.00, or a file created under release G06.00, D46.00, or later, but with a size that does not exceed two gigabytes (less one megabyte).

format 2 file. A large-format file, or a file that can contain larger partitions than a file created under software releases preceding G06.00 or D46.00. A format 2 file has the potential of exceeding the 2 GB (less 1 MB) size limit of a format 1 file.

front end. The Enform Plus process that compiles Enform Plus programs and prints reports. See also [compiler/report writer](#).

generic file. A file used to store some class of Enform Plus output.

group field. A collection of one or more fields that can be accessed with a single name.

group name. Name of one or more fields that can be accessed with a single name.

home terminal. The terminal from which the ENFORM command is entered.

link. Specifies a relationship between records in a relational database to be used in an Enform Plus query.

literal. One or more numeric or alphanumeric characters. See also [string literal](#), and [numeric literal](#).

logical expression. An expression that returns a true or false value.

normalized. Data that has been described in such a manner that only one value exists for every field position in a record.

numeric literal. Composed of the digits 0 through 9. Numeric literals cannot be larger than 32765 and must be enclosed in parentheses unless they appear in a logical expression or a TAB, SPACE, SKIP, or FORM clause.

option variable. An Enform Plus supplied variable that defines certain operational values.

OUT file. The physical device specified in the OUT option of the ENFORM command.

physical file name. Guardian file name in the form `\system.$volume.subvol.file-name`.

predefined aggregate. One of the Enform Plus aggregates: AVG, COUNT, MAX, MIN, or SUM.

primary key. The field or group of fields that uniquely identifies a record.

program. A sequence of related Enform Plus commands and statements.

qualification aggregate. An aggregate that appears in a request-qualification. See also [aggregate](#).

qualified field name. A name that uniquely identifies a field as a component of a record description.

query. A complete Enform Plus LIST or FIND statement specifying which fields and records to retrieve.

query processor (QP). The Enform Plus process that opens the files and retrieves the records from a relational database for a report or a new file.

record. A related set of field values.

record description. The dictionary description of a record, including the record name, record type, field names and data types, and key definitions.

record name. Name given a record description in a DDL RECORD statement.

record type. A record's structure, including field names and data types.

relational database. A database in which records are related through fields with common formats and comparable values.

report. The printed output of an Enform Plus query using an Enform Plus LIST statement.

request-qualification. The condition or conditions that a database element must satisfy to contribute to the target-record; begins with a WHERE clause followed by a logical expression.

reserved words. Keywords with specific meaning and reserved by Enform Plus.

server query processor. Specific query processor specified by an ?ATTACH command, initiated separately from the compiler.

session. Period of interaction with Enform Plus.

source code. The Enform Plus statements, clauses, and commands that make up the query specifications.

source file. The Edit-format file that contains the source code. See also [source code](#).

statement. The main instruction of an Enform Plus program.

string literal. One or more alphanumeric characters enclosed in quotation marks (").

subscript. A value used to select a particular element.

system variable. An Enform Plus supplied variable that returns the current time, date, line number, or page number.

target aggregate. An aggregate that appears as a part of the *target-record*.

target-file. The file produced by the Query Processor that contains records with all the information requested in the query specifications.

target-item. The record names, field names, expressions, variables, aggregates, and literals, including *by-items*, whose values appear in a *target-record*.

target-list. The record names, field names, expressions, variables, aggregates, and literals following the keywords LIST or FIND that contribute to the *target-record*. A *target-list* consists of *target-items*, some of which are *by-items*. See also [by-item](#) and [target-item](#).

target-record. The records generated by the Query Processor from which your Enform Plus output is produced.

unnormalized. Data that has been described such that more than one value exists for each field position in a record.

user aggregate. A user-declared aggregate. See also [aggregate](#).

user variable. A user-declared element that can be used to store numeric or string literals, field values, and the results of arithmetic or aggregate calculations.

?OUT file. The physical device specified in the ?OUT command.

Index

A

Accessing the Editor from Enform Plus [6-9](#)

Adding a character string to a target-item [5-18](#)

AFTER CHANGE clause

description [5-4](#)

examples [5-4](#), [5-5](#)

print-list elements [5-4](#)

spacing considerations [5-4](#)

syntax [5-4](#)

with a field name [5-4](#)

Aggregates

and null values created by LINK

OPTIONAL [4-29](#)

and scale [3-19](#)

description [3-11](#)

excluding duplicate values
(UNIQUE) [3-13](#)

predefined

AVG [3-13](#)

COUNT [3-13](#)

MAX [3-13](#)

MIN [3-13](#)

SUM [3-13](#)

qualification aggregates

and target records [3-18](#)

description [3-17](#)

examples [3-17](#), [3-18](#)

with embedded WHERE
clause [3-19](#)

with OVER ALL syntax [3-17](#)

with OVER syntax [3-17](#)

syntax [3-11](#)

target aggregates

description [3-15](#)

examples [3-15](#), [3-16](#)

rules for specifying [3-15](#)

target aggregates (continued)

with OVER ALL syntax [3-15](#)

with OVER syntax [3-16](#)

user aggregates [3-13](#)

with embedded WHERE clause [3-17](#)

Alphanumeric display format [5-10](#)

Alphanumeric edit descriptor

description [5-10](#)

examples [5-10](#)

overflow character modifier [5-16](#)

syntax [5-10](#)

Altering the effect of edit descriptors [5-15](#)

Arithmetic expressions [3-21](#)

description [3-21](#)

evaluation order [3-22](#)

operators [3-21](#)

rules for specifying [3-21](#)

scale factor of the result [3-22](#)

Arithmetic operators [3-21](#)

Arithmetic overflow conditions [3-29](#)

AS clause

decorations [5-18](#)

default display format [5-9](#)

description [5-7](#)

examples [5-10](#), [5-11](#), [5-12](#), [5-13](#), [5-14](#),
[5-16](#), [5-17](#), [5-18](#), [5-21](#)

modifiers

field blanking [5-15](#)

fill character [5-16](#)

justification [5-17](#)

overflow character [5-16](#)

permissible combinations with edit
descriptors [5-15](#)

symbol substitution [5-17](#)

nonrepeatable edit descriptors

optional plus sign [5-14](#)

scale factor [5-14](#)

AS clause (continued)

repeatable edit descriptors

alphanumeric [5-10](#)fixed format [5-11](#)integer [5-10](#)mask [5-12](#)syntax [5-7](#)**AS DATE clause**default display format [5-22](#)description [5-22](#)examples [5-23](#)syntax [5-22](#)with JULIAN-DATE Conversion
clause [5-44](#)**AS TIME clause**default display format [5-24](#)description [5-24](#)examples [5-25](#)syntax [5-24](#)time keywords [5-25](#)**ASCD clause**description [5-6](#)example [5-6](#)sorting precedence [5-6](#)syntax [5-6](#)**Ascending order, sorting [5-6](#)****ASSIGN command**and generic files [2-4](#), [2-8](#), [2-10](#), [2-12](#)description [2-7](#)for a server query processor [2-7](#)maximum file assignments [2-2](#)syntax [2-5](#)used for logical file assignments [2-4](#)ASSIGN internal table information,
displaying [6-17](#), [6-18](#)**Assigning**a FIND file to a process [2-10](#)a generic file to a process name [2-10](#)files before running Enform Plus [2-4](#)**Assigning (continued)**files to a server query processor [2-6](#)output to generic files [2-10](#)**Assignment syntax**description [3-4](#)in a FIND statement [4-18](#)used for a user variable [3-29](#)Associating a physical file with a record
description [2-4](#), [6-3](#)**AT END PRINT clause**description [5-26](#)examples [5-26](#)overriding session-wide AT END [5-27](#)print-list elements [5-26](#)spacing considerations [5-26](#)syntax [5-26](#)with a field name [5-26](#)**AT END statement**canceling [4-4](#)description [4-3](#)examples [4-3](#), [4-4](#)overriding [4-4](#)resetting [4-4](#)spacing considerations [4-3](#)syntax [4-3](#)with a field name [4-3](#)**AT START PRINT clause**description [5-28](#)examples [5-28](#), [5-29](#)

overriding session-wide

AT START [5-29](#)print-list elements [5-28](#)spacing considerations [5-28](#)syntax [5-28](#)with a field name [5-28](#)**AT START statement**canceling [4-6](#)description [4-5](#)examples [4-5](#), [4-6](#)

AT START statement (continued)

- overriding [4-6](#)
- resetting [4-6](#)
- spacing considerations [4-5](#)
- syntax [4-5](#)
- with a field name [4-5](#)

Attaching a query processor [6-6](#)**Average value, finding** [3-13](#)**AVG** [3-13](#)**B****Backspacing** [5-66](#)**BEFORE CHANGE clause**

- description [5-30](#)
- examples [5-31](#)
- print-list elements [5-30](#)
- spacing considerations [5-30](#)
- syntax [5-30](#)
- with a field name [5-30](#)

BEGINS WITH operator [3-24](#)**Blanking fields on reports** [3-26](#), [5-16](#), [5-49](#)**Boolean operators** [3-24](#)**Break key** [2-4](#), [5-49](#)**BUILDMK messages** [B-26](#)**BY clause**

- description [5-32](#)
- example [5-32](#)
- sorting precedence [5-32](#)
- syntax [5-32](#)

BY DESC clause

- description [5-32](#)
- sorting precedence [5-32](#)
- syntax [5-32](#)

By-item

- AFTER CHANGE clause [5-4](#)
- BEFORE CHANGE clause [5-30](#)
- BY and BY DESC clauses [5-32](#)
- CENTER clause [5-33](#)
- CUM clause [5-34](#)

By-item (continued)

- description [1-2](#)
- displayed in report columns [4-32](#)
- FIND statement [4-16](#)
- FORM clause [5-38](#)
- HEADING clause [5-39](#)
- LIST statement [4-31](#)
- NOHEAD clause [5-45](#)
- NOPRINT clause [5-46](#)
- PCT clause [5-54](#)
- SKIP clause [5-56](#)
- SPACE clause [5-58](#)
- SUBTOTAL clause [5-63](#)
- TAB clause [5-66](#)
- TOTAL clause [5-71](#)

C**Calculating**

- grand total [5-71](#)
- percentage value [5-54](#)
- running total [5-34](#)
- running total for grouped elements [5-34](#)
- subtotal [5-63](#)

Canceling

- AT END statement [4-4](#)
- AT START statement [4-6](#)
- FOOTING statement [4-24](#)
- SUBFOOTING statement [4-44](#)
- SUBTITLE statement [4-46](#)
- TITLE statement [4-48](#)

CENTER clause

- description [5-33](#)
- specifying in a clause
 - AFTER CHANGE [5-5](#)
 - AT END PRINT [5-27](#)
 - AT START PRINT [5-29](#)
 - BEFORE CHANGE [5-31](#)
 - FOOTING [5-37](#)
 - SUBFOOTING [5-60](#)

CENTER clause (continued)

specifying in a clause (continued)

SUBTITLE [5-62](#)

TITLE [5-70](#)

specifying in a statement [4-36](#)

AT END [4-4](#)

AT START [4-6](#)

FOOTING [4-24](#)

SUBFOOTING [4-44](#)

SUBTITLE [4-46](#)

TITLE [4-48](#)

syntax [5-33](#)

Centering

a print list [5-33](#)

a single report item [5-33](#)

all items in a report [5-33](#)

body of a report [5-49](#)

Changing

default fill character [5-16](#)

default output width [5-53](#)

default overflow character [5-16](#), [5-51](#)

default underline character [5-53](#)

session-wide default date display
format [5-50](#)

session-wide default time display
format [5-53](#)

Characters, special [3-4](#)

Clauses

AFTER CHANGE [5-4](#)

AS [5-7](#)

AS DATE [5-22](#)

AS TIME [5-24](#)

ASCD [5-6](#)

AT END PRINT [5-26](#)

AT START PRINT [5-28](#)

BEFORE CHANGE [5-30](#)

BY [5-32](#)

BY DESC [5-32](#)

CENTER [5-33](#)

Clauses (continued)

CUM [5-34](#)

DESC [5-6](#)

description [3-5](#)

FOOTING [5-36](#)

FORM [5-38](#)

HEADING [5-39](#)

INTERNAL [5-42](#)

JULIAN-DATE Conversion [5-43](#)

NOHEAD [5-45](#)

NOPRINT [5-46](#)

Option Variable [5-48](#)

PCT [5-54](#)

SKIP [5-56](#)

SPACE [5-58](#)

SUBFOOTING [5-59](#)

SUBTITLE [5-61](#)

SUBTOTAL [5-63](#)

summary of [5-1](#)

SUPPRESS [5-64](#)

System Variable [5-65](#)

TAB [5-66](#)

TIMESTAMP-DATE [5-67](#)

TIMESTAMP-TIME [5-68](#)

TITLE [5-69](#)

TOTAL [5-71](#)

WHERE [5-72](#)

with LIST statement but not FIND
statement [5-1](#)

Clearing linked record descriptions [4-12](#)

Clearing the internal table [4-14](#)

CLOSE statement [4-7](#)

DELINK statement [4-12](#)

DICTIONARY statement [4-13](#)

?DICTIONARY command [6-8](#)

CLOSE statement

description [4-7](#)

effect on the internal table [4-7](#)

syntax [4-7](#)

Column headings

specifying [5-39](#)suppressing [5-45](#)Combining percentages and subtotals [5-55](#)

Commands

Enform Plus commands [3-5](#)in an Edit-format file [6-1](#), [6-15](#)summary [6-1](#)?ASSIGN [2-4](#), [6-3](#)?ATTACH [2-6](#), [6-6](#)?COMPILE [6-7](#)?DICTIONARY [6-8](#)?EDIT [6-9](#)?EXECUTE [6-10](#)?EXIT [6-11](#)?HELP [6-12](#)?OUT [6-14](#)?RUN [6-15](#)?SECTION [6-16](#)?SHOW [6-17](#)?SOURCE [6-19](#)

TACL commands

ASSIGN [2-4](#), [2-7](#)ENFORM [2-1](#)FC [2-2](#)PARAM [2-5](#), [2-8](#), [4-39](#)PAUSE [2-4](#)QP [2-9](#)

Comments

description [3-4](#)examples [3-4](#)

Comparison template

mask edit descriptor [5-12](#)pattern match in a logical
expression [3-26](#)Compilation output [2-11](#)

Compiled query file

commands not saved [6-1](#)creating [6-7](#)

Compiled query file (continued)

defining parameters for [4-40](#)executing [6-10](#)passing parameters to [2-5](#), [4-40](#)@BREAK-KEY [5-49](#)Compiler directives [3-5](#), [6-1](#)

Compiler/report writer

description [1-3](#)error and warning messages [2-11](#)

Compiling

and executing [6-15](#)to a compiled query file [6-7](#)

Compound logical expression

description [3-24](#)effect of Boolean operators [3-24](#)effect of parentheses [3-24](#)Computing [3-29](#)Condition specifiers [5-19](#)

Conditional operators

BEGINS WITH [3-22](#), [3-24](#)CONTAINS [3-22](#), [3-24](#)description [3-22](#)EQ [3-22](#)EQUAL [3-22](#)GE [3-22](#)GREATER THAN [3-22](#)GT [3-22](#)IS [3-22](#)LE [3-22](#)LESS THAN [3-22](#)LT [3-22](#)NE [3-22](#)Conjunctive normal form [C-3](#)CONTAINS operator [3-24](#)Controlling the printing of a plus sign [5-14](#)Converting data values [5-7](#)Converting dates to internal format [5-44](#)Cost tolerance levels [5-49](#)COUNT [3-13](#)

Creating a new physical file [4-17](#)

Creating a server query processor

 ASSIGN command [2-7](#)

 description [2-6](#)

 example [2-10](#)

 PARAM command [2-8](#)

 QP command [2-9](#)

CUM clause

 description [5-34](#)

 example [5-34](#)

 restrictions [5-35](#)

 syntax [5-34](#)

 with a user variable [5-35](#)

Cumulative operations [3-11](#), [5-34](#), [5-63](#),
[5-71](#)

Current Enform Plus session,
terminating [6-11](#)

Current output listing file

 Break key [2-4](#)

 description [2-3](#)

 generic files [2-10](#)

 terminating [2-4](#)

Current values of option variables [6-18](#)

D

Data Definition Language (DDL) [1-3](#), [4-13](#),
[4-17](#), [6-8](#)

 data description [1-3](#)

 dictionaries produced by [6-8](#)

Database [1-3](#)

Database group [3-9](#), [4-19](#)

Database tables

 nesting [3-11](#)

 subscripting [3-9](#)

Date

 changing the display format
 in the current query [5-22](#)

 session-wide [5-50](#)

 default display format [5-22](#)

Date (continued)

 obtaining the current date [5-67](#)

Date keywords [5-22](#)

DDL

See Data Definition Language (DDL)

DECLARE statement

 description [4-9](#)

 examples [4-10](#), [4-11](#)

 syntax [4-9](#)

 user aggregate [4-10](#)

 user table [4-10](#)

 user variable [4-10](#)

Declaring user elements [4-9](#)

Decorations

 condition [5-19](#)

 default [5-20](#)

 description [5-18](#)

 examples [5-21](#)

 location [5-19](#)

 processing order [5-20](#)

 rules for use [5-19](#)

 syntax [5-18](#)

Default

 date display format [5-22](#)

 decorations [5-20](#)

 fill character [5-16](#)

 heading [5-39](#)

 input file [2-1](#)

 internal storage format [5-42](#)

 margin width [5-51](#)

 new line character [5-51](#)

 output file [2-3](#)

 output width [5-53](#)

 overflow character [5-16](#), [5-51](#)

 spacing between columns [5-53](#)

 subtotal label [5-53](#)

 target-item display format [5-9](#)

 time display format [5-24](#), [5-53](#)

 underline character [5-53](#)

Default (continued)

- value of a user variable [3-29](#)
- values for option variables [5-48](#)

Delimiters [3-4](#)

DELINK statement

- description [4-12](#)
- examples [4-12](#)
- syntax [4-12](#)

DESC clause

- description [5-6](#)
- sorting precedence [5-6](#)
- syntax [5-6](#)

Descending order, sorting [5-6](#)

Descriptors

- See also* AS clause
- alphanumeric [5-10](#)
- altering the effect of [5-15](#)
- combined with modifiers [5-15](#)
- fixed format [5-11](#)
- integer [5-10](#)
- mask [5-12](#)
- nonrepeatable [5-13](#)
- optional plus sign [5-14](#)
- repeatable [5-9](#)
- scale factor [5-14](#)

Diagnostic messages [B-1](#)

Dictionary

- description [1-3](#)
- identifying [4-13](#), [6-8](#)
- produced by DDL [1-3](#), [4-13](#), [6-8](#)
- record description of FIND file [4-16](#)

DICTIONARY statement

- clearing the internal table [4-14](#)
- description [4-13](#)
- example [4-13](#)
- identifying dictionary location [4-13](#)
- syntax [4-13](#)

Display format

- and scale [5-12](#)
- date
 - changing session-wide [5-50](#)
 - specifying for the current query [5-22](#)
- target-item [5-7](#)
- template [5-12](#)
- time
 - changing session-wide [5-53](#)
 - specifying for the current query [5-24](#)

Displaying environmental information

- description [6-17](#)
- internal table space [6-17](#), [6-18](#)
- links in effect [6-17](#)
- option variables [6-18](#)
- record descriptions [6-17](#)

Duplicate field names [3-7](#)Duration of Enform Plus statements [3-5](#)

E

Edit descriptors

- alphanumeric [5-10](#)
- altering the effect of [5-15](#)
- combined with modifiers [5-15](#)
- fixed format [5-11](#)
- integer [5-10](#)
- mask [5-12](#)
- nonrepeatable [5-13](#)
- optional plus sign [5-14](#)
- repeatable [5-9](#)
- scale factor [5-14](#)

Editor

- and the Break key [2-4](#)
- entering from within Enform Plus [6-9](#)
- exiting [6-9](#)

Edit-format files

- containing Enform Plus queries [2-3](#)
- reading [6-19](#)
- storing source code in [6-9](#), [6-15](#), [6-19](#)

Eliminating

- field values [5-46](#)
- headings [5-45](#)
- records from a report [5-64](#)

Embedded WHERE clause [3-19](#)ENFOLD macro [2-2](#)

ENFORM command

- description [2-1](#)
- syntax [2-1](#)

ENFPLUS macro [2-2](#)

Entering commands and statements

- directly [2-2](#)
- indirectly [2-3](#)

Entry-sequenced files [3-7](#)

Environmental information, displaying

- internal table space [6-18](#)
- links in effect [6-17](#)
- open record descriptions [6-17](#)
- option variable values [6-18](#)
- record-and-file assignments [6-18](#)

Error messages

- Enform Plus initialization [B-2](#)
- Enform Plus trap [B-26](#)
- types of [B-1](#)
- !!! errors and *** warnings [B-4](#)
- *** file errors [B-22](#)

Errors, reported [3-5](#)

Evaluation order

- arithmetic expressions [3-22](#)
- compound logical expressions [3-24](#)
- decorations [5-19](#)
- user variables [3-29](#)

Exclusion mode

- and a server query processor [2-6](#)
- and generic files [2-10](#)

Exclusion mode (continued)

- ASSIGN command [2-8](#)
- ?ASSIGN command [6-4](#)

Exclusion specification, changing [6-5](#)

Executing

- a compiled query file [6-10](#)
- and compiling a query [6-15](#)
- source code in an Edit-format file [6-15](#), [6-19](#)

EXIT statement

- description [4-15](#)
- syntax [4-15](#)

Exiting

Enform Plus

- interactive mode [2-3](#)
- noninteractive mode [2-3](#)
- Enform Plus session [4-15](#), [6-11](#)
- the Editor [6-9](#)

Expressions

- arithmetic [3-21](#)
- evaluation order [3-22](#)
- scale factor of the result [3-22](#)
- IF/THEN/ELSE [3-26](#)
- logical [3-22](#)
- conditional operators [3-22](#)
- effect of parentheses on evaluation [3-24](#)
- pattern-match in [3-26](#)
- range of values in [3-25](#)

Extent size, specifying

- primary [5-51](#)
- secondary [5-52](#)

FFC command [2-2](#)

Field blanking modifiers

- description [5-15](#)
- examples [5-16](#)

Field grouping clauses [5-32](#)

Field name

qualification required [3-7](#)references [3-6](#)

specifying in a clause

AFTER CHANGE [5-4](#)AT END PRINT [5-26](#)AT START PRINT [5-28](#)BEFORE CHANGE [5-30](#)FOOTING [5-36](#)SUBFOOTING [5-59](#)SUBTITLE [5-61](#)TITLE [5-69](#)

specifying in a statement

AT END [4-3](#)AT START [4-5](#)FOOTING [4-23](#)SUBFOOTING [4-43](#)SUBTITLE [4-45](#)TITLE [4-47](#)with subscripts [3-9](#)Field sorting clauses [5-6](#), [5-32](#)File assignments, logical [2-4](#)File error messages [B-22](#)File type and the LIST statement [4-31](#)

Files

Edit-format [2-1](#)entry-sequenced [3-7](#)format 1 [Glossary-2](#)format 2 [Glossary-2](#)generic [2-10](#)output [2-14](#)relative [3-7](#)unstructured [3-7](#)

Fill character modifier

description [5-16](#)examples [5-16](#)syntax [5-16](#)Filling fields with blanks [3-26](#), [5-15](#), [5-49](#)

FIND statement

description [4-16](#)examples [4-17](#), [4-18](#), [4-19](#), [4-20](#), [4-21](#)file type of generated file [4-17](#)grouping and sorting records [4-17](#)input elements [4-19](#)output fields [4-18](#)

output record dictionary

description [4-17](#)request-qualification [4-21](#)statements and clauses that do not apply [4-22](#)summary records [4-21](#)syntax [4-16](#)Finding the highest number in a set [3-13](#)Finding the lowest number in a set [3-13](#)

Fixed format edit descriptor

changing the special symbols [5-18](#)description [5-11](#)examples [5-11](#)syntax [5-11](#)

Fixed format number

specifying display format [5-11](#)specifying storage format [5-42](#)

FOOTING clause

description [5-36](#)examples [5-36](#)overriding a FOOTING statement [5-37](#)spacing considerations [5-36](#)syntax [5-36](#)with a field name [5-36](#)

FOOTING statement

canceling [4-24](#)description [4-23](#)examples [4-23](#), [4-24](#)overriding [4-24](#)resetting [4-24](#)spacing considerations [4-23](#)syntax [4-23](#)

FOOTING statement (continued)

with a field name [4-23](#)

FORM clause

description [5-38](#)example [5-38](#)

specifying in a clause

AT END PRINT [5-27](#)AT START PRINT [5-29](#)FOOTING [5-37](#)SUBFOOTING [5-60](#)SUBTITLE [5-62](#)TITLE [5-70](#)

specifying in a statement

AT END [4-4](#)AT START [4-6](#)FOOTING [4-24](#)LIST [4-31](#)SUBFOOTING [4-44](#)SUBTITLE [4-46](#)TITLE [4-48](#)syntax [5-38](#)with a by-item [5-38](#)with a target-item [5-38](#)within a print list [5-38](#)Format 1 files [Glossary-2](#)Format 2 files [Glossary-2](#)Formatter [5-7](#)**G**

Generic files

and a dedicated query processor [2-12](#)and the current output listing file [2-13](#)and the server query processor [2-13](#)ASSIGN command [2-4](#)description [2-10](#)exclusion mode [2-10](#)forms of output [2-13](#)output record length [2-11](#), [2-12](#)QUERY-COMPILER-LISTING [2-11](#)

Generic files (continued)

QUERY-QPSTATISTICS [2-12](#)QUERY-QPSTATUS-MESSAGES [2-12](#)QUERY-REPORT-LISTING [2-11](#)QUERY-SORT-AREA [2-12](#)QUERY-STATISTICS [2-11](#)QUERY-STATUS-MESSAGES [2-11](#)QUERY-WORK-AREA [2-11](#)Getting help [6-12](#)Gregorian dates [5-43](#)Group specification in a LIST statement [4-33](#)Grouping target-records by field values [4-16](#), [4-33](#), [5-32](#)Guardian Formatter [5-7](#)Guardian procedure TIMESTAMP [5-68](#)**H**

HEADING clause

description [5-39](#)examples [5-40](#)multiple-line headings [5-39](#)printing a / in a column heading [5-39](#)syntax [5-39](#)Heading, suppressing printing of [5-46](#)Highest number in a set, finding [3-13](#)

Home terminal

and the Break key [2-4](#)as the default input file [2-4](#)description [2-1](#)Horizontal spacing [5-52](#), [5-58](#)**I**Identifying a command [6-1](#)Identifying a specific query processor [6-6](#)

IF/THEN/ELSE expressions

description [3-26](#)example [3-26](#)syntax [3-26](#)

IF/THEN/ELSE expressions (continued)
 value keywords [3-26](#)

IN option
 and the current output listing file [2-3](#)
 description [2-1](#)

Initial value of a user variable [3-27](#)

Input file [2-3](#)

Integer edit descriptor
 description [5-10](#)
 examples [5-11](#)
 syntax [5-10](#)

Interactive mode
 ASSIGN command [2-5](#)
 description [2-2](#)
 getting help [6-12](#)

INTERNAL clause
 description [5-42](#)
 example [5-42](#)
 internal format types [5-42](#)
 syntax [5-42](#)

Internal table
 clearing [4-7](#), [4-12](#), [4-14](#), [6-8](#)
 description [4-7](#)

J

JULIAN-DATE Conversion clause
 converting dates to internal format [5-43](#)
 description [5-43](#)
 display format [5-44](#)
 examples [5-43](#), [5-44](#)
 Gregorian dates [5-43](#)
 syntax [5-43](#)

Justification modifiers
 description [5-17](#)
 examples [5-17](#)
 syntax [5-17](#)

K

Keeping files open
 for a dedicated query processor [2-4](#)
 for a server query processor [2-6](#)

Keys
 See primary keys

L

Language elements [3-1](#)
 Left justification [5-17](#)
 Left margin size [5-51](#)
 Limiting the number of records per query [5-53](#)
 Link diagrams [C-1](#)
 LINK OPTIONAL statement
 and other links [C-1](#)
 considerations [4-26](#)
 duration [4-26](#)
 examples of invalid links [C-7](#)
 examples of valid links [C-9](#)
 links initiated by [C-2](#)
 rules [4-27](#)
 syntax [4-25](#)

LINK statement
 clearing a link [4-26](#)
 compared to WHERE clause [C-4](#)
 considerations [4-26](#)
 description [4-25](#)
 duration of link [4-26](#)
 examples [4-25](#), [4-26](#), [4-27](#), [4-29](#)
 links initiated by [C-1](#)
 noncontributing record descriptions [4-28](#)
 syntax [4-25](#)

Linking relationships
 clearing [4-12](#), [4-14](#)
 combining [C-5](#)
 one-directional [C-2](#)
 two-directional [C-1](#)

Links

- clearing [4-7](#), [4-12](#), [4-14](#), [6-8](#)
- defined by Enform Plus
 - LINK OPTIONAL statement [C-2](#)
 - LINK statement [C-1](#)
 - WHERE clause [C-2](#)
- maximum number [4-26](#)
- session-wide [4-26](#)

LIST statement

- conditional printing [4-34](#)
- description [4-30](#)
- displaying values in report columns [4-32](#)
- examples [4-32](#), [4-33](#), [4-34](#), [4-35](#), [4-36](#)
- grouping and sorting target-records [4-32](#)
- input record description [4-31](#)
- optional clauses [4-36](#)
- request-qualification [4-34](#)
- summary reports [4-34](#)
- syntax [4-30](#)

Literals

- description [3-20](#)
- examples [3-20](#), [3-21](#)
- numeric literals [3-20](#)
- string literals [3-20](#)

Location of temporary work files [2-11](#)

Location specifiers [5-19](#)

Logical expressions

- BEGINS WITH operator [3-24](#)
- Boolean operators [3-24](#)
 - compound [3-24](#)
 - conditional operators [3-22](#)
- CONTAINS operator [3-25](#)
- description [3-22](#)
- effect of parentheses on evaluation [3-24](#)
- pattern match [3-23](#), [3-26](#)
- range of values [3-24](#), [3-25](#)
- simple [3-24](#)

Logical expressions (continued)

- syntax [3-23](#)

Logical file assignments

- and a server query processor [2-6](#)
- description [2-4](#)

Lowest number in a set, finding [3-13](#)

M

Margin [5-51](#)

Mask edit descriptor

- changing the special symbols [5-18](#)
- description [5-12](#)
- examples [5-12](#), [5-13](#)
- syntax [5-12](#)

MAX [3-13](#)

Maximum

- number of links [4-26](#)
- number of pages per report [5-51](#)
- number of records to be selected [5-53](#)
- number of requestors for a server query processor [2-8](#)
- number of user table elements [3-29](#)

Message table file [2-1](#), [6-12](#)

Messages

- BUILDMK messages [B-26](#)
- Enform Plus initialization [B-2](#)
- Enform Plus trap messages [B-26](#)
- initialization messages [B-2](#)
- summary [B-1](#)
- types of [B-1](#)
- !!! ERROR messages [B-4](#)
- !!! errors and *** warnings [B-4](#)
- *** FILE ERROR messages [B-22](#)
- *** file errors [B-22](#)
- *** WARNING messages [B-4](#)

MIN [3-13](#)

Modifiers

- combined with edit descriptors [5-15](#)
- description [5-15](#)

Modifiers (continued)

- field blanking [5-15](#)
- fill character [5-16](#)
- justification [5-17](#)
- overflow character [5-16](#)
- symbol substitution [5-17](#)
- syntax [5-7](#), [5-9](#)

N

Naming

- a collection of source code [6-16](#)
- a section [6-16](#)
- a server query processor [2-9](#)
- a subvolume containing a dictionary [2-1](#), [6-8](#)
- fields [3-6](#)
- parameters [3-6](#)
- records [3-6](#)
- the message table file [2-1](#)
- user-defined elements [3-6](#)

Nested

- arithmetic expressions [3-22](#)
- database tables [3-11](#)
- IF/THEN/ELSE expressions [3-26](#)
- logical expressions [3-24](#)
- ?SOURCE commands [6-19](#)

New line character [5-51](#)

NOHEAD clause

- description [5-45](#)
- examples [5-45](#)
- syntax [5-45](#)

Noncontributing records

- and a WHERE clause [4-28](#)
- and aggregates [4-29](#)

Noninteractive mode

- ASSIGN command [2-4](#)
- description [2-3](#)

Nonrepeatable edit descriptors

- description [5-13](#)
- optional plus sign [5-14](#)
- scale factor [5-14](#)
- syntax [5-7](#)

NOPRINT clause

- description [5-46](#)
- examples [5-46](#)
- syntax [5-46](#)

Numeric literals

- description [3-20](#)
- examples [3-20](#)
- rules for specifying [3-20](#)

OObtaining the current date or time [5-65](#)

OPEN AS COPY OF

- description [4-38](#)
- syntax [4-38](#)

Open record description information [6-17](#)

OPEN statement

- description [4-38](#)
- syntax [4-38](#)

Opening record descriptions [4-38](#)

Operators

- arithmetic [3-21](#)
- Boolean [3-24](#)
- conditional [3-22](#)

Option Variable clauses

- description [5-48](#)
- displaying [6-18](#)
- resetting [4-42](#)
- setting [4-42](#)
- syntax [5-48](#)
- @BLANK-WHEN-ZERO [5-49](#)
- @BREAK-KEY [5-49](#)
- @CENTER-PAGE [5-49](#)
- @COPIES [5-49](#)
- @COST-TOLERANCE [5-49](#)

Option Variable clauses (continued)

[@DATE-FORMAT 5-50](#)
[@DECIMAL 5-50](#)
[@DISPLAY-COUNT 5-50](#)
[@HEADING 5-51](#)
[@LINES 5-51](#)
[@MARGIN 5-51](#)
[@NEWLINE 5-51](#)
[@NONPRINT-REPLACE 5-51](#)
[@OVERFLOW 5-51](#)
[@PAGES 5-51](#)
[@PRIMARY-EXTENT-SIZE 5-51](#)
[@READS 5-52](#)
[@SECONDARY-EXTENT-SIZE 5-52](#)
[@SPACE 5-52](#)
[@STATS 5-52](#)
[@SUBTOTAL-LABEL 5-52](#)
[@SUMMARY-ONLY 5-53](#)
[@TARGET-RECORDS 5-53](#)
[@TIME-FORMAT 5-53](#)
[@UNDERLINE 5-53](#)
[@VSPACE 5-53](#)
[@WARN 5-53](#)
[@WIDTH 5-53](#)

Optional plus sign edit descriptor

[description 5-14](#)
[syntax 5-14](#)

OUT option

[and the current output listing file 2-3](#)
[description 2-1](#)

Output

[assigning to a generic file 2-5, 2-10](#)
[compilation 2-11](#)
[current output listing file 2-3](#)
[default width 5-53](#)
[files 2-14](#)
[specifying an output device for a report 2-11, 6-14](#)
[suspending 2-4, 5-49](#)

Output (continued)

[terminating 2-4](#)

Output record dictionary description [4-17](#)Overflow character [5-16, 5-51](#)

Overflow character modifier

[description 5-16](#)

[examples 5-17](#)

[syntax 5-16](#)

Overflow condition [3-29, 5-16](#)

Overriding

[ASSIGN command 6-5](#)

[AT END statement 4-4, 5-27](#)

[AT START statement 4-6, 5-29](#)

[FOOTING statement 4-24, 5-37](#)

[physical file named in DDL FILE IS clause 6-4](#)

[SUBFOOTING statement 4-44, 5-60](#)

[SUBTITLE statement 4-46, 5-62](#)

[TITLE statement 4-48, 5-70](#)

[?ASSIGN command 6-5](#)

P

Page

[numbers 5-65](#)

[specifying new 5-38](#)

Paginating a report [5-38](#)PARAM command [2-8](#)

[for a server query processor 2-8](#)

[syntax 2-5](#)

[to pass parameters 2-5](#)

PARAM statement

[description 4-39](#)

[examples 4-39, 4-40](#)

[syntax 4-39](#)

Parameter

[as a numeric literal 4-39](#)

[as a string literal 4-39](#)

[clearing 4-7, 4-14](#)

[defining 4-39](#)

Parameter (continued)

- deleting from the internal table [4-7](#)
- description [3-27](#)
- displaying current value [6-18](#)
- handling by Enform Plus [3-27](#), [4-39](#)
- in a print-list [4-39](#)
- maximum number allowed [4-39](#)
- passing to a compiled query file [2-5](#), [4-39](#)
- rules for naming [3-6](#)
- variable-length [3-25](#)

Passing parameters [2-5](#)Pattern match [3-23](#), [3-26](#)

See also Logical expressions

PAUSE command [2-4](#)

PCT clause

- combining percentages and subtotals [5-55](#)
- description [5-54](#)
- examples [5-54](#)
- restrictions [5-55](#)
- syntax [5-54](#)
- with a by-item [5-54](#)
- with a user variable [5-55](#)

Percentage of the grand total [5-54](#)Percentage values and truncation [5-55](#)

Physical file

- associated with a record description [6-3](#)
- creating [6-7](#)

Precision of arithmetic operations [3-22](#)

Predefined aggregates

- AVG [3-13](#)
- COUNT [3-13](#)
- MAX [3-13](#)
- MIN [3-13](#)
- SUM [3-13](#)

Pressing the Break key [2-4](#)Preventing specific records from printing [5-72](#)Primary extent size [5-51](#)

Primary keys

- description [3-7](#)
- entry-sequenced files [3-7](#)
- key-sequenced files [3-7](#)
- relative files [3-7](#)
- unstructured files [3-7](#)

Print list

- and a record name [3-6](#)
- centering [5-33](#)

Processing order

- of compound logical expressions [3-24](#)
- of decorations [5-19](#)
- of nested arithmetic expressions [3-22](#)

Prompt

- Enform Plus [2-1](#)
- TACL [2-1](#)

Q

QP command [2-9](#)

Qualification aggregates

- and target records [3-18](#)
- description [3-17](#)
- rules for specifying [3-17](#)
- with an embedded WHERE clause [3-19](#)
- with OVER ALL syntax [3-17](#)
- with OVER syntax [3-17](#)

Qualifying field names [3-6](#)

Query

- description [1-2](#)
- in a compiled query file [2-3](#)
- stored in an Edit-format file [2-3](#)

Query processor

- description [1-3](#)
- error messages [2-12](#)
- identifying with the ?ATTACH command [6-6](#)

Query specifications

- description [1-2](#)
- statements [3-4](#)

QUERY-COMPILER-LISTING

and the current output listing file [2-3](#)
description [2-11](#)

QUERY-QPSTATISTICS [2-12](#)**QUERY-QPSTATUS-MESSAGES** [2-12](#)**QUERY-REPORT-LISTING**

and the current output listing file [2-3](#)
description [2-11](#)

QUERY-SORT-AREA [2-12](#)**QUERY-STATISTICS** [2-11](#)**QUERY-STATUS-MESSAGES** [2-11](#)**QUERY-WORK-AREA** [2-11](#)**R****Range of values**

description [3-25](#)
syntax [3-25](#)

Reading an Edit-format file [6-19](#)**Reassigning a physical file** [2-5](#), [6-4](#)**Reclaiming table space** [4-7](#), [6-8](#)**Record**

displaying environmental
information [6-17](#), [6-18](#)
referencing a record name [3-6](#)

Record description

accessing [4-38](#)
clearing links between [4-12](#)
for a LIST statement [4-31](#)
linking [4-25](#), [5-72](#)
list of opened [6-17](#)
output record of a FIND statement [4-16](#)
removing from the internal table [4-7](#),
[4-14](#), [6-8](#)

Records with duplicate field names [3-7](#)**Referencing database elements**

field names [3-6](#)
primary keys [3-7](#)
record names [3-6](#)
rules [3-6](#)

Referencing database elements (continued)

using subscripts [3-8](#)

Relative files [3-7](#)**Repeatable edit descriptors**

alphanumeric [5-10](#)
changing the special edit symbols [5-18](#)
description [5-9](#)
fixed format [5-11](#)
integer [5-10](#)
mask [5-12](#)
syntax [5-7](#)

Replacing symbols used in edit descriptors [5-17](#)**Report lines** [5-51](#), [5-65](#)**Report writer**

See Compiler/report writer

Reporting error messages to a generic file [2-11](#)**Reporting statistics to a generic file** [2-11](#)**Reports**

adding character strings to target-items [5-18](#)
centering [5-33](#), [5-49](#)
creating a running total [5-34](#)
footings [4-23](#), [5-36](#)
headings [5-39](#)
headings for subscripted elements [5-40](#)
horizontal spacing [5-52](#), [5-58](#)
including the current date [5-22](#), [5-65](#),
[5-67](#)
including the current time [5-24](#), [5-65](#),
[5-68](#)
information at the end [4-3](#), [5-26](#)
information at the start [4-5](#), [5-28](#)
information within a report [5-30](#)
line numbers [5-65](#)
page numbers [5-51](#), [5-65](#)
preventing records from appearing [5-72](#)
selecting information [4-31](#)
starting a new line [5-51](#), [5-56](#)

Reports (continued)

- starting a new page [5-38](#)
- subfootings [4-43](#), [5-59](#)
- subtitles [4-45](#), [5-61](#)
- summary [4-34](#)
- suppressing zeros [5-49](#)
- title [4-47](#), [5-69](#)
- title for the current report [5-70](#)
- values in report columns [4-32](#)
- vertical spacing [5-53](#), [5-56](#)

REQUESTORS parameter [2-8](#)

Request-qualification

- aggregates [3-17](#)
- description [1-3](#)
- FIND statement [4-21](#)
- literals [3-20](#)
- user variables [3-29](#)
- WHERE clause [5-72](#)

Reserved words [3-2](#)

Resetting

- AT END statement [4-4](#)
- AT START statement [4-6](#)
- FOOTING statement [4-24](#)
- SUBFOOTING statement [4-44](#)
- SUBTITLE statement [4-46](#)
- TITLE statement [4-48](#)

Restricting records [5-72](#)

Result of arithmetic operations

- assigning to a user variable [3-22](#)
- scale factor [3-22](#)

Right justification [5-17](#)

Rules

- for decorations [5-19](#)
- for input elements for FIND files [4-20](#)
- for LINK OPTIONAL statements [4-27](#), [C-6](#)
- for literals [3-20](#)
- for naming a section [6-16](#)
- for naming user-defined elements [3-6](#)

Rules (continued)

- for output fields in FIND files [4-18](#)
- for parameter names [3-6](#)
- for qualification aggregates [3-17](#)
- for referencing database elements [3-6](#)
- for referencing field names [3-6](#)
- for referencing primary keys [3-7](#)
- for referencing record names [3-6](#)
- for target aggregates [3-15](#)
- for target-items in a LIST statement [4-32](#)
- for user aggregates [3-14](#)

Running Enform Plus [2-1](#)

- interactive mode [2-2](#)
- noninteractive mode [2-3](#)

S

Scale factor edit descriptor

- description [5-14](#)
- examples [5-14](#)
- rules for naming [6-16](#)
- syntax [5-14](#)

Scale factor of arithmetic results [3-22](#)Secondary extent size [5-52](#)

Section

- identifying [6-16](#)
- rules for naming [6-16](#)

Semicolon, ending a statement with [4-1](#)

Server query processor

- ASSIGN command [2-7](#)
- creating [2-6](#)
- creation example [2-10](#)
- description [2-6](#)
- PARAM command [2-8](#)
- QP command [2-9](#)

Session

- description [1-1](#)
- example [1-2](#)
- terminating [6-11](#)

SET statement

- and option variables [4-42](#)
- and user-defined elements [4-41](#)
- description [4-41](#)
- examples [4-42](#)
- syntax [4-41](#)

Setting

- option variables [4-41](#), [4-42](#)
- the left margin [5-51](#)
- the number of report lines [5-51](#)
- user elements [4-41](#)
- values of user-defined elements [4-41](#)

Sharing a server query processor [2-6](#)**Size**

- determining for a target-item [5-8](#)
- left margin [5-51](#)
- running total value [5-34](#)
- subtotal value [5-63](#)
- total value [5-71](#)

SKIP clause

- and @VSPACE [5-56](#)
- description [5-56](#)
- example [5-56](#)
- specifying in a clause
 - AFTER CHANGE [5-5](#)
 - AT END PRINT [5-26](#)
 - AT START PRINT [5-28](#)
 - BEFORE CHANGE [5-31](#)
 - FOOTING [5-36](#)
 - SUBFOOTING [5-59](#)
 - SUBTITLE [5-61](#)
 - TITLE [5-69](#)
- specifying in a statement
 - AT END [4-3](#)
 - AT START [4-5](#)
 - FOOTING [4-23](#)
 - LIST [4-31](#)
 - SUBFOOTING [4-43](#)
 - SUBTITLE [4-45](#)

SKIP clause (continued)

- specifying in a statement (continued)
 - TITLE [4-47](#)
- syntax [5-56](#)

Sorting

- and the Break key [2-4](#)
- in a FIND statement [4-17](#)
- in ascending order [5-6](#), [5-32](#)
- in descending order [5-6](#), [5-32](#)
- target-records [5-32](#)
- sort key [5-6](#)
- target records [5-6](#)

Source code

- entering [2-2](#)
- in an Edit-format file [6-9](#), [6-16](#), [6-19](#)

SPACE clause

- description [5-58](#)
- specifying in a clause
 - AFTER CHANGE [5-4](#)
 - AT END PRINT [5-26](#)
 - AT START PRINT [5-28](#)
 - BEFORE CHANGE [5-30](#)
 - FOOTING [5-36](#)
 - SUBFOOTING [5-59](#)
 - SUBTITLE [5-61](#)
 - TITLE [5-69](#)
- specifying in a statement
 - AT END [4-3](#)
 - AT START [4-5](#)
 - FOOTING [4-23](#)
 - LIST [4-31](#)
 - SUBFOOTING [4-43](#)
 - SUBTITLE [4-45](#)
 - TITLE [4-47](#)
- syntax [5-58](#)
- with a print list [5-58](#)
- with a target-item or by-item [5-58](#)

Special characters [3-4](#)**Special edit symbols [5-17](#)**

Statements

applicable only to queries containing a
LIST statement [4-1](#)

AT END [4-3](#)

AT START [4-5](#)

CLOSE [4-7](#)

DECLARE [4-9](#)

DELINK [4-12](#)

description [3-4](#)

DICTIONARY [4-13](#)

duration of effect [3-5](#), [4-1](#)

EXIT [4-15](#)

FIND [4-16](#)

FOOTING [4-23](#)

LINK [4-25](#)

LIST [4-30](#)

OPEN [4-38](#)

PARAM [4-39](#)

SET [4-41](#)

SUBFOOTING [4-43](#)

SUBTITLE [4-45](#)

summary [4-1](#)

terminating with a semicolon [3-5](#), [4-1](#)

TITLE [4-47](#)

Statistics

description [5-52](#)

output device for [2-11](#)

Storage format

default for a user table [4-10](#)

default for a user variable [4-10](#)

for dates [5-43](#)

Storing

a date in internal format [5-67](#)

a time in internal format [5-68](#)

source code in an Edit-format file [6-9](#),
[6-19](#)

Strategy cost

and a server query processor [2-8](#)

description [5-52](#)

String literals

description [3-20](#)

examples [3-21](#)

rules for specifying [3-20](#)

SUBFOOTING clause

description [5-59](#)

examples [5-59](#)

for the current report [5-60](#)

spacing considerations [5-59](#)

syntax [5-59](#)

with a field name [5-59](#)

SUBFOOTING statement

canceling [4-44](#)

description [4-43](#)

examples [4-43](#), [4-44](#)

overriding [4-44](#)

resetting [4-44](#)

spacing considerations [4-43](#)

syntax [4-43](#)

with a field name [4-43](#)

Subordinate field [3-10](#)

Subscripts

description [3-8](#)

examples [3-9](#), [3-10](#)

for a range [3-10](#)

headings [5-40](#)

syntax [3-9](#)

valid values [3-9](#)

SUBTITLE clause

description [5-61](#)

for the current report [5-62](#)

overriding a session-wide subtitle [5-62](#)

spacing considerations [5-61](#)

syntax [5-61](#)

with a field name [5-61](#)

SUBTITLE statement

canceling [4-46](#)

description [4-45](#)

examples [4-45](#), [4-46](#)

SUBTITLE statement (continued)

- overriding [4-46](#)
- resetting [4-46](#)
- spacing considerations [4-45](#)
- syntax [4-45](#)
- with a field name [4-45](#)

SUBTOTAL clause

- description [5-63](#)
- syntax [5-63](#)

SUM [3-13](#)Summary of Enform Plus statements [4-1](#)Summary records [4-21](#), [5-53](#)Summary report [4-35](#), [5-53](#)

SUPPRESS clause

- description [5-64](#)
- example [5-64](#)
- syntax [5-64](#)

Suppressing

- column headings [5-46](#)
- column headings for all reports [5-51](#)
- printing of report items [5-46](#)
- selected records from a report [5-64](#)

Suspending output [2-4](#), [5-49](#)

Symbol substitution modifier

- description [5-17](#)
- examples [5-18](#)
- syntax [5-17](#)

Syntax summary [A-1](#)

System Variable clauses

- @DATE [5-65](#)
- @LINENO [5-65](#)
- @PAGENO [5-65](#)
- @TIME [5-65](#)

T

TAB clause

- description [5-66](#)

TAB clause (continued)

specifying in a clause

- AFTER CHANGE [5-4](#)
- AT END PRINT [5-26](#)
- AT START PRINT [5-28](#)
- BEFORE CHANGE [5-30](#)
- FOOTING [5-36](#)
- SUBFOOTING [5-59](#)
- SUBTITLE [5-61](#)
- TITLE [5-69](#)

specifying in a statement

- AT END [4-3](#)
- AT START [4-5](#)
- FOOTING [4-23](#)
- LIST [4-31](#)
- SUBFOOTING [4-43](#)
- SUBTITLE [4-45](#)
- TITLE [4-47](#)

syntax [5-66](#)with a LIST target-item or by-item [5-66](#)with a print list [5-66](#)Tabbing to a report column [5-66](#)

TACL commands

- ASSIGN [2-4](#), [2-7](#)
- ENFORM [2-1](#)
- FC [2-2](#)
- PARAM [2-5](#), [2-8](#), [4-39](#)
- PAUSE [2-4](#)
- QP [2-9](#)

TACL macros

- ENFOLD [2-2](#)
- ENFPLUS [2-2](#)

Tallying instances of an element [3-13](#)

Target aggregates

- description [3-15](#)
- examples [3-15](#), [3-16](#)
- rules for specifying [3-15](#)
- with OVER ALL syntax [3-15](#)
- with OVER syntax [3-16](#)

Target-items

- aggregates [3-11](#)
- centering [5-33](#)
- description [1-2](#)
- display format [5-7](#)
- displayed in report columns [4-32](#)
- headings [5-39](#)
- in a LIST statement [4-30](#)
- record names [3-6](#)
- user tables [3-29](#)
- user variables [3-27](#)

Target-list

- aggregates [3-15](#)
- description [1-2](#)
- literals [3-20](#)

Target-records

- description [1-3](#)
- generated by a LIST statement [4-31](#)
- grouping by field values [5-32](#)
- sorting [5-6](#), [5-32](#)
- summary records [4-21](#)
- summary reports [4-34](#)

Temporarily changing

- fill character [5-16](#)
- overflow character [5-16](#)

Temporary work files [2-11](#)**Terminating**

- current output listing file [2-4](#)
- Enform Plus session [2-3](#), [2-9](#), [4-15](#), [6-11](#)
- query output [2-4](#)

Terms of a WHERE clause [C-2](#)**Time**

- changing the display format
 - current report [5-24](#)
 - session-wide [5-53](#)
- current [5-68](#)
- default display format [5-24](#)
- obtaining the current time [5-65](#)

Time keywords [5-25](#)**Timestamp field** [5-67](#), [5-68](#)**TIMESTAMP-DATE clause**

- description [5-67](#)
- example [5-67](#)
- Guardian procedure TIMESTAMP [5-67](#)
- syntax [5-67](#)
- timestamp field [5-67](#)

TIMESTAMP-TIME clause

- description [5-68](#)
- example [5-68](#)
- Guardian procedure TIMESTAMP [5-68](#)
- syntax [5-68](#)
- timestamp field [5-68](#)

Timing out a server query processor [2-9](#)**TITLE clause**

- description [5-69](#)
- examples [5-69](#)
- for the current report [5-70](#)
- overriding a session-wide title [5-70](#)
- spacing considerations [5-69](#)
- syntax [5-69](#)
- with a field name [5-69](#)

TITLE statement

- canceling [4-48](#)
- description [4-47](#)
- examples [4-47](#), [4-48](#)
- overriding [4-48](#)
- resetting [4-48](#)
- spacing considerations [4-47](#)
- syntax [4-47](#)
- with a field name [4-47](#)

TOTAL clause

- description [5-71](#)
- syntax [5-71](#)
- width of the element modified [5-71](#)

Totaling a set of numbers [3-13](#)**Transitive property of links** [C-5](#)**Translating a date to internal format** [5-43](#)

Trap messages [B-26](#)

Two-directional links [C-1](#)

U

Underline character [5-53](#)

Unstructured files [3-7](#)

User aggregates

 declaring [4-9](#)

 deleting from the internal table [4-7](#),
 [4-13](#), [6-8](#)

 description [3-13](#)

 examples [3-14](#), [3-15](#)

 initial value [4-9](#)

 naming [3-6](#)

 rules for using [3-15](#), [4-10](#)

 syntax [3-14](#), [4-9](#)

User tables

 as target-items [3-29](#)

 assignment syntax [3-29](#)

 declaring [4-9](#)

 default display format [4-11](#)

 default storage format [4-10](#)

 deleting from the internal table [4-7](#),
 [4-12](#), [6-8](#)

 description [3-29](#)

 displaying current value of [6-17](#)

 initializing [4-41](#)

 maximum number of elements [3-29](#)

 naming [3-6](#)

 subscripts [3-8](#)

 user-specified display format [4-11](#)

 user-specified storage format [4-10](#)

User variables

 as target-items [3-27](#)

 assigning the result of an arithmetic
 expression [3-22](#)

 assignment syntax [3-28](#)

 declaring [4-9](#)

 default display format [4-11](#)

User variables (continued)

 default storage format [4-10](#)

 deleting from the internal table [4-7](#),
 [4-12](#), [6-8](#)

 description [3-27](#)

 displaying current value of [6-17](#)

 examples [3-28](#), [3-29](#)

 in request-qualifications [3-29](#)

 initial value [3-28](#)

 initializing [4-41](#)

 naming [3-6](#)

 used to hold percentage values [5-55](#)

 used to hold running totals [5-35](#)

 user-specified display format [4-11](#)

 user-specified storage format [4-10](#)

User-defined elements

 aggregates [3-13](#)

 declaring [4-9](#)

 initializing [4-41](#)

 naming [3-6](#)

 parameters [4-39](#)

 tables [3-29](#)

 variables [3-27](#)

V

Value keywords

 BLANK [3-26](#)

 BLANKS [3-26](#)

 NULL [3-26](#)

 ZERO [3-26](#)

 ZEROS [3-26](#)

Variables

See User variables, System Variables, or
 Option variables

Vertical spacing [5-56](#)

W

Warning messages

appearing on a terminal [5-53](#)

listing [B-4](#)

WHERE clause

and a qualification aggregate [3-19](#)

compared to LINK OPTIONAL statement [C-11](#)

compared to LINK statement [C-4](#), [C-11](#)

description [5-72](#)

examples [5-72](#)

links initiated by [C-2](#)

specifying a link [5-72](#)

syntax [5-72](#)

terms [4-28](#), [C-2](#)

Z

Zeroing fields on reports [3-26](#)

Special Characters

!!! ERROR messages [B-4](#)

*** FILE ERROR messages [B-22](#)

*** WARNING messages [B-4](#)

?ASSIGN command [2-4](#)

and generic files [6-3](#)

changing the exclusion specification [6-5](#)

description [6-3](#)

examples [6-4](#), [6-5](#)

syntax [6-3](#)

?ATTACH command

and a server query processor [2-6](#)

description [6-6](#)

failure [6-6](#)

syntax [6-6](#)

?COMPILE command

creating a physical file [6-7](#)

description [6-7](#)

identifying the physical file [6-7](#)

?COMPILE command (continued)

syntax [6-7](#)

?DICTIONARY command

clearing the internal table [6-8](#)

description [6-8](#)

syntax [6-8](#)

where the dictionary resides [6-8](#)

?EDIT command

description [6-9](#)

storing Enform Plus programs [6-9](#)

syntax [6-9](#)

?EXECUTE command

description [6-10](#)

syntax [6-10](#)

?EXIT command

description [6-11](#)

syntax [6-11](#)

?HELP command

description [6-12](#)

examples [6-12](#), [6-13](#)

syntax [6-12](#)

?OUT command

description [6-14](#)

syntax [6-14](#)

?OUT file [2-3](#), [6-14](#)

?RUN command

description [6-15](#)

restrictions [6-15](#)

syntax [6-15](#)

?SECTION command

description [6-16](#)

rules for naming a section [6-16](#)

syntax [6-16](#)

?SHOW command

description [6-17](#)

information displayed [6-17](#), [6-18](#)

syntax [6-17](#)

- ?SOURCE command
 - description [6-19](#)
 - nesting of [6-19](#)
 - syntax [6-19](#)
- @BLANK-WHEN-ZERO
 - description [5-49](#)
 - setting [4-42](#)
- @BREAK-KEY
 - and the Break key [2-4](#)
 - description [5-49](#)
 - setting [4-42](#)
- @CENTER-PAGE
 - description [5-49](#)
 - setting [4-42](#)
- @COPIES
 - description [5-49](#)
 - setting [4-42](#)
- @COST-TOLERANCE
 - description [5-49](#)
 - setting [4-42](#)
- @DATE [5-65](#)
 - See also* System Variable clauses
- @DATE-FORMAT
 - description [5-50](#)
 - setting [4-42](#)
- @DECIMAL
 - description [5-50](#)
 - setting [4-42](#)
- @DISPLAY-COUNT
 - description [5-50](#)
 - setting [4-42](#)
- @HEADING
 - description [5-51](#)
 - setting [4-42](#)
- @LINENO [5-65](#)
 - See also* System Variable clauses
- @LINES
 - description [5-51](#)
 - setting [4-42](#)
- @MARGIN
 - description [5-51](#)
 - setting [4-42](#)
- @NEWLINE
 - description [5-51](#)
 - setting [4-42](#)
- @NONPRINT-REPLACE
 - description [5-51](#)
 - setting [4-42](#)
- @OVERFLOW
 - description [5-51](#)
 - setting [4-42](#)
- @PAGENO [5-65](#)
 - See also* System Variable clauses
- @PAGES
 - description [5-51](#)
 - setting [4-42](#)
- @PRIMARY-EXTENT-SIZE
 - description [5-51](#)
 - setting [4-42](#)
- @READS
 - description [5-52](#)
 - setting [4-42](#)
- @SECONDARY-EXTENT-SIZE
 - description [5-52](#)
 - setting [4-42](#)
- @SPACE
 - description [5-52](#)
 - setting [4-42](#)
- @STATS
 - description [5-52](#)
 - setting [4-42](#)
- @SUBTOTAL-LABEL
 - description [5-52](#)
 - setting [4-42](#)
- @SUMMARY-ONLY
 - and FIND files [4-21](#)
 - and summary records [4-21](#)
 - and summary reports [4-34](#)

@SUMMARY-ONLY (continued)description [5-53](#)setting [4-42](#)**@TARGET-RECORDS**description [5-53](#)setting [4-42](#)**@TIME** [5-65](#)*See also* System Variable clauses**@TIME-FORMAT**description [5-53](#)setting [4-42](#)**@UNDERLINE**description [5-53](#)setting [4-42](#)**@VSPACE**description [5-53](#)setting [4-42](#)**@WARN**description [5-53](#)setting [4-42](#)**@WIDTH**description [5-53](#)setting [4-42](#)

