Eloquence Report Writer Manual
B.06.32

Edition E1202
© Copyright 2002 Marxmeier Software AG.

Legal Notices

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

MARXMEIER SOFTWARE AG MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Marxmeier Software AG shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

This document contains proprietary information which is protected by copyright. All rights reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws

Restricted Rights Legend

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

Acknowledgments

© Copyright Marxmeier Software AG 2002. All Rights Reserved.

Marxmeier Software AG
Besenbruchstrasse 9
42285 Wuppertal
Germany

Eloquence is a trademark of Marxmeier Software AG in the US and other countries.

© Copyright Hewlett-Packard Company 1990-2002. All Rights Reserved.

This software and documentation are based in part on HP software and documentation under license from Hewlett-Packard Company. HP is a trademark of Hewlett-Packard Company.

Printing History

The manual printing date indicates its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. New editions are complete revisions of the manual.The dates on the title page change only when a new edition or a new update is published.

Manual updates may be issued between editions to correct errors or document product changes. Manuals that are published on the Eloquence website (www.hp-eloquence.com/doc) may be updated more often, please visit this website periodically for the most recent versions. To ensure that you receive the updated or new editions, you should also subscribe to the appropriate product support service.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition	Apr 1990	A.01.00
Second Edition	July 1991	A.01.00
Third Edition	January 1997	A.06.00
Fourth Edition	October 1997	A.06.00
Fifth Edition (E1202)	December 2002	B.06.32

Printed in the Federal Republic of Germany.

Printing History

Table of Contents

Contents

Contents

Contents

1

Introduction

Eloquence Report Writer is a combination of statements and functions which aid the programmer in producing reports. Standard report features such as headings, page numbers, totals, averages and logical breaks can be programmed using Eloquence Report Writer. The operations are broken into three distinct groups:

Report Description Statements

Described in page 15 , these statements appear in the report description section of a program. They describe the image of the report, including headings, breaks, trailing comments and page size.

Report Execution Statements

Described in page 43 , these statements appear anywhere within the program, although they are only executed when a report is active. They cause the report description statements to be evaluated and cause the report details to be printed.

Functions

Described in page 53 , these functions are used to hold information about the report including totals, averages, the last break or detail executed, and the page and line numbers last written.

The Eloquence Report Writer statements work together to produce a report. One statement will cause another statement to be executed which in turn may cause another to be executed. To understand the relationship between statements, you should read the entire manual. Later, when you understand Eloquence Report Writer, you can use page 77 for a quick reference to the syntax of individual statements.

NOTE: This manual describes operation of the Eloquence Report Writer software. The Eloquence Report Writer software is packaged with the Eloquence software.

Definitions

Some of the terminology used with the Eloquence Report Writer may be unfamiliar to you. Several terms are defined here for your understanding.

HEADER A header is a description of a series of operations to perform at the time a new break level is begun. For example, when a new page is begun the header might print the report name at the top of the page.

TRAILER A trailer is a description of a series of operations to perform at the end of the current break level after the break condition has been triggered. It is executed prior to the execution of the header routine for the new break level. For example, when a page break occurs, the trailer might print the page number at the bottom of the page before printing the heading for the new page.

BREAK A break is caused by a change in the value of one of the variables. You define this variable and the change required to the Eloquence Report Writer. A break indicates special action is to take place. For example, assume you specify 60 lines to a page. When the Eloquence Report Writer is about to output the 61st line, a break will occur.

LEVEL Level is the means of establishing a hierarchical order to the report breaks. The highest level number represents the most frequent breaks and the lowest level number represents the least frequent breaks. For example, in a payroll report the highest level break might be a department and the lowest level be a division. There are nine possible levels, one being the lowest and nine being the highest.

DETAIL LINE A detail line is the lowest logical group of data in a report. For example, in an inventory report the detail line is the output associated with a particular item (i.e., part number, description, quantity, etc.). The detail line is the primary trigger for all report breaks.

BLOCK STATEMENTS These statements follow a header or trailer statement. They

may be thought of as a subroutine automatically invoked by a break condition. The end of the block is indicated by the next execution of an Eloquence Report Writer header, trailer or end statement.

NOTE:

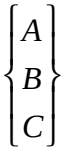
The PRINT USING statement is not a specific Eloquence Report Writer statement. However, its syntax is similar to many of the output statements in Eloquence Report Writer. It is therefore very important that you have a thorough understanding of PRINT USING and IMAGE as described in the output chapter in the *Eloquence Manual*.

Conventions

The statements in this manual use the same syntax conventions as in the *Eloquence Manual*.

- **Bold type** is used when a new term is introduced.
- **`Computer font`** indicates text to be input exactly as shown or text that is output from the system.
- *Italic type* is used for emphasis and titles of publications. It is also used to indicate parameters that are user defined.
- KEYCAP represents a key on the keyboard.
- Shading represents the softkeys displayed on the computer screen.
- …indicates that the previous variable can be repeated.
- [ ] indicates that information inside the brackets is optional. If there are brackets within brackets, the information within the inner bracket may only be specified if the information in the outer bracket is specified. Information may also be stacked in brackets. For example, A or B or neither may be selected when the following is shown:

- { } indicates that one of the choices stacked within the braces must be selected. For example, A or B or C must be selected when the following is shown:



NOTE: Notes contain important information that is set off from the text.

2

Report Description Statements

All the statements in this chapter must be in a program section that begins with the REPORT HEADER statement and ends with the END REPORT DESCRIPTION statement. This block or group of statements may be anywhere within the program. If more than one report is to be written, several description blocks can be within the program.

During execution of the program, when the REPORT HEADER statement is read, Eloquence Report Writer looks for the END REPORT DESCRIPTION statement and skips execution to the statement following it. Therefore, a branching statement is not required before the report description section.

In page 65 , an example report program and resultant report are given. Many of the examples are taken from this report. Note that the example is on a fold-out page to allow for better referencing while you read.

The REPORT HEADER Statement

The REPORT HEADER statement has two purposes. The first is to indicate the beginning of the report description section. The second is to write a one-time heading on the first page of the report. The header is printed when the first DETAIL LINE, TRIGGER PAGE BREAK or TRIGGER BREAK statement is executed. (These statements are described in page 43 .) It is printed before the first PAGE HEADER.

The syntax of this statement is:

[*label*:]REPORT HEADER [WITH *number* LINES]

$$\left[\ \text{USING}\ \left\{\begin{matrix} \textit{line id} \\ \textit{image string} \end{matrix}\right\}\ [;\textit{list}]\ \right]\ [\textit{block statements}]$$

The parameters for this statement are defined as follows:

label	This is the standard Eloquence line label. This label is referred to in the BEGIN REPORT statement. If not used, the BEGIN REPORT statement must refer to the line number of the REPORT HEADER statement.
number	The number of lines required for the header. If not entered, the value of number defaults to 1.
line id	A line number or label referencing an IMAGE statement.
image string	A list of image specifiers describing the output format, and enclosed in quotes.
list	This may be a list of string, numeric or array variables, literals, constants or expressions to be printed according to the image referenced.
block statements	Statements may follow the REPORT HEADER statement. They will be evaluated and executed immediately after the HEADER statement.

Some examples of these statements follow.

```
60  Inv: REPORT HEADER USING R_head
.
.
```

.

```
600 R_head: IMAGE 20X,"XYZ COMPANY INVENTORY",2/
```

The initial heading (XYZ COMPANY INVENTORY) is printed beginning at column 21. Two blank lines follow. The BEGIN REPORT statement (described in page 43) may reference line 60 or label Inv.

```
20    REPORT HEADER
```

With no block statements, there is no initial heading. The BEGIN REPORT statement must reference line 20.

```
90  Inv: REPORT HEADER USING "20X,K,2/ ";"XYZ COMPANY INVENTORY"
```

This example does exactly the same thing as the first example.

The PAGE HEADER Statement

The PAGE HEADER statement defines what is to be done at the top of each page. It is triggered by a page break condition. The syntax is:

PAGE HEADER [WITH *number* LINES]

$$\left[\ \text{USING}\ \left\{\begin{matrix}\textit{line id}\\ \textit{image string}\end{matrix}\right\}\ [;\textit{list}]\ \right]\ [\textit{block statements}]$$

The parameters are:

number	The number of lines required for the header. If not entered, the value of number defaults to 1. The number is evaluated during execution of the BEGIN REPORT statement. The number of lines for the PAGE HEADER and the PAGE TRAILER are used in conjunction with PAGE LENGTH to determine the effective page size.
line id	A line number or label referencing an IMAGE statement.
image string	A list of image specifiers describing the output format and enclosed in quotes.
list	This may be a list of string, numeric or array variables, literals, constants or expressions to be printed according to the image referenced.
block statements	Statements may follow the PAGE HEADER statement. They will be evaluated and executed immediately after the PAGE HEADER statement.

An example of this statement follows.

```
130  PAGE HEADER USING P_head;Date$
140  PRINT USING P_2hd
.
.
.
620  P_head:IMAGE 60X,8A,2/,10X,"PART","QUANTITY",10X,"UNIT",/,8X,
     "NUMBER",8X,"ON HAND",9X,"PRICE",8X,"VALUE"
630  P_2hd:  IMAGE 9X,"====", 8X,"=======", 8X,"====",8X,"===="
```

Since two program lines of image specifiers are needed to show the format, the PAGE HEADER statement refers to the first line (P_head) and the PRINT USING statement, which is the “block statement”, refers to the second line (P_2hd).

The result of this example is shown in page 65 “Example Report Program”.

The HEADER Statement

The HEADER statement defines what is to be done as a heading routine for a specified break condition. Each HEADER statement is directly associated with a break condition. There may be only one active HEADER statement for each of the nine possible break levels.

HEADER *level* [WITH *number* LINES]

$$\left[\ \text{USING}\ \left\{\begin{matrix}\textit{line id}\\ \textit{image string}\end{matrix}\right\}\ [;\textit{list}]\ \right]\ [\textit{block statements}]$$

The parameters are:

level	A numeric expression evaluating to an integer from 1 to 9. This integer associates the header to a break condition. If level is 0, the HEADER statement will be ignored.
number	The number of lines required for the header. If not entered, the value of number defaults to 1. The number is evaluated every time the HEADER statement is executed and can be dynamically changed.
line id	A line number or label referencing an IMAGE statement.
image string	A list of image specifiers describing the output format and enclosed in quotes.
list	This may be a list of string, numeric or array variables, literals, constants or expressions to be printed according to the image referenced.
block statements	Statements may follow the HEADER statement. They will be evaluated immediately after the HEADER statement.

Some examples of this statement follow.

```
170  HEADER 1
180    TOTALS ON Qty*Price
190    PRINT USING P_head1;Div$
.
.
.
670  P_head1:  IMAGE /,2A,2X, "DIVISION"
```

The TOTALS ON statement is described later in this chapter.

```
220  HEADER 3 USING P_head3
230    TOTALS ON Qty*Price
.
.
.
710  P_head3:  IMAGE /
```

The REPORT TRAILER Statement

The REPORT TRAILER statement is used to print a one-time trailer just prior to report termination. It is invoked by execution of the END REPORT statement, and is executed after all TRAILERS but prior to the last PAGE TRAILER.

REPORT TRAILER [WITH *number* LINES]

$$\left[\ \text{USING}\ \left\{\begin{matrix}\textit{line id}\\ \textit{image string}\end{matrix}\right\}\ [;\textit{list}]\ \right]\ [\textit{block statements}]$$

The parameters are:

number	The number of lines required for the trailer. If not entered, the value of number defaults to 1. If the report trailer requires more than one line, it is important that this parameter be used to ensure that the trailer be printed on one page. If a page break occurs in the middle of a trailer, an error will occur.
line id	A line number or label referencing an IMAGE statement.
image string	A list of image specifiers describing the output format and enclosed in quotes.
list	This may be a list of string, numeric or array variables, literals, constants or expressions to be printed according to the image referenced.
block statements	Statements may follow the REPORT TRAILER statements. They will be evaluated and executed immediately after the TRAILER statement.

An example of this statement follows:

```
160  REPORT TRAILER WITH 5 LINES USING  R_trail; TOTAL(0,1), TOTAL
(0,1),
     /NUMBREAK(1)
170  PRINT USING R_2tr; AVG(0,1)
.
.
.
650  R_trail:  IMAGE 2/,22x,"TOTAL COMPANY" 10x,DCDDDCDDDPDD,/
,22x,
               "AVG PER " "DIVISION",7x,DCDDDCDDDPDD
656  R_2tr:    IMAGE 22X,"AVG PER ITEM", 11X, DDDCDDDPDD
```

The TOTAL, NUMBREAK, and AVG functions are described in page 53 .

The PAGE TRAILER Statement

The PAGE TRAILER statement defines the course of action to be taken at the bottom of each page. It is triggered by a page break condition.

PAGE TRAILER [WITH *number* LINES]

$$\left[\ \text{USING}\ \left\{\begin{matrix}\textit{line id}\\ \textit{image string}\end{matrix}\right\}\ [;\textit{list}]\ \right]\ [\textit{block statements}]$$

The parameters for this statement are defined as follows:

number	The number of lines required for the trailer. If not entered, the value of number defaults to 1. If the page trailer requires more than one line, it is important that this parameter be used to ensure correct page alignment. The number is evaluated during execution of the BEGIN REPORT statement. The number of lines for the PAGE HEADER and the PAGE TRAILER are used in conjunction with PAGE LENGTH to determine the effective page size.
line id	A line number or label referencing an IMAGE statement.
image string	A list of image specifiers describing the output format and enclosed in quotes.
list	This may be a list of string, numeric or array variables, literals, constants or expressions to be printed according to the image referenced.
block statements	Statements may follow the PAGE TRAILER statement. They will be evaluated and executed immediately after the PAGE TRAILER statement.

An example of this statement follows.

```
150  PAGE TRAILER WITH 2 LINES USING P_trail;NUMPAGE
.
.
.
650  P_trail: IMAGE 60X,"PAGE",X,2D
```

This page trailer only requires one line but an extra blank line will follow because two lines are reserved by the WITH 2 LINES parameter. The NUMPAGE function is described in page 53 .

The TRAILER Statement

The TRAILER statement defines what is to be done as a follow up operation for a specified break condition. Each TRAILER statement is directly associated with a break condition. There may be only one active TRAILER statement for each of the nine possible break levels.

TRAILER *level* [WITH *number* LINES]

$$\left[\ \text{USING}\ \left\{\begin{matrix}\textit{line id}\\ \textit{image string}\end{matrix}\right\}\ [;\textit{list}]\ \right]\ [\textit{block statements}]$$

The parameters are:

level	A numeric expression evaluating to an integer from 0 to 9. This integer associates the trailer to a break level.
number	The number of lines required for the trailer. If not entered, the value of number defaults to 1. If the trailer requires more than one line, it is important that this parameter be used to ensure that a page break does not occur in the middle of the trailer. If a page break occurs in the middle of a trailer (because the maximum number of lines per page has been printed), an error will result. The number is evaluated every time the TRAILER statement is executed and can be dynamically changed.
line id	A line number or label referencing an IMAGE statement.
image string	A list of image specifiers describing the output format and enclosed in quotes.
list	This may be a list of string, numeric or array variables, literals, constants or expressions to be printed according to the image referenced.
block statements	Statements may follow the TRAILER statement. They will be evaluated and executed immediately after the TRAILER statement.

Some examples of this statement follow.

```
200  TRAILER 1 USING P_trail1;OLDCV$(1),TOTAL(1,1)
.
```

```
.
.
690  P_trail: IMAGE ”TOTAL”,X,2A,X,”DIVISION”,26X,DDDCDDDCDDDPDD
```

The OLDCV$ and TOTAL functions are described in page 53 . This trailer only requires one line, so the WITH…LINES parameter is not entered.

```
240  TRAILER 3 WITH 2 LINES USING P_trail3;OLDCV(3),TOTAL(3,1)
.
.
.
730  P_trail3: IMAGE /,”TOTAL DEPT”,X,2A,29X,DDDCDDDCDDDPDD
```

This trailer requires two lines as stated in line 240. If it were not stated and a page break occurred after the first line was printed, an error would occur.

The BREAK WHEN Statement

The BREAK WHEN statement establishes the criteria for determining the level break condition. This break, in turn, will invoke the break condition for any higher numbered levels. While active, a break condition will trigger execution of the appropriate header/trailer statement blocks. There may be one active BREAK WHEN statement for each of the nine possible break levels.

BREAK *level* WHEN *control* CHANGES [BY *increment*]

The parameters are:

level A numeric expression evaluating to an integer from 0 to 9. This break level associates the BREAK WHEN statement to header/trailer statement. If level is 0, the BREAK WHEN statement will be ignored.

control A numeric or string variable or expression evaluated during execution of the DETAIL LINE statement which is used to trigger the break condition.

A specific amount of space is reserved for a string expression control. If the string expression has a current length of greater than zero at execution of the BEGIN REPORT statement, this length is used. Otherwise, a default length of 18 will be used. For string variable controls, the dimensioned length will always be used. If that length is exceeded during the execution of the Eloquence Report Writer, an error will occur.

increment A numeric expression evaluated at the execution of the BEGIN REPORT statement which establishes the incremental change necessary in control to trigger the break condition. The increment parameter may only be used for a numeric control. If not present, any change will trigger the break condition. The value of increment is rounded to the nearest integer (i.e., .01 becomes 0 which will cause an error; 1.6 becomes 2 and 2.1 becomes 2).

The break limit is set when the control variable is first given a value. The value of the increment parameter is multiplied by an integer such that it is just larger (or just smaller if increment is negative) than the value of the control variable. For example, if increment =10 and the control variable =27, then increment is multiplied by 3 for a break limit of 30. If increment is -10 and control is 5, then the break limit is 0.

Once the break limit is set, it does not change until after a break occurs. (A break occurs either because the control variable changes past the increment, or because a TRIGGER BREAK statement was executed). The break limit changes by a multiple of the increment value so that it is just larger (or smaller) than the value of the control variable.

Examples of this statement follow.

```
100  BREAK 1 WHEN Div$ CHANGES
110  BREAK 3 WHEN Dept$ CHANGES
.
.
.
460  Div$=C$[1,2]
470  Dept$=C$[3,4]
```

Lines 460 and 470 are not in the description section.

```
200  BREAK 2 WHEN Dim CHANGES BY 10
```

If the value of Dim is initially 4, a break will occur when the value of Dim is equal to or greater than 10.

```
300  BREAK 4 WHEN Sam CHANGES BY 2
```

If the value of Sam is initially 5 then changes to 1, no break will occur. When the value changes to 6 or greater, a break will occur.

```
400  BREAK 5 WHEN Alpha CHANGES BY -1
```

If the initial value of Alpha is 3 then the value changes to 2 or less, a break will occur.

The PAGE LENGTH Statement

The PAGE LENGTH statement sets the vertical size of the page. The number of blank lines to appear at the top or bottom of the page may optionally be established. To arrive at effective or usable page size, subtract the total number of blank lines plus the number of lines required for the page header and page trailer from the lines per page. The result must be greater than three. All expressions are evaluated at the time of the BEGIN REPORT statement's execution.

PAGE LENGHT *lines per page* [,*blank lines top* ,*blank lines bottom*]

The parameters are:

lines per page The number of lines per page. If not entered, it defaults to 66 lines with two blank lines at the top and bottom of the page. If lines per page evaluates to zero, no blanks are printed at the top or bottom of a page, a page heading is printed when the first detail line is executed, a page trailer is printed with the END REPORT, the user is responsible for invoking all page breaks by using the TRIGGER PAGE BREAK statement and the NUMLINE function will accumulate the number of lines printed up to 32767.

blank lines top The number of blank lines to be output at the top of every page. The default is 2. The maximum is 255.

blank lines bottom The number of blank lines to be output at the bottom of every page. The default is 2. The maximum is 255.

The LEFT MARGIN Statement

The LEFT MARGIN statement may be used to horizontally shift a report on a page. It establishes the leftmost column of a line which may be used for printing. This statement only applies when output is going to a printer. For reports to a CRT display, the left margin will always be 1.

LEFT MARGIN *column*

The parameter is:

column A numeric expression which evaluates to an integer from 1 to the current printer width less one (or 132, whichever is less). It defines the leftmost column from which to reference all image execution. The value of column - 1 will be added to the length of each printed line. If LEFT MARGIN is not present, column defaults to 1.

The PAUSE AFTER Statement

The PAUSE AFTER statement temporarily pauses the execution of the report, allowing either the printing of reports on individual pages or viewing on the display. To resume execution of the report, you must press the RETURN key.

PAUSE AFTER *number* PAGES

The parameter is:

number A numeric expression which evaluates to an integer between 0 and 32767. If evaluated to be zero, the PAUSE AFTER will have no effect.

The statement PAUSE AFTER 1 PAGES causes one page to print followed by a pause, the next page to print followed by a pause, etc.

The SUPPRESS PRINT FOR Statement

The SUPPRESS PRINT FOR statement provides a means of inhibiting print for a specified number of pages at the beginning of a report. Its primary purpose is to provide for restart in case of power or mechanical printer failures. Note that it is still necessary to re-run the report from its beginning to that totals may be accumulated correctly even though printout will not appear for the specified number of pages.

The SUPPRESS PRINT FOR statement performs its execution by altering the select code for the standard printer as set in the PRINTER IS statement. Use of the PRINTER IS statement while a report is active will cause an error 270 to occur.

SUPPRESS PRINT FOR *number* PAGES

The parameter is:

number A numeric expression evaluating to a positive integer. If set to 0, the statement is ignored.

The SUPPRESS PRINT AT Statement

The SUPPRESS PRINT AT statement allows reports to be produced at summary levels. Headers and trailers for levels lower than the SUPPRESS PRINT AT level are executed. Those with equal or higher levels are not. Except for the printout reduction, the report is produced exactly as if all details were being printed (i.e., the detail line must still be executed). Only one SUPPRESS PRINT AT statement may appear in a report description.

SUPPRESS PRINT AT *level*

The parameter is:

level A numeric expression evaluating to an integer from 1 to 9. Only headers and trailers with a lower level will be executed.

An example of this statement follows.

```
220  SUPPRESS PRINT AT 4
```

The headers and trailers with levels 1, 2 and 3 are printed. No detail lines are printed. Headers and trailers with levels 4 through 9 are not printed. Page headers and page trailers are not affected. Example 2 in page 65 shows a report with the SUPPRESS PRINT AT statement used.

The PRINT DETAIL IF Statement

The PRINT DETAIL IF statement allows printing conditions for detail lines to be specified without affecting the totaling functions.

PRINT DETAIL IF *conditional exp*

The parameter is:

conditional exp An expression which will evaluate to zero if false. Otherwise, it will be assumed to be true. It is evaluated at every execution of the DETAIL LINE statement and if zero, the printout for that line will be suppressed. The line's data, however, will be included in all totaling functions.

If the DETAIL LINE statement requires more than 180 characters (including non-printing characters) to be defined, use a DETAIL LINE 0 to continue the statement. This will ensure that a PRINT DETAIL IF statement will execute correctly.

Some examples of this statement follow.

```
90  PRINT DETAIL IF Amt>200
```

When the DETAIL LINE is encountered, the variable Amt is evaluated. If Amt is greater than 200, the detail line is printed.

```
90  PRINT DETAIL IF Security%<10
```

If the variable Security is equal or greater than 10, the detail line is not printed.

The GRAND TOTALS ON Statement

The GRAND TOTALS ON statement provides a means of having the Eloquence Report Writer do automatic totaling for the entire report. The GRAND TOTALS ON statement may only appear once, directly following the REPORT HEADER statement. Once executed, the GRAND TOTALS ON statement allows two special functions to be used: TOTAL and AVG (described in page 53). All totals are zero at the beginning of the report execution and are accessible at any time while the report is active. Totals are incremented during the execution of a DETAIL LINE 1-9 after all breaks are serviced but before control is passed to the USING part of the statement.

GRAND TOTALS ON exp_1 [,exp_2 ...]

The parameter is:

exp Any valid number variable or expression.

An example of this statement follows.

```
60  Inv: REPORT HEADER
70       GRAND TOTALS ON Qty,Qty$\ast$Price
```

Each time a DETAIL LINE statement (with a number of 1 through 9) is encountered the values of Qty and Qty$\ast$Price are added to the totals kept by the GRAND TOTALS ON statement.

The TOTALS ON Statement

The TOTALS ON statement provides a means of having the Eloquence Report Writer do automatic totaling. The TOTALS ON statement may only appear within a header statement block and additionally must immediately follow the header statement. Only one TOTALS ON statement may be present in a header statement block. Once executed, the TOTALS ON statement allow two special functions to be used: TOTAL and AVG (described in page 53). Totals are incremented during execution of a DETAIL LINE (1-9) statement after all breaks are serviced but before control is passed to the USING part of the statement. Totals are zeroed before control is passed to their associated HEADER but they remain available and unchanged in all TRAILERS when a level break occurs.

TOTALS ON exp_1 [,exp_2 ...]

The parameter is:

exp Any valid numeric variable or expression.

An example of this statement follows.

```
170  HEADER 1
180    TOTALS ON Qty$\ast$Price
.
.
.
220  HEADER 3
230    TOTALS ON Qty$\ast$Price,Qty
```

The TOTALS ON statement keeps a running total of Qty$\ast$Price beginning when HEADER 1 is first executed and resetting each time thereafter at each execution of HEADER 1. The statement in line 230 keeps the same total and a running total of Qty but is governed by HEADER 3.

The REPORT EXIT Statement

The REPORT EXIT statement defines what action is to be taken when a report is prematurely stopped. It is triggered by the execution of a STOP REPORT statement in the local program. After the REPORT EXIT section is completed, blank lines are printed to ensure that a complete page is ejected.

REPORT EXIT (*exec flag*) [WITH *number* LINES]

[USING { *line id* / *image string* } [;*list*]] [*block statements*]

The parameters are:

exec flag	An integer expression which is evaluated after the STOP REPORT is executed. If it evaluates to zero, the REPORT EXIT will not be executed. If it evaluates to any non-zero value, the REPORT EXIT will be performed.
number	The number of lines required for the exit routine. If not entered, the value of number defaults to 1. If the routine requires more than one line, it is important that this parameter be used to ensure correct page alignment.
line id	A line number or label referencing an IMAGE statement.
image string	A list of image specifiers describing the output format and enclosed in quotes.
list	This may be a list of string, numeric or array variables, literals, constants or expressions to be printed according to the image referenced.
block statements	Statements may follow the REPORT EXIT statement. They will be evaluated and executed immediately after the REPORT EXIT statement.

An example of this statement follows:

```
100    REPORT EXIT (NUMLINES(0)>0) WITH 2 LINES USING"/
,20X,K" " *** USER TERMINATED REPORT ***"
110    END REPORT DESCRIPTION
.
.
.
```

```
200    KEY #8 ”EXIT REPORT” GOTO Rexit
.
.
.
300 Rexit:  OFF KEY #8
310         STOP REPORT
320         GOTO Eoj
```

The END REPORT DESCRIPTION Statement

The END REPORT DESCRIPTION statement's purpose is to terminate the report description section begun by a REPORT HEADER statement. It is required for all reports.

The syntax for this statement is:

```
END REPORT DESCRIPTION
```

3

Report Execution Statements

The execution statements, beginning with BEGIN REPORT and ending with END REPORT, activate the report. Execution statements, with the exception of STOP REPORT, may only be executed in the program that initiated the current report by executing a BEGIN REPORT statement.

In page 65 an example report program and resultant report are given. Many of the examples are taken from this report. Note that the example is on a fold out page to allow for better referencing while you read.

The BEGIN REPORT Statement

The BEGIN REPORT statement initiates execution. When executed, the Eloquence Report Writer scans the report description section referenced by this statement and evaluates the non-dynamic report options in the section. If no errors are detected, all Eloquence Report Writer variables and controls are established. Only the variables which can change during the report execution are evaluated later (such as PRINT DETAIL IF condition, or header and trailer WITH number LINES parameters). Only one report may be active at a time.

BEGIN REPORT *line id*

The parameter is:

line id The line identifier (line number or line label) referencing a REPORT HEADER statement.

The DETAIL LINE Statement

The DETAIL LINE statement is the foundation around which the Eloquence Report Writer runs. When the DETAIL LINE statement is executed, all break conditions are tested and triggered if appropriate before any output associated with the DETAIL LINE is printed. In addition, this statement causes all totals to be incremented. Upon the first execution of the DETAIL LINE statement, the Eloquence Report Writer triggers the report header, page header and all header statements in ascending level-number sequence.

DETAIL LINE *line* [WITH *number* LINES]

$$\left[\ \text{USING}\ \left\{\begin{matrix} \textit{line id} \\ \textit{image string} \end{matrix}\right\}\ [;\textit{list}]\ \right]$$

The parameters are:

line	A numeric expression evaluating to an integer between 0 and 9. If greater than zero, totals are incremented, Eloquence Report Writer counters are incremented and a check for break conditions is made. If equal to zero, the above items are not done. If a DETAIL LINE statement will not fit on two program lines, a second DETAIL LINE statement can be used by setting the line parameter equal to one (or greater) in the first statement and equal to 0 in the second statement.
number	The number of lines required for output. If not entered, the value defaults to 1. If more than one line is required, it is important to enter the number so that a page break does not occur which would cause an error.
line id	A line number or label referencing an IMAGE statement.
image string	A string that contains an image format.
list	A list of numeric, string or array variables, constants, literals or expressions to be printed according to the image referenced.

Examples of this statement follow.

```
520  DETAIL LINE 1 USING D_line;Part$, Qty, Price, Qty*Price
.
.
```

```
.
770  D_line:  IMAGE 9X,8A,8X,DCDDD,5X,DDCDDDPDD,3X,DDDCDDDPDD
```

This example causes the values of Part$, Qty, Price and Qty$\ast$Price to be printed in the format shown.

```
100  DETAIL LINE 1
110  IF A=1 THEN GOTO X
120  IF A=2 THEN GOTO Y
130  IF A=3 THEN GOTO Z
.
.
.
200  X: DETAIL LINE 0 USING D1;A,B,C,D
.
.
.
300  Y: DETAIL LINE 0 USING D2;A,B,C,D
.
.
.
400  Z: DETAIL LINE 0 USING D3;A,B,C,D
```

In this example, all break conditions are checked and totals incremented at line 100, but no detail is output. The value of A determines the image used for the detail line. Line 200, 300, or 400 is used to print the line. When it is printed, totals are not incremented nor are break conditions checked.

The TRIGGER BREAK Statement

The TRIGGER BREAK statement provides a means of forcing a break condition that cannot be described with a BREAK WHEN statement. The break condition is treated in the same manner as if the control in the BREAK WHEN statement had invoked the break condition. The TRIGGER BREAK statement may not be executed while a level or page break is active.

TRIGGER BREAK *level*

The parameter is:

level A numeric expression evaluating to an integer from 1 to 9. This number associates the TRIGGER BREAK or BREAK WHEN header and/or trailer statements of the same level. All trailer and header statements with greater or equal levels are triggered. Refer to page 61 , Execution Hierarchy, for the order each statement is triggered.

The TRIGGER PAGE BREAK Statement

The TRIGGER PAGE BREAK statement allows a page break condition to be forced under circumstances other than the end of page. This statement initiates an immediate break to a new page. Page trailers and page headers are printed. The TRIGGER PAGE BREAK statement may not be executed if a page break is already active.

```
TRIGGER PAGE BREAK
```

The NUMPAGE = Statement

The NUMPAGE = statement allows the value of the Eloquence Report Writer page counter to be changed during the execution of the report. The page counter is normally incremented by Eloquence Report Writer just prior to the printing of a page header.

NUMPAGE = *number*

The parameter is:

number A numeric expression evaluating to an integer value between 0 and 32767.

The END REPORT Statement

The END REPORT statement terminates the active report. It triggers all trailer statements in descending level number sequence, the report trailer and lastly the page trailer.

```
END REPORT
```

The STOP REPORT Statement

The STOP REPORT statement terminates the active report. The REPORT EXIT statement is executed if the following three conditions are true:

1. the STOP REPORT was executed from the program (not from the keyboard),
2. STOP REPORT is executed in the same program environment that initiated the current report (not a subprogram or function) and
3. the REPORT EXIT execution flag evaluates to a non-zero value.

In all other cases, STOP REPORT immediately terminates any further printing.

The STOP REPORT statement can be executed even if a report is not active. The only time an error will be generated is while the REPORT EXIT section is active.

```
STOP REPORT
```

4

Functions

The Eloquence Report Writer keeps track of totals, page numbers, line numbers, break conditions and the number of detail and breaks executed. These values can be accessed through the Eloquence Report Writer functions.

Eloquence Report Writer functions may be accessed by any subprogram and/or function internal to the program that initiated the current report by executing a BEGIN REPORT statement.

In page 65 , an example report program and resultant report are given. Many of the examples are taken from this report. Note that the example is on a fold out page to allow for better referencing while you read.

The AVG Function

The AVG function returns the average for the specified expression in the TOTALS ON statement. The value returned is a cumulative average since the header statement block, containing the TOTALS ON statement, was last invoked by break condition.

AVG (*level, sequence*)

The parameters are:

level A numeric expression evaluating to an integer from 0 to 9. It associates the AVG function to a TOTALS ON statement in a header with the same level number. If level equals zero, the AVG function will reference the GRAND TOTALS ON statement.

sequence A numeric expression evaluating to a positive integer. It corresponds to the sequential position of the desired expression in the TOTALS ON statement.

An example of this function follows.

```
70   GRAND TOTALS ON Qty*Price
.
.
.
265  REPORT TRAILER WITH 5 LINES USING R_trail;AVG(0,1)
```

The value returned is the grand total of Qty$\ast$Price divided by the number of detail lines (with number of one to nine) that have been executed.

The TOTAL Function

The TOTAL function returns the running total for the specified expression in a TOTALS ON statement. The value returned is the cumulative value since the header statement block, containing the TOTALS ON statement, was last invoked by a break condition.

TOTAL (*level*, *sequence*)

The parameters are:

level A numeric expression evaluating to an integer from 0 to 9. It associates the TOTAL function to a TOTALS ON statement in a header with the same level number. If level equals zero, the TOTAL function will reference the GRAND TOTALS ON statement.

sequence A numeric expression evaluating to a positive integer. It corresponds to the sequential position of the desired expression in the TOTALS ON statement.

An example of this function follows.

```
220  HEADER 3
230    TOTALS ON Qty*Price
240  TRAILER 3 USING P_trail3;TOTAL(3,1)
```

The NUMDETAIL Function

The NUMDETAIL function returns the number of DETAIL LINE statements (with number between one and nine) executed since the header statement at the specified level was executed. The number is incremented during execution of a DETAIL LINE (1-9) statement after all breaks are serviced but before control is passed to the USING part of the statement. When a level break occurs, the number is zeroed before control is passed to the associated HEADER but remains available and unchanged in all TRAILERS.

NUMDETAIL (*level*)

The parameter is:

level A numeric expression that evaluates to an integer from 0 to 9. It associates the NUMDETAIL with the header statement of the same level. If set to 0, the total number of detail lines (with number between one and nine) executed since the report became active is returned.

An example of this function follows.

```
60   HEADER 1
70     TOTALS ON Qty*Price
.
.
.
260  TRAILER 1 USING R_trail;TOTAL(1,1)/NUMDETAIL(1)
```

The NUMBREAK Function

The NUMBREAK function returns the number of times a break condition at the specified level has occurred. The NUMBREAK counter is incremented when the break is detected which is before any header routine is activated.

NUMBREAK (*level*)

The parameter is:

level A numeric expression evaluating to an integer from 1 to 9. It associates the NUMBREAK function with the BREAK WHEN statement of the same level.

An example of this function follows.

```
290  REPORT TRAILER WITH 5 LINES USING R_trail;TOTAL(0,1)/NUM-
BREAK(1)
```

The OLDCV Function

The OLDCV function returns the value of the control variable as it was evaluated in the last break condition. (The control variable is the parameter “control” in the BREAK WHEN statement.) Its primary purpose is to get the correct value of the control variable in trailer routines. The value changes when a break condition occurs. The old value of the control variable is changed to the new break value after all TRAILERS have been processed and before any HEADERS are called.

OLDCV [$] (*level*)

The parameters are:

$ The $ indicates that the control variable is a string. Without a $, it is assumed the control variable is numeric.

level A numeric expression that evaluates to an integer from 1 to 9. It associates the OLDCV function with the BREAK WHEN statement of the same level containing the control variable.

An example of this function follows.

```
240  TRAILER 3 WITH 2 LINES USING P_trail3;OLDCV(3),TOTAL(3,1)
```

The NUMPAGE Function

The NUMPAGE function returns the current Eloquence Report Writer page number. The Eloquence Report Writer page counter is incremented just before the PAGE HEADER s called. If the report contains a TRIGGER PAGE BREAK command, the page counter is incremented during the page break setup.

An example follows.

```
150  PAGE TRAILER USING P_trail;NUMPAGE
.
.
.
650  P_trail:  IMAGE 60X,"PAGE",2D
```

The NUMLINE function

The NUMLINE function returns the current Eloquence Report Writer line count.

An example follows.

```
500  DETAIL LINE 1 USING D_line;NUMLINE,Part*,Qty
```

The LAST BREAK Function

The LAST BREAK function returns the current value of the last break condition level number detected. If the value is zero, the break was triggered by the Eloquence Report Writer initialization. If the value is ten, the break was triggered by the Eloquence Report Writer termination (END REPORT statement).

An example follows.

```
250  IF LAST BREAK=3 THEN GOTO 100
```

The RWINFO Function

The general information function, RWINFO, provides a means of retrieving Eloquence Report Writer information. If a report is not active, a –1 is returned.

RWINFO (*integer*)

Integer Value	Information Returned
1	Page size.*
2	Effective page = Page size – (blank line top + blank lines bottom + lines in PAGE TRAILER).
3	Number of lines used in current page (same as NUMLINE).
4	Number of lines left in current page.*
5	Number of lines left in effective page.*
6	Page break cause flag: 0 = not caused by DETAIL LINE; 1 = caused by DETAIL LINE
7	Page count (same as NUMPAGE).
8	Number of pages left to suppress.
9	Number of logical pages produced.
10	Same as LAST BREAK.
11	Current LEFT MARGIN.
12	Current HEADER/TRAILER level if not in a break condition.

* Will return a zero if pagination is turned off (PAGE LENGTH = 0).

5

Execution Hierarchy

Three statements, BEGIN REPORT, END REPORT and DETAIL LINE, control the execution of the Eloquence Report Writer. These statements invoke break conditions which in turn trigger special actions.

The Eloquence Report Writer has a specific hierarchical order in which it executes the break conditions:

1 Page breaks always have the highest priority. Page trailers are usually processed before

page headers. Page breaks are generally triggered by the internal line counter independent of any other break condition.

2 Trailers have the second highest priority. They are processed in descending sequence from the highest level to the current break level.

3 Headers have the next priority. They are processed in ascending sequence from the current break level to the highest level.

4 A detail line has the lowest priority. It is processed after all break conditions have been serviced.

This hierarchy is shown in the following table.

Table 1 **Break Execution Hierarchy**

1st Detail Line	Detail Line Break	End Report
Report Header	Trailer 9	Trailer 9
Page Header	.	.
Header 1	.	.
.	.	.
.	Trailer n	Trailer 1
.	Header n	Report Trailer
Header 9	.	Page Trailer
	.	
	.	
	Header 9	

n=current break level

When the first DETAIL LINE statement is executed, the statements in column one are also executed (before the detail line is printed). When following DETAIL LINE statements (with a line not equal to zero) are executed and cause a break to occur, the statements in column two are executed. When the END REPORT statement is executed, the statements in column three are executed.

Between Trailer n and Header n, Eloquence Report Writer does the following for all break levels triggered:

- Resets the value of the control variables.

- Sets the Total On counters to zero.
- Sets the NUMDETAIL counters to zero.
- Increments the break counters.

After Header 9 is executed, Eloquence Report Writer does the following for Detail Lines with line greater than zero:

- Increments the detail line counter.
- Increments totals.
- Checks Print Suppression conditions (also done for line = 0).
- Checks for a Page Break condition (also done for line = 0).
- Prints the Detail Line (if no Print Suppression condition is valid) (also done for line = 0).

6

Example Report Programs

Example 1

The following program is an example of how the Eloquence Report Writer statements can be used to produce a report. This program accesses a file which is sorted first by division (Div$) and second by the department (Dept$).

```
10    !   This program accesses the file “INV” for data.
20    !   The data is output in a report.
30    !
40    !   Report Description Section
50    !
60 Inv:       REPORT HEADER USING R_head
70            GRAND TOTALS ON Qty*Price
80            PAGE LENGTH 66
100           BREAK 1 WHEN Div$ CHANGES
110           BREAK 3 WHEN Dept$ CHANGES
120   !
130           PAGE HEADER USING P_head;Date$
140                PRINT USING P_2hd
150           PAGE TRAILER WITH 2 LINES USING P_trail;NUMPAGE
160   !
170           HEADER 1
180             TOTALS ON Qty*Price
190             PRINT USING P_head1;Div$
200           TRAILER 1 USING P_trail1;OLDCV$(1),TOTAL(1,1)
210   !
220           HEADER 3 USING P_head3
230             TOTALS ON Qty*Price
240           TRAILER 3 WITH 2 LINES USING P_trail3;OLDCV$(3),
TOTAL(3,1)
250   !
260          REPORT TRAILER WITH 5 LINES USING R_trail;TOTAL(0,1),
TOTAL(0,1)/NUMBREAK(1)
265             PRINT USING R_2tr:AVG(0,1)
270   !
280           END REPORT DESCRIPTION
290   !
300   !
310   !
320           PRINT “ENTER DATA AS MM/DD/YY”
330           INPUT Date$
340           PRINTER IS 0
350           ASSIGN #1 TO “INV,FILES”
360           READ #1,1;X
370   !
380   !
390   !
400   !       Report Execution Section
410   !
420   !
430           BEGIN REPORT Inv
440           FOR I=2 TO X
450               READ #1,I;C$
460               Div$=C$[1,2]          ! Set value for BREAK 1
470               Dept$=C$[3,4]         ! Set value for BREAK 3
480               Part$=C$[5,9]
```

```
490               Qty=VAL(C$[10,12])
500               Price=VAL(C$[13,17])
510   !
520            DETAIL LINE 1 USING D_line1;Part$,Qty,Price,Qty*Price
530   !
540           NEXT I
550           END REPORT
560           PRINTER IS 8
570   !
580   !       Line Image Section
590   !
600 R_head:       IMAGE 20X,"XYZ COMPANY INVENTORY",2/
610   !
620 P_head:       IMAGE 60X,8A,2/,10X,"PART",9X,"QUANTY",10X,
"UNIT",/,9X,"NUMBER",8X,"ON HAND",9X,"PRICE",8X,"VALUE,/
630      P_2hd:                                                    IMAGE
9X,"======",8X,"========",8X,"=====",8X,"====="
640   !
650 P_trail:      IMAGE 60X,"PAGE",X,2D
660   !
670 P_head1:      IMAGE /,2A,2X,"DIVISION"
680   !
690 P_trail1:      IMAGE ,22X,"TOTAL",X,2A,X,"DIVISION",6X,
DCDDDCDDDPDD,/47X,10("-")
700   !
710 P_head3:      IMAGE /
720   !
730 P_trail3:     IMAGE /,22X,"TOTAL DEPT",2A,9X,DDCDDDCDDDPDD,/,
46X,"----------"
740   !
750 R_trail:      IMAGE 2/,22X,"TOTAL COMPANY"10X,DCDDDCDDDPDD,
/22X,"AVG PER", "DIVISION",7X,DCDDDCDDDPDD
755 R_2tr:        IMAGE 22X,"AVG PER ITEM",11X,DCDDDCDDDPDD
760   !
770 D_line1:      IMAGE 9X,8A,8X,DCDDD,5X,DDCDDDPDD,3X,DDDCDDDPDD
780   !
790    END
```

```
                                XYZ COMPANY INVENTORY

                                                                      4/25/91

        PART            QUANTY                UNIT
        NUMBER          ON HAND               PRICE           VALUE

        ======          ========              =====           =====

AA DIVISION
       A-123              15                 41.15             617.25
       K-573             125                 10.90           1,362.50
       B-115             982                    75             736.50
       B-125              99                 37.50           3,712.50

                       TOTAL DEPT 1                          6,428.75
                                                            ----------

       A-111             782                  1.99           1,556.18
       D-286             306                  5.28           4,783.68
       N-742              94                 28.67           2,694.92
       A-524             120                    44              52.80

                       TOTAL DEPT 2                          9,087.64
                                                            ----------

       K-419             668                  9.03           6,032.04

                       TOTAL DEPT 3                          6,032.04
                                                            ----------
                       TOTAL AA DIVISION                    21,548.43
                                                            ----------

BB DIVISION
       F-395              50                 78.75           3,937.50
       C-974             225                    32              72.00
       J-156              12                108.05           1,296.60
       K-238              41                 97.87           4,012.67

                       TOTAL DEPT 1                          9,318.77
                                                            ----------
                       TOTAL BB DIVISION                     9,318.77
                                                            ----------


                       TOTAL COMPANY                        30,867.20
                       AVG PER DIVISION                     15,433.60
                       AVG PER ITEM                          2,374.40

                                                                PAGE 1
```

Example 2

When the statement SUPPRESS PRINT AT 3 is added to the report description section of the previous example, the following is the resultant output:

```
                              XYZ COMPANY INVENTORY

                                                                      4/25/91

        PART              QUANTY                UNIT
        NUMBER            ON HAND               PRICE            VALUE

        ======            ========              =====            =====

AA DIVISION
                          TOTAL AA DIVISION                      21,548.43
                                                                 ----------
BB DIVISION
                          TOTAL BB DIVISION                       9,318.77
                                                                 ----------

                          TOTAL COMPANY                          30,867.20
                          AVG PER DIVISION                       15,433.60
                          AVG PER ITEM                            2,374.40

                                                                    PAGE 1
```

Example 3

This example is a subprogram which uses data from an Eloquence database. The statements which access the database are described in the *Eloquence DBMS Manual*. However, the Eloquence Report Writer portions of the program can be understood without knowledge of the Eloquence database management system.

The breaks are set on product and region. Products must have a lower break level than region since product has less frequent breaks. These breaks are shown in lines 1430 and 1440. The file which has the data must be sorted first by product then by region. This is done in line 1240.

The report description section sets up the headers, trailers, and totals. The header (B in the listing and report) for the product looks up the product description and prints the product number and description every time the product changes. The trailer (C) is the total number of orders, the total amount and the average price.

The header for the region (D) lists the region every time the product or region changes. The trailer (E) is the total number of orders (the number of detail lines printed) and the total dollar amount sold.

The report header (F) is the name of the report which is printed on top of the first page. The report trailer (G) lists the grand totals: the number of orders (the number of detail lines), the amount and the average price printed on the last report page.

To have a total printed in the trailer, the TOTALS ON (or GRAND TOTALS ON) statement must be used in the header. Line 1710, the report trailer, is directly connected to line 1350, the report header.

The detail line, line 1900, prints the order number, customer, city, date, and price.

The page header (H) gives the date and report name then the column headings on every page. The page trailer gives the page.

The page length, margin, level and page suppression is set up in I of the listing. These values are passed to the subprogram through the COM statements.

```
1010   !      RP301 - Report Writer demo - produces bicycle
by region rpt
1030    COM Base$[8],Pselect,Pass$[8],Buff$[200],Lst$[2],
Clr$[41],Dat$[8]
1040    COM INTEGER Cur_seg,Menu_no,Curovly$[16]
1050   !
1060   COM I$[160],INTEGER S(0:9),Restart,Spool,Code,Msg$[80],
K$(1:8)
1070   !
1080   COM #1,Sort(1:4),Options,Level,Left,Value
1090   !
1100   DIM Name$[30]
1110   INTEGER Order_date,Product
1120   !
1130   !
1140           ON ERROR GOTO R98_error
1150           LOAD SUB "RWUTIL";7,9        ! Rwhalt,Error,FNCrt
1160           ON HALT CALL Rwhalt
1170   !
1180           CURSOR (1,19)
1190           LDISP "One moment while data is being sorted."
1200         IN DATA SET "CUSTOMER" USE SKP 1,Name$,SKP 1,City$,
SKP 3,Order_date,SKP 1,Region$,Product
1210         IN DATA SET "OPTION" USE Order$,Option$,Option_price
1220           ASSIGN "WORKF" TO #1
1230         WORKFILE IS #1;THREAD IS "CUSTOMER","ORDER","OPTION"
1240           SORT BY Product,Region$,Order$,Option$
1250   !
1260           PRINTER IS Pselect
1270           IF NOT Spool THEN R05
1280              PRINTER IS "SPOOL"
1290              CURSOR (1,15)
1300              LDISP " ";
1310              CURSOR (25,10)
1320              LDISP "*** REPORT IS BEING SPOOLED ***"
1330   !
1340   R05:    REPORT HEADER
1350              GRAND TOTALS ON Option_price
1360              PRINT SPA(27),"Eloquence REPORT WRITER DEMO"
1370   !
1380              PAGE LENGTH 66-45*(Pselect=8),2*(Pselect%<>8),
1+3*(Pselect%<>8)
1390              LEFT MARGIN Left
1400              SUPPRESS PRINT AT Level
1410              SUPPRESS PRINT FOR Restart PAGES
1420   !
1430              BREAK 3 WHEN Product CHANGES
1440              BREAK 6 WHEN Region$ CHANGES
1450   !
1460           PAGE HEADER WITH 5 LINES USING Ph1;Dat$
1470              PRINT USING Ph2
1480              PRINT USING Ph3
1490              PRINT USING Ph4
1500   !
1510           PAGE TRAILER WITH 2 LINES USING Pt1;Last$,VAL$
(NUMPAGE)
1520            IF (Pselect=8) AND NOT Spool THEN ON FNCrt(Last_
flag,NUMPAGE,Restart,5) GOTO R20,R90_exit
1530              IF NOT Spool THEN R20
1540                 CURSOR (34,11)
```

```
 1550                DISP “CURRENT PAGE “;VAL$(NUMPAGE)
 1560   !
 1570   !
 1580           HEADER 3 WITH 7 LINES
 1590              TOTALS ON Option_price
1600            DBGET (Base$,”PRODUCT”,7,S(*),Lst$,Buff*,Product)
 1610              IF S(0) THEN R95_dberror
 1620              PRINT USING Hd3;Product,Buff$[3;30]
 1630   !
 1640           HEADER 6 WITH 4 LINES USING Hd6;Region$
 1650              TOTALS ON Option_price
 1660   !
1670          TRAILER 3 WITH 5 LINES USING Tr3;OLDCV(3),NUMDETAIL
(3),TOTAL(3,1),AVG(3,1)
 1680   !
1690         TRAILER 6 WITH 4 LINES USING Tr6;OLDCV$(6),NUMDETAIL
(6),TOTAL(6,1)
 1700   !
1710          REPORT TRAILER WITH 5 LINES USING Rt1;NUMDETAIL(0).
TOTAL(0,1),AVG(0,1)
 1720              Last$=”LAST”
 1730              Last_flag=1
 1740   !
 1750   R20:    END REPORT DESCRIPTION
 1760   !
 1770           ON KEY #8:”EXIT” GOTO R90_exit
 1780           ON KEY #16 GOTO R90_exit
 1790           IF NOT Spool AND (Pselect=8) THEN DISP “   “;
 1800   !
 1810           BEGIN REPORT R05
 1820           FOR I=1 TO WFLEN(1)
 1830              READ #1,I;Cnptr,Orptr,Opptr
 1840              IF Cnptr=Lcnptr THEN R40
 1850              Lcnptr=Cnptr
1860            DBGET (Base$,”CUSTOMER”,4,S(*),Lst$,Buff$,Cnptr)
 1870              IF S(0) THEN R95_dberror
 1880             DBGET (Base$,”OPTION”,4,S(*),Lst$,Buff$,Opptr)
 1890              IF S(0) THEN R95_dberror
1900             DETAIL LINE 1 USING Dl1;Order$,Name$,City$,Val$
(Order_date MOD 100)&”/”&VAL$(INT(Order_date/100)),Option_price
 1910   R40:       NEXT I
 1920           IF NUMDETAIL (0) THEN END REPORT
 1930   !
 1940   R90_exit:!
 1950           ASSIGN * TO #1
 1960           STOP REPORT
 1970           LOAD “RM00”
 1980   !
 1990   R95_dberror:!
 2000           LOAD “SALES”,Dberror
 2010   !
 2020   R98_error:!
 2030           OFF ERROR
 2040           OFF KEY #
 2050           CALLRwerror (ERRM$,Base$,S(*))
 2060           PAUSE
 2070           STOP
 2080   !

 2090   !      ***** PRINT USING IMAGE STATEMENTS *****
 2100   !
```

```
 2110   Ph1:   IMAGE “DATE:  “,21A,”BICYCLES BY REGION REPORT”,/
 2120   !
 2130   Ph2:   IMAGE 6X,”ORDER”,51X,”ORDER”
 2140   !
 2150   Ph3:   IMAGE 6X,”NUMBER”,11X,”CUSTOMER”,19X,”CITY”,8X,
“DATE”,7X,”PRICE”
 2160   !
2170   Ph4:   IMAGE 4X,”=========”,X,”==========================
===”,2X,”================”,X,”=====”,2X,”==========”,/
 2180   !
 2190   Pt1:   IMAGE /,65X,5A,”PAGE “,3A
 2200   !
 2210   Hd3:   IMAGE /,”FOR PRODUCT: “,K,” - “,K,/
 2220   !
 2230   Hd6:   IMAGE “  FOR REGION: “,K,/
 2240   !
2250   Tr3:   IMAGE /,19X,”**** TOTAL ORDERS FOR “,4D,28X,DDDDD,/,
 19X,”**** TOTAL AMOUNT”,31X,DDDDCDDD.DD,/,19X,”**** AVERAG
 PRICE”,30X,DDDDCDDD.DD,/
 2260   !
 2270   Tr6:   IMAGE /,19X,”*** TOTAL ORDERS FOR REGION: “,25A,
 DDDDD,/,19X,”*** TOTAL REGION AMOUNT”,25X,DDDDCDDD.DD,/
 2280   !
 2290   Rt1:   IMAGE 2/,19X,”***** GRAND TOTALS OF ORDERS”,25X,
7D,/,19X,”***** GRAND AMOUNT”,30X,DDDDCDDD.DD,/,19X,”***** GRAND
 AVERAGE PRICE”,23X,DDDDCDDD.DD
 2300   !
 2310   D11:   IMAGE 4X,11A,31A,17A,6A,DDCDDD.DD
```

```
DATE: 07/10/91                    DEMONSTRATION BICYCLE COMPANY
                                  BICYCLES BY REGION REPORT

    ORDER                                                          ORDER  NUMBER
CUSTOMER                CITY         DATE      PRICE
    =======   ====================     ==============   =====   ======


FOR PRODUCT: 100 - Standard Bicycle

  FOR REGION:  MSR

       101       Noname, Joseph           Loveland            5/91     75.00

                     *** TOTAL ORDERS FOR REGION:  MSR                    1
                     *** TOTAL REGION AMOUNT                          75.00

  FOR REGION:  SA

       103       Hernandes, Jose          Mexico City         6/91     75.00
       108       Arauja, Luciano A.       Rio de Janeiro      8/91     75.00

                     *** TOTAL ORDERS FOR REGION:  SA                     2
                     *** TOTAL REGION AMOUNT                         150.00


                     **** TOTAL ORDERS FOR 100                            3
                     **** TOTAL AMOUNT                               225.00
                     **** AVERAGE PRICE                               75.00

FOR PRODUCT: 300 - 3-Speed Bicycle

  FOR REGION:  FE

       104       Houseman, Sean           Sidney              6/91    110.00

                     *** TOTAL ORDERS FOR REGION:  FE                     1
                     *** TOTAL REGION AMOUNT                         110.00


                     **** TOTAL ORDERS FOR 300                            1
                     **** TOTAL AMOUNT                               110.00
                     **** AVERAGE PRICE                              110.00
```

```
FOR PRODUCT: 500 - 5-Speed Bicycle

  FOR REGION:  AFR

       105       Sono, Jomo A.            Addis Ababa         5/91    125.00

                     *** TOTAL ORDERS FOR REGION:  AFR                    1
                     *** TOTAL REGION AMOUNT                         125.00

                                                                     PAGE 1


DATE: 07/10/91                    BICYCLES BY REGION REPORT

       ORDER                                                        ORDER
       NUMBER            CUSTOMER                 CITY             DATE     PRICE
      =======     ====================     =============   =====   ========

FOR REGION:  ESR

       100       Smith, Thomas A.         Ft. Collins         6/91    125.00

                     *** TOTAL ORDERS FOR REGION:  ESR                    1
                     *** TOTAL REGION AMOUNT                         125.00


FOR REGION:  FE

       109       Bekker, Bart             Kilbirnie           6/91    125.00

                     *** TOTAL ORDERS FOR REGION:  FE                     1
                     *** TOTAL REGION AMOUNT                         125.00


                     **** TOTAL ORDERS FOR 500                            3
                     **** TOTAL AMOUNT                               375.00
                     **** AVERAGE PRICE                              125.00


FOR PRODUCT: 1000 - 10-Speed Bicycle

FOR REGION:  EUR

       106       Heining, Heinz           Boeblingen          7/91    150.00

                     *** TOTAL ORDERS FOR REGION:  EUR                    1
                     *** TOTAL REGION AMOUNT                         150.00


FOR REGION:  MSR

       107       Dalling, Jimmy           Ft. Collins         8/91    150.00

                     *** TOTAL ORDERS FOR REGION:  MSR                    1
                     *** TOTAL REGION AMOUNT                         150.00


FOR REGION: WSR

       102       Johnson, Sam             San Francisco       7/91    150.00

                     *** TOTAL ORDERS FOR REGION:  WSR                    1
                     *** TOTAL REGION AMOUNT                         150.00
```

```
                    **** TOTAL ORDERS FOR 1000                          3
                    **** TOTAL AMOUNT                                450.00
                    **** AVERAGE PRICE                               150.00

                                                                   PAGE 2


DATE: 07/10/91                  BICYCLES BY REGION REPORT

       ORDER                                                        ORDER
       NUMBER          CUSTOMER              CITY                   DATE     PRICE
      =======   ====================    =================   =====  ========

                    ***** GRAND TOTALS OF ORDERS                         10
                    ***** GRAND AMOUNT                            1,160.00
                    ***** GRAND TOTALS OF ORDERS                    116.00

                                                              LAST PAGE 3
```

A

Summary of Statements and Functions

Description Statements

BREAK *level* WHEN *control* CHANGES [BY *increment*]

The BREAK statement establishes the criteria for determining the level break condition.

END REPORT DESCRIPTION

The END REPORT DESCRIPTION statement must be used to end the description section.

GRAND TOTALS ON exp_1[,exp_2…]

The GRAND TOTALS ON statement provides automatic totaling for the entire report.

HEADER *level* [WITH *number* LINES]

$$\left[\ \text{USING}\ \left\{\begin{matrix}\textit{line id}\\ \textit{image string}\end{matrix}\right\}\ [;\textit{list}]\ \right]\ [\textit{block statements}]$$

The HEADER statement defines what is to be done as a heading when the specified level break occurs. The USING parameters are the same as in a PRINT USING statement.

LEFT MARGIN *column*

The LEFT MARGIN statement sets the column in which each line of the report will begin.

PAGE HEADER [WITH *number* LINES]

$$\left[\ \text{USING}\ \left\{\begin{matrix}\textit{line id}\\ \textit{image string}\end{matrix}\right\}\ [;\textit{list}]\ \right]\ [\textit{block statements}]$$

The PAGE HEADER statement defines what is to be done at the top of every page. The USING parameters are the same as in a PRINT USING statement.

PAGE LENGTH *lines per page* [,*blank top* , *blank bottom*]

The PAGE LENGTH statement specifies the number of lines there are on the page. The number of blank lines to be printed at the top and bottom of the page may also be specified.

PAGE TRAILER [WITH *number* LINES]

[USING { *line id* / *image string* } [;*list*]] [*block statements*]

The PAGE TRAILER statement defines what is to be done at the bottom of each page. If more than one line is used, that number should be specified. The USING parameters are the same as in a PRINT USING statement.

PAUSE AFTER *number* PAGES

The PAUSE AFTER statement causes a pause to occur after the specified number of pages has been output. The [ENTER] key is pressed to resume output.

PRINT DETAIL IF *condition expression*

The PRINT DETAIL IF statement causes exceptional detail lines only to be printed without affecting the totaling functions.

REPORT EXIT (*exec flag*) [WITH *number* LINES]

[USING { *line id* / *image string* } [;*list*]] [*block statements*]

The REPORT EXIT statement defines what action is to be taken when the report is prematurely stopped.

[*label*:]REPORT HEADER [WITH *number* LINES]

[USING { *line id* / *image string* } [;*list*]] [*block statements*]

The REPORT HEADER statement begins the description section. It specifies what is to be done at the beginning of the report. The USING parameters are the same as in a PRINT USING statement.

REPORT TRAILER [WITH *number* LINES]

[USING { *line id* / *image string* } [;*list*]] [*block statements*]

The REPORT TRAILER statement defines what is to be done at the end of the report. If more than one line is required, that number should be specified. The USING parameters are the same as in PRINT USING statement.

SUPPRESS PRINT AT *level*

The SUPPRESS PRINT AT statement specifies the level of headers and trailers that will be printed. Those with equal or higher levels will not be printed.

SUPPRESS PRINT FOR *number* PAGES

The SUPPRESS PRINT FOR statement causes the first specified number of pages not to be printed.

TOTALS ON exp_1[,exp_2…]

The TOTALS ON statement provides automatic totaling for a break level. It immediately follows a header statement.

TRAILER *level* [WITH *number* LINES]

[USING { *line id* / *image string* } [;*list*]] [*block statements*]

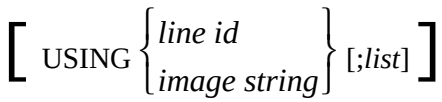
The TRAILER statement defines what is to be done as a trailer for the specified break level. If more than one line is required, that number should be specified. The USING parameters are the same as in a PRINT USING statement.

Execution Statements

BEGIN REPORT *line id*

The BEGIN REPORT statement initiates execution of a report, the description section of which is referenced by the line id.

DETAIL LINE *line* [WITH *number* LINES]



The DETAIL LINE statement causes all break conditions to be tested, totals to be incremented, and data to be printed. If more than one line is required for data output, that number should be specified. The USING parameters are the same as in a PRINT USING statement.

END REPORT

The END REPORT statement causes final trailers to be executed and terminates the Eloquence Report Writer.

NUMPAGE=*expression*

The NUMPAGE=statement causes the page counter to take the specified value..

STOP REPORT

The STOP REPORT statement immediately terminates an active report. No trailing statements are printed.

TRIGGER BREAK *level*

The TRIGGER BREAK statement forces a break condition at the specified level.

TRIGGER PAGE BREAK

The TRIGGER PAGE BREAK statement forces a page to break.

Functions

AVG (*level, sequence*)

The AVG function returns the average for the specified expression in a TOTALS ON statement. The level of the TOTALS ON statement and the sequential position of the expression are specified.

NUMBREAK (*level*)

The NUMBREAK function returns the number of times the specified level break condition has occurred.

NUMDETAIL (*level*)

The NUMDETAIL function returns the number of DETAIL LINE statements that have been executed since the specified level header was last executed.

OLDCV [$] (*level*)

The OLDCV function returns the value of the control variable as it was in the last break condition. If the control variable is a string, the $ is appended to OLDCV.

TOTAL (*level, sequence*)

The TOTAL function returns the running total for the specified expression in a TOTALS ON statement. The level of the TOTALS ON and the sequential position of the expression are specified.

LAST BREAK

The LAST BREAK function returns the value of the last break condition level number detected.

NUMLINE

The NUMLINE function returns the current line number to which output will go.

NUMPAGE

The NUMPAGE function returns the current page number to which output will go.

RWINFO (*integer*)

The RWINFO function returns Eloquence Report Writer information.

B
Error Messages

250 The BEGIN REPORT statement does not reference a report header. You should check the line id specified in the parameter and the line id of the line containing the REPORT HEADER statement.

251 A BEGIN REPORT statement was encountered when the Eloquence Report Writer was already active. The BEGIN REPORT statement must be outside the execution loop.

252 An END REPORT DESCRIPTION statement is missing as a terminator to the description section. This error may occur when the BEGIN REPORT is executed. If the description section precedes the execution section, the Eloquence Report Writer reads the header statement then skips over any statements until the END REPORT DESCRIPTION statement. If it is missing, this error will occur. You should check the program listing carefully. Every REPORT HEADER statement must have a corresponding END REPORT DESCRIPTION statement.

253 A duplicate description section statement has been encountered. You should check that the level number of each header, trailer or break statement is not duplicated. All other description statements can only be used once in a description section.

254 The number of blank lines in the PAGE LENGTH statement is invalid. This number must be a non-negative integer and may not exceed page length minus 1.

255 Expression in a statement evaluates to an unacceptable value. The acceptable values for each expression can be found in the appropriate chapter.

256 A GRAND TOTALS ON or TOTALS ON is improperly positioned in the description section. These statements must immediately follow the header statement. Only one totals statement is allowed per header.

257 A. An Eloquence Report Writer operation was requested while

a report was not active. B. An END REPORT statement was not executed before a subprogram terminated.

258 Effective page size is less than three lines. Effective page size is the number of blank lines plus the number of lines required for the page header and page trailer subtracted from the page length.

259 Invalid execution of a description section statement. This may be caused by a description section statement not being within the description section, or by the program branching into the description section.

260 Insufficient space for printed output within the current printer page. Most likely caused because the WITH number LINES parameter was too small for a header, trailer, or detail line.

261 Left margin specified is less than one or greater than the current printer width – 1, or is greater than 132.

262 A character string in BREAK WHEN control variable has a length greater than was found at the execution fo a BEGIN REPORT. Refer to BREAK WHEN in page 15 for a description of the control variable and its length.

263 A DETAIL LINE statement may not appear within the report description section.

264 The level parameter is out of range for an Eloquence Report Writer statement. Level must be from zero to nine.

265 TOTALS ON or GRAND TOTALS ON statement is not active for the level requested. A TOTAL or AVG function refers to a level which has no GRAND TOTALS ON or TOTALS ON statement.

266 The sequence parameter is out of range for GRAND TOTALS ON or TOTALS ON statement at the level requested.

267 The WITH number LINES parameter is a header, trailer, or detail line is greater than effective page size or is negative.

268 The OLDCV function references a level which has no corresponding break level.

269 The OLDCV function does not match the data types for the control variable in a BREAK WHEN statement at the level requested. You should check that the level specified is correct

and check the data types of the control variable.

270 The PRINTER IS statement may not be executed while the Eloquence Report Writer is active.

271 Eloquence Report Writer statement may not be used recursively.

Index