

HYPERION® SQR® PRODUCTION REPORTING

RELEASE 11.1.1

DEVELOPER'S GUIDE
VOLUME 2: LANGUAGE REFERENCE

ORACLE®
ENTERPRISE PERFORMANCE
MANAGEMENT SYSTEM

Production Reporting Developer's Guide, 11.1.1

Copyright © 1996, 2008, Oracle and/or its affiliates. All rights reserved.

Authors: EPM Information Development Team

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable: U.S. GOVERNMENT RIGHTS: Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services.

Contents

Chapter 1. Introduction	11
About the Production Reporting Language	11
Production Reporting Language Program Structure	11
Production Reporting Language Syntax Conventions	12
Production Reporting Language Syntax Abbreviation Conventions	13
Rules for Entering Production Reporting Commands	13
Production Reporting Data Elements	14
Columns	14
Literals	14
Variables	14
Chapter 2. Production Reporting Command-line	21
Production Reporting Command-line Flags	21
Production Reporting Command-line Arguments	31
Chapter 3. Production Reporting Command Reference	33
About Production Reporting Commands	36
ADD	36
ALTER-COLOR-MAP	37
ALTER-CONNECTION	39
ALTER-LOCALE	41
ALTER-PRINTER	48
ALTER-REPORT	50
ALTER-TABLE	55
ARRAY-ADD, ARRAY-DIVIDE, ARRAY-MULTIPLY, ARRAY-SUBTRACT	58
ASK	60
BEGIN-DOCUMENT	61
BEGIN-EXECUTE	62
BEGIN-FOOTING	66
BEGIN-HEADING	68
BEGIN-PROCEDURE	69
BEGIN-PROGRAM	72

BEGIN-SELECT	72
BEGIN-SETUP	76
BEGIN-SQL	77
BREAK	80
CALL, CALL SYSTEM	80
CLEAR-ARRAY	85
CLOSE	86
CLOSE-RS	86
COLUMNS	88
COMMIT	88
CONCAT	89
CONNECT	91
CREATE-ARRAY	92
CREATE-COLOR-PALETTE	94
CREATE-LIST	95
CREATE-TABLE	96
#DEBUG	99
DECLARE-CHART	100
Attributes Argument	112
DECLARE-COLOR-MAP	126
DECLARE-CONNECTION	128
DECLARE-IMAGE	130
DECLARE-LAYOUT	132
DECLARE-PRINTER	137
DECLARE-PROCEDURE	143
DECLARE-REPORT	144
DECLARE-TABLE	146
DECLARE-TOC	148
DECLARE-VARIABLE	150
#DEFINE	153
DISPLAY	155
DIVIDE	158
DO	159
DRAW	160
DUMP-TABLE	163
#ELSE	164
ELSE	164
ENCODE	164

END-DECLARE, END-DOCUMENT, END-EVALUATE, END-FOOTING, END-HEADING	165
#END-IF, #ENDIF	166
END-IF	167
END-PROCEDURE, END-PROGRAM, END-SELECT, END-SETUP, END-SQL, END-WHILE, END-EXECUTE	167
EVALUATE	168
EXECUTE	170
EXIT-SELECT	175
EXTRACT	176
FILL-TABLE	177
FIND	179
GET	180
GET-COLOR	181
GOTO	183
#IF	184
IF	186
#IFDEF	187
#IFNDEF	188
#INCLUDE	188
INPUT	189
LAST-PAGE	192
LET	192
Operands	193
Operators	194
Functions	196
LOAD-LOOKUP	217
LOOKUP	222
LOWERCASE	223
MBTOSBS	224
MOVE	224
MULTIPLY	228
NEW-PAGE	229
NEW-REPORT	230
NEXT-COLUMN	231
NEXT-LISTING	232
OPEN	233
OPEN-RS	235
PAGE-NUMBER	237

POSITION	238
PRINT	239
BOLD	241
BOX	241
BOX-FILL-COLOR	242
BOX-LINE-COLOR	242
CENTER	242
CENTER-WITHIN	242
CODE-PRINTER	242
DATE	243
DELAY	243
EDIT	244
FILL	252
FONT	252
FOREGROUND/BACKGROUND	253
ITALIC	253
MATCH	253
MONEY	254
NOP	254
NUMBER	254
ON-BREAK	254
POINT-SIZE	257
SHADE	257
UNDERLINE	257
URL	257
URL-TARGET	258
WRAP	258
PRINT-BAR-CODE	260
PRINT-CHART	263
PRINT-DIRECT	270
PRINT-IMAGE	271
PRINT-TABLE	273
PUT	274
READ	275
ROLLBACK	277
SBTOMBS	278
SECURITY	278
SET-COLOR	280
SET-DELAY-PRINT	282

SHOW	283
STOP	287
STRING	288
SUBTRACT	289
TOC-ENTRY	290
UNSTRING	291
UPPERCASE	292
USE	293
USE-COLUMN	293
USE-PRINTER-TYPE	294
USE-PROCEDURE	295
USE-REPORT	296
WHILE	297
WRITE	299
WRITE-RS	300
Chapter 4. HTML Procedures	303
About HTML Procedures	303
HTML General Purpose Procedures	303
HTML Heading Procedures	305
HTML Highlighting Procedures	307
HTML Hypertext Link Procedures	309
HTML List Procedures	309
HTML Table Procedures	312
Chapter 5. Encoding in Production Reporting	315
Encoding Methods	315
Encoding Keys in SQR.INI	315
Encoding Keys in the [Default-Settings] Section	315
Encoding Keys in the [Environment] Section	317
Encodings Supported without Using Unicode Internally	318
Encodings Supported in Production Reporting	320
Chapter 6. SQR.INI	327
Installation of SQR.INI	327
For Windows Platforms Only	327
For All Other Platforms	328
[Default-Settings] Section	328
[Environment: environment] Section	333
Using the Java Virtual Machine	334

DDO Variables	334
Encoding Keys	335
[SQR Extension] Section	335
[Locale:local-name] Section	335
[Fonts] Section	337
Adding [Fonts] Entries	337
Specifying Character Sets in Windows	338
[PDF Fonts] Section	338
Embedding Fonts	338
Available Fonts	339
[PDF Settings] Section	340
[HTML Fonts] Section	341
[HTML:Images] Section	342
[Enhanced-HTML] Section	343
[Color Map] Section	344
[MAP-ODBC-DB] Section	345
[MAP-DDO-DB] Section	345
[SQR Remote] Section	345
Chapter 7. Production Reporting Samples	347
Chapter 8. Production Reporting Messages	351
Unnumbered Messages	351
Numbered Messages	353
Appendix A. Production Reporting Language Quick Reference	421
Appendix B. Deprecated Information	435
Deprecated Production Reporting Command-line Flags	435
Deprecated SQR.INI Entries	436
Values for the FullHTML Keyword in the [Enhanced-HTML] Section	436
[Processing-Limits] Section	436
Values for PDFCompressionText and PDFCompressionGraphics in the [Default-Settings] Section	438
Deprecated Transforms	438
Deprecated Production Reporting Commands	438
BEGIN-REPORT	439
DATE-TIME	440
DECLARE PRINTER	441
DECLARE PROCEDURE	445
DOLLAR-SYMBOL	446

GRAPHIC BOX	448
GRAPHIC FONT	449
GRAPHIC HORZ-LINE	450
GRAPHIC VERT-LINE	450
MONEY-SYMBOL	451
NO-FORMFEED	452
PAGE-SIZE	453
PRINT ...CODE	454
PRINTER-DEINIT	454
PRINTER-INIT	455
Index	457

1

Introduction

In This Chapter

About the Production Reporting Language	11
Production Reporting Data Elements.....	14

About the Production Reporting Language

The Oracle's Hyperion® SQR® Production Reporting language is a specialized programming language for accessing, manipulating, and reporting enterprise data. Using the Production Reporting language, you can build complex procedures that execute multiple calls to multiple datasources and implement nested, hierarchical, or object-oriented program logic.

The Production Reporting language has several key benefits:

- Flexibility and scalability
- Comprehensive facilities for combined report and data processing
- Multiple platform availability
- Multiple datasource compatibility

With the Production Reporting language, you can design custom reports by defining the page size, headers, footers, and layout. The Production Reporting language enables you to generate a wide variety of output such as complex tabular reports, multiple page reports, and form letters. You can display data in columns, produce special formats such as mailing labels, and create HTML, PDF, or customized output for laser printers and phototypesetters.

The high-level programming capabilities that the Production Reporting language provides enable you to add procedural logic and control to datasource calls. You can use the Production Reporting language to write other types of applications, such as database manipulation and maintenance, table load and unload, and interactive query and display.

Production Reporting Language Program Structure

The Production Reporting language processes source code from standard text files and generates reports. Text files containing source code contain sections delimited with `BEGIN-section` and `END-section` commands. The following examples show the general structure of the Production Reporting language.

The `SETUP` section describes overall characteristics of the report:

```
BEGIN-SETUP
  {setup commands}...
END-SETUP
```

The HEADING and FOOTING sections specify what is printed in the header and footer on each page of the report:

```
BEGIN-HEADING {heading_lines}
  {heading commands}...
END-HEADING
BEGIN-FOOTING {footing_lines}
  {footing commands}...
END-FOOTING
```

The PROGRAM section executes the procedures contained in the report:

```
BEGIN-PROGRAM
  {commands}...
END-PROGRAM
```

The PROCEDURE section accomplishes the tasks associated with producing the report:

```
BEGIN-PROCEDURE {procedure_name}
  {procedure commands}...
END-PROCEDURE
```

Production Reporting Language Syntax Conventions

Table 1 Syntax Conventions

Symbol	Description
{ }	Braces enclose required items.
[]	Square brackets enclose optional items.
...	An ellipsis shows that the preceding parameter can be repeated.
	A vertical bar separates alternatives within brackets, braces, or parentheses.
'	A single quote starts and ends a literal text constant or any argument with more than one word. Caution: If you copy codes directly from the examples in the pdf file, make sure you change the slanted quotes to regular quotes or else you will get an error message.
!	An exclamation point begins a single-line comment that extends to the end of the line. Each comment line must begin with an exclamation point. Do not use != to delineate a comment block unless it starts in the first column. The characters "!=" denotes a relational operator, and Production Reporting could confuse it with a comment where a relational argument could occur.
,	A comma separates multiple arguments.
()	Parentheses must enclose an argument or element.

Symbol	Description
UPPERCASE	Production Reporting commands and arguments are specified in UPPERCASE.
Italics	Information and values that you must supply are specified in <i>italics</i> .

Production Reporting Language Syntax Abbreviation Conventions

Table 2 Syntax Abbreviation Conventions

Abbreviation	Description	Example
any_col	A column of any type.	&string &number &date
date_col	Date or datetime column retrievable from a database.	&date1
num_col	Numeric column retrievable from a database.	&price
txt_col	Text column retrievable from a database.	&address
any_lit	A literal of any type.	'abc' 12
int_lit	Integer literal defined in a program.	12345
num_lit	Numeric literal defined in a program.	12345.67
txt_lit	Text literal defined in a program.	'Company Confidential'
any_var	A variable of any type.	\$string #number \$date
date_var	A variable explicitly defined as a date variable.	\$date1
num_var	Numeric variable defined in a program.	#total_cost
txt_var	String variable defined in a program.	\$your_name
nn	Integer literal used as an argument to a command.	123
position	The position qualifier, which consists of the line, column, and length specification. The minimum position, (), means to use the current line and column position on the page for the length of the field being printed.	(5,10,30)

Rules for Entering Production Reporting Commands

- Production Reporting commands are *not* case sensitive. Common practice is to use upper case for commands, but Production Reporting ignores case when compiling source code.
- Separate command names and arguments by at least one space or tab character.
- Begin each command on a new line; however, you can develop commands that extend beyond one line.

- Break a line in any position between words *except* within a quoted string.
- Use a hyphen (-) at the end of a line to indicate that it continues on the next line. (Production Reporting ignores hyphens and carriage returns within commands.)
- Begin each comment line with an exclamation point (!).
- To display the ! or ' symbols in a report, type the symbols twice to indicate that they are text. For example, DON'T is typed DON"\"T.

Note:

You do not need to type quotation and exclamation marks twice in the DOCUMENT section of form letter reports

Production Reporting Data Elements

Production Reporting data elements include columns, literals, and variables. Each element begins with a special character that denotes the type of data element.

Columns

Columns are fields defined in the database.

- & begins a database column or expression name. It can be any type of column as long as it is a standard SQL datatype. Except for dynamic columns and database or aggregate functions, it is declared automatically for columns defined in a query.

Literals

Literals are text or numeric constants.

- A single quote begins and ends a text literal. For example, 'Hello'.
- 0-9 begin any numeric literals. Numerals that include digits with an optional decimal point and leading sign are acceptable numeric literals. For example, -543.21. Numeric literals can also be expressed in scientific form. For example, 1.2E5.

Variables

Variables are storage places for text or numbers.

- \$ begins a text or date variable
- # begins a numeric variable
- % begins a list variable
- @ begins a variable name for a marker location. Marker locations are used to identify positions to begin printing inside a BEGIN-DOCUMENT paragraph.

Variable Rules

- Variables can be almost any name of almost any length. For example, \$state_name or #total_cost. For exceptions, see “[Production Reporting Reserved Variables](#)” on page 16.
- Do not use “_” or “:” as the first character of a two variable name.
- Variable names are *not* case sensitive. That is, you can use a name as uppercase on one line and lowercase on the next; both refer to the same variable.
- Production Reporting initializes variables to null (text and date) or zero (numeric).
- Commands can grow to whatever length the memory of your computer can accommodate.
- Numeric variables can be FLOAT, INTEGER, or DECIMAL. See “[DECLARE-VARIABLE](#)” on page 150 for more information.
- Variables and columns are known globally throughout a report, except if used in a local procedure (one with arguments or declared with the LOCAL argument) in which case they are known in that procedure only. See “[BEGIN-PROCEDURE](#)” on page 69 for more information.

List Variables

List variables contain an ordered collection of Production Reporting variables and are nonrecursive (you cannot nest lists within lists).

Indicate list variables with the % symbol. Create list variables with the LET command along with a list of variables. For example,

```
LET %LIST1 = LIST (num_var1|str_var1, num_var2|str_var2,...)
```

Note:

List variables are used in Production Reporting DDO only.

You can perform the following actions with list variables:

- **Define a list variable**—List variables can hold multiple rows of information. Before you assign a list variable, define it using the following syntax:

```
let %listname=LIST(col_var|num_var|str_var|str_lit|num_lit[,...])
```

or

```
let %listname[num_lit]=list(NUMBER|DATE|TEXT$colname  
|'.colname'[,...])
```

- **Assign a list variable**—Use the following syntax:

```
let %listname|%listname[num_var|num_lit]=list(col_var|str_var  
|num_var|str_lit|num_lit[,...])
```

- **Access a list variable**—Use the following syntax:

```
let str_var|num_var=%listname[num_var|num_lit].#colname
```

- **Modify a list variable**—When you modify a list variable, you can modify a specific row element of any list item.

In list-variable arguments, the value between the brackets indicates either the number of rows in the list for the definition case or the row within the list to be modified or assigned.

If no brackets exist, there is no need to predefine; assign the types based on the given variable types. For multirow lists, the assignment must be compatible with the types given in the definition.

A `NUMBER` field has the same characteristics as an undeclared `#var`; the underlying storage depends on the contents, and the `DEFAULT-NUMERIC` setting applies.

The usual Production Reporting rules for variable assignment apply to list access. Assignment is prohibited only between Date and Numeric types. Assignment of a numeric column to a string variable returns the string representation of the numeric value; assignment of a date variable to a string variable returns the default-edit-mask representation of the date.

Production Reporting Reserved Variables

When you create multiple reports, the variables apply to the current report. Production Reporting reserves a library of predefined variables for general use.

Note:

All of Production Reporting's reserved variables are global variables. Reference reserved variables in local procedures with a leading underscore. For example, `#_sqlstatus` or `$_sql-error`. See “[BEGIN-PROCEDURE](#)” on page 69 for information on defining local procedures.

Table 3 Production Reporting Reserved Variables

Variable	Description
<code>#current-column</code>	Current column on the page.
<code>\$current-date</code>	Current date-time on the local machine when Production Reporting starts running the program.
<code>#current-line</code>	Current line on the page. This value is the physical line on the page, not the line in the report body. Line numbers are referenced in PRINT and other Production Reporting commands used for positioning data on the page. Optional page headers and footers, defined BEGIN-HEADING and BEGIN-FOOTING , have their own line sequences. Line 2 of the heading is different from line 2 of the report body or footing.

Variable	Description
#end-file	Set to one (1) if end of file occurs when reading a flat file; otherwise, set to zero (0). Your program should check this variable after each READ command. (See “READ” on page 275 for more information.)
#page-count	Current page number.
#return-status	Value returned to the operating system when Production Reporting exits. Can be set in your report. #return-status is initialized to the “success” return value for the operating system.
#sql-count	Count of rows affected by a DML statement (INSERT, UPDATE, or DELETE). This is equivalent to ROWCOUNT in Oracle and Sybase.
\$sql-error	Text message from the database explaining an error. This variable is rewritten when a new error is encountered.
#sql-status	Value of #SQL-STATUS set whenever BEGIN-SELECT executes. Normally this variable is checked from within an ON-ERROR procedure so its value describes the error condition (whereas the \$SQL-ERROR variable contains the error message). The actual meaning of #SQL-STATUS is database dependent. Consult the proper database manual to fully interpret its meaning.
\$sql-text	<p>Last SQL statement sent to the database by Production Reporting.</p> <p>The variable contents are valid after Production Reporting processes BEGIN-SQL, BEGIN-SELECT, or LOAD-LOOKUP, or within the ON-ERROR-procedure for BEGIN-SQL and BEGIN-SELECT.</p> <p>The variable is populated for ODBC, Sybase, Oracle, Informix, Teradata, DB2, and DDO.</p>
\$sqr-connected-db {sqr-connected-db}	<p>Class of the backend database from the initial connection or from CONNECT.</p> <p>For ODBC, the class is: ODBC, Sybase, Oracle, Informix, Redbrick, Teradata, or DB2.</p> <p>For DDO, the class is: XML, CSV, SAP, MSOLAP, Essbase, SQLServer, Sybase, Oracle, Informix, or DB2.</p> <p>The information is derived from \$sqr-connected-db-name / {sqr-connected-db-name}.</p>
\$sqr-connected-db-name {sqr-connected-db-name}	Name of the database driver from the initial connection or from CONNECT.

Variable	Description
	For ODBC, the value is derived from the ODBC Driver Manager. For DDO, the value is derived from the driver used. For all other databases, the value is the same as the value in <code>\$sqr-database / {sqr-database}</code> .
<code>\$sqr-database {sqr-database}</code>	Database type for which Production Reporting was compiled. Valid values are: DB2, ODBC, Sybase, Informix, and Oracle.
<code>\$sqr-dbc {sqr-dbc}</code>	Defines whether Production Reporting recognizes double-byte character strings. The value can be either YES or NO.
<code>\$sqr-encoding {sqr-encoding}</code>	Name of the default encoding as defined by the ENCODING environment variable when Production Reporting is invoked.
<code>\$sqr-encoding-console {sqr-encoding-console}</code>	Name of encoding for character data written to the log file or console.
<code>\$sqr-encoding-database {sqr-encoding-database}</code>	Character data retrieved from and inserted into the database.
<code>\$sqr-encoding-file-input {sqr-encoding-file-input}</code>	Name of encoding for character data read from files used with OPEN .
<code>\$sqr-encoding-file-output {sqr-encoding-file-output}</code>	Name of encoding for character data written to files used with OPEN .
<code>\$sqr-encoding-report-output {sqr-encoding-report-output}</code>	Report generated by Production Reporting (for example, an LIS file or a PostScript file).
<code>\$sqr-encoding-source {sqr-encoding-source}</code>	Name of encoding for Production Reporting source files and include files.
<code>\$sqr-hostname {sqr-hostname}</code>	Name of the computer on which Production Reporting is currently executing.
<code>\$sqr-locale</code>	Name of the current locale. A + at the end of the name indicates an argument used in the locale has changed.
<code>#sqr-max-columns</code>	Maximum number of columns as determined by the layout. When a new report is selected, this variable is automatically updated to reflect the new layout.

Variable	Description
<i>#sqr-max-lines</i>	Maximum number of lines as determined by the layout. When a new report is selected, this variable is automatically updated to reflect the new layout.
<i>#sqr-pid</i>	Process ID of the current Production Reporting process. #sqr-pid is unique for each run of Production Reporting. This variable is useful in creating unique temporary names.
<i>\$sqr-platform {sqr-platform}</i>	Hardware/operating system type for which Production Reporting was compiled. Valid values are WINDOWS and UNIX.
<i>\$sqr-program</i>	Name of the Production Reporting program file.
<i>\$sqr-report</i>	Name of the report output file. \$sqr-report reflects the actual name of the file to be used (as specified by the -F flag or the NEW-REPORT command).
<i>\$sqr-ver</i>	Text string shown with the -ID flag. Production Reporting version.
<i>\$username</i>	Database user name specified on the command line.

2

Production Reporting Command-line

In This Chapter

Production Reporting Command-line Flags.....21
 Production Reporting Command-line Arguments31

Production Reporting Command-line Flags

Production Reporting command-line flags begin with a dash (-). When a flag takes an argument, the argument must follow the flag with no intervening space.

Table 4 Production Reporting Command-line Flags

Flag	Description	Program	Database
-A	<p>Appends output to a file with the same name as the source. If the file does not exist, a new one is created. Useful for running a report multiple times with a single output file.</p> <p>Restrictions:</p> <ul style="list-style-type: none"> ● Only works with LIS files. Does not work with SPF files. ● Only applies to line printers (-PRINTER:LP). Ignored for other printer types and output formats. ● Only applies to SQR and SQRP in non-Windows environments. 	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-Bnn	<p>Defines the number of rows to buffer when retrieving data. The default is 10 rows. Regardless of the setting, all rows are retrieved. On the command line, -B controls the setting for all BEGIN-SELECT commands. In a program, each BEGIN-SELECT command can have its own -B flag for further optimization.</p>	Production Reporting Server Production Reporting Execute	ODBC Oracle Sybase
-BURST:{xx}	<p>Specifies the bursting type.</p> <ul style="list-style-type: none"> ● -BURST:T generates the Table of Contents. ● -BURST:S generates report output according to the symbolic Table of Contents entries defined with the <i>level</i> argument in TOC-ENTRY. In -BURST:S[{l}], {l} is the level at which to burst. -BURST:S is equivalent to -BURST:S1. ● -BURST:P generates report output by report page numbers. In -BURST:P[{l} , {s} [, {s}....]] , {l} is the number of logical report pages that each HTML file contains and {s} is the page selection: {n}, {n}-{m}, -{m}, or {n}-. -BURST:P is equivalent 	Production Reporting Server Production Reporting Execute Production Reporting Print	All

Flag	Description	Program	Database
	<p>to -BURST:P0,1- when using -PRINTER:HT or -BURST:P1 when using -PRINTER:EH.</p> <p>See “Bursting and Demand Paging” in Volume 1 of the <i>Hyperion SQR Production Reporting Developer's Guide</i> for more information.</p> <p>Note: -BURST:P and -BURST:S require -PRINTER:EH or -PRINTER:HT. The Page range selection feature of -BURST:P requires -PRINTER:HT. -BURST:T requires -PRINTER:HT.</p>		
-C	(Windows) Displays the Cancel dialog box while the program runs to terminate program execution.	Production Reporting Server Production Reporting Execute	All
-CB	(Windows, Callable Production Reporting) Forces the communication box to be used.	Production Reporting Server	All
-Dnn	<p>(non-Windows) Displays the report output at the same time it is written to the output file. <i>nn</i> is the maximum number of lines to display before pausing. If no number is entered after -D, the display scrolls continuously.</p> <p>Note: The printer type must be <i>LP</i> or the display is ignored. If the program produces multiple reports, the display is for the first report only.</p>	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-DBdatabase	Uses the specified database, which overrides any USE command in the Production Reporting program.	Production Reporting Server Production Reporting Execute	Sybase
-DEBUG[xxx]	<p>Compiles lines preceded by #DEBUG. Without this flag, Production Reporting ignores these lines.</p> <p>(See “#DEBUG” on page 99.)</p>	Production Reporting Server	All
-DNT:{xx}	Defines the default behavior for numeric variables. The value for <i>xx</i> can be INTEGER, FLOAT, DECIMAL, or V30. To specify a precision for DECIMAL, append it with a colon delimiter (:)—for example, -DNT:DECIMAL:20. See the DEFAULT argument for “DECLARE-VARIABLE” on page 150 for a detailed explanation. If used, the DEFAULT argument in DECLARE-VARIABLE takes precedence.	Production Reporting Server	All
-E[file]	Directs error messages to the named file, or to the default file <i>program.err</i> . If no errors occur, no file is created.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_APPLETS:dir	<p>Defines the directory for Enhanced HTML applets. (The default directory for applets is IMAGES.)</p> <p>Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</p>	Production Reporting Server Production Reporting Execute	All

Flag	Description	Program	Database
		Production Reporting Print	
-EH_BQD	Generates a {report}.bqd file from the report data and associates a BQD (Brio Query Format File) icon with {report}.bqd in the navigation bar. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_BQD:file	Associates BQD icons with the specified file. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_BROWSER:xx	Defines the browser and generates the HTML. <ul style="list-style-type: none"> ● BASIC—Generates HTML suitable for all browsers. ● IE—Generates HTML for Internet Explorer. ● NETSCAPE —Generates HTML for Netscape. ● ALL—If necessary, Production Reporting generates Basic, IE, and Netscape HTML files. Report_fm.htm contains Javascript to “sense” the browser on the user’s machine and display the appropriate version. (In this case, the user’s machine is the machine of the person reading the report, not the person writing it.) Note: Only applicable with -PRINTER:EH or -PRINTER:EP. Only recognized when combined with -EH_FULLHTML.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_CSV	Generates a {report}.csv file from report data. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_CSV:file	Associates the CSV icon with the specified file. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_CSVONLY	Creates a CSV file but does not create an HTML file. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute	All

Flag	Description	Program	Database
		Production Reporting Print	
-EH_DEBUG[:opts]	Produces a DBG output file containing compiler and internal error messages. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_ICONS:dir	Defines the directory where HTML should look for the referenced icons. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_IMAGES:dir	Defines the directory path for the GIF files used by the Navigation Bar. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_KEEP	Copies (does not move) the files when used in conjunction with -EH_ZIP. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_LANGUAGE:xx	Defines the HTML navigation bar language. You can specify English, French, German, Italian, Japanese, Korean, Portuguese, Spanish, SChinese, or TChinese. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_PDF	Associates a PDF icon with {report}.pdf in the navigation bar. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-EH_SCALE:{nn}	Sets the scaling factor from 50 to 200. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute	All

Flag	Description	Program	Database									
		Production Reporting Print										
-EH_XIMG	Specifies to not remove the directory path from the IMAGE reference. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All									
-EH_XML:file	Associates an XML icon with a file. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All									
-EH_ZIP[:file]	Moves generated files to the specified file or {report}.zip if {file} is not specified. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.	Production Reporting Server Production Reporting Execute Production Reporting Print	All									
-F[file directory]	Overrides the default output file name, <i>program.lis</i> . The default action places the <i>program.lis</i> file in the same directory as the <i>program.sqr</i> file. To use the current directory, specify -F without an argument. To change the name of the output file, specify -F with the new name. If the new name does not specify a directory, the file is created in the current directory. The output file is not created until data actually prints on the page. If no data prints, no output file is created. The following shows how to specify file names and directories for different operating systems.	Production Reporting Server Production Reporting Execute Production Reporting Print	All									
	<table border="1"> <thead> <tr> <th>Operating System</th> <th>Directory Character</th> <th>Example</th> </tr> </thead> <tbody> <tr> <td>UNIX</td> <td>/</td> <td>-F\$HOME/reports/</td> </tr> <tr> <td>Windows</td> <td>\</td> <td>-FC:\Oracle\Files\</td> </tr> </tbody> </table>	Operating System	Directory Character	Example	UNIX	/	-F\$HOME/reports/	Windows	\	-FC:\Oracle\Files\		
Operating System	Directory Character	Example										
UNIX	/	-F\$HOME/reports/										
Windows	\	-FC:\Oracle\Files\										
-ID	(non-Windows) Displays the copyright banner.	Production Reporting Server Production Reporting Execute Production Reporting Print	All									
-Idir_list	Specifies the directories that Production Reporting searches when processing the #INCLUDE directive if the include file does not exist in the current directory and no path was specified for the file. The	Production Reporting Server	All									

Flag	Description	Program	Database
	<p>directory names must be separated by either commas (,) or semicolons (;).</p> <p>In UNIX, if your shell uses semicolons as command delimiters, precede each semicolon with a backslash (\). Always append the directory character to the end of each directory name. See the -F flag for a list of directory characters by operating system. For example, under UNIX:</p> <pre>sqr myreport sammy/baker -I/home/sqr/inc/, /usr/sqr/incl/</pre>		
-KEEP	Creates SPF files in addition to LIS files for each report generated. (See Chapter 28, "Printing Issues," in Volume 1 of the <i>Hyperion SQR Production Reporting Developer's Guide</i> for more information on LIS and SPF files.)	Production Reporting Server Production Reporting Execute	All
-LL{s d}{c i}	LOAD-LOOKUP: s = SQR, d = DB, c = case-sensitive, i = case-insensitive (See "LOAD-LOOKUP" on page 217)	Production Reporting Server	All
-NOLIS	Prevents the creation of all Production Reporting output file types. SPF output is created instead.	Production Reporting Server Production Reporting Execute	All
-O[file]	Directs log messages to the specified file or to <i>program.log</i> if no file is specified. By default, the file <i>sqr.log</i> is used in the current working directory.	Production Reporting Server Production Reporting Execute	All
-PB	Retains trailing blanks in column data.	Production Reporting Server Production Reporting Execute	Informix
-PRINTER:xx	Uses printer type <i>xx</i> when creating output files.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
	xx	Printer Type	Example
	EH	Enhanced HTML	-PRINTER:EH
	EP	Enhanced HTML/PDF	-PRINTER:EP
	GD -GD_DRIVER:name defines the name	Generic Output Driver	-PRINTER:GD To generate Excel documents, enter: -PRINTER:GD -GD_DRIVER:EXCEL

Flag	Description		Program	Database
	<p>of the Generic Output Driver.</p> <p>-GD_OPTION:opts defines optional parameters to pass to the Generic output Driver. (The optional parameters are used to pass command line data to your personal application. The parameters entered depend on your application.)</p>			<p>To generate Word documents, enter:</p> <pre>-PRINTER:GD -GD_DRIVER:WORD</pre> <p>To generate Power Point documents, enter:</p> <pre>-PRINTER:GD -GD_DRIVER:PP</pre> <p>These commands create Office HTML files that can be opened by Word 2003, Word XP, Power Point 2003, Power Point XP, Excel 2003, and Excel XP. (For Office 2000 products, you can download a Microsoft tool to convert Office HTML files into something that Office 2000 products can open.)</p> <p>Note: You can use the -F flag to change the output file name to report.doc, report.ppt, or report.xls.</p>
	HP	HP LaserJet	<p>-PRINTER:HP</p> <p>Note: Production Reporting does not support color for -PRINTER:HP (HP PCL) output generation.</p>	
	LP	Line Printer	-PRINTER:LP	
	PD	PDF	-PRINTER:PD	
	PS	PostScript	-PRINTER:PS	
	WP	Windows	-PRINTER:WP	
	<p>Notes:</p> <ul style="list-style-type: none"> ● LP, HP, and PS produce files with the .lis extension. ● EH and HT produce .htm file output. <ul style="list-style-type: none"> ○ HT is controlled by the PrinterHT setting in the [Default-Settings] section of SQR.INI. If PrinterHT is set to <i>standard</i>, HT produces version 2.0 HTML files with the report content inside <PRE></PRE> tags. If PrinterHT is set to <i>enhanced</i>, HT is mapped to EH. (See “[Default-Settings] Section” on page 328 on page 333 for additional information.) ○ EH produces reports fully formatted with version 1.1 XHTML tags. ● PD produces PDF 1.3 compliant files. When -PRINTER:PD is used, PRINT-DIRECT, PRINT ...Code, and Print with CODE-PRINTER commands are processed but ignored. ● On Windows systems, WP sends output to the default Windows printer. To specify a non-default Windows printer, use -PRINTER:WP:{Printer Name}. The {Printer Name} can be the name assigned to a printer; or, if the operating system permits 			

Flag	Description	Program	Database
	it, the UNC name (i.e. \\Machine\ShareName). For example, to send output to a Windows printer named <i>NewPrinter</i> , you could use -PRINTER:WP:NewPrinter. If your printer name has spaces, enclose the entire command in double quotes.		
-PSnn	Sets the TDS (Tabular Data Stream) packet size to the specified value.	Production Reporting Server Production Reporting Execute	Sybase
-RS	Saves the program in a run-time file. The program is scanned, compiled, and checked for correct syntax. Queries are validated and compiled. Then, the executable version is saved in a file with the name <i>program.sqt</i> . Note that Production Reporting does not prompt for ASK variables after compilation.	Production Reporting Server	All
-RT	Uses the run-time file saved with -RS. Skips syntax and query checking and begins processing immediately. Note that Production Reporting does not prompt for ASK variables after compilation.	Production Reporting Server	All
-S	Displays the status of all cursors at the end of the report run. Status includes the text of each SQL statement, the number of times each was compiled and executed, and the total number of rows selected. The output goes directly to the screen. This information can be used for debugging SQL statements and enhancing performance and tuning.	Production Reporting Server Production Reporting Execute	All
-Tnn	Specifies that you want to test your report for <i>nn</i> pages. To save time during testing, Production Reporting ignores all ORDER BY clauses in SELECT statements. For multiple reports, Production Reporting stops after the specified number of pages defined for the first report.	Production Reporting Server Production Reporting Execute	All
-T{B}	Trims trailing blanks from database character columns.	Production Reporting Server Production Reporting Execute	DB2 Sybase ODBC Teradata
-T{Z}	Trims trailing zeros from the decimal portion of numeric columns.	Production Reporting Server Production Reporting Execute	DB2 Teradata
-U{priv_connectivity}	Directs SQR to connect to a privileged user and then proxy as the "user" from the normal connectivity piece. <pre>sqr {program} {connectivity} - Upriv_connectivity [flags] [args]</pre> For example: <pre>sqr scott {prg} -Upriv/priv@instance</pre> Logs in in as "priv/priv@instance" and then proxies to user "scott". All access is then be based "scott" and not "priv".		

Flag	Description	Program	Database
-Vserver	Uses the named server.	Production Reporting Server Production Reporting Execute	Sybase
-XB	(non-Windows) Suppresses the Production Reporting banner and the "Production Reporting... End of Run" message.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-XC	(Callable Production Reporting) Suppresses the database commit when the report finishes running.	Production Reporting Server	All
-XCB	(Windows) Defines to not use the communication box. Requests for input are made in Windows dialog boxes.	Production Reporting Server Production Reporting Execute	All
-XFRM	Prevents Production Reporting from creating a frame in HTML files generated with -PRINTER:EH.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-XI	Prevents user interaction during a program run. If <code>ASK</code> or <code>INPUT</code> requires user input, an error displays and the program ends.	Production Reporting Server Production Reporting Execute	All
-XL	Prevents Production Reporting from logging onto the database. Programs run in this mode cannot contain any SQL statements. -XL lets you run Production Reporting without accessing the database. You still must supply at least an empty slash (/) on the command line as a placeholder for the connectivity information. For example: <code>sqr myprog / -xl</code>	Production Reporting Server Production Reporting Execute	All
-XLFF	Prevents trailing form feed.	Production Reporting Server Production Reporting Execute	All
-XMB	(Windows) Disables error message display so programs can run without interruption by error messages/warnings generated by Production Reporting, or by user generated messages (SHOW/DISPLAY). All messages still go to their designated output files (SQR.ERR or -E{filename} / SQR.LOG or -O{filename})	Production Reporting Server	All
-XNAV	Prevents Production Reporting from creating the Navigation Bar in HTML files generated with -PRINTER:HT and -PRINTER:EH. This	Production Reporting Server	All

Flag	Description	Program	Database
	occurs when only a single HTML file is produced. Multiple HTML files generated from a single report always contain the Navigation Bar.	Production Reporting Execute Production Reporting Print	
-XP	Prevents Production Reporting from creating temporary stored procedures. See "BEGIN-SELECT" on page 72 for more information.	Production Reporting Server Production Reporting Execute	Sybase
-XTB	Preserves trailing blanks in LIS files at the end of a line.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-XTOC	Prevents Production Reporting from generating the Table of Contents for the report. This flag is ignored when -PRINTER:HT or -PRINTER:EP is also specified.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-ZEN{name}	Sets the default encoding name.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-ZIF{file}	Sets the full path and name of the SQR initialization file, SQR.INI.	Production Reporting Server Production Reporting Execute Production Reporting Print	All
-ZIV	Invokes the SPF Viewer after generating the program.spf file. Implicitly invokes -KEEP to create program.spf. For multiple output files, only the first report file passes to the Production Reporting Viewer.	Production Reporting Server Production Reporting Execute	All
-ZMF{file}	Defines the full path and name of the Production Reporting error message file, sqrrr.dat.	Production Reporting Server Production Reporting Execute Production Reporting Print	All

Flag	Description	Program	Database
-ZRF{file}	<p>(DDO) Sets the full path and name of an alternate <i>registry.properties</i> file. Following is a common default path to the <i>registry.properties</i> file on a Windows system:</p> <p>c:\program files\hyperion\properties\registry.properties</p> <p>The <i>registry.properties</i> file lists datasources that Production Reporting can access. The information in the <i>registry.properties</i> file makes it possible for Production Reporting to access datasources for which DDO drivers have been loaded and configured.</p>	<p>Production Reporting Server</p> <p>Production Reporting Execute</p>	All

Production Reporting Command-line Arguments

Table 5 Production Reporting Command-line Arguments

Argument	Description	Program
<i>program</i>	Name of the text file containing source code. The default file type or extension is .sqr. If entered as “?” or omitted, Production Reporting prompts for the report program name. On UNIX, if your shell uses the question character as a WILD CARD character, precede it with a backslash (\).	<p>Production Reporting Server</p> <p>Production Reporting Execute</p>
-flags	Any flag in Production Reporting Command-line Flags .	<p>Production Reporting Server</p> <p>Production Reporting Execute</p> <p>Production Reporting Print</p>
<i>pars...</i>	Report parameters for ASK and INPUT commands. Enter the parameters on the command line in the same sequence they are expected by the program—first ASK parameters in order and then INPUT parameters.	<p>Production Reporting Server</p> <p>Production Reporting Execute</p>
@file...	File containing program arguments, one argument per line. Arguments are processed one at a time—first ASK arguments in order and then INPUT arguments. For non-Windows platforms, the command-line arguments program, connectivity, and args can be specified in this file.	<p>Production Reporting Server</p> <p>Production Reporting Execute</p>
<i>connectivity</i>	Information needed to connect to the database. If entered as “?” or omitted, Production Reporting prompts for the information.	<p>Production Reporting Server</p> <p>Production Reporting Execute</p>
	DB2	<p><i>[Database]/[Username]/ [Password]</i></p> <ul style="list-style-type: none"> ● <i>Database</i>—Database name. ● <i>Username</i>—User name for the database. ● <i>Password</i>—Password for the database.
	Informix	<p><i>Database[/username/password]</i></p> <ul style="list-style-type: none"> ● <i>Database</i>—Database name. ● <i>Username</i>—User name for the database. ● <i>Password</i>—Password for the database.
	ODBC	<i>Data_Source_Name/[Username]/[Password]</i>

Argument	Description	Program
		<ul style="list-style-type: none"> ● <i>Data_Source_Name</i>—ODBC driver name. ● <i>Username</i>—User name for the database. ● <i>Password</i>—Password for the database. <p>Note: This port has been certified against Microsoft SQL Server.</p>
	Oracle	<p><i>[Username]/[Password[@Database]]</i></p> <ul style="list-style-type: none"> ● <i>Database</i>—(Optional) Database connection string (for example, @sales.2cme.com). ● <i>Username</i>—Username for the database. ● <i>Password</i>—Password for the database.
	Sybase	<p><i>Username/[Password]</i></p> <ul style="list-style-type: none"> ● <i>Username</i>—User name for the database. ● <i>Password</i>—Password for the database.
	Teradata	<p><i>[tdpid/]username[,password]</i></p> <ul style="list-style-type: none"> ● <i>tdpid</i>—Teradata tdp name. ● <i>username</i>—User name for the database. ● <i>password</i>—Password for the database.

3

Production Reporting Command Reference

In This Chapter

About Production Reporting Commands.....	36
ADD	36
ALTER-COLOR-MAP	37
ALTER-CONNECTION	39
ALTER-LOCALE.....	41
ALTER-PRINTER.....	48
ALTER-REPORT	50
ALTER-TABLE	55
ARRAY-ADD, ARRAY-DIVIDE, ARRAY-MULTIPLY, ARRAY-SUBTRACT.....	58
ASK.....	60
BEGIN-DOCUMENT.....	61
BEGIN-EXECUTE	62
BEGIN-FOOTING	66
BEGIN-HEADING	68
BEGIN-PROCEDURE.....	69
BEGIN-PROGRAM	72
BEGIN-SELECT.....	72
BEGIN-SETUP.....	76
BEGIN-SQL.....	77
BREAK.....	80
CALL, CALL SYSTEM	80
CLEAR-ARRAY	85
CLOSE.....	86
CLOSE-RS.....	86
COLUMNS.....	88
COMMIT.....	88
CONCAT	89
CONNECT	91
CREATE-ARRAY	92
CREATE-COLOR-PALETTE	94
CREATE-LIST	95
CREATE-TABLE.....	96
#DEBUG	99

DECLARE-CHART	100
DECLARE-COLOR-MAP	126
DECLARE-CONNECTION	128
DECLARE-IMAGE	130
DECLARE-LAYOUT	132
DECLARE-PRINTER	137
DECLARE-PROCEDURE	143
DECLARE-REPORT	144
DECLARE-TABLE	146
DECLARE-TOC	148
DECLARE-VARIABLE	150
#DEFINE	153
DISPLAY	155
DIVIDE	158
DO	159
DRAW	160
DUMP-TABLE	163
#ELSE	164
ELSE	164
ENCODE	164
END-DECLARE, END-DOCUMENT, END-EVALUATE, END-FOOTING, END-HEADING	165
#END-IF, #ENDIF	166
END-IF	167
END-PROCEDURE, END-PROGRAM, END-SELECT, END-SETUP, END-SQL, END-WHILE, END-EXECUTE	167
EVALUATE	168
EXECUTE	170
EXIT-SELECT	175
EXTRACT	176
FILL-TABLE	177
FIND	179
GET	180
GET-COLOR	181
GOTO	183
#IF	184
IF	186
#IFDEF	187
#IFNDEF	188
#INCLUDE	188
INPUT	189
LAST-PAGE	192
LET	192
LOAD-LOOKUP	217
LOOKUP	222

LOWERCASE	223
MBTOSBS	224
MOVE.....	224
MULTIPLY.....	228
NEW-PAGE.....	229
NEW-REPORT.....	230
NEXT-COLUMN.....	231
NEXT-LISTING.....	232
OPEN.....	233
OPEN-RS.....	235
PAGE-NUMBER.....	237
POSITION	238
PRINT.....	239
PRINT-BAR-CODE	260
PRINT-CHART	263
PRINT-DIRECT.....	270
PRINT-IMAGE	271
PRINT-TABLE.....	273
PUT	274
READ	275
ROLLBACK	277
SBTOMBS	278
SECURITY	278
SET-COLOR	280
SET-DELAY-PRINT	282
SHOW	283
STOP	287
STRING	288
SUBTRACT	289
TOC-ENTRY.....	290
UNSTRING	291
UPPERCASE	292
USE.....	293
USE-COLUMN	293
USE-PRINTER-TYPE.....	294
USE-PROCEDURE	295
USE-REPORT.....	296
WHILE	297
WRITE	299
WRITE-RS	300

About Production Reporting Commands

This chapter describes each command in the Production Reporting lexicon. The commands follow the conventions in [Table 1, “Syntax Conventions,” on page 12](#) and use the abbreviations in [Table 2, “Syntax Abbreviation Conventions,” on page 13](#).

Caution!

If you copy codes directly from the examples in the PDF file, change the slanted quotes to regular quotes to avoid error messages.

Note:

For information on commands that are DDO-specific or have special instructions for DDO, see “DDO” in the index.

ADD

Function

Adds one number to another.

Syntax

```
ADD {src_num_lit|_var|_col} TO dst_num_var [ROUND=nn]
```

Arguments

src_num_lit|*_var*|*_col*

Numeric source value added to *dst_num_var*.

dst_num_var

Numeric destination variable containing results after execution.

ROUND

Results rounded to the number of digits to the right of the decimal point. Float variables: valid values are 0 to 15. Decimal variables: valid values are 0 to the precision of the variable. Integer variables: N/A.

Description

The source value is added to the destination variable and the result is placed in the destination. Source is always first and destination is always second. Money-related values (dollars and cents) use decimal variables.

Examples

```
add 10 to #counter
add #counter to #new_count
add &price to #total round=2
```

See Also

[LET](#) for information on complex arithmetic expressions

ALTER-COLOR-MAP

Function

Dynamically alters a color.

Syntax

```
ALTER-COLOR-MAP
NAME={color_name_lit|_var|_col}
VALUE=({color_name_lit|_var|_col} | {rgb})
```

Arguments

NAME

Name of the color. For example, *light blue*.

VALUE

RGB value of the color. For example, (193, 233, 230).

{color_name_lit|_var|_col}

- Includes alphanumeric characters (A-Z, 0-9), the underscore (_), and the dash (-)
- Must start with an alpha (A-Z) character
- Case insensitive
- A name in the format (RGBredgreenblue) cannot be assigned a value.
- 'None' and 'default' are reserved. 'Default' is used during execution when a referenced color is not defined in the runtime environment.

{rgb}

red_lit|_var|_col, *green_lit|_var|_col*, *blue_lit|_var|_col* where each component is a value in the range of 000 to 255. In `BEGIN-SETUP`, only literal values are allowed.

Default colors implicitly installed with Production Reporting:

black= (0,0,0)

white=(255,255,255)

gray=(128,128,128)

```
silver=(192,192,192)
red=(255,0,0)
green=(0,255,0)
blue=(0,0,255)
yellow=(255,255,0)
purple=(128,0,128)
olive=(128,128,0)
navy=(0,0,128)
aqua=(0,255,255)
lime=(0,128,0)
maroon=(128,0,0)
teal=(0,128,128)
fuchsia=(255,0,255)
```

Description

ALTER-COLOR-MAP is allowed wherever PRINT is allowed. ALTER-COLOR-MAP dynamically alters a color; it does not define a color.

Examples

```
begin-setup
  declare-color-map
    light_blue = (193, 222, 229)
  end-declare
end-setup

begin-program
  alter-color-map name = 'light_blue' value = (193, 233, 230)
  print 'Yellow Submarine' ()
  foreground = ('yellow')
  background = ('light_blue')
  get-color print-text-foreground = ($print-foreground)
  set-color print-text-foreground = ('purple')
  print 'Barney' (+1,1)
  set-color print-text-foreground = ($print-foreground)
end-program
```

See Also

[DECLARE-COLOR-MAP](#), [SET-COLOR](#), and [GET-COLOR](#)

ALTER-CONNECTION

Function

Alters data source logon parameters prior to logon. Can be used to override default connection logon parameters.

Note:

ALTER-CONNECTION is specific to Production Reporting DDO ports only.

Syntax

```
ALTER-CONNECTION
NAME=connection_name
[DSN={uq_txt_lit|_var}]
[USER={uq_txt_lit|_var}]
[PASSWORD={uq_txt_lit|_var}]
[PARAMETERS=keyword_str=attr_str;
[, keyword_str=attr_str;...]]
[NO-DUPLICATE=TRUE|FALSE]
SET-GENERATIONS=( [{dimension1, hierarchy1}
[,dimensioni, hierarchyi] ...])
SET-LEVELS=( [{dimension1, level1} [,dimensioni, leveli] ...])
SET-MEMBERS=( [{dimension1, level1} [,dimensioni, leveli] ...])
```

Arguments

NAME

User-defined name for describing a data source connection.

DSN

Logical data source name recorded in *Registry.properties*.

USER, PASSWORD

Traditional logon semantics.

PARAMETERS

Keyword-attribute pairs required by data source drivers for logon. Syntax restrictions include delimiting semi-colons (;) and equal signs (=). Keywords must match logon property names for data sources in property files.

NO-DUPLICATE=TRUE|FALSE (default is FALSE)

(Optional) Prevents Production Reporting from creating additional logins to data sources handling previous queries. Creating a new login is the default behavior, allowing a single CONNECTION declaration in a subquery. This behavior, while allowing dynamic logins as needed, causes difficulties when doing DDL (BEGIN-SQL) and DML (BEGIN-SELECT) against temporary tables in some data sources. In such cases, you must fetch from the temporary table

using the same login in which it was created. Here, you should code CONNECTION as NO-DUPLICATE=TRUE, and then use that connection in both the table creation logic of BEGIN-SQL and the row fetching logic of BEGIN-SELECT.

SET-GENERATIONS

Dimension hierarchy for the previously-declared dimension. In the following example:

```
set-generations=( 'product',5, 'time',1 )
```

SET-GENERATIONS:

- Returns the members in the ‘product’ dimension at the 5th generation in the hierarchy.
For example, returns all ‘Brand Name’ members (Generation Level 5) under the product hierarchy of ‘all products.drink.alcoholic beverages.beer and wine’. This increases the result set to a list of beers and wines.
- Returns the members in the ‘time’ dimension at the 1st generation in the hierarchy.
For example, returns all ‘Year’ members (Generation Level 1) under the time hierarchy of ‘1997.Q.2.’ This reduces the result set to ‘1997’.

To clear values defined with SET-GENERATIONS, use:

```
set-generations=()
```

SET-LEVELS

Extends dimension hierarchies for previously-declared dimensions. Dimensions and hierarchies defined with SET-LEVELS can be *literal* values only. In the following example:

```
set-levels=( 'product',2 )
```

- SET-LEVELS used with the previous SET-MEMBERS returns all members under the product hierarchy and the next two generations (Product SubCategory and Brand Name) for the product hierarchy of ‘all products.drink.alcoholic beverages.beer and wine’.
- SET-LEVELS used with the previous SET-MEMBERS *and* SET-GENERATIONS returns all members for generation levels 5 through 7 under the product hierarchy of ‘all products.drink.alcoholic beverages.beer and wine.’

To clear values defined with SET-LEVELS, use:

```
set-levels=()
```

SET-MEMBERS

Returns the members in a dimension, level, or hierarchy whose name is specified by a string. Dimensions and hierarchies defined with SET-MEMBERS can be *literal* values only. In the following example:

```
set-members=( 'product', 'all products.drink.alcoholic beverages.beer and wine', 'time', '1997.Q1.2' )
```

SET-MEMBERS:

- Returns the members in the 'product' dimension, at the 'all products' hierarchy, at the 'drink', 'alcoholic beverages', and 'beer and wine' levels.
- Returns the members in the 'time' dimension, at the '1997' hierarchy, at the 'Q1' and '2' levels.

To clear values defined with SET-MEMBERS, use:

```
set-members=()
```

Examples

```
alter-connection
  name=SAPR3-1
  password=psswd
parameters=logon.client=600;logon.ashost=starfish;logon.sysnr=
00;logon.language=EN;
```

Note:

Do not wrap the lines in the 'parameters=' line. Space restrictions dictate the wrapped line in the preceding example.

See Also

[DECLARE-CONNECTION](#)

ALTER-LOCALE

Function

Selects a locale or changes locale parameters for printing date, numeric, and money data and for data accepted by INPUT. Locales are preferences for language, currency, and the presentation of charts and numbers.

Syntax

```
ALTER-LOCALE
[LOCALE={txt_lit _var|DEFAULT|SYSTEM}]
[NUMBER-EDIT-MASK={txt_lit|_var|DEFAULT|SYSTEM}]
[MONEY-EDIT-MASK={txt_lit|_var|DEFAULT|SYSTEM}]
[DATE-EDIT-MASK={txt_lit|_var|DEFAULT|SYSTEM}]
[INPUT-DATE-EDIT-MASK={txt_lit|_var|DEFAULT|SYSTEM}]
[MONEY-SIGN={txt_lit|_var|DEFAULT|SYSTEM}]
[MONEY-SIGN-LOCATION={txt_var|DEFAULT|SYSTEM|LEFT|RIGHT}]
[THOUSAND-SEPARATOR={txt_lit|_var|DEFAULT|SYSTEM}]
[DECIMAL-SEPARATOR={txt_lit|_var|DEFAULT|SYSTEM}]
[DATE-SEPARATOR={txt_lit|_var|DEFAULT|SYSTEM}]
[TIME-SEPARATOR={txt_lit|_var|DEFAULT|SYSTEM}]
[EDIT-OPTION-NA={txt_lit|_var|DEFAULT|SYSTEM}]
[EDIT-OPTION-AM={txt_lit|_var|DEFAULT|SYSTEM}]
[EDIT-OPTION-PM={txt_lit|_var|DEFAULT|SYSTEM}]
[EDIT-OPTION-BC={txt_lit|_var|DEFAULT|SYSTEM}]
```

```
[EDIT-OPTION-AD={txt_lit|_var|DEFAULT|SYSTEM}]
[DAY-OF-WEEK-CASE={txt_var|DEFAULT|SYSTEM|UPPER|LOWER
|EDIT|NO-CHANGE}]
[DAY-OF-WEEK-FULL=({txt_lit1|_var1}...{txt_lit7|_var7})]
[DAY-OF-WEEK-SHORT=({txt_lit1|_var1}...{txt_lit7|_var7})]
[MONTHS-CASE={txt_var|DEFAULT|SYSTEM|UPPER|LOWER|EDIT|NO-CHANGE}]
[MONTHS-FULL=({txt_lit1|_var1}...{txt_lit12|_var12})]
[MONTHS-SHORT=({txt_lit1|_var1}...{txt_lit12|_var12})]
```

Arguments

Note:

Many of the settings can have a value of `DEFAULT` or `SYSTEM`. `DEFAULT` retrieves values from the corresponding setting of the *default* locale in the [Default-Settings] section of `SQR.INI`. `SYSTEM` retrieves values from the corresponding setting of the *system* locale. You can alter the *system* locale using [ALTER-LOCALE](#); however, you cannot define it in `SQR.INI`.

LOCALE

Locale name. This name must be defined in `SQR.INI`. If omitted, the current locale is used. The locale name is case-insensitive and is limited to A-Z, 0-9, underscore, or hyphen. To determine the current locale, print the reserved variable `$sqr-locale`.

NUMBER-EDIT-MASK

Numeric edit mask used with the keyword `NUMBER` in [PRINT](#), [MOVE](#), [SHOW](#), or [DISPLAY](#).

MONEY-EDIT-MASK

Numeric edit mask used with the keyword `MONEY` in [PRINT](#), [MOVE](#), [SHOW](#), or [DISPLAY](#).

DATE-EDIT-MASK

Date edit mask used with the keyword `DATE` in [PRINT](#), [MOVE](#), [SHOW](#), or [DISPLAY](#), or the [LET](#) functions `datetostr()` or `strtodate()`.

INPUT-DATE-EDIT-MASK

Default date format to use with [INPUT](#) when `TYPE=DATE` is specified or the input variable is a date variable.

Note:

For more information on Edit Masks, see “[“Edit Masks” on page 247](#)”.

MONEY-SIGN

Character(s) that replace \$ or other currency symbols used in edit masks.

MONEY-SIGN-LOCATION

Where to place the MONEY-SIGN character(s). Valid values are LEFT and RIGHT.

THOUSAND-SEPARATOR

Character to replace the ',' edit character.

DECIMAL-SEPARATOR

Character to replace the '.' edit character.

DATE-SEPARATOR

Character to replace the '/' character.

TIME-SEPARATOR

Character to replace the ':' character.

EDIT-OPTION-NA

Character(s) to use with the 'na' option.

EDIT-OPTION-AM

Character(s) to replace 'AM'.

EDIT-OPTION-PM

Character(s) to replace 'PM'.

EDIT-OPTION-BC

Character(s) to replace 'BC'.

EDIT-OPTION-AD

Character(s) to replace 'AD'.

DAY-OF-WEEK-CASE

How the case for DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT is affected when used with 'DAY' or 'DY'. Valid values are:

- UPPER, LOWER—Forces output to all uppercase or lowercase, ignoring the case of the format code in the edit mask.
- EDIT—Uses the case specified with the format code in the edit mask.
- NO-CHANGE—Ignores the case of the format code and outputs the day of week defined in DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT.

DAY-OF-WEEK-FULL

Full names for the days of the week. Production Reporting considers the first day to be Sunday. You must specify all seven days.

DAY-OF-WEEK-SHORT

Abbreviated names for the days of the week. Production Reporting considers the first day to be Sunday. You must specify all seven abbreviations.

MONTHS-CASE

How the case for MONTHS-FULL or MONTHS-SHORT is affected when used with 'MONTH' or 'MON'. Valid values are:

- UPPER, LOWER—Forces output to all uppercase or lowercase, ignoring the case of the format code in the edit mask.
- EDIT—Uses the case specified with the format code in the edit mask.
- NO-CHANGE—Ignores the case of the format code and outputs the month defined in MONTHS-FULL or MONTHS-SHORT

MONTHS-FULL

Full names for the months of the year. Production Reporting considers the first month to be January. You must specify all 12 months.

MONTHS-SHORT

Abbreviated names for the months of the year. Production Reporting considers the first month to be January. You must specify all 12 abbreviations.

Description

When you install Production Reporting, the default locale is set to SYSTEM.

Table 6 SYSTEM Locale Settings

Keyword	Value
NUMBER-EDIT-MASK	PRINT prints two digits to the right of the decimal point and left justifies the number. MOVE, SHOW, and DISPLAY format the number with six digits to the right of the decimal point and left justifies the number.
MONEY-EDIT-MASK	Same default values as those in NUMBER-EDIT-MASK.
DATE-EDIT-MASK	Date formats in Table 61, “Default Formats by Database,” on page 251.
INPUT-DATE-EDIT-MASK	Date edit masks in Table 59, “Sample Date Edit Masks,” on page 247.
MONEY-SIGN	'\$'
MONEY-SIGN-LOCATION	LEFT
THOUSAND-SEPARATOR	','
DECIMAL-SEPARATOR	'.'
DATE-SEPARATOR	'/'
TIME-SEPARATOR	':'
EDIT-OPTION-NA	'n/a'

Keyword	Value
EDIT-OPTION-AM	'am'
EDIT-OPTION-PM	'pm'
EDIT-OPTION-BC	'bc'
EDIT-OPTION-AD	'ad'
DAY-OF-WEEK-CASE	EDIT
DAY-OF-WEEK-FULL	('Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday')
DAY-OF-WEEK-SHORT	('Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat')
MONTHS-CASE	EDIT
MONTHS=FULL	('January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December')
MONTHS-SHORT	('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')

Examples

The following code:

```
!
! The following program segments will illustrate the various
! ALTER-LOCALE features.
!
begin-setup
declare-variable
date $date $date1 $date2 $date3
end-declare
end-setup
!
! Set default masks
!
alter-locale
number-edit-mask = '9,999,999.99'
money-edit-mask = '$999,999,999.99'
date-edit-mask = 'Mon DD, YYYY'

let #value = 123456
let $edit = 'Mon DD YYYY HH:MI:SS'
let $date = strtodate('Jan 01 1995 11:22:33', $edit)
show 'With NUMBER option #Value = ' #value number
show 'With MONEY option #Value = ' #value money
show 'Without NUMBER option #Value = ' #value
show 'With DATE option $Date = ' $date date
show 'Without DATE option $Date = ' $date
```

Produces the following output:

```
With NUMBER option #Value = 123,456.00
With MONEY option #Value = $123,456.00
```

```
Without NUMBER option #Value = 123456.000000
With DATE option      $Date = Jan 01, 1995
Without DATE option   $Date = 01-JAN-95
```

The following code:

```
!
! Reset locale to Production Reporting defaults and assign a multi-
character
! money-sign.
!
alter-Locale
locale = 'System'
money-sign = 'AU$'           ! Australian dollars

let #value = 123456
show #value edit '$999,999,999,999.99'
show #value edit '$$$$,$$$$999,999.99'
```

Produces the following output:

```
AU$      123,456.00
        AU$123,456.00
```

The following code:

```
!
! Move the money-sign to the right side of the value. Note
! the leading space.
!
alter-locale
  money-sign = ' AU$'           ! Australian dollars
  money-sign-location = right
let #value = 123456
show #value edit '$999,999,999,999.99'
show #value edit '$$$$,$$$$999,999.99'
```

Produces the following output:

```
123,456.00 AU$
123,456.00 AU$
```

The following code:

```
!
! Reset locale to Production Reporting defaults and flip the thousand and
! decimal separator characters.
!
alter-locale
  locale = 'System'
  thousand-separator = '.'
  decimal-separator = ','

let #value = 123456
show #value edit '999,999,999,999.99'
```

Produces the following output:

123.456,00

The following code:

```
!  
! Reset locale to Production Reporting defaults and change the date and  
time  
! separators  
!  
alter-locale  
  locale = 'System'  
  date-separator = '-'  
  time-separator = '.'  
  
let $edit = 'Mon/DD/YYYY HH:MI:SS'  
let $date = strtodate('Jan/01/1995 11:22:33', $edit)  
show $date edit :$edit
```

Produces the following output:

```
Jan-01-1995 11.22.33
```

The following code:

```
!  
! Reset locale to Production Reporting defaults and change the text used  
with  
! the edit options 'na', 'am', 'pm', 'bc', 'ad'  
!  
alter-locale  
  locale = 'System'  
  edit-option-na = 'Not/Applicable'  
  edit-option-am = 'a.m.'  
  edit-option-pm = 'p.m.'  
  edit-option-bc = 'b.c.'  
  edit-option-ad = 'a.d.'  
  
let $value = ''  
let $edit = 'Mon DD YYYY HH:MI'  
let $date1 = strtodate('Jan 01 1995 11:59', $edit)  
let $date2 = strtodate('Feb 28 1995 12:01', $edit)  
show $value edit '999,999,999,999.99Na'  
show $date1 edit 'Mon DD YYYY HH:MI:SS PM'  
show $date2 edit 'Mon DD YYYY HH:MI:SS pm'
```

Produces the following output:

```
Not/Applicable  
Jan 01 1995 11:59:00 A.M.  
Feb 28 1995 12:01:00 p.m.
```

The following code:

```
!  
! Input some dates using the 'system' locale and  
! output using other locales from the SQR.INI file.  
!  
alter-locale
```

```

    locale = 'System'
let $date1 = strtodate('Jan 01 1995', 'Mon DD YYYY')
let $date2 = strtodate('Feb 28 1995', 'Mon DD YYYY')
let $date3 = strtodate('Mar 15 1995', 'Mon DD YYYY')
show 'System:'
show
show $date1 edit 'Month DD YYYY' ' is ' $date1 edit 'Day'
show $date2 edit 'Month DD YYYY' ' is ' $date2 edit 'Day'
show $date3 edit 'Month DD YYYY' ' is ' $date3 edit 'Day'
alter-locale
    locale = 'German'
show
show 'German:'
show
show $date1 edit 'DD Month YYYY' ' ist ' $date1 edit 'Day'
show $date2 edit 'DD Month YYYY' ' ist ' $date2 edit 'Day'
show $date3 edit 'DD Month YYYY' ' ist ' $date3 edit 'Day'
alter-locale
    locale = 'Spanish'
show
show 'Spanish:'
show
show $date1 edit 'DD Month YYYY' ' es ' $date1 edit 'Day'
show $date2 edit 'DD Month YYYY' ' es ' $date2 edit 'Day'
show $date3 edit 'DD Month YYYY' ' es ' $date3 edit 'Day'

```

Produces the following output:

```

System:
January 01 1995 is Sunday
February 28 1995 is Tuesday
March 15 1995 is Wednesday
German:
01 Januar 1995 ist Sonntag
28 Februar 1995 ist Dienstag
15 März 1995 ist Mittwoch
Spanish:
01 enero 1995 es domingo
28 febrero 1995 es martes
15 marzo 1995 es miércoles

```

See Also

- [DISPLAY, LET, MOVE, PRINT, and SHOW](#)
- [Chapter 6, “SQR.INI”](#)

ALTER-PRINTER

Function

Alters printer parameters at run time.

Syntax

```
ALTER-PRINTER
```



```
[POINT-SIZE={point_size_num_lit|_var}]  
[FONT-TYPE={font_type|txt_var}]  
[SYMBOL-SET={symbol_set_id|txt_var}]  
[FONT={font_int_lit|_var}]  
[PITCH={pitch_num_lit|_var}]
```

Arguments

POINT-SIZE

New font point size.

FONT-TYPE

New font type. (PROPORTIONAL or FIXED)

SYMBOL-SET

New symbol set identifier.

FONT

New font as a number. (For example, 3 = Courier and 4 = Helvetica.)

PITCH

New pitch in characters per inch.

Note:

See [Table 32, “DECLARE-PRINTER Command Arguments,” on page 138](#) for more information on ALTER-PRINTER arguments.

Description

You can place ALTER-PRINTER in any part of an Production Reporting program except the SETUP section.

ALTER-PRINTER changes the attributes of the *current* printer for the *current* report. Attributes that do not apply to the *current* printer, are ignored. For example, ALTER-PRINTER is ignored if it specifies proportional fonts for a report printed on a line printer. When a program creates multiple reports and shares the printer with another report, the attributes are changed for that report as well.

Examples

Change the font and symbol set for the current printer.

```
alter-printer  
font=4      ! Helvetica  
symbol-set=12U    ! PC-850 Multilingual
```

If the output prints to a PostScript printer, SYMBOL-SET ignored; however, if the SPF file is kept (see the -KEEP command line flag) and later printed on an HP LaserJet, the symbol set 12U can be used.

See Also

[DECLARE-PRINTER](#)

ALTER-REPORT

Function

Alters report-specific functionality.

Syntax

```
ALTER-REPORT
[HEADING={heading_name_txt_lit|_var|_col}
[HEADING-SIZE={heading_size_int_lit|_var|_col}
[FOOTING={footing_name_txt_lit|_var|_col}
[FOOTING-SIZE={footing_size_int_lit|_var|_col}]
[PDF-APPEARANCE=(appearance_lit|_var|_col)]
[PDF-INFORMATION=(information_lit|_var|_col, value_lit|_var|_col
[,information_lit|_var|_col, value_lit|_var|_col]...)]
[PDF-OPEN-ACTION=(openaction_lit|_var|_col,
[,name_lit|_var|_col,value_lit|_var|_col]...)]
[PDF-PAGE-TRANSITION=(transition_lit|_var|_col, duration_lit|_var|_col)]
[PDF-SECURITY=(security_lit|_var|_col, value_lit|_var|_col[,security_lit|
_var|_col, value_lit|_var|_col]...)]
[PDF-VIEWER-PREFERENCE=(preference_lit|_var|_col, value_lit|_var|_col
[,preference_lit|_var|_col, value_lit|_var|_col]...)]
```

Arguments

HEADING

Name of the BEGIN-HEADING section.

HEADING-SIZE

Amount of space occupied by the BEGIN-HEADING section.

FOOTING

Name of the BEGIN-FOOTING section.

FOOTING-SIZE

Amount of space occupied by the BEGIN-FOOTING section.

PDF-APPEARANCE

Appearance of the document when opened.

Table 7 Appearance Values

Appearance	Description
None	Neither bookmarks nor thumbnails are visible. (Default value when the document does not contain bookmarks.)
Bookmarks	Opens documents with bookmarks visible. (Default value when the document contains bookmarks.)
Thumbnails	Opens documents with thumbnails visible.
Fullscreen	Opens in full-screen mode. (This value does not work in the browser.)

For example:

```
ALTER-REPORT
  PDF-APPEARANCE= ( 'None ' )
```

```
PDF-INFORMATION
```

The information name to address and the data to apply to the information parameter. You can specify any of the standard information names shown in [Table 8](#), or you can specify any user-defined name with the exception of the following names which are reserved by the PDFlib: *CreationDate*, *Producer*, *ModDate*, or *Trapped*.

Table 8 Standard PDF Information Names

Name	Description
Subject	Document subject.
Title	Document title. (The default is the product name and version string.)
Creator	Software used to create the document.
Author	Document author.
Keywords	Keywords describing the document contents.

For example:

```
ALTER-REPORT
  PDF-INFORMATION=( 'Author', 'Peter Burton', 'Keywords',
  'Sample Private' )
```

```
PDF-OPEN-ACTION
```

Action the PDF viewer takes when opening a file, name of the additional value, and value to apply.

Table 9 Open Actions

Open Action	Description
Fixed	Use a fixed destination view defined with <i>Zoom</i> , <i>Left</i> , and <i>Top</i> .

Open Action	Description
Window	Fit the complete page to the window.
Width	Fit the page width to the window defined with <i>Top</i> .
Height	Fit the page height to the window defined with <i>Left</i> .
Rectangle	Fit the rectangle defined with <i>Left</i> , <i>Bottom</i> , <i>Right</i> , and <i>Top</i> .
Visible	Fit the visible contents of the page to the window.
VisibleWidth	Fit the visible contents of the page to the window defined with <i>Top</i> .
VisibleHeight	Fit the visible contents of the page to the window defined with <i>Left</i> .

Table 10 Additional Values

Value	Description
Page	Initial page to display. (default = 1)
Zoom	Zoom factor to use when the page displays. (default =100)
Left	Column number of the page to position at the left edge of the window. (default = 1)
Top	Line number of the page that will be positioned at the top edge of the window. (default = 1)
Bottom	Last line of the page to display. (default = entire page)
Right	Last column of the page to display. (default = entire page)

For example:

```
ALTER-REPORT
  PDF-OPEN-ACTION=('Fixed', 'Zoom, 75, 'Page', 2)
```

```
PDF-PAGE-TRANSITION
```

Page transition for current and future pages and the duration (in seconds) for the transition.

Table 11 Transitions

Transition	Description
Split	Two lines sweeping across the screen reveal the page.
Blinds	Multiple lines sweeping across the screen reveal the page.
Box	A box reveals the page.
Wipe	A single line sweeping across the screen reveals the page.
Dissolve	The old page dissolves to reveal the new page.
Glitter	The dissolve effect moves from one screen edge to another.

Transition	Description
Replace	The old page is replaced by the new page. (default value)

For example:

```
ALTER-REPORT
  PDF-PAGE-TRANSITION=( 'Wipe' , 3.25)
```

```
PDF-SECURITY
```

Security parameter to address and value to apply to the security parameter.

Table 12 Security Parameters

Security	Description
User-Password	User level password applied to the generated PDF. Up to 32 characters. Needed to open the document.
Master-Password	Master level password applied to the generated PDF. Up to 32 characters. Used to open documents or override the permissions.
Permissions	Permissions applied to the generated PDF. <ul style="list-style-type: none"> ● NoPrint—Prevents printing the file. ● NoModify—Prevents users from adding form fields or making any other changes. ● NoCopy—Prevents copying and extracting text and graphics and disables the accessibility interface. ● NoAnnots—Prevents adding or changing comments or form fields. ● NoForm—Prevents form field filling, even if NoAnnots is not specified. (Requires Acrobat 5 or higher) ● NoAccessible—Prevents extracting text or graphics for accessibility purposes. For example, a screen reader program.(Requires Acrobat 5 or higher) ● NoAssemble—Prevents inserting, deleting, or rotating pages and creating bookmarks and thumbnails, even if NoModify is not specified. (Requires Acrobat 5 or higher) ● NoHiResPrint—Prevents high-resolution printing. If NoPrint is not specified, printing this setting is restricted to the <i>Print As Image</i> feature, which prints a low-resolution rendition of the page. (Requires Acrobat 5 or higher)

Note:

Default documents have no passwords and all permissions.

For example:

```
ALTER-REPORT
  PDF-SECURITY=( 'User-Password' , $User_Password,
  'Master-Password' , &Master_Password, 'Permissions' , 'NoPrint NoCopy')
```

```
PDF-VIEWER-PREFERENCE
```

Viewer preference to address and value to apply to the preference.

Table 13 Viewer Preferences

Preference	Description
Toolbar	<i>True</i> hides Acrobat's tool bar. Default = <i>False</i>
MenuBar	<i>True</i> hides Acrobat's menu bar. Default = <i>False</i>
WindowUI	<i>True</i> hides Acrobat's windows controls. Default = <i>False</i>
FitWindow	<i>True</i> resizes the document's window to the size of the first page. Default = <i>False</i>
CenterWindow	<i>True</i> positions the document's windows in the center of the screen. Default = <i>False</i>
DisplayDocTitle	<i>True</i> displays the document information field in Acrobat's title bar; <i>False</i> displays the file name. Default = <i>False</i>
NonFullscreen-PageMode	How to display the document on exiting full-screen mode: <ul style="list-style-type: none"> ● UseOutlines–Displays page and document outlines. ● UseThumbs–Displays page and thumbnails. ● UseNone–Displays neither document outlines nor thumbnails. Default = <i>UseNone</i>
Direction	Reading order of the document. (Affects scroll ordering in double-page view.) <ul style="list-style-type: none"> ● L2R–Left to right ● R2L–Right to left (including vertical writing systems). Default = <i>L2R</i>

For example:

```
ALTER-REPORT
  PDF-VIEWER-PREFERENCE=('Direction', 'R2L')
```

Description

ALTER-REPORT dynamically changes how much space active heading and/or footing sections occupy for the current report. For PDF reports, you can add information about the report, control the display and appearance of the report, and control security settings.

If HEADING or FOOTING = 'NONE', the section is disabled for the current report.

If HEADING or FOOTING = 'DEFAULT', the section reverts to whatever was in effect when the report was initiated.

If no HEADING or FOOTING value is set HEADING-SIZE and/or FOOTING-SIZE values affect the HEADING/FOOTING currently used.

When HEADING, HEADING-SIZE, FOOTING, or FOOTING-SIZE is defined: the command is not invoked in a BEGIN-HEADING and/or BEGIN-FOOTING section, the page is not written to, and the assignment takes effect immediately. Otherwise, it takes effect for the next page.

Examples

```
begin-footing 2 name=confidential
```

```

    print 'Company Confidential' (1,1,0) center
    page-number (2,37,0)
end-footing

begin-footing 2 name=proprietary
    print 'Company Proprietary' (1,1,0) center
    page-number (2,37,0)
end-footing

begin-program
    .
    .
    .
    alter-report
        footing = 'Proprietary'
        footing-size = 6          ! Increase depth
    .
    .
    .
end-program

```

See Also

[BEGIN-FOOTING](#) and [BEGIN-HEADING](#)

ALTER-TABLE

Function

Manipulates table attributes.

Syntax

```

ALTER-TABLE
NAME=table_name_var|_lit|_col
ACTION=action_lit
[COUNT=count_var|_lit|_col]
[ROW=row_var|_lit|_col]
[ATTRIBUTES=({keyword1},{value1}, ..., {keywordn},{valuen})]

```

Arguments

NAME

Name of the table created by CREATE-TABLE.

ACTION

Action to perform on the table.

- When ACTION=ERASE, the underlying table data is removed from memory. When this action is specified, no other keywords can be specified.
- When ACTION=INFO, table information will be retrieved as defined. The following keywords are allowed (you must specify at least one):

- ROW–The last table row acted upon.
- COUNT–Number of rows in the table.
- When ACTION=ATTRIBUTES, table rows will have all unassigned columns set to the specified attributes. If an attribute is not specified, the current setting is retained. The following keywords are allowed:
 - ROW–The row to be affected.
 - COUNT–Number of rows. (Default=1)
 - ATTRIBUTES–Attributes to apply to the rows.

COUNT

Number of affected rows.

ROW

The insertion row.

ATTRIBUTES

Attributes to apply to the row.

Table 14 ALTER-TABLE Attributes

Attribute	Description
DEFAULT	Causes all attributes to be set to their default values as defined by DECLARE-TABLE and CREATE-TABLE before applying any other attributes.
BACKGROUND	Background color name or RGB triplet
BOLD	YES NO
CENTER	YES NO
COLUMN-LEADING	Expressed in decipoints
COLUMN-LINE COLOR	Color name or RGB triplet
COLUMN-LINE-THICKNESS	Expressed in decipoints
COLUMN-LINE-STYLE	SOLID SQUARE-DOT DASH DASH-DOT LONG-DASH LONG-DASH-DOT LONG-DASH-DOT-DOT
HEADER	YES NO
FILL-COLOR	Fill color name or RGB triplet. Default=NONE
FONT	Font number
FOOTER	YES NO
FOREGROUND	Foreground color name or RGB triplet
GROUP-HEADER	YES NO

Attribute	Description
GROUP-FOOTER	YES NO
ITALIC	YES NO
POINT-SIZE	Point size of the font
ROW-BORDER-COLOR	Color name or RGB triplet
ROW-BORDER-LINE-STYLE	SOLID SQUARE-DOT DASH DASH-DOT LONG-DASH LONG-DASH-DOT LONG-DASH-DOT-DOT
ROW-BORDER-THICKNESS	Expressed in decipoints
ROW-LINE-COLOR	Color name or RGB triple
ROW-LINE-STYLE	SOLID SQUARE-DOT DASH DASH-DOT LONG-DASH LONG-DASH-DOT LONG-DASH-DOT-DOT
ROW-LINE-THICKNESS	Expressed in decipoints
UNDERLINE	YES NO
WRAP	YES NO maximum number of lines
WRAP-HEIGHT	Number of lines between each wrapped line
WRAP-ON	Characters on which to force a <code>WRAP</code>
WRAP-STRIP	Characters to change to a space before the <code>WRAP</code> is done

Description

Use `ALTER-TABLE` in any section except `BEGIN-SETUP`, `BEGIN-SQL`, and `BEGIN-DOCUMENT` to manipulate table objects.

Example

```
alter-table name='customers' action='attributes' count=1 row=1
  attributes=('foreground', ('green'), 'background', ('red'),
    'center', 'no', 'italic', 'yes', 'row-border-color', 'black',
    'row-border-thickness', 4, 'fill-color', 'yellow', 'point-size', 8,
    'bold', 'yes')
```

See Also

[CREATE-TABLE](#), [DECLARE-TABLE](#), [DUMP-TABLE](#), [FILL-TABLE](#), [PRINT-TABLE](#)

ARRAY-ADD, ARRAY-DIVIDE, ARRAY-MULTIPLY, ARRAY-SUBTRACT

Function

Performs arithmetic on array elements.

Syntax

```
ARRAY-ADD{src_num_lit|_var|_col}...TO  
dst_array_name (element_lit|_var|_col) [field  
[ (occurs_lit|_var|_col) ] ] ...  
ARRAY-DIVIDE{src_num_lit|_var|_col}...INTO  
dst_array_name (element_int_lit|_var|_col) [field  
[ (occurs_lit|_var|_col) ] ] ...  
ARRAY-MULTIPLY{src_num_lit|_var|_col}...TIMES  
dst_array_name (element_int_lit|_var|_col) [field  
[ (occurs_lit|_var|_col) ] ] ...  
ARRAY-SUBTRACT{src_num_lit|_var|_col}...FROM  
dst_array_name (element_int_lit|_var|_col) [field  
[ (occurs_lit|_var|_col) ] ] ...
```

Arguments

src_num_lit|*_var*|*_col*

Source value(s) are added, divided, multiplied, or subtracted from the respective destination array fields. All variables must be numeric.

dst_array_name (*element_int_lit*|*_var*|*_col*) [*field* [(*occurs_lit*|*_var*|
_col)]]

Destination array field(s) contain the results after the operation. All variables must be numeric.

Description

The following information applies to the array arithmetic commands:

- The array must be created with CREATE-ARRAY.
Array arithmetic commands perform on one or more source numbers and place the result into the corresponding array field.
- Array element and field occurrence numbers can be numeric literals (123) or numeric variables (*#j*) and can be from zero (0) to one less than the size of the array.
- If fields are not listed, the results are placed into consecutively defined fields in the array. If fields are listed, results are placed into those fields, at the specified occurrence of the field. If an occurrence is not specified the zeroth (0) occurrence is used.
- All fields must be of the type NUMBER, DECIMAL, FLOAT, or INTEGER. They cannot be of type DATE, CHAR, or TEXT.
- If division by zero is attempted, a warning message is displayed, the result field is unchanged, and Production Reporting continues executing.

Examples

```
array-add &salary #comm to emps(#j)
```

Adds &salary and #comm to the first two fields defined in the emps array. The #j'th element of the array is used.

```
array-subtract #lost #count 1 from stats(#j2) loses tot sequence
```

Subtracts #lost, #count, and 1 from the fields loses, tot and sequence of the #j2'th element of the stats array.

```
array-multiply 2 2 2 times percentages(#i) p(0) p(1) p(2)
```

Multiplies occurrences 0 through 2 of the field p in the #i'th element of the percentages array by 2.

```
array-divide 100 into commissions(#j) salesman(#i2)
```

Divides the #i2'th occurrence of the salesman field of the #j'th element of the commissions array by 100.

The following example uses ARRAY-ADD in an Production Reporting program.

```
begin-setup
  ! declare arrays
  create-array name=emps size=1 ! one row needed for this example
  field=Salary:number=35000 ! initialize to 35,000
  field=Comm:number=5000 ! initialize to 5,000
end-setup

begin-program
  do Main
end-program

begin-procedure Main local
  ! Show original contents of the arrays, then the modified arrays
  ! array-add
  ! retrieve values from the only row of array "emps"
  get #sal #com FROM emps(0) Salary Comm
  print 'Array-Add' (+1, 1)

  print 'Add 1000 to each column' (+1, 1)
  print 'Salary' (+1, 3) bold underline
  print 'Comm' (,25) bold underline

  print #sal (+1, 1) money
  print #com (,22) money

  let #salary = 1000
  let #commission = 1000
  let #j = 0 ! address the array row with variable "#j"
  ! Add 1000 (in variables) to each column of row 0 (the 1st and only row)
  array-add #salary #commission TO emps(#j)
  ! retrieved the new "added" values
  get #sal #com FROM emps(0) Salary Comm
  print #sal (+1,1) money
  print #com (,22) money
```

end-procedure

See Also

- [CREATE-ARRAY](#) for information on creating an array.
- [CLEAR-ARRAY](#) for information on clearing or initializing an array.
- [GET](#), [PUT](#), and [LET](#) for information on using arrays.

ASK

Function

Retrieves values for compile-time substitution variables. Retrieval can be by user input, command-line arguments, or as entries in @file on the command line. (See [“Production Reporting Command-line Arguments” on page 31](#) for more information.)

Syntax

```
ASK substitution_variable [prompt]
```

Arguments

substitution_variable

Variable to use as the substitution variable.

prompt

(Optional) Literal text string displayed as a prompt if the substitution variable value is not entered on the command line or in an argument file.

Description

The value of the substitution variable replaces the reference variable in the program. Variables are referenced by enclosing the variable name in braces, for example, '{state_name}'. If the substitution variable is text or date, surround the brackets by single quotes. Substitutions are made as the program is compiled and are saved in the SQT file. Each variable can be referenced multiple times.

ASK is used only in the SETUP section and must appear prior to any substitution variable references.

You cannot break ASK across program lines.

Examples

In the following example, *state* takes the user-supplied value in response to the prompt *Enter state for this report*.

```
begin-setup
  ask state 'Enter state for this report'
end-setup
```

```
...
begin-select
  name, city, state, zip
  from customers where state = '{state}'
end-select
```

See Also

- [INPUT](#) for information on input at run time
- “Compiling Programs and using Production Reporting Execute” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*

BEGIN-DOCUMENT

Function

Begins a `DOCUMENT` paragraph. Document paragraphs allow you to write free-form text (for example, form letters and invoices).

Syntax

```
BEGIN-DOCUMENT position
.
.
.
END-DOCUMENT
```

Arguments

position

Location on the page where the document begins. Can be fixed or relative to the current position.

Description

Database columns, Production Reporting variables, and document markers can be referenced within documents. Their location determines where they are printed. Do not use tabs in document paragraphs. To indent text or fields, use the spacebar. If variables printed inside a document paragraph are variable in length, manipulate the variables outside the `DOCUMENT` paragraph.

Note:

Documents must be executed before referencing their document markers. Since documents can be printed at relative positions on the page, the location of document markers may not be known by Production Reporting until the document is executed.

Examples

```
begin-document (1,1)
```

```
.b
Dear $firstname
...
end-document
```

See Also

- END-DOCUMENT
- “Creating Form Letters” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide* for a full example of BEGIN-DOCUMENT

BEGIN-EXECUTE

Note:

BEGIN-EXECUTE is specific to Production Reporting DDO ports only.

Function

Begins a new query or procedure execution. BEGIN-EXECUTE is only required when additional information about the DDO datasource or query is needed. In a BEGIN-EXECUTE paragraph, the syntax of BEGIN-SELECT varies as shown below.

Syntax

```
BEGIN-EXECUTE
[CONNECTION=uq_txt_lit]
[ON-ERROR=sqr_procedure(arg1[,argi]...)]
[RSV=num_var]
[STATUS=list_var|num_var|txt_var]
[PROPERTIES=(key_txt_lit|_var)=(value_txt_lit|_var|_col)|(num_lit|_var|_col),...)]
[SCHEMA={txt_lit|_var}]
[
  PROCEDURE={txt_lit|_var}
  [PARAMETERS=(arg1 [IN|INOUT])|NULL] [,argi [IN|INOUT]]|NULL] ... )]
  (or)
  COMMAND={txt_lit|_var}
  (or)
  GETDATA={txt_lit|_var}
]
[BEGIN-SELECT [BEFORE=sqr_procedure(arg1[,argi]...)]
  [AFTER=sqr_procedure(arg1[,argi]...)]]
col-name TYPE=CHAR|TEXT|NUMBER|DATE [edit-mask]
[on-break]...
[FROM ROWSETS=(m|m-n|m-|-n) [, ...]]|{ALL})|
  {FROM PARAMETER={txt_lit|_var}|
  {FROM {table_name}}]
END-SELECT]
END-EXECUTE
```

Arguments

CONNECTION

Name previously defined using `DECLARE-CONNECTION`. If not specified, Production Reporting uses the default connection defined by the command-line entries for data source (`DSN`), username (`USER`), and password (`PASSWORD`). Name is not case-sensitive.

ON-ERROR

Procedure to execute if errors occur.

RSV

Row Set Variable. Global Production Reporting variable containing the row set retrieved.

STATUS

List or scalar variable that receives the stored procedure status.

PROPERTIES

Keyword/value pair(s) that represent modifications made to the Properties of the data source (defined by `CONNECTION=`).

SCHEMA

Data source location of the object queried. Valid options include:

- `PROCEDURE`—Name of the data source/stored procedure to execute. If the data source is SAP R/3, this procedure is a BAPI. The name can include spaces.
- `PARAMETERS`—Scalar and/or list variables of the form `list_var|num_lit|txt_lit|txt_var|num_var|any_col`. If you do not specify the keywords `IN` or `INOUT`, `IN` is the default. Define all parameters in order; leaving any parameters unnamed causes a syntax error. To ignore a parameter, fill its position with the keyword `NULL`.
- `COMMAND`—Text string passed to the data source without modification by Production Reporting. Can include embedded Production Reporting variables.
- `GETDATA`—Supports the Java (DDO) `GetData` paradigm for data access.

BEFORE/AFTER

Production Reporting procedure to execute before or after the row set. The procedure is not performed unless at least one row is returned from the specified rowset(s).

FROM ROWSETS

Special case addition to `BEGIN-SELECT`. Available for use with all data source types, including SAP R/3 and JDBC. Names the rowset(s) from which to retrieve the column variables. For multiple row sets, use identical column name/type signatures. Row set numbers must be sequential from left-to-right within the parentheses, and they must not overlap as in this example: (1-3, 2-4). Numeric literals or #variables are allowed.

In `FROM ROWSETS`, “m” and “n” are integer values (1, 2, 3, 4, 5). “m-n” is 3-5 (rowsets 3, 4, 5). “m-” is 4- (rowsets 4, 5). “-n” is -3 (rowset 1, 2, 3).

FROM PARAMETER

Special case addition to BEGIN-SELECT. Available only for SAP R/3 data sources. Use only with the PROCEDURE keyword. Names an output parameter containing one or more rows from which column variables are to retrieve.

Note:

This is similar to the PARAMETERS= statement in DECLARE-CONNECTION and ALTER-CONNECTION, except the properties specified here alter the flow of returned information, as opposed to simply setting login properties. Can be used with any data-access model (Procedure, Command, Getdata). An application of this statement would be in the MDB setting, where it might be used to specify such things as Level, Generation, or Include-Column. For example, PROPERTIES = ('SetColumn' = 5)

FROM {table_name}

Relational data source table name. Literals or variables are *not* allowed.

Examples

```
begin-setup
  declare-variable
    date $when_ordered
    text $ship_method
    integer #theRow
    integer #theStatus
    integer #howMany
  end-declare
end-setup

input #howMany type=integer
input $pword
let %parm1 = list($when_ordered, $ship_method, #howMany)

declare-connection SAPR3
  user=scott
  parameters=clientno=5;node=starfish;
end-declare

alter-connection
  name=SAPR3
  password=$pword

begin-execute
  connection=SAPR3
  rsv=#theRow
  status=#theStatus
  on-error=it_failed(#theStatus)
  procedure='CreditHistory version 5'
  parameters=(%parm1, 'recalculate')
  print 'proc ran OK, status is '(+1,1)
  print #theStatus (,+5) edit 999
```



```

begin-select before=do_eject after=cleanup
  city &col=char (1,1) on-break level=1 after=city-tot
  keyval type=number (1,+1)
  rcvd type=date (0,+2)
  from Rowsets=(1)
end-select
end-execute

```

Tip:

When you set up DECLARE-CONNECTION, you must use the same name defined in *Registry.properties*. For example, if *Registry.properties* contains:

```

XML_DATA.desc=Sample XML files
XML_DATA.class=com.scribe.xmlacc.XMLDataSource
XML_DATA.lib=
XML_DATA.load=
XML_DATA.conn=D:\\SampleData\\XML_DATA

```

Then the Production Reporting code should look similar to:

```

begin-setup
declare-connection default
  DSN=XML_DATA          ! Use the same name as specified in the
end-declare            ! Registry.properties file. Case sensitive.
end-setup

begin-procedure domystuff
  begin-execute
    GetData='sample' ! The filename is sample.xml. Substitute
the                    ! filename of your xml file here. The path to
the                    ! file is in the Registry.properties file.
    begin-select
      CUSTOMERS.cust_num type=num (+1,1) edit 099999
      CUSTOMERS.name type=char (,30)
      from customers
    end-select
  end-execute
end-procedure

```

The previous Production Reporting code produces the following output:

```

<CUSTOMERS>
  <Customer cust_num='100013'>
    <CUSTOMERS.CUST_NUM>100013</CUSTOMERS.CUST_NUM>
    <CUSTOMERS.NAME>Gregory Stonehaven</CUSTOMERS.NAME>
    <CUSTOMERS.ADDR1>Middlebrook Road</CUSTOMERS.ADDR1>
    <CUSTOMERS.ADDR2>Grey Quarter</CUSTOMERS.ADDR2>
    <CUSTOMERS.CITY>Everrettsville</CUSTOMERS.CITY>
    <CUSTOMERS.STATE>OH</CUSTOMERS.STATE>
    <CUSTOMERS.ZIP>402331000</CUSTOMERS.ZIP>
    <CUSTOMERS.PHONE>2165553109</CUSTOMERS.PHONE>
    <CUSTOMERS.TOT>39</CUSTOMERS.TOT>
  </Customer>
</CUSTOMERS>

```

Tip:

BEGIN-EXECUTE is only required when additional information about the DDO data source or query is needed, such as 'Connection', 'Schema', 'Command', 'GetData', 'Procedure', or 'Parameters'. The following example does not require BEGIN-EXECUTE since it does not require information about the DDO datasource or query.

```
begin-setup
page-size 58 80
  declare-connection ORACLE_CONNECTION
    dsn=saw806
    user=jerryh
    password=canttellyou
  end-declare
end-setup

begin-procedure print_customers
  print 'FULL CUSTOMER LIST BY Customer Number' (+1) center

  begin-select
  cust_num  (+1,1,6) edit 099999
  name      (0,+2,30)
  addr1     (+1,12,30)
  addr2     (0,+4,30)
  city      (+1,12,16)
  state     (0,+2,2)
  zip       (0,+2,10)
  phone     (0,+2,0) edit (xxx)bxxx-xxxx

  !Edit phone number for' easy reading.

  next-listing  skiplines=2 need=3

  !Skip 2 lines between listings. Since each listing takes 3 lines,
we      !specify 'need=3' to prevent a customer's data from being broken  !
across two pages.

  from customers
  order by cust_num
  end-select

end-procedure
```

See Also

[EXECUTE](#)

BEGIN-FOOTING

Function

Begins the FOOTING section.

Syntax

```
BEGIN-FOOTING footing_lines_int_lit
[FOR-REPORTS=(report_name1[,report_namei]...)]
[FOR-TOCS=(toc_name1[,toc_namei]...)]
[NAME={footing_name}]
END-FOOTING
```

Arguments

footing_lines_int_lit

Number of lines to reserve at the bottom of each page.

FOR-REPORTS

Reports to which this footing applies. Required only for programs with multiple reports.

FOR-TOCS

Table of Contents to which this heading applies.

NAME

Name associated with this footing section. Used with [ALTER-REPORT](#). Cannot be NONE or DEFAULT.

Description

FOOTING sections define and control information printed at the bottom of each page.

Define *report_name* in DECLARE-REPORT. If you do not use DECLARE-REPORT, the footing is applied to all reports. You can also specify FOR-REPORTS=(ALL). (The parentheses are required).

There can only be one BEGIN-FOOTING section for each report. A BEGIN-FOOTING section with FOR-REPORTS=(ALL) can be followed by other BEGIN-FOOTING sections for specific reports, which override ALL.

Define *toc_name* in DECLARE-TOC. You can also specify FOR-TOCS=(ALL). (The parentheses are required.)

There can only be one BEGIN-FOOTING section for each Table of Contents. A BEGIN-FOOTING section with FOR-TOCS=(ALL) can be followed by other BEGIN-FOOTING sections for a specific Table of Contents, which override ALL.

BEGIN-FOOTING sections can be shared between reports and Table of Contents.

You can print outside the Footing area of the report from the Footing, but you cannot print into the Footing area from the body.

Examples

```
begin-footing 2 for-reports=(customer, summary)
  print 'Company Confidential' (1,1,0) center
  page-number (2,37,0)
end-footing
```

```

begin-footing 2                                ! For all reports
  print 'Division Report' (1,1,0) center
  page-number (2,37,0)
end-footing

begin-footing 2 for-tocs=(all)
  print 'Table of Contents' (2,1)
  let $page = roman(#page-count)      ! ROMAN numerals
  print $page (,64)
end-footing

```

See Also

- [ALTER-REPORT](#) for information on dynamic headings/footings
- [DECLARE-LAYOUT](#) for information on page layout
- [DECLARE-REPORT](#) for information on programs with multiple reports
- [DECLARE-TOC](#) for information on Table of Contents
- [END-FOOTING](#)

BEGIN-HEADING

Function

Begins a HEADING section.

Syntax

```

BEGIN-HEADING heading_lines_int_lit
[FOR-REPORTS=(report_name1[, report_namei]...)]
[FOR-TOCS=(toc_name1[, toc_namei]...)]
[NAME={heading_name}]
END-HEADING

```

Arguments

heading_lines_int_lit

Number of lines to reserve at the top of each page.

FOR-REPORTS

Reports to which this heading applies. Only required for programs with multiple reports.

FOR-TOCS

Table of Contents to which this heading applies.

NAME

Name associated with this heading section. Cannot use if FOR-REPORTS or FOR-TOCS is also defined. Used in conjunction with [ALTER-REPORT](#). Cannot be NONE or DEFAULT.

Description

The `HEADING` section defines and controls information printed at the top of each page.

Define `report_name` in `DECLARE-REPORT`. If you do not use `DECLARE-REPORT`, the heading is applied to all reports. You can also define `FOR-REPORTS=(ALL)`. (The parentheses are required.)

There can only be one `BEGIN-HEADING` section for each report. A `BEGIN-HEADING` section with `FOR-REPORTS=(ALL)` can be specified followed by other `BEGIN-HEADING` sections for specific reports, which override `ALL`.

Define `toc_name` in `DECLARE-TOC`. You can also specify `FOR-TOCS=(ALL)`. (The parentheses are required.)

There can only be one Table of Contents for each `BEGIN-HEADING` section. A `BEGIN-HEADING` section with `FOR-TOCS=(ALL)` can be specified followed by other `BEGIN-HEADING` sections for specific Table of Contents, which override `ALL`.

`BEGIN-HEADING` sections can be shared between reports and Table of Contents.

You can print outside the heading area of the report from the heading, but you cannot print into the heading area from the body.

Examples

```
begin-heading 2                                ! Use 2 lines for heading,  
print $current-date (1,1) edit MM/DD/YY      ! 2nd is blank.  
  print 'Sales for the Month of ' (1,30)  
  print $month ()  
end-heading
```

```
begin-heading 2 for-tocs=(all)  
  print 'Table of Contents' (1,1) bold center  
end-heading
```

See Also

- [ALTER-REPORT](#) for information about dynamic headings/footings
- [DECLARE-LAYOUT](#) for information on page layout
- [DECLARE-REPORT](#) for information on programs with multiple reports
- [DECLARE-TOC](#) for information on Table of Contents
- `END-HEADING`

BEGIN-PROCEDURE

Function

Begins a procedure.

Syntax

```
BEGIN-PROCEDURE procedure_name [LOCAL|(arg1 [, argi]...)]  
END-PROCEDURE
```

Arguments

procedure_name

Procedure name. Not case-sensitive.

LOCAL

Defines that this is a local procedure.

arg1 [, *argi*]...

Arguments passed to or returned from the procedure. Can be string variables (*\$arg*), numeric variables (*#arg*), or date variables (*\$arg*). To return a value passed back to the calling **DO**, place a colon (:) before the variable name. Arguments in **BEGIN-PROCEDURE** and **DO** must match in number, order, and type.

Description

The procedure name must be unique. The name is referenced in **DO**. Procedures contain other commands and paragraphs (for example, **SELECT**, **SQL**, **DOCUMENT**).

By default, procedures are global. That is, variables or columns defined within a procedure are known and can be referenced outside the procedure.

A procedure is local when the word **LOCAL** appears after the procedure name or when the procedure is declared with arguments. That is, variables declared within the procedure are available only within the procedure, even when the same variable name is used elsewhere in the program. Queries defined in a local procedure have local database column variable names assigned that do not conflict with similarly named columns defined in queries in other procedures.

Production Reporting procedures can be called recursively. However, unlike C or Pascal, Production Reporting only maintains one copy of the local variables and they are persistent.

Arguments passed by **DO** to a procedure must match in number:

- Database text or date columns, string variables, and literals can be passed to procedure string arguments. If passing a date string to a date argument, the date string must be in the format specified by **SQR_DB_DATE_FORMAT**, a database dependent format (see [Table 61, “Default Formats by Database,” on page 251](#)), or the database-independent format **SYYYYYMMDD [HH24 [MI [SS [NNNNNN]]]]**.
- Database numeric columns, numeric variables, and numeric literals can be passed to procedure numeric arguments.
- Numeric variables (**DECIMAL**, **INTEGER**, **FLOAT**) can be passed to procedure numeric arguments without regard to the argument type of the procedure. Production Reporting automatically converts the numeric values upon entering and leaving the procedure as required.

- Date variables or columns can be passed to procedure date or string arguments. When passing a date variable or column to a string argument, the date is converted to a string according to the following rules:
 - For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by `SQR_DB_DATE_FORMAT`. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,” on page 251](#).
 - For DATE columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252](#).
 - For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252](#).

To reference or declare global variables from local procedures, add a leading underscore to the variable name, after the initial \$, #, or &. (Example: `#_amount`)

Note:

All Production Reporting reserved variables, such as `#sql-status` and `$sql-error`, are global variables. Within local procedures, they must be referenced using the leading underscore: `#_sql-status` or `$_sql-error`.

Examples

The following example shows a `BEGIN-PROCEDURE MAIN`, that also executes the procedure `PRINT-LIST`, for each row returned from the `SELECT` statement. No parameters are passed to `PRINT-LIST`.

```
begin-procedure main
  begin-select
    name
    address
    phone
    do print_list
  from custlist order by name
  end-select
end-procedure      ! main
```

In the following example, five arguments are passed to the `CALC` procedure:

```
do Calc (&tax, 'OH', &county_name, 12, #amount)

begin-procedure Calc(#rate, $state, $county, #months, :#answer)
.
.
.
  let #answer = ...
end-procedure
```

In the preceding example the value for `:#answer` is returned to `#amount` in the `DO` command.

The following example references global variables:

```
begin-procedure print-it ($a, $b)
  print $_deptname (+2,5,20) ! $deptname is
  print $a          (+,1) ! declared outside
  print $b          (+,1) ! this procedure
end-procedure
```

See Also

[DO](#) and [END-PROCEDURE](#)

BEGIN-PROGRAM

Function

Begins the program section of an Production Reporting program.

Syntax

```
BEGIN-PROGRAM
END-PROGRAM
```

Description

After processing the commands in [SETUP](#), Production Reporting starts program execution at [BEGIN-PROGRAM](#). The [PROGRAM](#) section typically contains a list of [DO](#) commands, though other commands can be used. This is the only required section in an Production Reporting program.

Examples

```
begin-program
  do startup
  do main
  do finish
end-program
```

See Also

[BEGIN-SETUP](#) and [END-PROGRAM](#)

BEGIN-SELECT

Function

Begins a [SELECT](#) paragraph. A [SELECT](#) paragraph is the principal means of retrieving data from the database and printing it in a report. A [SELECT](#) paragraph must be inside a [PROCEDURE](#) or [BEGIN-PROGRAM](#) section.

Syntax

```
BEGIN-SELECT [DISTINCT] [-Cnn] [-Bnn] [-XP] [-NR] [-SORTnn]
[-LOCK{RR|CS|RO|RL|XX}] [-DBdatabase]
[-DBconnectionstring]
[LOOPS=nn] [ON-ERROR=procedure[ (arg1[, argi] ... )]]
{column} [&synonym]
{expression &synonym}
[{$columnname} &synonym = (char|number|date)]
[sqr_commands]
FROM {table,... | [table:$tablename]}
      [additional SQL]
      [$variable]

END-SELECT
```

Arguments

Note:

Arguments can span multiple lines; however, do not use the first character position unless the continuation character terminated the previous line. Otherwise, the argument will be misconstrued as a SELECT column.

DISTINCT

Eliminates duplicate query rows.

-Cnn

(Oracle) Sets the context area size (buffer size for query) to larger or smaller than the default.

-Bnn

(ODBC, Oracle, Sybase) Sets the number of rows to retrieve at once. For performance purposes only. Regardless of this setting, all rows are selected. The default, without using -B, is 10 rows. An overall setting for a program can be indicated on the Production Reporting command line with -B, which can be overridden by a separate -B flag on each BEGIN-SELECT command.

-XP

(Sybase) Prevents the creation of stored procedures for the SELECT paragraph. When specified, Production Reporting generates a new SQL statement using the current value of any bind variables each time BEGIN-SELECT is executed.

Use -XP if you change variables frequently during execution and do not want Production Reporting to automatically create stored procedures. You can also use -XP if users do not have permission to create stored procedures. If you do not change variables frequently during execution, stored procedures may optimize program performance. In this case, do not use -XP.

-XP improves performance when using bind variables and dynamic query variables in the same query. Each time the dynamic query variable changes in value, a new stored procedure is created.

If the dynamic query variable changes often and the query contains bind variables, you create many stored procedures if you do not use `-XP`.

`-XP` is available as a command-line flag.

`-DBconnectionstring`

(ODBC) The ODBC connection string for this `SELECT` paragraph only. A connection string has the following syntax:

```
DSN=data_source_name[;keyword=value[;keyword=value [...]]]
```

Combines data from multiple databases in one program. For example, a connection string for an Oracle database named “ora8” might look like the following:

```
'DSN=ora7;UID=scott;PWD=tiger'
```

where `DSN`, `UID`, and `PWD` are keywords common to all drivers (representing: name, user ID, and password, respectively). Connection string options are always separated by a semicolon (;). Other driver-specific options may be added to the connection string using driver-defined keywords. See your ODBC driver documentation for available options.

LOOPS

Number of rows to retrieve. After processing the specified number, the `SELECT` loop exits.

ON-ERROR

Procedure to execute if errors occur due to incorrect SQL syntax. Use error trapping with dynamic query variables. `SELECT` paragraphs without dynamic variables are checked for errors before programs are processed and do not require special error procedures.

You can optionally specify arguments to pass to the `ON-ERROR` procedure. Arguments can be any variable, column, or literal.

Note:

Production Reporting invokes `ON-ERROR` when it safely can. If Production Reporting can recover from a database error, users are given the chance to fix the error. If Production Reporting cannot recover from a database error, it will exit the program.

Description

`BEGIN-SELECT` can be placed inside a `BEGIN-PROGRAM` section. Note that `SELECT * FROM` is not a valid Production Reporting SQL statement.

Note:

In **Production Reporting DDO**, you can name data source-specific aggregation functions in place of column names in a `BEGIN-SELECT` block. This shifts the processing burden from Production Reporting to the data' source host and usually improves performance. The aggregation function feature also makes it possible to use literals (such as *empty column*) and simple mathematical operations (such as *5+10*) in place of column names.

In **Production Reporting DDO-SAP**, the `TYPE=datatype` qualifier used in a `BEGIN-SELECT` block is optional. When you report on data sources that provide adequate metadata (such as SAP), withholding the `TYPE` qualifier allows Production Reporting to generate code that is more efficient and portable than it would be otherwise.

You can use the `intersect`, `union`, and `minus` SQL operators in Production Reporting queries by adding them to the SQL statement that follows the `FROM` and `WHERE` clauses.

The `SELECT` list for the secondary SQL statement in the `union`, `intersect`, or `minus` query must match the data type, number of columns, and length of columns selected in the first query. If you select string expressions or literals, ensure that the lengths of the fields in both `SELECT` lists are the same.

Note that `intersect` and `minus` are not available with SYBASE's Transact SQL.

Enter the part of the SQL statement following the `union`, `minus`, or `intersect` clauses normally; that is, with commas between column names and without alias names, as shown below:

```
begin-select
  cust_num (1,1) edit 099999
  co_name  (,9,30)
  name    (,+2,25)
  city    (,+2,18)
  state   (,+2,2)
  zip     (,+1) edit xxxxx-xxxx
next-listing
from customers where state in ('OH', 'IN', 'MI')
union select cust_num, co_name, name, city, state, zip
from prospects where state in ('OH', 'IN', 'MI')
and first_contact >= '01-JAN-88'
order by 2
end-select
```

Examples

In this example, duplicate rows are not selected for the city, state, and zip columns because of the “distinct” keyword. The numbers within parentheses accompanying City, State, and Zip define the column positions of these rows. Column names can not have spaces in front of them. See “Column Variables” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*.

```
begin-select distinct
  city (1,1,30)
  state (0,+2,2)
  zip (1,+3,6)
  from custlist order by city
end-select
```

In this example, the first two columns may, or may not, be present when the statement is compiled. The column `cust_id` is declared to be a number. A runtime error is produced if the database table, as identified by the variable `$table_name`, declares it to be something other than a number.

```
begin-select          loops=100
  [$col_var_char]    &col1=char
```

```
    [$col_var_num]      &col2=number
    cust_id             &id=number
  from [$table_name]
  [$where clause]
  [$order_by_clause]
end-select
```

See Also

- “Selecting Data” and “Dynamic SQL and Error Checking” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*
- END-SELECT and [EXIT-SELECT](#)

BEGIN-SETUP

Function

Begins a SETUP section. This section is optional, but if included, it is processed prior to BEGIN-PROGRAM, BEGIN-HEADING, and BEGIN-FOOTING.

Syntax

```
BEGIN-SETUP
END-SETUP
```

Description

SETUP should be the first section in the program. It contains commands that determine overall program characteristics. The commands used in SETUP cannot be used elsewhere unless specified. SETUP can include the following commands:

ASK

BEGIN-SQL (can also be used in BEGIN-PROCEDURE.)

CREATE-ARRAY (can also be used in other Production Reporting programs sections)

DECLARE-CHART

DECLARE-IMAGE

DECLARE-LAYOUT

DECLARE-PRINTER

DECLARE-PROCEDURE

DECLARE-REPORT

DECLARE-TOC

DECLARE-VARIABLE (can also be used in LOCAL)

LOAD-LOOKUP (can also be used in the other Production Reporting program sections)

USE (Sybase and ODBC only)

Examples

```
begin-setup
  declare-layout customer_list
    paper-size=(8.5, 11)
    left-margin=1.0
    right-margin=1.0
  end-declare
end-setup
```

See Also

[ASK](#), [BEGIN-SQL](#), [CREATE-ARRAY](#), [LOAD-LOOKUP](#) , and [USE](#)

BEGIN-SQL

Function

Begins an SQL paragraph, which can reside in `BEGIN-PROCEDURE`, `BEGIN-SETUP`, or `BEGIN-PROGRAM`.

Syntax

```
BEGIN-SQL [-Cnn] [-XP] [-NR] [-SORTnn]
[-LOCK{RR|CS|RO|RL|XX}]
[-DBdatabase] [-DBconnectionstring]
[ON-ERROR=procedure [ (arg1 [, argi] ... ) ] (non-setup)
| [ON-ERROR={STOP|WARN|SKIP}] (SETUP)
END-SQL
```

Arguments

`-Cnn`

(Oracle) Sets the context area size (buffer size for query) to larger or smaller than the default.

`-XP`

(Sybase) Prevents the creation of stored procedures for SQL paragraphs. When specified, Production Reporting generates a new SQL statement using the current value of the bind variables each time `BEGIN-SQL` is executed. This disables the performance optimization created by stored procedures.

Use `-XP` if you change variables frequently during execution and do not want Production Reporting to automatically create stored procedures. You can also use `-XP` if users do not have permission to create stored procedures. If you do not change variables frequently during execution, stored procedures may optimize program performance. In this case, do not use `-XP`.

`-XP` improves performance when using bind variables and dynamic query variables in the same query. Each time the dynamic query variable changes in value, a new stored procedure is created. If the dynamic query variable changes often and the query contains bind variables, you create many stored procedures if you do not use `-XP`.

-DBconnectionstring

(ODBC) The ODBC connection string for this SQL paragraph only. A connection string has the following syntax:

```
DSN=data_source_name[;keyword=value[;keyword=value[;...]]]
```

Combines data from multiple databases in one program. For example, a connection string for an Oracle named “ora8” might appear as:

```
'DSN=ora8;UID=scott;PWD=tiger'
```

where *DSN*, *UID*, and *PWD* are keywords common to all drivers (representing name, user ID, and password, respectively). Connection string options are always separated by a semicolon (;). Other driver-specific options may be added to the connection string using driver-defined keywords. See your ODBC driver documentation for available options.

Connection=connstr

Used with Production Reporting DDO. The name of a data source previously declared using *DECLARE-CONNECTION*. If not specified, the default connection is used. (See [BEGIN-EXECUTE](#) for the behavior of the default connection.)

ON-ERROR

Procedure to execute if an error occurs due to incorrect SQL syntax except when executed in a *BEGIN-SETUP* section. By default, Production Reporting reports any error and then halts. If an error procedure is declared, you can trap errors, report or log them, and continue processing. The procedure is invoked when an error occurs in any SQL statement in the paragraph. After the error procedure ends, control returns to the next SQL statement, if any.

You can optionally specify arguments to pass to *ON-ERROR*. Arguments can be any variable, column, or literal.

If *ON-ERROR* is used in *SETUP*, it is a condition flag supporting the following conditions:

- *STOP*—Do not run the program.
- *WARN*—Run the program with a warning message.
- *SKIP*—Ignore any errors and run the program.

Note:

Production Reporting invokes the *ON-ERROR* procedure when it safely can. If Production Reporting can recover from a database error, users are given the chance to fix the error. If Production Reporting cannot recover from a database error, it will exit from the program.

Description

BEGIN-SQL starts all SQL statements except *SELECT*, which has its own *BEGIN-SELECT* paragraph. If a single paragraph contains more than one SQL statement, terminate each statement (except the last) by a semicolon (;).

If a single paragraph contains more than one SQL statement, and the `-C` flag is used, all are assigned the same context area size or logical connection number.

Only non-`SELECT` statements can be used (except `SELECT INTO` for Sybase and Microsoft SQL Server backends). Reference columns and variables in SQL statements.

Examples

```
begin-sql
  update orders set invoice_num = #next_invoice_num
  where order_num = &order_num
end-sql
```

```
begin sql
  delete orders
  where order_num = &order_num;
  insert into orders values ($customer_name, #order_num,...)
end-sql
```

Stored Procedures

For Sybase, and Microsoft SQL Server, Production Reporting supports stored procedures with [EXECUTE](#). For Oracle, stored procedures are implemented using PL/SQL in the `BEGIN-SQL` paragraph.

For some databases such as ORACLE, using DDL statements in `BEGIN-SQL` causes a commit of outstanding inserts, updates, and deletes and releases cursors. For this reason, ensure that these are done in the proper order or unpredictable results may occur.

Oracle PL/SQL

For Oracle, PL/SQL is supported in a `BEGIN-SQL` paragraph. This requires an additional semicolon at the end of each PL/SQL statement.

For Oracle PL/SQL:

```
begin-sql
  declare
    var1 varchar2 (25);;
    var2 number (8,2);;
  begin
    var1 :='abcdefg';;
    $v1 :=var1;;
    $v2 :='1230894asd';;
    var2 :=1234.56;;
    #v :=var2;;
  end;;
end-sql
```

For Oracle stored procedures:

```
begin-sql
  begin
    #dept_number :=get_dept_no($dept_name);;
  end;;
end-sql
```

See Also

- “Dynamic SQL and Error Checking” and “Using DML and DDL” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*
- END-SQL, [BEGIN-PROCEDURE](#), and [EXECUTE](#)
- The `-S` command-line flag.

BREAK

Function

Exits from EVALUATE or WHILE and continues to the command immediately following END-WHILE or END-EVALUATE.

Syntax

```
BREAK
```

Description

BREAK is used inside a WHILE . . . END-WHILE loop or within an EVALUATE command.

See Also

[WHILE](#) and [EVALUATE](#)

CALL, CALL SYSTEM

Function

Issues an operating system command or calls a subroutine written in another language such as C or COBOL and passes the specified parameters.

Note:

CALL is available in all Production Reporting environments except Oracle's Hyperion® SQR® Production Reporting Studio. With Oracle's Hyperion® SQR® Production Reporting Studio, use CALL SYSTEM instead.

Syntax

```
CALL subroutine USING {src_txt_lit|_var|_col} | {src_num_lit|_var|_col}  
{dst_txt_var|_num_var} [param]
```

To issue operating system commands in an Production Reporting program, use the following syntax:

```
CALL SYSTEM USING command status [WAIT|NOWAIT]
```


Arguments

subroutine

Name of the subroutine.

src_txt_lit|_var|_col

Text column, variable, or literal to input into the called subroutine.

src_num_lit|_var|_col

Numeric column, variable (decimal, float, or integer), or literal to input into the called subroutine.

dst_txt_var|_num_var

Text or numeric variable (decimal, float, or integer) into which the called subroutine places the return result. (See [Table 16 on page 82](#).)

param

(Optional) Alphanumeric string of characters passed as a parameter to the subroutine.

SYSTEM

Defines that this CALL command issues an operating system command.

command

Operating system command to execute. The *command* can be a quoted string, string variable, or column.

status

Status returned by the operating system. The *status* must be a numeric variable. The value returned in *status* is system-dependent as shown in [Table 15](#).

Table 15 Operating System Status Values for the CALL Command

System	Value Returned
UNIX	Zero (0) indicates success. Any other value is the system error code.
PC/Windows	A value less than 32 indicates an error.

WAIT|NOWAIT

(Windows) - WAIT suspends execution until CALL SYSTEM finishes processing. NOWAIT starts CALL SYSTEM while continuing its own processing.

For Windows, the default is NOWAIT. On UNIX operating systems the behavior is always WAIT.

Description

You can write your own subroutines to perform tasks that are awkward in Production Reporting. Subroutines can be written in any language.

Caution!

If `ucall` uses database calls, it may cause erroneous results.

Used in an Production Reporting program, `CALL` has the following format:

```
CALL your_sub USING source destination [param_literal]  
CALL SYSTEM USING command status [WAIT|NOWAIT]
```

`CALL SYSTEM` is a subroutine provided with Production Reporting to allow the program to issue operating system commands. Its arguments, `command`, `status`, and `WAIT|NOWAIT` are described above.

The values of the source and destination variables and the parameter's literal value are passed to the subroutine. Upon return from the subroutine, a value is placed in the destination variable. If the arguments passed to the function are longer than the limit, they are truncated. (No warning message displays, but the call proceeds.)

Write the subroutine and call it in one of the supplied `ucall` routines. (Optionally, you could rewrite `ucall` in another language).

The source file `UCALL.C` contains sample subroutines written in C. :

Table 16 `UCALL` Subroutine Arguments

Argument	Description	How Passed
callname	Subroutine name.	By reference with a maximum of 31 characters, null terminated.
strsrc	Source string.	By reference with a maximum of 511 characters, null terminated.
strdes	Destination string.	By reference with a maximum of 511 characters, null terminated.
dblsrc	Source double floating point.	By reference.
dbldes	Destination double floating point.	By reference.
param	Subroutine parameter string. It must be a literal.	By reference with a maximum of 2047 characters, null terminated.

`CALL` arguments are handled as follows:

- Arguments are copied into variables depending on argument type. Strings are placed in `strsrc`, and numerics are placed in `dblsrc`.

- Return values are placed in *strdes* or *dbldes* depending on whether the destination argument for `CALL` is a string or numeric variable.
- Destination arguments can be used to pass values to a subroutine.
- To access a subroutine, add a reference to it in `UCALL`, passing along the needed arguments.
- Relink Production Reporting to `CALL` after compiling a user-defined function that becomes another Production Reporting function.
- Add subroutines to the link command file (UNIX: `SQRMAKE`, Windows: `SQREXT.MAK`) for new object files. (Alternatively, you could add the routine to the bottom of the `UCALL` source module included in the link).
- Subroutines and calling Production Reporting programs are responsible for passing correct string or numeric variables and optional parameter strings to the subroutine. No checking is performed.

Examples

Sample subroutines included in the `UCALL` source file:

- `TODASH` shows how strings can be manipulated.
- `SQROOT` demonstrates how to access numerics.
- `SYSTEM` invokes a secondary command processor.

The following code calls these subroutines:

```
call todash using $addr $newaddr './.',! Convert these to dashes
call sqroot using #n #n2           ! Put square root of #n into #n2
call sqroot using &hnvr #j         ! Hnvr is numeric database column
call system using 'dir' #s         ! Get directory listing
```

The following example uses the `SYSTEM` argument to issue an operating system command. Some operating systems let you invoke a secondary command processor to enter one or more commands and then return to Production Reporting.

```
! Unix (Type 'exit' to return to Production Reporting)
!
let $shell = getenv('SHELL')
if isblank($shell)
  let $shell = '/bin/sh'
end-if
call system using $shell #unix_status
!Windows 98/NT (Type 'exit' to return to Production Reporting)
!
let $comspec = getenv('COMSPEC')
let $cmd = comspec || '/c' || $comspec || ' /k'
call system using $cmd #win_status wait
```

The following example adds a user-defined subroutine to Production Reporting so that it can be invoked using `CALL`. For this example, the C function `initcap`, which uppercases the first letter of a string, is added. The function accepts two parameters. The first parameter is the string to which `initcap` is applied. The second is the resultant string.

1. Add the `initcap` prototype

```
static void initcap CC_ARGS((char *, char *));
```

2. Modify `ucall` in `UCALL.C`. Specifically, add an `else if` statement at the end of the `if` statement to check for `initcap`:

```
void ucall CC_ARGL((callname, strsrc, strdes, dblsrc, dbldes, params))
...
    /* If other subroutines, add "else if..." statement for each */
    else if (strcmp(callname,"initcap") == 0)
        initcap(strsrc, strdes);
    else
        sq999("Unknown CALled subroutine: %s\n", callname);
    return;
}
```

3. At the end of `UCALL.C`, add the `initcap` routine listed in the following example. The routine name must be lower case; however, in your Production Reporting program it can be referenced with either upper or lower case.

```
static void initcap CC_ARGL((strsrc, strdes))
CC_ARG(char *, strsrc) /* Pointer to source string */
CC_LARG(char *, strdes) /* Pointer to destination string */
{
    int nIndex;
    int nToUpCase;
    char cChar;

    nToUpCase = 1;
    for (nIndex = 0; cChar = strsrc[nIndex]; nIndex++)
    {
        if (isalnum(cChar))
        {
            if (nToUpCase)
                strdes[nIndex] = islower(cChar) ? toupper(cChar) : cChar;
            else
                strdes[nIndex] = isupper(cChar) ? tolower(cChar) : cChar;
            nToUpCase = 0;
        }
        else
        {
            nToUpCase = 1;
            strdes[nIndex] = cChar;
        }
    }
    strdes[nIndex] = '\0';
}
```

Note:

`CC_ARG` macros are defined in the `UCALL.C` source module. The macros allow you to define a fully prototyped function without having to worry if the C compiler supports the feature.

After these modifications, recompile `UCALL.C` and relink Production Reporting. See “Interoperability” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide* for details.

The following is an example of a simple Production Reporting program using `initcap`:

```
begin-program
  input $name 'Enter the first name '
    ! Get the first name from the user

  lowercase $name
    ! Set the first name to all lower case

  call initcap using $name $capname
    ! Now set the first character to upper case

  input $last 'Enter the last name '
    ! Get the last name from the user

  lowercase $last
    ! Set the last name to all lower case

  call initcap using $last $caplast
    ! Now set the first character to upper case
.
.
```

See Also

[LET](#) for information on user-defined functions using `UFUNC.C` that can be used in the context of an expression and that can pass and/or return any number of arguments.

CLEAR-ARRAY

Function

Resets array fields to their initial values.

Syntax

```
CLEAR-ARRAY NAME=array_name
```

Arguments

NAME

Name of array to clear.

Description

`CLEAR-ARRAY` resets each field of the named array to its initial value specified in `CREATE-ARRAY`. If no initial value was specified, numeric fields are reset to zero, text fields are reset to null, and date fields are reset to null. `CLEAR-ARRAY` releases all memory used by the specified array.

Examples

```
clear-array name=custs
```

See Also

[CREATE-ARRAY](#)

CLOSE

Function

Closes a file, specified by its file number.

Syntax

```
CLOSE {filenum_lit|_var|_col}
```

Arguments

filenum_lit|*_var*|*_col*

Number assigned to the file in OPEN.

Description

Closes flat files previously opened with OPEN.

Examples

```
close 5  
close #j
```

See Also

[OPEN](#), [READ](#), and [WRITE](#)

CLOSE-RS

Function

Closes a row set.

Syntax

```
CLOSE-RS  
NAME=row_set_name_var|_lit|_col
```

Arguments

NAME

Name of the row set.

Description

CLOSE-RS can reside in any section except [BEGIN-SETUP](#), [BEGIN-SQL](#), and [BEGIN-DOCUMENT](#). The row set specified by *row_set_name* must be active, or an exception is thrown.

The row set file is an XML file. You can define whether to create the XML file in a BI Publisher (BIP) format or an SQR format in the `FormatForRowsetXML` entry in the [Default-Settings] section of SQR.INI.

Example

```
Begin-Report
  Open-RS Name='customer' FileName='customer.xml'
    Column = ('cust_num', 'integer')
    Column = ('name', 'string')
    Column = ('addr1', 'string')
    Column = ('addr2', 'string')
    Column = ('city', 'string')
    Column = ('state', 'string')
    Column = ('zip', 'string')
    Column = ('phone', 'string')
    Column = ('tot', 'integer')
  Begin-Select
  cust_num
  name
  addr1
  addr2
  city
  state
  zip
  phone
  tot
  Write-RS Name='customer'
    Value = ('cust_num', &cust_num)
    Value = ('name', &name)
    Value = ('addr1', &addr1)
    Value = ('addr2', &addr2)
    Value = ('city', &city)
    Value = ('state', &state)
    Value = ('zip', &zip)
    Value = ('phone', &phone)
    Value = ('tot', &tot)
  from customers
  order by cust_num
  End-Select
  Close-RS Name='customer'
End-Report
```

See Also

[OPEN-RS](#), [WRITE-RS](#)

COLUMNS

Function

Defines logical columns to use for PRINT commands.

Syntax

```
COLUMNS {int_lit|_var|_col}[int_lit|_var|_col]. . .
```

Arguments

int_lit|*_var*|*_col*

Left margin position of each column.

Description

COLUMNS defines the left-most position of one or more columns in the page layout. It sets the first column as current.

COLUMNS can be used for printing data either down the page or across the page, depending on how you use NEXT-COLUMN and USE-COLUMN.

COLUMNS only applies to the current report. To print columns in multiple reports, specify COLUMNS for each report.

USE-COLUMN 0 turns off columns.

See Also

[NEXT-COLUMN](#), [NEXT-LISTING](#), [NEW-PAGE](#), [USE-COLUMN](#), and [USE-REPORT](#)

COMMIT

Function

Causes a database commit.

Syntax

```
COMMIT
```

Description

COMMIT is useful for multiple inserts, updates, or deletes in SQL paragraphs. A database commit releases the locks on inserted, updated, or deleted records. If used in an active SELECT paragraph, unpredictable results may occur.

When the application completes, COMMIT is performed automatically unless ROLLBACK was done or, for callable Production Reporting, the -XC flag was set.

Other commands or options, such as `CONNECT` and the use of DDL statements for some databases with a `BEGIN-SQL` paragraph, can also cause the database to do a commit.

`COMMIT` is an Production Reporting command and *should not* be used within an SQL paragraph. If used in an SQL paragraph, unpredictable errors can occur.

Note:

`COMMIT` can be used with DB2, ODBC, DDO, Teradata, and Oracle. For Sybase, use `BEGIN TRANSACTION` and `COMMIT TRANSACTION` within SQL paragraphs as in the following code segment.

Examples

```
add 1 to #updates_done
if #updates_done > 50
    commit
    move 0 to #updates_done
end-if
```

For Sybase:

```
...    ! Begin Transaction occurred previously
begin-sql
    insert into custlog values (&cust_num, &update_date)
end-sql
add 1 to #inserts
if #inserts >= 50
    begin-sql
        commit transaction;! Commit every 50 rows
        begin transaction ! Begin next transaction
    end-sql
    move 0 to #inserts
end-if

...    ! One more Commit Transaction is needed
```

Caution!

Any data changed by a current transaction is locked by the database and cannot be retrieved in a `SELECT` paragraph until the transaction is completed by a `COMMIT` or `ROLLBACK` statement (or `COMMIT TRANSACTION` or `ROLLBACK TRANSACTION` statement for Sybase and Microsoft SQL Server backends).

CONCAT

Function

Concatenates variables, columns, or literals with string variables.

Syntax

```
CONCAT {src_any_lit|_var|_col} WITH dst_txt_var[:$]edit_mask
```

Arguments

src_any_lit|_var|_col

Source field to concatenate with *dst_txt_var*.

dst_txt_var

Result after execution.

edit_mask

Optional edit mask.

Description

The contents of the source field are appended to the end of the destination field.

CONCAT can optionally edit the source field before appending it. To dynamically change an edit mask, place it in a string variable and reference the variable name preceded by a colon (:). (See [“Edit Masks” on page 247.](#))

The source can be a date variable or column. If an edit mask is not specified, the date is converted to a string according to the following rules:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by SQR_DB_DATE_FORMAT. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,” on page 251.](#)
- For DATE columns, Production Reporting uses the format specified by SQR_DB_DATE_ONLY_FORMAT. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252.](#)
- For TIME columns, Production Reporting uses the format specified by SQR_DB_TIME_ONLY_FORMAT. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252.](#)

Examples

```
concat &zip_plus_4 with $zip '-xxxx' ! Edit zip plus 4.  
concat &descrip with $rec :$desc_edit ! Edit mask in variable.  
concat $date1 with $string ! Concatenate a date.
```

See Also

- [PRINT](#) or information on edit masks
- [LET](#) for string functions.
- [STRING](#) and [UNSTRING](#)

CONNECT

Function

Logs off the database and logs on under a new user name and password.

Syntax

```
CONNECT {txt_lit|_var|_col} [ON-ERROR=procedure [(arg1
[, argi]...)]]
```

Arguments

txt_lit|*_var*|*_col*

Username and password for logon.

ON-ERROR

Procedure to execute logon fails. If no ON-ERROR procedure is specified and the logon fails, Production Reporting halts with an error message.

You can optionally specify arguments to pass to ON-ERROR. Arguments can be any variable, column, or literal.

Note:

CONNECT is the same as the {connectivity} portion of the Production Reporting command line as follows:

DB2: DB[/username/password]

DDO: DSN[/username/password]

INFORMIX: DB[/username/password][@InformixServer]

ODBC: DSN[/username/password]

ORACLE: [username/password][@OracleServer]

TERADATA: [TDPID/]username[,password]

Description

New connectivity information can be stored in a string variable, column, or literal.

After each CONNECT, the reserved variable *\$username* is set to the new username.

All database cursors or logons are closed before the CONNECT occurs. Do not issue a CONNECT within a SELECT or an SQL paragraph while a query is actively fetching or manipulating data from the database.

Examples

```
connect $new-user on-error=bad-logon($new_user)
```

```
connect 'sqr/test'
```

Caution!

Connectivity information is not encrypted, so beware of security issues.

CREATE-ARRAY

Function

Creates an array of fields to store and process data.

Syntax

```
CREATE-ARRAY NAME=array_name SIZE=nn  
[EXTENT=nn]  
{FIELD=name:type[:occurs]  
[={init_value_txt_lit|_num_lit|_binary_lit}]}...
```

Arguments

NAME

Name of the array. Referenced in other array commands.

SIZE

Number of array elements.

EXTENT

Number of array elements used to incrementally extend the array size beyond the initial allocation defined in SIZE. The value entered for EXTENT must be a *numeric literal*.

FIELD

Defines each field or column in the array.

- DECIMAL[(p)]—Decimal numbers with an optional precision (p).
- FLOAT—Double precision floating point numbers.
- INTEGER—Whole numbers.
- NUMBER—Uses the DEFAULT-NUMERIC type. (See [DECLARE-VARIABLE](#).)
- CHAR (or TEXT)—Character string.
- DATE—Same as date variable.

You can specify an initialization value for each field. Each field is set to this value when the array is created and when CLEAR-ARRAY is executed. If no initialization value is specified, numeric fields (DECIMAL, FLOAT, INTEGER) are set to zero, character fields are set to null, and date fields are set to null. All occurrences of a multiple occurring field are set to the same value. For dates,

the initialization string must be formatted as 'SYYYYMMDD [HH24 [MI [SS [NNNNNNN]]]]'. See [Table 57 on page 245](#) for a description of the format codes.

OCCURS

Fields can optionally have a number of occurrences (*occurs*), that is, they can be repeated any number of times.

Description

You can define arrays to store intermediate results or data retrieved from the database. For example, a `SELECT` paragraph can retrieve data, store it in an array, and gather statistics all at the same time. When the query finishes, a summary could be printed followed by the data previously stored in the array.

Production Reporting creates arrays before a program starts to execute. `CREATE-ARRAY` can be used in any section of a program.

Commands to process arrays include:

`CREATE-ARRAY`

`CLEAR-ARRAY`

`GET`

`PUT`

`ARRAY-ADD`

`ARRAY-SUBTRACT`

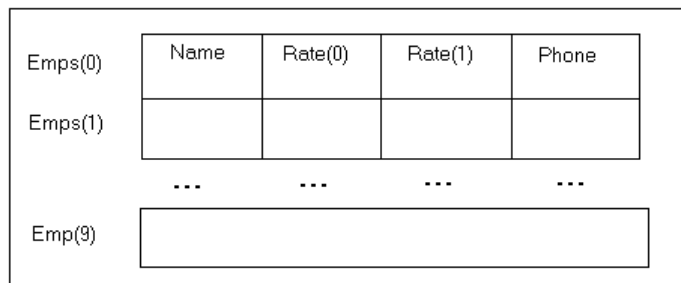
`ARRAY-MULTIPLY`

`ARRAY-DIVIDE`

`LET`

The maximum number of arrays in a program is 128; the maximum number of fields per array is 200.

Figure 1 Sample Array with Three Fields



The following `CREATE-ARRAY` command defines the array:

```
create-array name=emps size=10
  field=name:char='Unknown'
```

```
field=rate:number:2=10.50
field=phone:char='None'
```

The *name* is a simple field (one occurrence), *rate* has two occurrences, and *phone* is a simple field. Both array elements and field occurrences are referenced beginning with zero (0). The *rate* is referenced by `rate(0)` or `rate(1)`. The *emps* array contains 10 elements, 0 through 9. All *name* fields are initialized to “Unknown”, all *phone* fields are initialized to “None”, and all *rate* fields are initialized to 10.50.

Examples

The following example defines an array `names custs` with 100 elements that can be incrementally extended by 25 elements:

```
create-array name=custs size=100 extent=25
  field=name:char
  field=no:number
  field=state:char
  field=zip:char
  field=contacts:char:5
  field=last-contacted:date
```

The following example defines point labels as part of a data array.

```
create-array name=multi_series_radar_data_with_labels size=7
  field=label:char ! point label
  field=theta:number:1 ! angle
  field=radius:number:2 ! two series of point
```

See Also

- The sample report `CUSTOMR4.SQR` included with Production Reporting
- [LOAD-LOOKUP](#) for an alternative way to store database table(s) in memory
- [DECLARE-VARIABLE](#), [ARRAY-ADD](#), [ARRAY-DIVIDE](#), [ARRAY-MULTIPLY](#), [ARRAY-SUBTRACT](#), [GET](#), [PUT](#), [LET](#), and [CLEAR-ARRAY](#)

CREATE-COLOR-PALETTE

Function

Create a color palette.

Syntax

```
CREATE-COLOR-PALETTE
NAME={palette_name_txt_lit}
COLOR_1={rgb}
COLOR_2={rgb}
[COLOR_n]={rgb}
```

Arguments

NAME

Name of the color palette.

COLOR_1

First color in the palette.

COLOR_2

Second color in the palette.

COLOR_n

The *n*'th color in the palette. You can specify up to 64 colors in the palette.

{*rgb*}

A color reference. This can be expressed as (r,g,b) where r, g, and b are either a numeric literal (0 to 255), a numeric variable, or a numeric column. It can also be expressed as a (c) where c is a string literal, column, or variable that is the name of a color.

Description

This command creates a palette of colors. There is no limit to the number of palettes that can be defined in a program. No gaps are permitted in the palette.

Examples

```
begin-report
  create-color-palette
    name = 'funky'
    color_1 = ('blue')
    color_2 = ('red')
    color_3 = ('orange')
  print-chart Groovy
    color-palette = 'Funky'
end-report
```

See Also

- [DECLARE-CHART](#)
- [PRINT-CHART](#)

CREATE-LIST

Function

Creates a named list.

Syntax

```
CREATE-LIST  
NAME=list_name_txt_lit|_var|_col  
LIST=(value_lit|var|_col|(r,g,b)...)
```

Arguments

NAME

Name of the list.

LIST

Values included in the list.

Description

This command creates named lists of items. This command may be used anywhere within an Production Reporting program and will override a previously declared named list. The list and internal copies of the variables placed into the structure used to maintain the list are established at run-time. Updates to the elements used to define the list do not change the contents of the list after the list has been established. To add an item to a list, reenter the complete list.

Examples

```
begin-report  
create-list name='linestyle'  
list=('solid', 'longdash', 'dot')  
end-report
```

CREATE-TABLE

Function

Creates a table from a template.

Syntax

```
NAME=table_name_var|_lit|_col  
USING=table_template_var|_lit|_col  
[COLUMN-COUNT=number_of_columns_var|_lit|_col  
[COLUMN-ATTRIBUTES=({column-number},{keyword1},{value1}, ..., {keywordn},  
{valuen})]  
[ROW-ATTRIBUTES=({keyword1},{value1}, ..., {keywordn},{valuen})]  
[TABLE-ATTRIBUTES=({keyword1},{value1}, ..., {keywordn},{valuen})]
```

Arguments

NAME

Table name used by ALTER-TABLE, DUMP-TABLE, FILL-TABLE, and PRINT-TABLE. Valid values include alphanumeric characters(A-Z, 0-9), underscore (_), and dash (-).

USING

Name of the table template. The template must be defined with `DECLARE-TABLE`. `NONE` indicates that the table is defined solely by `CREATE-TABLE` parameters.

COLUMN-COUNT

Number of table columns.

COLUMN-ATTRIBUTES

Attributes to apply to column cells. The values defined are only applicable when `USING=NONE`.

Table 17 Column Attributes

Attribute	Description
BACKGROUND	Background color name or RGB triplet. Default= <code>NONE</code> .
BOLD	YES NO
CENTER	YES NO
DEFAULT	When the <code>USING</code> argument is anything other than <code>NONE</code> , this attribute causes all the attributes to be set to default values prior to applying any other attributes.
FILL-COLOR	Fill color name or RGB triplet. Default= <code>NONE</code>
FONT	Font number. (Must be defined.)
FOREGROUND	Foreground color name or RGB triplet. Default= <code>BLACK</code> .
ITALIC	YES NO
LEADING	Expressed in decipoints
LINE-COLOR	Color name or RGB triplet of column line (line after column). Default= <code>NONE</code> (no line).
LINE-STYLE	Column line style (<code>SOLID</code> , <code>SQUARE-DOT</code> , <code>DASH</code> , <code>DASH-DOT</code> , <code>LONG-DASH</code> , <code>LONG-DASH-DOT</code> , <code>LONG-DASH-DOT-DOT</code>). Default= <code>SOLID</code> .
LINE-THICKNESS	Column line thickness expressed in decipoints. Default=two decipoints.
POINT-SIZE	Point size of the font. (Must be specified.)
UNDERLINE	YES NO
WIDTH	Width expressed in coordinate units. (Must be specified.)
WRAP	YES NO maximum number of lines
WRAP-HEIGHT	Number of lines between each wrapped line. Default=one line.
WRAP-ON	Characters on which to force a <code>WRAP</code> . The default is not to force a <code>WRAP</code> .
WRAP-STRIP	Characters to change to a space before the <code>WRAP</code> is done. The default is not to strip any characters.

ROW-ATTRIBUTES

Attributes to apply to rows.

Table 18 Row Attributes

Attribute	Description
BORDER-COLOR	Color name or RGB triplet. Default=NONE (no border)
BORDER-LINE-STYLE	The border line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT, LONG-DASH-DOT-DOT). Default=SOLID
BORDER-THICKNESS	Border thickness expressed in decipoints. Default=two decipoints
DEFAULT	When the USING argument is anything other than NONE, this attribute causes all the attributes to be set to default values prior to applying any other attributes.
FILL-COLOR	Fill color name or RGB triplet. Default=NONE
HEIGHT	Number of lines between each printed row. Default=one line
LINE-COLOR	Color name or RGB triplet. Default=NONE (no line)
LINE-STYLE	The line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT, LONG-DASH-DOT-DOT). Default=SOLID
LINE-THICKNESS	Line thickness expressed in decipoints. Default=two decipoints

TABLE-ATTRIBUTES

Attributes for the appearance of the table.

Table 19 Table Attributes

Attribute	Description
BORDER-COLOR	Color name or RGB triplet. Default=NONE (no border)
BORDER-LINE-STYLE	The border line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT, LONG-DASH-DOT-DOT). Default=SOLID
BORDER-THICKNESS	Border thickness expressed in decipoints. Default=two decipoints
DEFAULT	When the USING argument is anything other than NONE, this attribute causes all the attributes to be set to default values prior to applying any other attributes.
FILL-COLOR	Fill color name or RGB triplet. Default=NONE
LEADING	Expressed in decipoints. Default=0 decipoints

Description

Use CREATE-TABLE in any section except BEGIN-SETUP, BEGIN-SQL, and BEGIN-DOCUMENT to create a table from a template.

Example

```
create-table
  name='tab2'
  using='template4'
  column-count=4
  table-attributes=('border-thickness',&int4 'leading',&int2,
    'border-color',(&red), 'border-line-style', 'square-dot')
  row-attributes=('fill-color', (230,240,255))
  column-attributes=(0, 'italic', &yes, 'bold', &yes,
    'center', &no, 'font', &int3, 'point-size', &int10,
    'foreground', 'white', 'background', (&blue), 'width', &int25)
  column-attributes=(1, default, 'background', ('red'),
    'foreground', ('blue'))
```

See Also

[ALTER-TABLE](#), [DECLARE-TABLE](#), [DUMP-TABLE](#), [FILL-TABLE](#), [PRINT-TABLE](#)

#DEBUG

Function

Causes the current command to process during a debugging session.

Syntax

```
#DEBUG[x...]sqr_command
```

Arguments

x

Any letter or digit.

Description

A `-DEBUG[xx]` flag in the Production Reporting command line allows conditional compilation of Production Reporting commands. When this flag is used, any command (including other compiler directives) preceded by the word `#DEBUG` is processed; otherwise, the command is ignored.

This is useful for placing `DISPLAY`, `SHOW`, `PRINT` or other commands in your program for testing and for deactivating them when the report goes into production.

The `-DEBUG` flag can be suffixed by up to 36 letters or digits. These characters are used to match any letters or digits appended to the `#DEBUG` preprocess command inside the program.

`#DEBUG` commands with one or more matching suffix characters are processed; other commands are ignored. Commands without any suffix always match.

In addition, for each `-DEBUGxx` letter, a substitution variable is defined. For example, if the flag `-DEBUGab` is used on the command line, three substitution variables are defined: `debug`, `debuga`, and `debugb`. These variables can be referenced in `#IFDEF` commands to turn whole sections of code on or off for debugging.

Examples

The following Production Reporting command line contains the `-DEBUG` flag with no suffixes:

```
sqr myprog sammy/baker -debug
```

The following `SHOW` command in the program executes if invoked with the previous command line because the `-DEBUG` flag was used:

```
#debug show 'The total is ' #grand-tot 999,999,999
```

In the following example, the command line contains the `-DEBUG` flag with the suffixes `a`, `b`, and `c`:

```
sqr myprog sammy/baker -debugabc
```

In the following program segment, the first three `#DEBUG` commands are compiled, but the fourth, beginning “`#debugc`”, is not since its suffix does not match any of the suffixes on the `-DEBUG` flag:

```
#debuga show 'Now selecting rows...'  
#debug show 'Finished query.'  
#debugb show 'Inserting new row.'  
#debugc show 'Deleting row.'
```

The following example shows the use of an `#IF` with a `#DEBUG`:

```
#debuga #if {platform}='unix'  
#debuga show 'Platform is UNIX'  
#debuga #endif
```

See Also

[#IF](#), [#IFDEF](#), and [#IFNDEF](#)

DECLARE-CHART

Function

Defines the attributes of a chart that you can later display using `PRINT-CHART`.

Syntax

```
DECLARE-CHART chart_name  
[TYPE=chart_type_lit]  
[CHART-SIZE=(chart_width_int_lit,chart_depth_int_lit)]  
[TITLE=title_txt_lit]  
[SUB-TITLE=subtitle_txt_lit]  
[FILL=fill_lit]  
[3D-EFFECTS=3d_effects_lit]  
[BORDER=border_lit]  
[COLOR-PALETTE=color_palette_lit]  
[POINT-MARKERS=point_markers_lit]  
[ATTRIBUTES={selector_lit|  
LIST:{selector_list_name_lit|(selector_lit,...)},
```

```

        {decl_key_lit, {decl_value_lit|
        LIST: {decl_val_list_name_lit| (decl_val_lit, ...)}|
        PALETTE: {color_palette_lit}}}, ...}}]
[DATA-ARRAY=array_name]
[DATA-ARRAY-ROW-COUNT=row_count_num_lit]
[DATA-ARRAY-COLUMN-COUNT=column_count_num_lit]
[DATA-ARRAY-COLUMN-LABELS={NONE|array_name|(txt_lit,...)}]
[DATA-LABELS=data_labels_lit]
[FOOTER-TEXT=NONE|text_lit]
[SUB-FOOTER-TEXT=NONE|text_lit]
[ITEM-COLOR=(item_color_keyword_lit, {color_value_lit|(r,g,b)})]
[ITEM-SIZE=(item_size_keyword_lit, item_size_num_lit)]
[LEGEND=legend_lit]
[LEGEND-TITLE=legend_title_txt_lit]
[LEGEND-PLACEMENT=legend_placement_lit]
[LEGEND-PRESENTATION=legend_presentation_lit]
[PIE-SEGMENT-QUANTITY-DISPLAY=pie_segment_quantity_display_lit]
[PIE-SEGMENT-PERCENT-DISPLAY=pie_segment_percent_display_lit]
[PIE-SEGMENT-EXPLODE=pie_segment_explode_lit]
[X-AXIS-GRID=x_axis_grid_lit]
[X-AXIS-LABEL=x_axis_label_txt_lit]
[X-AXIS-MIN-VALUE={x_axis_min_value_lit|_num_lit}]
[X-AXIS-MAX-VALUE={x_axis_max_value_lit|_num_lit}]
[X-AXIS-MAJOR-INCREMENT={x_axis_major_increment_lit|_num_lit}]
[X-AXIS-MINOR-INCREMENT=x_axis_minor_increment_num_lit]
[X-AXIS-MAJOR-TICK-MARKS=x_axis_major_tick_marks_lit]
[X-AXIS-MINOR-TICK-MARKS=x_axis_minor_tick_marks_lit]
[X-AXIS-TICK-MARK-PLACEMENT=x_axis_tick_mark_placement_lit]
[X-AXIS-ROTATE=x_axis_rotate_num_lit]
[X-AXIS-SCALE=x_axis_scale_lit]
[Y-AXIS-GRID=y_axis_grid_lit]
[Y-AXIS-LABEL=y_axis_label_lit]
[Y-AXIS-MASK=mask_txt_lit]
[Y-AXIS-MIN-VALUE={y_axis_min_value_lit|_num_lit}]
[Y-AXIS-MAX-VALUE={y_axis_max_value_lit|_num_lit}]
[Y-AXIS-MAJOR-INCREMENT={y_axis_major_increment_lit|_num_lit}]
[Y-AXIS-MINOR-INCREMENT=y_axis_minor_increment_num_lit]
[Y-AXIS-MAJOR-TICK-MARKS=y_axis_major_tick_marks_lit]
[Y-AXIS-MINOR-TICK-MARKS=y_axis_minor_tick_marks_lit]
[Y-AXIS-TICK-MARK-PLACEMENT=y_axis_tick_mark_placement_lit]
[Y-AXIS-SCALE=y_axis_scale_lit]
[Y2-AXIS-LABEL=y2_axis_label_lit]
[Y2-AXIS-MASK=mask_txt_lit]
[Y2-AXIS-MIN-VALUE={y2_axis_min_value_lit|_num_lit}]
[Y2-AXIS-MAX-VALUE={y2_axis_max_value_lit|_num_lit}]
[Y2-AXIS-MAJOR-INCREMENT={y2_axis_major_increment_lit|_num_lit}]
[Y2-AXIS-MINOR-INCREMENT=y2_axis_minor_increment_num_lit]
[Y2-AXIS-MAJOR-TICK-MARKS=y2_axis_major_tick_marks_lit]
[Y2-AXIS-MINOR-TICK-MARKS=y2_axis_minor_tick_marks_lit]
[Y2-AXIS-SCALE=y2_axis_scale_lit]
[Y2-COLOR-PALETTE=color_palette_lit]
[Y2-DATA-ARRAY=array_name]
[Y2-DATA-ARRAY-ROW-COUNT=row_count_num_lit]
[Y2-DATA-ARRAY-COLUMN-COUNT=column_count_num_lit]
[Y2-DATA-ARRAY-COLUMN-LABELS={NONE|array_name|(txt_lit,...)}]
[Y2-TYPE=chart_type_lit]
END-DECLARE

```

Note:

If you do not define the CHART-SIZE in DECLARE-CHART, you must define it in PRINT-CHART.

Arguments

Table 20 describes the DECLARE-CHART arguments. (These arguments are also valid for PRINT-CHART.) Default values are underlined>.

Note:

Several of the arguments in Table 20 refer to NewGraphics. To invoke NewGraphics, change the NewGraphics entry in the [Default-Settings] section of SQR.INI to TRUE (NewGraphics=True). (See NewGraphics under “[Default-Settings] Section” on page 328 for more information.)

When you use NewGraphics, font values are interpreted as HTML text size values (*not* point size values). For example, assume you have the following point values:

- ITEM-SIZE= ('Title',12)
- ITEM-SIZE= ('SubTitle',10)
- ITEM-SIZE= ('XAxisLabel',8)

In this example, if NewGraphics=True, you would convert the points size values to HTML text size values similar to the following:

- ITEM-SIZE= ('Title',3)
- ITEM-SIZE= ('SubTitle',2)
- ITEM-SIZE= ('XAxisLabel',1)

HTML text size values are:

- 1 - very small
- 2 - small
- 3 - normal size
- 4 - large
- 5 - larger
- 6 - very large
- 7 - largest

Table 20 DECLARE-CHART Arguments

Argument	Choices	Description
3D-EFFECTS	YES NO	YES gives your chart depth. NO displays the chart in a two-dimensional mode.

Argument	Choices	Description
		Must be set to YES for other 3D parameters to function.
ATTRIBUTES	See “Attributes Argument” on page 112 for information on the valid choices.	<p>Defines the appearance of a chart.</p> <p>Production Reporting reads and processes the keywords in the ATTRIBUTES argument left-to-right and first-to-last. As a result, a subsequent value setting overrides a previously-established value. Values assigned with DECLARE-CHART are overridden by values assigned with PRINT-CHART.</p> <p>Setting the ATTRIBUTES for the ALL selector establishes a default value for all property values within a chart. Any subsequent entry for a specific area, such as Header, overrides the value previously established by the ALL selector. Any invalid combination of selectors, sub-selectors, or declarations produce an error.</p> <p>Note: Some of the keywords in the ATTRIBUTES argument replace the functionality previously provided by the ITEM-COLOR, ITEM-SIZE, LEGEND-PLACEMENT, FILL, and COLOR-PALETTE arguments. Production Reporting processes all old style keyword, value parameters prior to the new ATTRIBUTES argument. This may result in the new ATTRIBUTES argument overriding a value previously established with the old style keyword, value parameter pairs.</p>
BORDER	YES NO	Defines whether to draw a border around the chart.
chart_name	User defined chart name.	Name for referencing a chart.
CHART-SIZE	User defined chart size.	Size of the chart frame in standard Production Reporting coordinate units.
COLOR-PALETTE	palette_name	<p>Defines the color of the individual data points in charts (for example, bar, slice, point). Use CREATE-COLOR-PALETTE to define a valid Production Reporting color palette to use.</p> <p>Note: The defined color palette is only valid when NewGraphics=TRUE.</p> <p>Note: The FOREGROUND and BACKGROUND declaration keywords in the ATTRIBUTES argument replace the functionality provided by COLOR-PALETTE . As a result, a value set with FOREGROUND or BACKGROUND overrides a value set with COLOR-PALETTE.</p>

Argument	Choices	Description
DATA-ARRAY Y2-DATA-ARRAY	array_name	Name of the array containing the data to plot. This must be the name of an array defined with CREATE-ARRAY. Use DATA-ARRAY to define the data array for the Y-Axis. Use Y2-DATA-ARRAY to define the data array for the Y2-Axis. (Y2-Axis values are ignored for pie charts.) Y2-DATA-ARRAY is only available with NewGraphics.
DATA-ARRAY-ROW-COUNT	row_count	Number of rows or sets of data to use from the DATA-ARRAY. If DATA-ARRAY has a greater number of rows, only DATA-ARRAY-ROW-COUNT is included in the chart.
Y2-DATA-ARRAY-ROW-COUNT	row_count	(NewGraphics) Number of rows or sets of data to use from Y2-DATA-ARRAY. If Y2-DATA-ARRAY has a greater number of rows, only Y2-DATA-ARRAY-ROW-COUNT is included in the chart.
DATA-ARRAY-COLUMN-COUNT	column_count	Number of columns to use from DATA-ARRAY. If the DATA-ARRAY has a greater number of columns, only DATA-ARRAY-COLUMN-COUNT is included in the chart.
Y2-DATA-ARRAY-COLUMN-COUNT	column_count	(New Graphics) Number of columns to use from the Y2-DATA-ARRAY. If Y2-DATA-ARRAY has a greater number of columns, only Y2-DATA-ARRAY-COLUMN-COUNT is included in the chart.
DATA-ARRAY-COLUMN-LABELS	NONE array_name (label1,label2, ...)	Labels for each Y-Axis value of the data set (fields) in DATA-ARRAY. Labels are displayed in the legend box. Column labels are ignored for pie charts. See Table 65 on page 266 for applicable fields for each type of chart.
Y2-DATA-ARRAY-COLUMN-LABELS	NONE array_name (label1,label2, ...)	(New Graphics) Labels for each Y2-Axis value of the data set (fields) in Y2-DATA-ARRAY. Labels are displayed in the legend box. Column labels are ignored for pie charts. See Table 65 on page 266 for applicable fields for each type of chart.
DATA-LABELS	YES NO	(NewGraphics) If YES, Production Reporting prints the numeric value above the individual data points. If NO, no numeric values are displayed.
FILL	GRAYSCALE COLOR CROSSHATCH NONE	Type of filling applied to the shapes (for example, bars, pie-segments) that represent the data loaded in the chart. GRAYSCALE varies the density of black dots. COLOR

Argument	Choices	Description
		<p>sends color instructions to the current printer. If the current printer does not support color, then it could appear in GRAYSCALE. CROSSHATCH uses patterns to fill the shapes representing each data set. With NONE all graph shapes are filled with white.</p> <p>Note: The STYLE declaration keyword in the ATTRIBUTES argument replaces the functionality provided by FILL. As a result, a value set with STYLE overrides a value set with FILL.</p>
FOOTER-TEXT	NONE text	<p>Footer text for the chart. The text is placed at the bottom of the chart.</p> <p>Default value is NONE.</p>
ITEM-COLOR	<p>ChartBackground—Background color of entire chart area .</p> <p>ChartForeground—Text and Line color of chart area.</p> <p>HeaderBackground—Area within the text box specified for the title and sub-title.</p> <p>HeaderForeground—Text color of the Title and sub-title.</p> <p>LegendBackground—Area within the box defining the legend.</p> <p>LegendForeground—Text and Outline color of the legend.</p> <p>ChartAreaBackground—Area that includes the body of the chart.</p> <p>ChartAreaForeground—Text and Line colors of the chart area.</p> <p>PlotAreaBackground—Area within the X and Y Axis of a chart.</p> <p>Note: For more information see, “Changing Colors with New Graphics” in Volume 1 of the <i>Hyperion SQR Production Reporting Developer’s Guide</i>.</p>	<p>(NewGraphics) Color for individual chart items. Specify a chart item and a valid (r,g,b) color to set the color of the chart item.</p> <p>Note: The COLOR declaration keyword in the ATTRIBUTES argument replaces the functionality provided by ITEM-COLOR. As a result, a value set with COLOR overrides a value set with ITEM-COLOR.</p>
ITEM-SIZE	Title SubTitle XAxisLabel XAxisMarkers YAxisLabel YAxisMarkers Y2AxisLabel Y2AxisMarkers LegendText LegendTitle	<p>(NewGraphics) Size of the following chart objects. The value is based on HTML text sizes.</p> <ul style="list-style-type: none"> ● Title—Chart title ITEM-SIZE=('Title',value) ● SubTitle—Chart subtitle ITEM-SIZE= ('SubTitle',value)

Argument	Choices	Description
		<ul style="list-style-type: none"> ● XAxisLabel—Label below the X Axis of the chart ITEM-SIZE= ('XAxisLabel',value) ● XAxisMarkers—Point labels on the X Axis ITEM-SIZE= ('XAxisMarkers',value) ● YAxisLabel—Rotated label to the left of the chart ITEM-SIZE= ('YAxisLabel',value) ● YAxisMarkers—Point labels on the Y Axis ITEM-SIZE= ('YAxisMarkers',value) ● Y2AxisLabel—Rotated Label to the right of the chart ITEM-SIZE= ('Y2AxisLabel',value) ● Y2AxisMarkers—Point labels on the Y Axis ITEM-SIZE= ('Y2AxisMarkers',value) ● LegendText—Legend text ITEM-SIZE= ('LegendText',value) ● LegendTitle—Legend title ITEM-SIZE= ('LegendTitle',value) <p>Note: If you do not define an ITEM-SIZE value, Production Reporting uses the HTML text value of 3 (normal size).</p> <p>Note: The POINT-SIZE declaration keyword in the ATTRIBUTES argument replaces the functionality provided by ITEM-SIZE. As a result, values set with POINT-SIZE override values set with ITEM-SIZE.</p>
LEGEND	YES NO	Defines whether to display a legend.
LEGEND-PLACEMENT	CENTER-RIGHT CENTER-LEFT UPPER-RIGHT UPPER-LEFT UPPER-CENTER LOWER-RIGHT LOWER-LEFT LOWER-CENTER	<p>Places the legend in the specified location on the chart. The first portion of the placement parameter (CENTER, UPPER, or LOWER) is the vertical position, and the second portion is the horizontal.</p> <p>Note: The LOCATION declaration keyword in the ATTRIBUTES argument replaces the functionality provided by LEGEND-PLACEMENT. As a result, values set with LOCATION override values set with LEGEND-PLACEMENT.</p>
LEGEND-PRESENTATION	INSIDE OUTSIDE	If INSIDE, then the legend is presented inside the area defined by the two axes. If OUTSIDE, then the legend is presented within the chart border, but outside of the region represented by the two axes.

Argument	Choices	Description
LEGEND-TITLE	NONE text	Title for the legend. If NONE, then no title is displayed in the legend box.
PIE-SEGMENT-EXPLODE	NONE MAX MIN USE-3RD-DATA-COLUMN	Controls what pie segments are exploded (selected) within the pie chart. MAX selects the largest segment. MIN selects the smallest segment. If USE-3RD-DATA-COLUMN, then the third field in the DATA-ARRAY is used to determine which pie segments are exploded. This third field should be a CHAR and values of 'YES' or 'Y' indicate that the segment should be exploded.
PIE-SEGMENT-PERCENT-DISPLAY	YES NO	If YES, then percent-of-total figures is presented for each pie segment.
PIE-SEGMENT-QUANTITY-DISPLAY	YES NO	If YES, then the quantity is presented for each pie segment.
POINT-MARKERS	YES NO	If YES, then point markers are displayed for line charts. If NO, then point markers are not displayed.
SUB-FOOTER-TEXT	NONE text	Sub-footer text for the chart. The text is placed below the footer regardless of whether the FOOTER-TEXT is specified. Default value is NONE.
SUB-TITLE	NONE text	Subtitle for the chart. Text is placed below the title regardless of whether or not TITLE is specified.
TITLE	NONE text	Chart title. Text is placed at the top of the chart.
TYPE Y2-TYPE (no pie charts)	PIE BAR AREA LINE STACKED-BAR OVERLAPPED-BAR FLOATING-BAR HISTOGRAM 100%-BAR 100%-AREA STACKED-AREA HIGH-LOW-CLOSE COMBO XY-SCATTER-PLOT BUBBLE RADAR POLAR CANDLE-STICK AREA-RADAR NONE Note: NONE is only valid for Y2-TYPE.	Chart type. See “Business Charts” in Volume 1 of the <i>Hyperion SQR Production Reporting Developer's Guide</i> (Y2-TYPE is only available with NewGraphics.)
Y2-AXIS-COLOR-PALETTE	palette_name	(NewGraphics) Color palette used to color data points in charts (for example, bar, slice, point). You must define a valid Production Reporting color-palette with CREATE-COLOR-PALETTE.
X-AXIS-GRID Y-AXIS-GRID Y2-AXIS-GRID	YES NO YES NO YES NO	If YES, then a dashed grid line is drawn for each major tick-mark on the axis. If NO, then no grid line is drawn on this axis.

Argument	Choices	Description
X-AXIS-LABEL Y-AXIS-LABEL Y2-AXIS-LABEL	NONE text	Line of text to display below (or alongside) the tick-mark labels on the axis. (Y2-AXIS-LABEL is only available with NewGraphics.)
X-AXIS-MAX-VALUE	AUTOSCALE number	Maximum value on the X axis. Data values greater than X-AXIS-MAX-VALUE are not displayed. AUTOSCALE calculates an appropriate maximum value.
Y-AXIS-MAX-VALUE	AUTOSCALE number	Maximum value on the Y axis. Data values greater than Y-AXIS-MAX-VALUE are not displayed. AUTOSCALE calculates an appropriate maximum value.
Y2-AXIS-MAX-VALUE	AUTOSCALE number	(NewGraphics) Maximum value on the Y2 axis. Data values greater than Y2-AXIS-MAX-VALUE are not displayed. AUTOSCALE calculates an appropriate maximum value.
X-AXIS-MIN-VALUE	AUTOSCALE number	Minimum value on the X axis. Data values less than X-AXIS-MIN-VALUE are not displayed. AUTOSCALE calculates an appropriate minimum value.
Y-AXIS-MIN-VALUE	AUTOSCALE number	Minimum value on the Y axis. Data values less than Y-AXIS-MIN-VALUE are not displayed. AUTOSCALE calculates an appropriate minimum value.
Y2-AXIS-MIN-VALUE	AUTOSCALE number	(NewGraphics) Minimum value on the Y2 axis. Data values less than Y2-AXIS-MIN-VALUE are not displayed. AUTOSCALE calculates an appropriate minimum value.
X-AXIS-MAJOR-TICK-MARKS	YES NO	YES displays tick-marks along the axis between X-AXIS-MIN-VALUE and X-AXIS-MAX-VALUE, according to the X-AXIS-SCALE setting spaced by X-AXIS-MAJOR-INCREMENT.
Y-AXIS-MAJOR-TICK-MARKS	YES NO	YES displays tick-marks along the axis between Y-AXIS-MIN-VALUE and Y-AXIS-MAX-VALUE, according to the Y-AXIS-SCALE setting spaced by Y-AXIS-MAJOR-INCREMENT.
Y2-AXIS-MAJOR-TICK-MARKS	YES NO	(NewGraphics) YES displays tick-marks along the axis between Y2-AXIS-MIN-VALUE and Y2-AXIS-MAX-VALUE, according to the Y2-AXIS-SCALE setting spaced by Y2-AXIS-MAJOR-INCREMENT.
Y-AXIS-MASK	'\$999,999.99'	Numeric mask used to format the Y Axis. Follows the edit mask rules defined in Table 56 .

Argument	Choices	Description
Y2-AXIS-MASK	'099999'	(NewGraphics) Numeric mask used to format the Y2 Axis. Follows the edit mask rules defined in Table 56 .
X-AXIS-MINOR-TICK-MARKS	YES NO	YES displays tick-marks along the axis between X-AXIS-MIN-VALUE and X-AXIS-MAX-VALUE, according to the X-AXIS-SCALE setting spaced by X-AXIS-MINOR-INCREMENT.
Y-AXIS-MINOR-TICK-MARKS	YES NO	YES displays tick-marks along the axis between Y-AXIS-MIN-VALUE and Y-AXIS-MAX-VALUE, according to the Y-AXIS-SCALE setting spaced by Y-AXIS-MINOR-INCREMENT.
Y2-AXIS-MINOR-TICK-MARKS	YES NO	(NewGraphics) YES displays tick-marks along the axis between Y2-AXIS-MIN-VALUE and Y2-AXIS-MAX-VALUE, according to the Y2-AXIS-SCALE setting spaced by Y2-AXIS-MINOR-INCREMENT.
X-AXIS-MAJOR-INCREMENT Y-AXIS-MAJOR-INCREMENT Y2-AXIS-MAJOR-INCREMENT	AUTOSCALE number	Increment used to space major tick-marks on the axis. AUTOSCALE calculates an appropriate increment. (Y2-AXIS-MAJOR-INCREMENT is only available with NewGraphics)
X-AXIS-MINOR-INCREMENT Y-AXIS-MINOR-INCREMENT Y2-AXIS-MINOR-INCREMENT	number	Increment used to space minor tick-marks on the axis. This must be set for the X-AXIS-MINOR-TICK-MARKS, the Y-AXIS-MINOR-TICK-MARKS, or the Y2-AXIS-MINOR-TICK-MARKS to display. (Y2-AXIS-MINOR-INCREMENT is only available with NewGraphics)
X-AXIS-ROTATE	X-Axis-Rotate = 0 No Rotation X-Axis-Rotate = 1 Always Rotate X-Axis-Rotate = n Rotate Labels if: <i>Data-Array-Row-Count > n</i>	Defines the X Axis rotation of Markers. The default value is 5.
X-AXIS-TICK-MARK-PLACEMENT Y-AXIS-TICK-MARK-PLACEMENT	INSIDE OUTSIDE BOTH	INSIDE (or OUTSIDE) directs Production Reporting to place the tick marks on the inside (or outside) of the axis only. BOTH directs Production Reporting to draw the tick-marks such that they appear on both sides of the axis. These arguments have no meaning in NewGraphics.
X-AXIS-SCALE Y-AXIS-SCALE Y2-AXIS-SCALE	LOG LINEAR	LOG specifies a logarithmic scale for the axis. Otherwise, the scale is LINEAR. (Y2-AXIS-SCALE is only available with NewGraphics)

Description

DECLARE-CHART defines the attributes of a chart to print as part of a report. You can use the attributes in any order, with the exception of *chart-name*, which must follow the DECLARE-CHART keyword. DECLARE-CHART can only appear in the SETUP section.

A chart defined with DECLARE-CHART prints by referencing its name in PRINT-CHART. You can override some or all of the chart attributes at run-time with PRINT-CHART. As such, DECLARE-CHART is useful when the basic properties of a chart are common to many PRINT-CHART commands.

Tip:

All values declared for a chart in the DECLARE-CHART section of an Production Reporting program become the default values for that chart. To override an assigned value, you must set the value in the PRINT-CHART section of the Production Reporting program. The following example illustrates this functionality.

```
begin-setup
  declare-chart default-chart
    attributes= ('All', 'Font', 31, 'Font-Style', 'Plain', 'Point-Size', 12)
    attributes= ('Header', 'Font', 31, 'Font-Style', 'Bold', 'Point-Size', 18)
    attributes= ('chart1', 'start-angle', 0, 'threshold-method', 'percent',
      'threshold-value', 20)
    attributes= ('chart1', '3d-depth', 5, '3d-rotation', 65,
      '3d-elevation', 85)
    attributes= ('chart1', 'cluster-width', 75, 'cluster-overlap', 65)
    attributes= ('legend', 'location', 'lower-center')
    attributes= ('chart1.fill', 'style', LIST:('25per', '50per', '75per'))
    attributes= ('chart1.line', 'style', LIST:'linestyle')
    attributes= ('all.line', 'size', 2)
    attributes= ('all.line', 'color', LIST: (('red'), ('green'), ('blue')))
    attributes= ('chart1', 'sort-order', 'Largest')
    attributes= ('chart1', 'Background', (230, 230, 255))
    attributes= ('all-axis.marker', 'foreground', (230, 0, 100))
  end-declare
  create-array
    name = emp_sales
    size = 20
    field = col_name:char:1
    field = sales:number:3
end-setup

begin-report
  create-list
    name='linestyle'
    list=('longdash', 'shortdash', 'dashdot', 'longshort')
  put 'Madeline' 10 12 12
    into emp_sales(0) col_name(0) sales(0) sales(1) sales (2)
  put 'Jacob' 25 35 45
    into emp_sales(1) col_name(0) sales(0) sales(1) sales (2)
  put 'Evan' 18 28 38
    into emp_sales(2) col_name(0) sales(0) sales(1) sales (2)
  put 'Claire' 60 70 80
    into emp_sales(3) col_name(0) sales(0) sales(1) sales (2)
```

```

print-chart default-chart (4,1)
  chart-size = (50,50)
  title = 'Employee Sales'
  type = overlapped-bar
  3D-effects = yes
  x-axis-label = 'Employees'
  y-axis-label = 'Sales (in thousands)'
  sub-ytitle = 'Overlapped-Bar Chart'
  data-array-row-count = 4
  data-array-column-count = 4
  data-array-column-labels = ('June', 'July', 'August')
  data-array = emp_sales
  footer-text = 'Keep up the good work'
  sub-footer-text = 'my team'
  attributes= (LIST:('header', 'footer'),'Font',32,
    'font-style','bold italic','Point-Size',18)
end-report

```

The FILL specification in DECLARE-PRINTER can influence the appearance of the chart. Table 21 lists the final appearance of the chart with a combination of values for PRINTER.COLOR and CHART.FILL options.

Table 21 PRINTER.COLOR Setting Effect on CHART.FILL

CHART.FILL=	PRINTER.COLOR=Y	PRINTER.COLOR=N
GRAYSCALE	GRAYSCALE	GRAYSCALE
COLOR	COLOR	GRAYSCALE
CROSSHATCH	COLOR-CROSSHATCH	CROSSHATCH
NONE	NONE	NONE

Examples

This example declares a basic sales chart using DECLARE-CHART. Then, for each region, the SUB-TITLE, DATA-ARRAY, and other elements are overridden to provide the chart with the specific features desired.

```

begin-setup
  declare-chart base_sales_chart
    chart-size      = (30, 20 )
    title           = 'Quarterly Sales'
    sub-title       = none
    fill            = color
    3d-effects      = yes
    type            = stacked-bar
    legend-title    = 'Product'
    x-axis-grid     = yes
  end-declare
end-setup

begin-program

  print-chart base_sales_chart
    sub-title      = 'Region I'

```

```

data-array          = reg1_sales
data-array-row-count = #rows_reg1
data-array-column-count = 2
y-axis-max-value    = #max_of_all_regions
y-axis-min-value    = #min_of_all_regions
legend              = no

print-chart base_sales_chart
sub-title           = 'Region II'
data-array          = reg2_sales
data-array-row-count = #rows_reg2
data-array-column-count = 2
y-axis-max-value    = #max_of_all_regions
y-axis-min-value    = #min_of_all_regions
legend              = no

end-program

begin-procedure chart_region_sales ($sub, $ary,
    #rows, #cols,
    #max_of_all_regions,
    #min_of_all_regions)

    print-chart base_sales_chart (20, 15 )
    sub-title           = $sub
    data-array          = all sales
    data-array-row-count = #rows
    data-array-column-count = #cols
    data-array-column-labels = ('Q1', 'Q2', 'Q3', 'Q4' )
    y-axis-max-value    = #max_of_all_regions
    y-axis-min-value    = #min_of_all_regions
    chart-size          = (50, 30)
end-procedure

```

Attributes Argument

The Attributes argument allows you to override the individual default values of most chart elements. It consists of two sub-parameters: *selector* and *declaration*. The simplest form of the Attributes argument is:

```
ATTRIBUTES=(selector, declaration, declaration value)
```

The *selector* identifies an element of the chart, the *declaration* identifies a property, and the *declaration value* identifies the property's value. For example, the following statement sets the text point-size for the entire chart:

```
ATTRIBUTES=('all', 'point-size', 12)
```

If desired, you can override a specific chart element. For example, to override the point-size for the title, you could specify the following:

```
ATTRIBUTES=('title', 'point-size', 16)
```

You can specify multiple selectors (use either an inline list, or a named list created with [CREATE-LIST](#)) and multiple declarations. For example, the following statement sets the point-size and foreground text color for Title and Sub-Title.


```
ATTRIBUTES=(LIST:('title','sub-title'),'point-size',16,foreground, ('red'))
```

Review the following topics for information on the *selector* and *declaration* sub-parameters.

- [Selector/Sub-Selector Keywords](#)
- [Declaration Keywords](#)

Selector/Sub-Selector Keywords

The combination of *selector* and *sub-selector* identifies specific chart elements. The format is *selector.sub-selector* where a period is used as the delimiter. In most cases, each component is optional. If you do not specify a selector, ALL is assumed. (ALL implies the complete set of selectors.)

Possible selector values include: ALL, CHART1, CHART2, HEADER, TITLE, SUB-TITLE, FOOTER, FOOTER-TEXT, SUB-FOOTER-TEXT, CHART-AREA, PLOT-AREA, LEGEND, LEGEND-SYMBOL, LEGEND-TITLE, ALL-AXIS, X-AXIS, Y-AXIS, and Y2-AXIS.

Note:

For Combination charts, use CHART1 and CHART2 selectors to identify chart elements for the primary and secondary charts, respectively. For example, to set the grid color for the primary chart to red, enter: `ATTRIBUTES=('chart1.grid', 'color', ('red'))`

Possible sub-selector values include: FILL, GRADIENT, GRID, HILO-LINE, LABEL, LINE, MARKER, OTHER, OUTLINE, RISING-FILL/FALLING-FILL, and SYMBOL.

Table 22 Sub-selector Descriptions

Sub_Selector	Description
FILL	Color of the fill area on a chart. Example: <code>ATTRIBUTES=('fill','style','verticalstripe')</code> Creates a fill style of VerticalStripe for all charts.
GRADIENT	Color of the gradient fill area on a chart. Example: <code>ATTRIBUTES=('chart1.gradient','color','none')</code> Declares that fill areas are not gradient. <code>ATTRIBUTES=('chart2.gradient','color','black')</code> Declares that fill areas are gradient and that black will be used to create the fill area.
GRID	Grid attributes (color, style, and line thickness) on a chart. Example: <code>ATTRIBUTES=('x-axis.grid','style','solid')</code> Sets the style of the x-axis grid to a solid line.

Sub_Selector	Description
HILO-LINE	<p>For HiLo and Candle charts, defines color, style, and line thickness of the Hi-Low line.</p> <p>Example:</p> <pre>ATTRIBUTES=('chart1.hilo-line', 'color', ('blue'))</pre> <p>Sets the HILO-LINE (line) color to blue.</p>
LABEL	<p>Label attributes.</p> <p>Example:</p> <pre>ATTRIBUTES=('x-axis.label', 'foreground', ('yellow'))</pre> <p>Sets the X-Axis label to the colors specified for all charts.</p>
LINE	<p>For Line charts, defines color, style, and thickness of data lines.</p> <p>Example:</p> <pre>ATTRIBUTES=('chart1.line', 'color', list(('red'), ('blue'), 'green'))</pre> <p>Sets the first, second, and third data lines to red, blue, and green, respectively.</p>
MARKER	<p>Marker label attributes.</p> <p>Example:</p> <pre>attributes=('X-axis.marker', 'point-size', .10)</pre> <p>Sets the point size for X-axis markers to ten.</p>
OTHER	<p>For Pie charts, defines the “Other” pie chart slice.</p> <p>Example:</p> <pre>attributes=('other', 'color', ('red'))</pre> <p>Sets the “Other” pie slice to red.</p>
OUTLINE	<p>Attributes (color, style, border thickness) for outlines (bars, pie slices, legend, and legend symbols).</p> <p>Example:</p> <pre>ATTRIBUTES=('chart1.outline', 'style', 'shortdash')</pre> <p>Sets the outline of chart areas to be a short dashed line.</p>
RISING-FILL/FALLING-FILL	<p>For Candle charts, attributes (fill pattern, color, width of candle) for the fill area in the rising and falling candle.</p> <p>Example:</p> <pre>ATTRIBUTES=('rising-fill', 'size', 5)</pre> <p>Sets the Rising Fill size to 5 pixels.</p>
SYMBOL	<p>For Line charts, defines the symbols that appear.</p> <p>Example:</p> <pre>attributes=('chart1.symbol', 'style', list('diamond', 'circle', 'square'), 'size', 20)</pre>

Sub_Selector	Description
	Identifies the symbols to be used for data points in a line chart. Diamonds, circles and squares are used for first, second and third data line, respectively. Each symbol is 20x20 pixels.

Table 23 Valid Selector/Sub-selector Combinations

	MARKER	LABEL	LINE	GRID	SYMBOL	OTHER	OUTLINE	GRADIENT	FILL	HILO-LINE	RISING-FILL	FALLING-FILL
ALL	•	•	•	•	•	•	•	•	•	•	•	•
CHART1	•	•	•	•	•	•	•	•	•	•	•	•
CHART2	•	•	•	•	•	•	•	•	•	•	•	•
ALL-AXIS	•	•	•									
X-AXIS	•	•	•									
Y-AXIS	•	•		•								
Y2-AXIS	•	•		•								
LEGEND							•					
LEGEND-SYMBOL							•					

Some examples of valid combinations using the selectors/sub-selectors in [Table 23](#) include `Chart1.Line`, `X-Axis.Marker`, and `Y2-Axis.Label`.

Declaration Keywords

A *declaration* identifies a chart property followed by its value. [Table 24](#) lists the declaration keywords available for the `ATTRIBUTES` argument.

Table 24 ATTRIBUTES Declaration Keywords

Declaration Keyword	Choices	Description
3D-DEPTH	0 - 100 Default = 10	Depth of the three-dimensional display in percent. Note: You must set the <code>3d-EFFECTS</code> argument to YES for this keyword to work. Example: <code>ATTRIBUTES= ('All', '3D-DEPTH', 15)</code> Displays all three-dimensional charts with a depth of 15 percent.

Declaration Keyword	Choices	Description
3D-ELEVATION	0 - 180 Default = 45	Elevation of the three-dimensional display in degrees. Note: You must set the 3D-EFFECTS argument to YES for this keyword to work. Example: <code>ATTRIBUTES= ('All', '3D-ELEVATION', 45)</code> Displays all three-dimensional charts with an elevation of 45 degrees.
3D-ROTATION	0 - 90 Default = 40	Rotation of the three-dimensional display in degrees. Note: You must set the 3D-EFFECTS argument to YES for this keyword to work. Example: <code>ATTRIBUTES= ('All', '3D-ROTATION', 50)</code> Displays all three-dimensional charts with a rotation of 50 degrees.
BACKGROUND	Named color or value in the range of RGB. Default = Transparent See “DECLARE-COLOR-MAP” on page 126 for an explanation of RGB values.	Background color of a selected chart area. Example: <code>ATTRIBUTES= ('All', 'BackGround', ('White'))</code> Sets the background color for all charts to white. Note: Along with FOREGROUND, BACKGROUND replaces the functionality provided by COLOR-PALETTE.
CLUSTER-OVERLAP	-100 - 100 Default = 0	Percentage of bar overlap. Negative values add space between bars. Positive values cause bars to overlap. Example: <code>ATTRIBUTES= ('All', 'CLUSTER-OVERLAP', 45)</code> Displays all bar charts with a cluster overlap of 45 percent.
CLUSTER-WIDTH	0 - 100 Default = 80	Percentage of available space between each bar cluster. Example: <code>ATTRIBUTES= ('All', 'CLUSTER-WIDTH', 60)</code> Displays all bar charts with a width of 60 percent between each bar cluster.

Declaration Keyword	Choices	Description
COLOR	<p>List generated from CREATE-LIST, or an in-line list of values.</p> <p>When referencing a list, the keyword LIST: must prefix the name of the list. When using a named_color_palette, the keyword PALETTE: must prefix the name of the color palette.</p>	<p>Defines a single color or a group of colors for an individual chart element.</p> <p>COLOR values are used in presentation order and are reused once the current list is exhausted. The COLOR keyword overrides the default values established for the color property of a chart.</p> <p>An error occurs if the contents of the list do not match the data type expected.</p> <p>COLOR can reference a color palette or a single color.</p> <p>Example:</p> <pre>ATTRIBUTES= ('CHART2.LINE', 'COLOR', LIST: (('red'), ('yellow'), ('maroon')))</pre> <p>Creates a group of line colors of 'red', 'yellow' and 'maroon' for chart2.</p> <pre>ATTRIBUTES= ('OTHER', 'COLOR', ('red'))</pre> <p>Sets the color for the 'Other' pie chart slice to 'red' for all pie charts.</p> <p>Note: COLOR replaces the functionality provided by ITEM-COLOR.</p>
FONT	<p>Values specified under [Fonts] in SQR.INI.</p> <p>Each entry consists of a font number assigned to a named font. For example, 3 may represent Courier.</p> <p>Default = Times New Roman</p>	<p>Font for all text and/or for specific text areas in a chart image.</p> <p>Example:</p> <pre>ATTRIBUTES= ('ALL', 'Font', 3, 'Point-Size', 12)</pre> <p>Sets the font typeface to the corresponding value from SQR.INI in the [Fonts] section for value 3 and the size to 12 point for all text items for all charts.</p>
FONT-STYLE	<p>PLAIN BOLD ITALIC UNDERLINE</p> <p>Default = PLAIN</p>	<p>Style of the font for all text and/or for specific text areas in a chart image. Separate multiple values with a space.</p> <p>Example:</p> <pre>ATTRIBUTES= ('Title', 'Font- Style', 'Bold Underline')</pre> <p>Sets the font style to 'BOLD' and 'Underline' for all charts.</p>
FOREGROUND	<p>Named color or values in the range of RGB.</p> <p>Default = Black</p>	<p>Text, outline, and line color for a chart area.</p> <p>Example:</p>

Declaration Keyword	Choices	Description
	See "DECLARE-COLOR-MAP" on page 126 in the for an explanation of RGB values.	<p>ATTRIBUTES= ('X-Axis.Label', 'ForeGround', ('Yellow'))</p> <p>Sets the X-Axis label for all charts to yellow.</p> <p>Note: Along with BACKGROUND, FOREGROUND replaces the functionality provided by COLOR-PALETTE.</p>
HALF-RANGE	<p>YES NO</p> <p>Default = NO</p>	<p>Determines how the x-axis is displayed in Polar charts.</p> <p>NO—X-axis is displayed as one full range fro 0 to 360 degrees.</p> <p>YES—X-axis is displayed as two half-ranges from –180 to 180 degrees.</p> <p>Example:</p> <p>ATTRIBUTES= ('Chart1', 'HALF-RANGE', 'Yes')</p> <p>Sets the axis label range from -180 to 180 degrees.</p>
HOLE-VALUE	<p>A value between the following minimum and maximum values:</p> <p>Min Value: -1.7976931348623157E+308</p> <p>Max Value: 1.7976931348623157E+308</p> <p>Default = None</p>	<p>Value in the data to ignore. You can only have one HOLE-VALUE per chart.</p> <p>Example:</p> <p>ATTRIBUTES= ('All', 'HOLE-VALUE', -1)</p> <p>Sets the HOLE-VALUE to -1. This means that if DECLARE-CHART or PRINT-CHART finds a value of -1 in the data to chart, the -1 value is ignored.</p>
LABEL-LOCATION	<p>INNER OUTER AUTO</p> <p>Default = AUTO</p>	<p>Location of labels in pie charts.</p> <p>INNER—Labels are placed within the chart area.</p> <p>OUTER—Labels are placed outside the chart area.</p> <p>AUTO—The label location is controlled by the charting application.</p> <p>Example:</p> <p>ATTRIBUTES= ('All', 'LABEL-LOCATION', 'OUTER')</p> <p>Displays the pie chart labels outside of the chart area.</p>
LEGEND-COLUMNS	<p>Any numeric value.</p> <p>Default = 0</p> <p>A value of 0 means the charting application will determine the proper value.</p>	<p>Number of columns to use when generating the chart legend. You can only have one LEGEND-COLUMNS value per chart.</p> <p>Example:</p>

Declaration Keyword	Choices	Description
		<p>ATTRIBUTES= ('All' , 'LEGEND-COLUMNS' , 4)</p> <p>Displays the chart legend with four columns.</p>
LEGEND-ROWS	<p>Any numeric value.</p> <p>Default = 0</p> <p>A value of 0 means the charting application will determine the proper value.</p>	<p>Number of rows to use when generating the chart legend. You can only have one LEGEND-ROWS value per chart.</p> <p>Example:</p> <p>ATTRIBUTES= ('All' , 'LEGEND-ROWS' , 2)</p> <p>Displays the chart legend with two rows.</p>
LOCATION	<p>LOWER-RIGHT CENTER-RIGHT UPPER-RIGHT LOWER-LEFT CENTER-LEFT UPPER-LEFT LOWER-CENTER UPPER-CENTER</p> <p>Note:CENTER-CENTER is reserved for ChartArea.</p>	<p>Positions the main chart elements of the Header, Footer, and Legend in pixels, or controls the specific location for the Legend.</p> <p>Each positional object is considered a rectangle, and the (x,y) pixel location of (1,1) relates to the upper left-hand coordinate of an image. The charting application positions the upper left-hand corner of the rectangle at the pixel location specified.</p> <p>Example:</p> <p>ATTRIBUTES= ('Legend' , 'LOCATION' , 'CENTER-RIGHT')</p> <p>Positions the chart legend in the center vertical and right horizontal location of the chart.</p> <p>Note: LOCATION replaces the functionality provided by LEGEND-PLACEMENT.</p>
ORIGIN-BASE-ANGLE	<p>0 - 360</p> <p>Default = 0</p>	<p>Position of the x-axis for Polar, Radar, and Area Radar charts in degrees.</p> <p>Example:</p> <p>ATTRIBUTES= ('CHART1' , 'ORIGIN-BASE-ANGLE' , '90')</p> <p>Sets the ORIGIN-BASE-ANGLE to 90 degrees.</p>
OTHER-LABEL	<p>Any value.</p> <p>Default = Other</p>	<p>Name of the label used for the “Other” pie chart segment.</p> <p>Example:</p> <p>ATTRIBUTES= ('ALL' , 'OTHER-LABEL' , 'OTHER-PRODUCTS')</p> <p>Declares the label used for the “Other” pie chart segment is “Other-Products”.</p>
PATTERN	<p>CIRCULAR WEBBED</p> <p>Default = CIRCULAR</p>	<p>Sets the gridlines in Radar and Area Radar charts to circular or webbed.</p>

Declaration Keyword	Choices	Description
		<p>Example:</p> <pre>ATTRIBUTES= ('CHART1.GRID', 'PATTERN', 'WEBBED')</pre> <p>Sets the Radar chart grid to webbed..</p>
PERCENTAGE-PRECISION	<p>0 - 15</p> <p>Default = 2</p>	<p>Number of digits to the right of the decimal point in a pie chart.</p> <p>Example:</p> <pre>ATTRIBUTES= ('All', 'PERCENTAGE-PRECISION', 5)</pre> <p>Displays the pie chart percentage with five digits to the right of the decimal point.</p>
POINT-SIZE	<p>Any point size.</p> <p>Default = 12</p>	<p>Font size (in points) for all areas in a chart image. Control can be for all text on a chart and/or for specific text areas on a chart.</p> <p>Example:</p> <pre>ATTRIBUTES= ('Title', 'Style', 'Bold Underline', 'Point-Size', 16)</pre> <p>Sets the font style to 'Bold' and 'Underline' and font size to 16 point for the 'Title' text item for all charts.</p> <p>Note: POINT-SIZE replaces the functionality provided by ITEM-SIZE.</p>
SIZE	<p>Depends on the SUB-SELECTOR:</p> <ul style="list-style-type: none"> ● LINE - 1 to 10 pixels ● SYMBOL - 1 to 100 pixels ● GRID - 1 to 10 pixels 	<p>Size of the lines, symbols, and grids for a chart element.</p> <p>The SIZE keyword overrides the default values established for the size property of a chart.</p> <p>The default line and grid width is 1 pixel. The default symbol width is a bounding box of 6x6 pixels.</p> <p>Example:</p> <pre>ATTRIBUTES= ('CHART1.LINE', 'SIZE', 2)</pre> <p>Sets the line size for all lines to 2 pixels for chart1.</p>
SORT-ORDER	<p>LARGEST SMALLEST DATAORDER</p> <p>Default = DATAORDER</p>	<p>Defines whether to display pie-chart slices largest-to-smallest, smallest-to-largest, or the order they appear in the data.</p> <ul style="list-style-type: none"> ● LARGEST—Largest-to-smallest ● SMALLEST—Smallest-to-largest ● DATAORDER—Order they appear in the data.

Declaration Keyword	Choices	Description
		<p>Example:</p> <pre>ATTRIBUTES= ('All', 'SORT-ORDER', 'Largest')</pre> <p>Sets the display order of the pie slices from largest to smallest for all charts.</p>
START-ANGLE	<p>0 - 359</p> <p>Default = 0</p>	<p>For Pie Charts:</p> <p>Position in the pie chart where the first pie slice is drawn.</p> <p>A value of zero degrees represents a horizontal line from the center of the pie to the right-hand side of the pie chart.</p> <p>A value of 90 degrees represents a vertical line from the center to the top of the pie.</p> <p>Slices are drawn clockwise from the specified angle.</p> <p>The default position for the first pie segment is 90 degrees.</p> <p>Example:</p> <pre>ATTRIBUTES= ('All', 'START-ANGLE', 45)</pre> <p>Sets the starting location of the first pie slice to a line 45 degrees to the right and down of a horizontal line.</p> <p>For Polar, Radar, and Area Radar Charts:</p> <p>Angle that the y-axis makes with the ORIGIN-BASE-ANGLE.</p> <p>Example:</p> <pre>ATTRIBUTES= ('Chart1', 'START-ANGLE', 135)</pre> <p>Sets the START-ANGLE to 135 degrees.</p>
STYLE	<p>Depends on the SUB-SELECTOR:</p> <ul style="list-style-type: none"> ● LINE - Solid LongDash ShortDash LongShort DashDot ● SYMBOL - None Dot Box Triangle Diamond Star VerticalLine HorizontalLine Cross Circle Square ● FILL - None Solid 25Per 50Per 75Per HorizontalStripe VerticalStripe 45Stripe 135Stripe DiagonalHatch CrossHatch 	<p>Group of styles for an individual chart element.</p> <p>This keyword can use the list generated from CREATE-LIST, or you can enter an in-line list of values. When using a named list, the keyword LIST: must prefix the name of the list. An error occurs if the contents of the list do not match the data type expected.</p> <p>STYLE values are used in presentation order and are reused once the current list is exhausted. The STYLE keyword overrides the default values established for the style property of a chart.</p> <p>The default line and fill style is Solid. The default symbol style is Dot.</p>

Declaration Keyword	Choices	Description
		<p>Example:</p> <pre>ATTRIBUTES= ('Chart1.Line', 'Style', LIST: ('shortdash', 'longdash', 'solid'))</pre> <p>Creates a group of line styles of 'shortdash', 'longdash', and 'solid' for chart1.</p> <p>Note: STYLE replaces the functionality provided by FILL.</p>
THRESHOLD-METHOD	VALUE PERCENT Default = VALUE	<p>Grouping method to use for the Other slice in a pie chart.</p> <ul style="list-style-type: none"> ● VALUE—Use when you know the data value to group into the Other slice. Value places only those items that are less than the Threshold-Value into the Other slice. ● PERCENT—Use when you want to devote a certain percentage of the pie to the Other slice. Percent places only those items that are accumulatively less than the Threshold-Value into the Other slice. <p>Example:</p> <pre>ATTRIBUTES= ('All', 'THRESHOLD- METHOD', 'Value')</pre> <p>Sets the THRESHOLD-METHOD to indicate that the THRESHOLD-VALUE is a data value for all charts.</p>
THRESHOLD- VALUE	<p>Depends on the THRESHOLD-METHOD:</p> <ul style="list-style-type: none"> ● When THRESHOLD-METHOD is set to <i>Value</i>, valid values are numbers greater than or equal to 0. ● When THRESHOLD-METHOD is set to <i>Percent</i>, valid values are from 0 to 100. <p>Default = 0 (No Other Slice)</p>	<p>Data value to group into the Other slice in a pie chart.</p> <p>Example:</p> <pre>ATTRIBUTES= ('All', 'THRESHOLD- VALUE', 10)</pre> <p>Together with THRESHOLD-METHOD establishes that values less than 10 will be grouped into the 'Other' pie slice for all charts.</p>
UNITS	DEGREES RADIANS GRADS Default = DEGREES	<p>Angular units used for Polar, Radar, and Area Radar charts. Affects the ORIGIN-BASE-ANGLE and START-ANGLE.</p> <p>Example:</p> <pre>ATTRIBUTES= ('Chart1', 'UNITS', 'Radians')</pre> <p>Displays the ORIGIN-BASE-ANGLE and START-ANGLE in radians.</p>

In certain instances, the declaration value must be a list (either an inline list, or a named list previously created using [CREATE-LIST](#)).

- A list is always required for the STYLE declaration. For example:

```
ATTRIBUTES=('Chart1.Fill', 'Style',
           LIST:('Solid', '45Stripe', 'DiagonalHatch'))
```

```
ATTRIBUTES=('Chart2.Line', 'Style',
           LIST:('LongDash', 'DashDot', 'Solid'))
```

```
ATTRIBUTES=('Symbol', 'Style',
           LIST:('Square', 'Diamond', 'Triangle'))
```

- A list can also be specified for the COLOR declaration when used with the LINE, SYMBOL, or FILL sub-selector. For example:

```
ATTRIBUTES=('Chart1.Symbol', 'Color',
           LIST:(('Red'), ('Blue'), (100, 200, 130)))
```

Selector/Sub-Selector - Declaration Keyword Combinations

Table 25 and Table 26 show which selectors/sub-selectors are valid for each declaration keyword. (The selectors and sub-selectors are listed in the first column, and the declaration keywords are listed in the first row.) Production Reporting does not allow invalid combinations of selectors, sub-selectors, and declarations.

Table 25 Valid Selector/Sub-selector -Declaration Keyword Combinations

	Color	Style	Size	Pattern	Font Font Style Point Size	Sort Order	Background Colors	ForeGround Colors
Selectors								
ALL					•	•	•	•
CHART1					•	•	•	•
CHART2					•	•	•	•
HEADER					•		•	•
SUBTITLE					•			•
FOOTER					•		•	•
FOOTER-TEXT					•			•
SUB-FOOTER-TEXT					•			•
CHART-AREA					•		•	•
PLOT-AREA					•		•	•
LEGEND					•		•	•

	Color	Style	Size	Pattern	Font Font Style Point Size	Sort Order	Background Colors	ForeGround Colors
LEGEND-TITLE					•			•
LEGEND-SYMBOL								
ALL-AXIS					•			•
X-AXIS					•			•
Y-AXIS					•			•
Y2-AXIS					•			•
Sub-selectors								
MARKER					•			•
LABEL					•			•
LINE	•	•	•					
GRID	•	•	•	•				
SYMBOL	•	•	•					
FILL	•	•						
OTHER	•							

Table 26 Valid Selector/Sub-selector -Declaration Keyword Combinations

	Start Angle	3d- Depth 3d- Elevation 3d- Rotation	Cluster- Width Cluster- Overlap	Threshold Value Threshold Method	Location	Hole Value	Legend Columns	Legend Rows	Percentage Precision	Label Location	Other Label	Units	Origin Base- Angle
ALL	•	•	•	•		•	•	•	•	•	•	•	•
CHART1	•	•	•	•		•	•	•	•	•	•	•	•
CHART2	•	•	•	•		•	•	•	•	•	•	•	•
HEADER					•								
FOOTER					•								
LEGEND					•								

	Start Angle	3d-Depth 3d-Elevation 3d-Rotation	Cluster-Width Cluster-Overlap	Threshold Value Threshold Method	Location	Hole Value	Legend Columns	Legend Rows	Percentage Precision	Label Location	Other Label	Units
Y-AXIS	●											

Sub-Selector - Declaration Keyword Value Ranges

Table 27 shows the value ranges for sub-selector - declaration keywords. These are the values assigned for each property not defined in DECLARE-CHART.

Table 27 Sub-selector - Declaration Keyword Value Ranges

Sub-selector	Declaration	Value
LINE	STYLE	Solid, LongDash, ShortDash, LongShort, Dot, DashDot Default value = Solid
LINE	SIZE	1 - 10 pixels Default value = 1
LINE	COLOR	Named colors or values in the range of RGB. See "DECLARE-COLOR-MAP" on page 88 in the for an explanation of RGB values.
SYMBOL	STYLE	None, Dot, Box, Triangle, Diamond, Star, VerticalLine, HorizontalLine, Cross, Circle, Square Default value = Dot
SYMBOL	SIZE	1 - 100 pixels Default value = bounding box 6x6
SYMBOL	COLOR	Named color or Values in the range of RGB. See "DECLARE-COLOR-MAP" on page 88 in the for an explanation of RGB values.
FILL	STYLE	None, Solid, 25Per, 50Per, 75Per, HorizontalStripes, VerticalStripe, 45Stripe, 135Stripe, DiagonalHatch, CrossHatch Default value = Solid
FILL	COLOR	Named color or values in the range of RGB. See "DECLARE-COLOR-MAP" on page 88 in the for an explanation of RGB values.
GRID	SIZE	1 - 10 pixels Default value = 1
GRID	COLOR	Named color or values in the range of RGB. See "DECLARE-COLOR-MAP" on page 88 in the for an explanation of RGB values.
GRID	STYLE	None, Solid, LongDash, ShortDash, LongShort, DashDot

Sub-selector	Declaration	Value
		Default value = Solid
OUTLINE	STYLE	None, Solid, LongDash, ShortDash, LongShort, DashDot Default value = Solid
OUTLINE	SIZE	1 - 10 pixels Default value = 1
OUTLINE	COLOR	Named color or values in the range of RGB. See "DECLARE-COLOR-MAP" on page 88 in the for an explanation of RGB values.
OTHER	COLOR	Named color or values in the range of RGB. See SQR Language Reference/Declare-Color-Map for explanation of RGB values. Default value = Purple - RGB(170,00,255)

See Also

[PRINT-CHART](#)

DECLARE-COLOR-MAP

Function

Defines colors in an Production Reporting report.

Syntax

In the SETUP section:

```
DECLARE-COLOR-MAP
```

```
    color_name=({rgb})
```

```
color_name=({rgb})
```

```
.
```

```
.
```

```
.
```

```
END-DECLARE
```

Arguments

color_name

A *color_name* is composed of the alphanumeric characters (A-Z, 0-9), the underscore (_) character, and the dash (-) character. It must start with an alpha (A-Z) character. It is case insensitive. The name 'none' is reserved and cannot be assigned a value. A name in the format (RGBredgreenblue) cannot be assigned a value. The name 'default' is reserved and can be

assigned a value. 'Default' is used during execution when a referenced color is not defined in the runtime environment.

{ rgb }

red_lit|_var|_col, green_lit|_var|_col, blue_lit|_var|_col where each component is a value in the range of 000 to 255. In the `BEGIN-SETUP` section, only literal values are allowed.

Default colors implicitly installed with Production Reporting include:

black = (0,0,0)

white=(255,255,255)

gray=(128,128,128)

silver=(192,192,192)

red=(255,0,0)

green=(0,255,0)

blue=(0,0,255)

yellow=(255,255,0)

purple=(128,0,128)

olive=(128,128,0)

navy=(0,0,128)

aqua=(0,255,255)

lime=(0,128,0)

maroon=(128,0,0)

teal=(0,128,128)

fuchsia=(255,0,255)

Description

`DECLARE-COLOR-MAP` in the `BEGIN-SETUP` section defines or redefines colors in an Production Reporting report. You can define an endless number of entries.

Examples

```
begin-setup
  declare-color-map
    light_blue = (193, 222, 229)
  end-declare
end-setup
```

See Also

[ALTER-COLOR-MAP](#), [GET-COLOR](#), and [SET-COLOR](#)

DECLARE-CONNECTION

Function

Defines data source logon parameters prior to logon. Can be used to override default connection logon parameters.

Note:

DECLARE-CONNECTION is specific to Production Reporting DDO ports only.

Syntax

In the SETUP section:

```
DECLARE-CONNECTION connection_name
DSN={uq_txt_lit}
[USER={uq_txt_lit}]
[PASSWORD={uq_txt_lit}]
[PARAMETERS=keyword_str=attr_str; [keyword_str=attr_str; ...]]
[NO-DUPLICATE=TRUE|FALSE]
SET-GENERATIONS=({dimension1, hierarchy1} [, dimensioni, hierarchyi] ...)
SET-LEVELS=({dimension1, level1} [, dimensioni, leveli] ...)
SET-MEMBERS=({dimension1, level1} [, dimensioni, leveli] ...)
END-DECLARE
```

Arguments

connection_name

User-defined name for describing a datasource connection.

DSN

Logical datasource name recorded in the DDO Registry (Registry.properties file).

USER, PASSWORD

Traditional logon semantics.

PARAMETERS=*keyword_str=attr_str*;

List of keyword-attribute pairs required by a datasource driver for logon. There is no syntax restriction on these entries apart from the delimiting semi-colons (;) and equal signs (=). The keywords must match the logon property names listed for a datasource.

NO-DUPLICATE=TRUE|FALSE (default is FALSE)

(Optional) Prevents Production Reporting from automatically creating additional logins to datasources that are busy handling a previous query. Creating a new login in such cases is the default behavior for Production Reporting, which allows a single CONNECTION declaration to use in a subquery. This behavior, while allowing dynamic logins as-needed, causes difficulties when doing both DDL (BEGIN-SQL) and DML (BEGIN-SELECT) against temporary tables in certain vendors datasources. In such cases, you must fetch from the temporary table using the

same login in which it was created. Here, you should code the `CONNECTION` as `NO-DUPLICATE=TRUE`, and then use that connection in both the table creation logic of `BEGIN-SQL` and the row fetching logic of `BEGIN-SELECT`.

SET-GENERATIONS

Dimension hierarchy for the previously-declared dimension. The dimension and hierarchy defined with `SET-GENERATIONS` can be a *literal* value only. Consider the following example:

```
set-generations=('product',5,'time',1 )
```

In this example, `SET-GENERATIONS`:

- Returns the set of members in the ‘product’ dimension that are at the 5th generation in the dimension’s hierarchy.

For example, returns all ‘Brand Name’ members (Generation Level 5) under the product hierarchy of ‘all products.drink.alcoholic beverages.beer and wine’. This would increase the result set to a list of beers and wines.

- Returns the set of members in the ‘time’ dimension that are at the 1st generation deep into the dimension.

For example, returns all ‘Year’ members (Generation Level 1) under the time hierarchy of ‘1997.Q.2.’ This reduces result set to ‘1997’.

See “Set Generations” in Volume 3 of the *Hyperion SQR Production Reporting Developer's Guide* for detailed examples of `SET-GENERATIONS`.

SET-LEVELS

Extends the dimension hierarchy for the previously-declared dimension. The dimension and hierarchy defined with `SET-LEVELS` can be a *literal* value only. Consider the following example:

```
set-levels=('product',2 )
```

In this example:

- `SET-LEVELS` used with only the previous `SET-MEMBERS` returns all members under the product hierarchy and the next two generations (Product SubCategory and Brand Name) for the product hierarchy of ‘all products.drink.alcoholic beverages.beer and wine’.
- `SET-LEVELS` used with the previous `SET-MEMBERS` and `SET-GENERATIONS` returns all members for generation levels 5 through 7 under the product hierarchy of ‘all products.drink.alcoholic beverages.beer and wine.’

See “Set Levels” in Volume 3 of the *Hyperion SQR Production Reporting Developer's Guide* detailed examples of `SET-LEVELS`.

SET-MEMBERS

Returns the set of members in a dimension, level, or hierarchy whose name is specified by a string. The dimension and hierarchy defined with `SET-MEMBERS` can be a *literal* value only. Consider the following example:

```
set-members=('product','all products.drink.alcoholic beverages.beer and wine','time','1997.Q1.2' )
```

In this example, SET-MEMBERS:

- Returns the set of members in the dimension ‘product’ at the specific hierarchy of ‘all products’, at a specific level of ‘drink’, at a specific level of ‘alcoholic beverages’, at a specific level of ‘beer and wine’.
- Returns the set of members in the dimension ‘time’ at the specific hierarchy of ‘1997’, at the specific level of ‘Q1’, at the specific level of ‘2’.

See “Set Members” in Volume 3 of the *Hyperion SQR Production Reporting Developer's Guide* for detailed examples of SET-MEMBERS.

Examples

```
declare-connection SAPR3-1
  dsn=SAPR3
  username=guest
  password=guest
end-declare
```

See Also

[ALTER-CONNECTION](#)

DECLARE-IMAGE

Function

Declares the type, size, and source of an image to print.

Syntax

```
DECLARE-IMAGE image_name
[TYPE=image_type_lit]
[IMAGE-SIZE=(width_num_lit,height_num_lit)]
[SOURCE=file_name_lit]
[[FOR-PRINTER=( {POSTSCRIPT|HPLASERJET|HTML|PDF|WINDOWS |PS|HP|HT|PD|
WP}, image_type_lit,file_name_lit) . . .]
END-DECLARE
```

Note:

DECLARE-IMAGE and PRINT-IMAGE work together to identify information about the image. The IMAGE-SIZE argument is required and must be defined in either DECLARE-IMAGE or PRINT-IMAGE. The SOURCE and TYPE arguments are optional; however, if you define one you must define the other.

Arguments

image_name

Unique name for referencing the image declaration.

TYPE

Image type. Types can be EPS-FILE, HPGL-FILE, GIF-FILE, JPEG-FILE, BMP-FILE, PNG-FILE, or AUTO-DETECT.

IMAGE-SIZE

Width and height of the image in Production Reporting coordinates.

SOURCE

Name of a file containing the image. The file must be in the SQRDIR directory, or you must specify the full path.

FOR-PRINTER

Separate image file for each report output type.

Tip:

The TYPE and SOURCE arguments contain the default values. You can override these defaults for a specific printer by using the FOR-PRINTER argument.

Note:

If the file is not in the SQRDIR directory, the full path or no path should be given. A relative path will not do, because you need to know where you execute the file from.

Description

DECLARE-IMAGE defines and names an image. This image can then be placed in a report at the position specified with PRINT-IMAGE.

If an image has not been declared, or if the image type is not supported for a particular report output type, or if the image file has incomplete header information, then a box (either shaded for HP printers or with a diagonal line through it for Postscript printers) appears where the image is expected. Table 28 illustrates the valid relationships between image type and report output type.

Table 28 Valid Images Types

	BMP	EPS	GIF	HPGL	JPEG	PNG
HPLaserJet				X		
HTML	X	X	X	X	X	X
PDF	X		X		X	X
Postscript		X				
Windows	X					

Examples

```
declare-image officer-signature
  type= eps-file
  source= 'off_sherman.eps'
  image-size= (40, 5)
end-declare
```

```
declare-image oracle-logo
  type='auto-detect'
  image-size=(40,10)
  source=$BLOB_Column
```

The following example defines separate image files for different printer types:

```
begin-setup
  declare-image oracle_logo
    type=GIF-FILE
    Image-size=(40,10)
    source=oracle.gif
    for-printer=(PS, EPS-FILE, 'oracle.eps')
    for-printer=(HP, HPGL-FILE, 'oracle.hppl')
  end-declare
end-setup

begin-report
  move 'hyperion.bmp' to $image_src
  print-image oracle_logo (10,15)
    for-printer=(WP, BMP-FILE, $image_src)
end-report
```

In this example, the image file used for each printer type is:

- HP—'oracle.hppl' (Identified using FOR-PRINTER in DECLARE-IMAGE)
- PS—'oracle.eps' (Identified using FOR-PRINTER in DECLARE-IMAGE)
- PD—'oracle.gif' (Declared as default using SOURCE= in DECLARE-IMAGE)
- HT—'oracle.gif' (Declared as default using SOURCE= in DECLARE-IMAGE)
- WP—'oracle.bmp' (Identified using FOR-PRINTER in PRINT-IMAGE)

See Also

- [PRINT-IMAGE](#)
- “Adding Graphics” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*.

DECLARE-LAYOUT

Function

Defines the attributes for the layout of an output file.

Syntax

```
DECLARE-LAYOUT layout_name
[PAPER-SIZE=({paper_width_num_lit[uom],paper_depth_num_lit[uom]} |
{paper_name})]
[FORMFEED=form_feed_lit]
[ORIENTATION=orientation_lit]
[LEFT-MARGIN=left_margin_num_lit[uom]]
[TOP-MARGIN=top_margin_num_lit[uom]]
[RIGHT-MARGIN=right_margin_num_lit[uom]]
|LINE-WIDTH=line_width_num_lit[uom]
|MAX-COLUMNS=columns_int_lit]
[BOTTOM-MARGIN=bottom_margin_num_lit[uom]]
|PAGE-DEPTH=page_depth_num_lit[uom]
|MAX-LINES=lines_int_lit]
[CHAR-WIDTH=char_width_num_lit[uom]]
[LINE-HEIGHT=line_height_num_lit[uom]]
END-DECLARE
```

Arguments

layout_name

Unique layout name used to reference the layout and its attributes.

uom

Optional suffix which denotes the unit of measure applied to the preceding value.

Table 29 Valid *uom* Suffixes

Suffix	Meaning	Definition
dp	decipoint	0.001388 inch
pt	point	0.01388 inch
mm	millimeter	0.03937 inch
cm	centimeter	0.3937 inch
in	inch	1.0000 inch

paper_name

An option of PAPER-SIZE. This name is associated with predefined dimensions.

Table 30 Valid Paper Names

Name	Width	Depth	Orientation
Letter	8.5 in	11 in	Portrait
Legal	8.5 in	14 in	Portrait
A4	8.27 in	11.69 in	Portrait

Name	Width	Depth	Orientation
A3	11.69 in	16.54 in	Portrait
Executive	7.25 in	10.5 in	Portrait
B5	7.17 in	10.12 in	Portrait
Com-10	4.125 in	9.5 in	Landscape
Monarch	3.875 in	7.5 in	Landscape
DL	4.33 in	8.66 in	Landscape
C5	6.378 in	9.016 in	Landscape

Table 31 DECLARE-LAYOUT Command Arguments

Argument	Choice or Default uom	Default Value	Description
PAPER-SIZE	inches	8.5 in, 11 in	Physical size of the page. The first parameter is the width of the page. The second parameter is the depth or length. It may also be a predefined name. (See Table 30.) Note: When ORIENTATION= LANDSCAPE the default values are 11 in, 8.5 in.
FORMFEED	YES, NO	YES	Whether to write form feeds at the end of each page.
ORIENTATION	PORTRAIT, LANDSCAPE	PORTRAIT	Portrait = vertical. Landscape = horizontal. Printing in landscape for the printer type HPLASERJET requires landscape fonts.
LEFT-MARGIN	inches	0.5 in	Amount of blank space to leave at the left side of the page.
TOP-MARGIN	inches	0.5 in	Amount of blank space to leave at the top of the page.
RIGHT-MARGIN	inches	0.5 in	Amount of blank space to leave at the right side of the page. If you specify LINE-WIDTH or MAX-COLUMNS, you cannot use this parameter.
LINE-WIDTH	inches	7.5 in	Length of the line. If you specify RIGHT-MARGIN or MAX-COLUMNS, you cannot use this parameter.
MAX-COLUMNS		75	Maximum number of columns in a line. If you specify RIGHT-MARGIN or LINE-WIDTH, you cannot use this parameter.
BOTTOM-MARGIN	inches	0.5 in	Amount of blank space to leave at the bottom of the page. If you specify PAGE-DEPTH or MAX-LINES, you cannot use this parameter.
PAGE-DEPTH	inches	10 in	Depth of the page. If you specify BOTTOM-MARGIN or MAX-LINES, you cannot use this parameter.
MAX-LINES		60	Maximum number of lines printed on the page. If you specify PAGE-DEPTH or BOTTOM-MARGIN, you cannot use this parameter.
LINE-HEIGHT	points	12 pt	Size of each line on the page. There are 72 points per inch. If LINE-HEIGHT is not specified, it follows the value for POINT-SIZE, if specified. The default value of 12 points yields 6 lines per inch. For the printer type LINEPRINTER, this value is used only to calculate the TOP-MARGIN and BOTTOM-MARGIN (for example, not in computing the position on the page).

Argument	Choice or Default uom	Default Value	Description
CHAR-WIDTH	points	7.2 pt	Size of each horizontal character column on the page (for example, the distance between the locations (1, 12) and (1, 13)). For the printer type LINEPRINTER, this value is used only to calculate the TOP-MARGIN and BOTTOM-MARGIN (not in computing the position on the page).

Description

DECLARE-LAYOUT describes the characteristics of a layout to use for an output file. A layout can be shared by more than one report. You can define as many layouts as are necessary for the requirements of the application. You can override the default layout attributes by defining a layout called DEFAULT in your program. Each layout name must be unique.

Production Reporting maps its line and column positions on the page by using a grid determined by the LINE-HEIGHT and CHAR-WIDTH arguments. That is, Production Reporting calculates the number of columns per row by dividing the LINE-WIDTH by the CHAR-WIDTH and calculates the number of lines by dividing the PAGE-DEPTH by the LINE-HEIGHT. Each printed segment of text is placed on the page using this grid. Because the characters in proportional fonts vary in width, it is possible that a word or string is wider than the horizontal space you have allotted, especially in words containing uppercase letters or bold characters. To account for this behavior, you can either move the column position in the PRINT or POSITION statements or indicate a larger CHAR-WIDTH in DECLARE-LAYOUT.

The ORIENTATION parameter selects the proper fonts. In addition, the parameter interacts with PAPER-SIZE as follows:

- When you do not specify ORIENTATION=LANDSCAPE or the PAPER-SIZE dimensions, Production Reporting creates a page with the dimensions set to 11 inch by 8.5 inch. This results in a page of 100 columns by 45 lines with 0.5 inch margins.
- When you specify PAPER-SIZE=(*paper_name*), the page orientation is set according to the *paper_name* specified. If you also specify ORIENTATION and the value differs from the PAPER-SIZE value, the ORIENTATION value overrides the PAPER-SIZE value.
- When you specify PAPER-SIZE=(*page_width*, *page_depth*), Production Reporting *does not* swap the page width and page depth if ORIENTATION=LANDSCAPE.

Note:

If none of the following commands are present in an Production Reporting report, none of the default values in [Table 31](#) take effect, and the report is created with 62 lines by 132 characters.

DECLARE-REPORT (Setup)

DECLARE-LAYOUT (Setup)

DECLARE-PRINTER(Setup)

DECLARE-PROCEDURE (Setup)

DECLARE-TOC (Setup)

USE-REPORT (Body)

USE-PROCEDURE (Body)
USE-PRINTER-TYPE (Body)
TOC-ENTRY (Body)
ALTER-PRINTER (Body)
BEGIN-HEADING For-Tocs=() (Construct)
BEGIN-FOOTING For-Tocs=() (Construct)
BEGIN-HEADING For-Reports=() (Construct)
BEGIN-FOOTING For-Reports=() (Construct)

Examples

The following example illustrates the ability to specify parameters using a metrics measurement system. (The syntax results in a paper size of 210mm by 297mm, a top margin of 12.7mm, a left margin of 12.7mm, a right margin of 25.4mm, a bottom margin of 12.7mm, a portrait orientation, 67 columns, and 63 lines.)

```
declare-layout my-layout
  paper-size=(a4)
  left-margin=12.7 mm
  right-margin=25.4 mm
end-declare
```

The following example changes the page dimensions. It also changes the left and right margins to one inch. (The syntax results in a paper size of 14 inches by 11 inches, a top margin of 0.5 inches, a left margin of one inch, a right margin of one inch, a bottom margin of 0.5 inches, a portrait orientation, 120 columns, and 60 lines.)

```
declare-layout large-paper
  paper-size=(14, 11)
  left-margin=1
  right-margin=1
end-declare
```

The following example retains the default page dimensions and changes the left and right margins to one inch. (The syntax results in a paper size of 8.5 inches by 11 inches, a top margin of 0.5 inches, a left margin of one inch, a right margin of one inch, a bottom margin of 0.5 inches, a portrait orientation, 65 columns, and 60 lines.)

```
declare-layout default
  left-margin=1
  right-margin=1
end-declare
```

The following example changes the orientation to landscape. The columns and rows are recalculated. All other values remain the same. (The syntax results in a paper size of 11 inches by 8.5 inches, a top margin of 0.5 inches, a left margin of 0.5 inches, a right margin of 0.5 inches, a bottom margin of 0.5 inches, 100 columns and 45 lines.)

```
declare-layout default
  orientation=landscape
```



```
end-declare
```

The following example changes the orientation to landscape. In addition the top margin is set to one inch. (The syntax results in a paper size of 11 inches by 8.5 inches, a top margin of 0.5 inches, a left margin of 0.5 inches, a right margin of 0.5 inches, a bottom margin of 0.5 inches, a landscape orientation, 100 columns and 43 lines.)

```
declare-layout my_landscape
  orientation=landscape
  top-margin=1
end-declare
```

The following example specifies the page dimensions using a predefined name. Note that the orientation changes since this example is an envelope. (The syntax results in a paper size of 4.125 inches by 9.5 inches, a top margin of 0.5 inches, a left margin of 0.5 inches, a right margin of 0.5 inches, a bottom margin of 0.5 inches, a landscape orientation, 85 columns and 18 lines.)

```
declare-layout envelope
  paper-size=(com-10)
end-declare
```

See Also

[DECLARE-REPORT](#)

DECLARE-PRINTER

Function

Overrides the printer defaults for specified printer type.

Syntax

```
DECLARE-PRINTER printer_name
[FOR-REPORTS=(report_name1[,report_namei]...)]
[TYPE=printer_type_lit]
[INIT-STRING=initialization_string_txt_lit]
[RESET-STRING=reset_string_txt_lit]
[COLOR=color_lit]
[POINT-SIZE=point_size_num_lit]
[FONT-TYPE=font_type_int_lit]
[SYMBOL-SET=symbol_set_id_lit]
[STARTUP-FILE=file_name_txt_lit]
[PITCH=pitch_num_lit]
[FONT=font_int_lit]
[BEFORE-BOLD=before_bold_string_txt_lit]
[AFTER-BOLD=after_bold_string_txt_lit]
END-DECLARE
```

Arguments

printer_name

Unique name used to reference a printer definition and its attributes.

Table 32 describes the other DECLARE-PRINTER arguments.

Description

Each printer has a set of defaults in Table 32. DECLARE-PRINTER overrides these defaults.

Use DECLARE-PRINTER in the SETUP section to define the characteristics of the printer or printers to use. If you need to change some of the arguments depending on the run-time environment, you can use ALTER-PRINTER in any part of the program except the PROGRAM and SETUP sections.

A program can contain no more than one DECLARE-PRINTER command for each printer type for each report. If you do not provide a printer declaration, the default specifications are used. The default printer attributes can be overridden by providing a DECLARE-PRINTER specification for each printer. Their names are: DEFAULT-LP for line printer, DEFAULT-HP for HP LaserJet, DEFAULT-HT for HTML, and DEFAULT-PS for PostScript.

Table 32 describes each of the arguments, the possible choices, and the default values.

Table 32 DECLARE-PRINTER Command Arguments

Argument	Choice or Measure	Default	Description
AFTER-BOLD	any string	(none)	See BEFORE-BOLD.
BEFORE-BOLD	any string	(none)	<p>BEFORE-BOLD and AFTER-BOLD are for line printers only. They specify the character string to turn bolding on and off. If the string contains blank characters, enclose it in single quotes ('...').</p> <p>To specify non-printable characters, such as ESC, enclose the decimal value inside angle brackets as follows:</p> <pre>BEFORE-BOLD=<27>[r ! Turn on bold AFTER-BOLD=<27>[u ! Turn it off</pre> <p>These arguments work with the BOLD argument of PRINT.</p>
COLOR	Yes, No	No	Defines whether the printer can print in color.
FONT	font_number	3	<p>Font number of the typeface to use. For HP LASERJET printers, this is the typeface value as defined by Hewlett-Packard. For a complete list of the typeface numbers, see the HP LaserJet <i>Technical Reference Manual</i>. For POSTSCRIPT printers, Production Reporting supplies a list of fonts and arbitrary font number assignments in the file POSTSCRI.STR.</p> <p>The font numbers are the same as those for HP LaserJet printers, wherever possible. You can modify the font list in POSTSCRI.STR to add or delete fonts. Read the POSTSCRI.STR file for</p>

Argument	Choice or Measure	Default	Description
			instructions. Table 33 lists the fonts available in Production Reporting internally. This table lists the fonts available in the Production Reporting POSTSCRI.STR file.
FONT-TYPE	PROPORTIONAL, FIXED	Depends on the font	Applies only to HP LASERJET printers and needs to be specified only for font types not defined in Table 33 .
FOR-REPORTS		ALL	Name of the reports that use this printer definition (default = ALL). Required only for programs with multiple reports. Ignore this argument for programs that produce a single report..
INIT-STRING		(none)	Sends control or other characters to the printer at the beginning of the report. Designed primarily for line printers and has limited use with other printer types. Specify non-display characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character.
PITCH	characters/inch	10	Required for HPLASERJET printers and SPF Viewer. Fixed–pitched fonts should indicate the pitch.
POINT-SIZE	points	12	Beginning size of the font. Does not apply to line printers
RESET-STRING		(none)	Sends control or other characters to the printer at the end of the report. Designed primarily for line printers and has limited use with other printer types. Specify non-display characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character.
STARTUP-FILE	filename	POSTSCRI.STR	POSTSCRIPT printers only. Defines an alternate startup file. Unless otherwise specified, the default startup file is located in the directory pointed to by the environment variable SQDIR.
SYMBOL-SET	HP defined sets	OU	HP LASERJET printers only. The default value of “OU” is for the ASCII symbol set. For a complete list of the symbol sets, see the HP LaserJet <i>Technical Reference Manual</i> .
TYPE	LINEPRINTER, POSTSCRIPT, HPLASERJET, HTML, LP, PS, HP, HT	LP	Production Reporting creates output specific to each printer. <ul style="list-style-type: none"> ● LINEPRINTER (LP) files generally consist of ASCII characters and can be viewed by a text editor. ● POSTSCRIPT (PS) files consist of ASCII characters, but you need to know

Argument	Choice or Measure	Default	Description
			PostScript to understand what is shown on the printer. <ul style="list-style-type: none"> ● HPLASERJET (HP) files are binary files and cannot be edited or viewed. ● HTML (HT) files consist of ASCII characters and can be viewed by a browser.

[Table 33](#) lists the fonts available in Production Reporting for use with the `FONT` argument for HP LaserJet printer types.

Table 33 Fonts Available for HP LaserJet Printers in Production Reporting

Value	Typeface	Style
0	Line printer	Fixed
1	Pica	Fixed
2	Elite	Fixed
3	Courier	Fixed
4	Helvetica	Proportional
5	Times Roman	Proportional
6	Letter Gothic	Fixed
8	Prestige	Fixed
11	Presentations	Fixed
17	Optima	Proportional
18	Garamondi	Proportional
19	Cooper Black	Proportional
20	Coronet Bold	Proportional
21	Broadway	Proportional
22	Bauer Bodini Black Condensed	Proportional
23	Century Schoolbook	Proportional
24	University Roman	Proportional

The font you choose—in orientation, typeface, and point size—must be an internal font, available in a font cartridge, or downloaded to the printer.

For fonts not listed in [Table 33](#), indicate the font style using the `FONT-TYPE` argument, or the correct typeface cannot be selected by the printer.

Table 34 lists the fonts available in Production Reporting for use with the FONT argument for PostScript printer types. Those for which bold face types are available are indicated by a “Y” in the Bold column.

Table 34 Fonts Available for PostScript Printers

Value	Typeface	Bold
3	Courier	Y
4	Helvetica	Y
5	Times Roman	Y
6	Avant Garde Book	
8	Palatino Roman	Y
11	Symbol	
12	Zapf Dingbats	
17	Zapf Chancery Medium Italic	
18	Bookman Light	
23	New Century Schoolbook Roman	Y
30	Courier Oblique	Y
31	Helvetica Oblique	Y
32	Times Italic	Y
33	Avant Garde Demi	
34	Avant Garde Book Oblique	
35	Avant Garde Demi Oblique	
36	Palatino Oblique	Y
37	New Century Schoolbook Italic	Y
38	Helvetica Narrow	Y
39	Helvetica Narrow Oblique	Y
40	Bookman Demi	
41	Bookman Light Italic	
42	Bookman Demi Italic	

Other type faces can be added to POSTSCRI.STR.

Table 35 lists the fonts available in Production Reporting when printing on Windows printer drivers using -PRINTER:WP. When you use -PRINTER:WP, your report is sent directly to the

default Windows printer. To specify a non-default Windows printer, use `-PRINTER:WP:{Printer Name}`. The `{Printer Name}` can be the name assigned to a printer; or, if the operating system permits it, the UNC name (i.e. `\\Machine\ShareName`). For example, to send output to a Windows printer named *NewPrinter*, you could use `-PRINTER:WP:NewPrinter`. If your printer name has spaces, enclose the entire command in double quotes.

Fonts are specified in the `ALTER-PRINTER FONT` qualifier by their number.

Table 35 Fonts Available for Windows Printers

Value	Windows Font/Name	Style
3	Courier New	Fixed
300	Courier New	Bold
4	Arial	Proportional
400	Arial	Bold
5	Times New Roman	Proportional
500	Times New Roman	Bold
6	AvantGarde	Proportional
8	Palatino	Proportional
800	Palatino	Bold
11	Symbol	Proportional

Note:

Fonts 6, 8, and 800 are not supplied with Windows. You can get these fonts by purchasing the ADOBE Type Manager (ATM). The advantage of using ATM fonts is the compatibility for PostScript printer fonts.

The Symbol font uses the `SYMBOL_CHARSET` instead of the usual `ANSI_CHARSET` character set.

To add more fonts, edit the `[Fonts]` section in `SQR.INI`.

Examples

```
declare-printer HP-definition ! Default HP definition
  type=HP                    ! for all reports
  font=4                     ! Helvetica
  symbol-set=12U             ! PC-850 Multilingual
end-declare
```

```
declare-printer PS-Sales ! PS definition
  for-reports=(sales)     ! for the Sales report
  type=PS
  font=5                  ! Times-Roman
```

end-declare

See Also

[ALTER-PRINTER](#) and [DECLARE-REPORT](#)

DECLARE-PROCEDURE

Function

Declares procedures triggered when a specified event occurs.

Syntax

```
DECLARE-PROCEDURE  
[FOR-REPORTS=(report_name1[,report_namei]...)]  
[BEFORE-REPORT=procedure_name[(arg1[,argi]...)]]  
[AFTER-REPORT=procedure_name[(arg1[,argi]...)]]  
[BEFORE-PAGE=procedure_name[(arg1[,argi]...)]]  
[AFTER-PAGE=procedure_name[(arg1[,argi]...)]]  
END-DECLARE
```

Arguments

FOR-REPORTS

Reports that use the given procedures. Required only for programs with multiple reports.

BEFORE-REPORT

Procedure executed at the time of the execution of the first command which causes output to be generated (PRINT). It can be used, for example, to create a report heading.

AFTER-REPORT

Procedure executed just before the report file is closed at the end of the report. It can be used to print totals or other closing summary information. If no report was generated, the procedure does not execute.

BEFORE-PAGE

Procedure executed at the beginning of every page, just before the first output command for the page. It can be used, for example, to set up page totals.

AFTER-PAGE

Procedure executed just before each page is written to the file. It can be used, for example, to display page totals.

You can optionally specify arguments to pass to any of the procedures. Arguments can be any variable, column, or literal.

Description

DECLARE-PROCEDURE can be used to define Production Reporting procedures to invoke before or after a report is printed or before the beginning or end of each page.

Issue DECLARE-PROCEDURE in the SETUP section. For multiple reports, you can use the command as often as required to declare procedures required by all the reports. If you issue multiple DECLARE-PROCEDURE commands, the last one takes precedence. In this way, you can use one command to declare common procedures for ALL reports and others to declare unique procedures for individual reports. The referenced procedures can accept arguments.

If no FOR-REPORTS is specified, ALL is assumed. Initially, the default for each of the four procedure types is NONE. If a procedure is defined in one DECLARE-PROCEDURE for a report, that procedure is used unless NONE is specified.

Use the USE-PROCEDURE command to change the procedures to use at run-time. To turn a procedure off, specify NONE in the USE-PROCEDURE statement.

Examples

```
declare-procedure          ! These procedures will
  before-report=report_heading ! be used by all reports
  after-report=report_footing
end-declare
```

```
declare-procedure          ! These procedures will
  for-reports=(customer)    ! be used by the customer
  before-page=page_setup    ! report
  after-page=page_totals
end-declare
```

See Also

[USE-PROCEDURE](#)

DECLARE-REPORT

Function

Defines reports and their attributes.

Syntax

```
DECLARE-REPORT report_name
[TOC=toc_name]
[LAYOUT=layout_name]
[PRINTER-TYPE=printer_type]
END-DECLARE
```

Arguments

report_name

Report name.

TOC

Name of the Table of Contents.

LAYOUT

Layout name. If none is specified, the default layout is used.

PRINTER-TYPE

Type of printer. If none is specified, the default is the `LINEPRINTER`. If no `DECLARE-PRINTER` is specified, `DEFAULT-LP` is used. Valid values for `PRINTER-TYPE` are `HT`, `HP`, `PD`, `PS`, `LP`, `HTML`, `HPLASERJET`, `POSTSCRIPT`, and `LINEPRINTER`.

Description

Issue `DECLARE-REPORT` in the `SETUP` section.

You can use `DECLARE-REPORT` to declare one or more reports to be produced in the application.

You must use this command when developing applications to produce more than one report.

Multiple reports can share the same layout and the same printer declarations or each report can use its own layout or printer definitions if the report has unique characteristics.

When you are printing multiple reports, unless report names are specified using the `-F` command-line flag, the first report declared is generated with the name of *program.lis*, where *program* is the application name.

Additional reports are generated with names conforming to the rules dictated by the `SQR.INI OUTPUT-FILE-MODE` setting.

When the `-KEEP` or `-NOLIS` flags are used, the first intermediate print file (SPF file) is generated with a name of *program.spf* and additional reports are generated with names conforming to the rules dictated by the `SQR.INI OUTPUT-FILE-MODE` setting.

Examples

```
declare-layout customer_layout
  left-margin
  right-margin
end-declare
```

```
declare-layout summary_layout
  orientation=landscape
end-declare
```

```
declare-report customer_detail
  toc=detailed
  layout=customer_layout
  printer-type=postscript
end-declare
```

```
declare-report customer_summary
  layout=summary_layout
  printer-type=postscript
end-declare
```

```

.
.
.
use-report customer_detail
...print customer_detail...
use-report customer_summary
...print customer_summary...

```

See Also

[USE-REPORT](#), [DECLARE-LAYOUT](#), [DECLARE-PRINTER](#), and [DECLARE-TOC](#)

DECLARE-TABLE

Function

Defines a template for a table.

Syntax

```

DECLARE-TABLE table_template_name
COLUMN-COUNT=number_of_columns
[COLUMN-ATTRIBUTES=({column number},{keyword1},{value1}, ..., {keywordn},
{valuen})]
[ROW-ATTRIBUTES=({keyword1},{value1}, ..., {keywordn},{valuen})]
[TABLE-ATTRIBUTES=({keyword1},{value1}, ..., {keywordn},{valuen})]

```

Arguments

table_template_name

Name of the table template. Valid values include alphanumeric characters (A-Z, 0–9), underscore (_), and dash (-). Do not use the reserved word NONE.

COLUMN-COUNT

Number of columns in the table.

COLUMN-ATTRIBUTES

Attributes to apply to column cells.

Table 36 Column Attributes

Attribute	Description
BACKGROUND	Background color name or RGB triplet. Default=NONE
BOLD	YES NO
CENTER	YES NO
FILL-COLOR	Fill color name or RGB triplet. Default=NONE

Attribute	Description
FONT	Font number. Cannot specify a default value
FOREGROUND	Foreground color name or RGB triplet. Default=BLACK
ITALIC	YES NO
LEADING	Expressed in decipoints
LINE-COLOR	Color name or RGB triplet of column line (line after column). Default=NONE (no line)
LINE-STYLE	Column line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT, LONG-DASH-DOT-DOT). Default=SOLID
LINE-THICKNESS	Thickness of the column line expressed in decipoints. Default=two decipoints
POINT-SIZE	Point size of the font. Cannot specify a default value.
UNDERLINE	YES NO
WIDTH	Width expressed in coordinate units. Cannot specify a default value
WRAP	YES NO maximum number of lines
WRAP-HEIGHT	Number of lines between each wrapped line. Default=one line
WRAP-ON	Characters on which to force a WRAP. The default is not to force a WRAP

ROW-ATTRIBUTES

Attributes to apply to rows.

Table 37 Row Attributes

Attribute	Description
BORDER-COLOR	Color name or RGB triplet. Default=NONE (no border)
BORDER-LINE-STYLE	The border line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT, LONG-DASH-DOT-DOT). Default=SOLID
BORDER-THICKNESS	Border thickness expressed in decipoints. Default=two decipoints
FILL-COLOR	Fill color name or RGB triplet. Default=NONE
HEIGHT	Number of lines between each printed row. Default=one line
LINE-COLOR	Color name or RGB triplet. Default=NONE (no line)
LINE-STYLE	Row line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT, LONG-DASH-DOT-DOT). Default=SOLID
LINE-THICKNESS	Line thickness expressed in decipoints. Default=two decipoints

TABLE-ATTRIBUTES

Attributes for the appearance of the table.

Table 38 Table Attributes

Attribute	Description
BORDER-COLOR	Color name or RGB triplet. Default=NONE (no border)
BORDER-LINE-STYLE	The border line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT, LONG-DASH-DOT-DOT). Default=SOLID
BORDER-THICKNESS	Border thickness expressed in decipoints. Default=two decipoints
FILL-COLOR	Fill color name or RGB triplet. Default=NONE
LEADING	Expressed in decipoints. Default=0 decipoints

Description

Use `DECLARE-TABLE` in the `BEGIN-SETUP` section to define a template for a table.

Example

```
declare-table template4
  column-count=3
  column-attributes=(0, 'line-color', ('blue'))
  column-attributes=(2, 'point-size', 16, 'font', 4,
    'bold', 'No', 'italic', 'yes', 'center', 'yes',
    'fill-color', ('red'))
  column-attributes=(3, 'point-size', 8, 'font', 300,
    'bold', 'YES', 'italic', 'yes', 'center', 'yes')
  row-attributes=('line-color', ('green'), 'height', 5)
end-declare
```

See Also

[ALTER-TABLE](#), [CREATE-TABLE](#), [DUMP-TABLE](#), [FILL-TABLE](#), [PRINT-TABLE](#),

DECLARE-TOC

Function

Defines the Table of Contents and its attributes.

Syntax

```
DECLARE-TOC toc_name
[FOR-REPORTS=(report_name1[,report_namei]...)]
[DOT-LEADER=YES|NO]
[INDENTATION=position_count_num_lit]
[BEFORE-TOC=procedure_name[(arg1[,argi]...)]]
[AFTER-TOC=procedure_name[(arg1[,argi]...)]]
[BEFORE-PAGE=procedure_name[(arg1[,argi]...)]]
[AFTER-PAGE=procedure_name[(arg1[,argi]...)]]
```

```
[ENTRY=procedure-name [(argi [,argi] ...)]]  
END-DECLARE
```

Arguments

toc_name

Name of the Table of Contents.

FOR-REPORTS

One or more reports that use this Table of Contents.

DOT-LEADER

Whether a dot leader precedes the page number. The default is NO.

INDENTATION

Number of spaces to indent each level. The default is 4.

BEFORE-TOC

Procedure executed before generating the Table of Contents. If no Table of Contents is generated, the procedure does not execute.

AFTER-TOC

Procedure executed after generating the Table of Contents. If no Table of Contents is generated, the procedure does not execute.

BEFORE-PAGE

Procedure executed at the start of every page.

AFTER-PAGE

Procedure executed at the end of each page.

ENTRY

Procedure that is executed to process each Table of Contents entry (instead of Production Reporting doing it for you). When this procedure is invoked, the following Production Reporting-reserved variables are populated with data about the TOC entry:

#SQR-TOC-LEVEL

Contains the level

\$SQR-TOC-TEXT

Contains the text

#SQR-TOC-PAGE

Contains the page number

These are global variables. If the procedure is local, you must precede it with an underscore (for example, `#_sqr-toc-page`). These three Production Reporting-reserved variables are only valid within the scope of the ENTRY procedure. They can be referenced outside the scope, but their contents are undefined.

Description

Use `DECLARE-TOC` in the `SETUP` section.

You can use `DECLARE-TOC` to declare one or more Table of Contents for the application.

A Table of Contents can be shared between reports.

Example

```
begin-setup
  declare-toc common
    for-reports=(all)
    dot-leader=yes
    indentation=2
  end-declare
end-setup
.
.
.
toc-entry level=1 text=$Chapter
toc-entry level=2 text=$Heading
.
.
```

See Also

[BEGIN-FOOTING](#), [BEGIN-HEADING](#), [DECLARE-REPORT](#), and [TOC-ENTRY](#)

DECLARE-VARIABLE

Function

Explicitly declares a variable type.

Syntax

```
DECLARE-VARIABLE
[DEFAULT-NUMERIC={DECIMAL[(prec_lit)]|FLOAT|INTEGER}]
[DECIMAL[(prec_lit)]num_var[(prec_lit)] [num_var[(prec_lit)]...]
[FLOAT num_var[num_var]...]
[DATE date_var[date_var]...]
[INTEGER num_var[num_var]...]
[TEXT string_var[string_var]...]
[BINARY binary_var[binary_var]...]
END-DECLARE
```


DECLARE-VARIABLE allows the user to determine the type of variables to use to fit their needs. This command can only appear in the SETUP section or as the first statement of a local procedure. The placement of the command affects its scope. When used in the SETUP section, it affects all variables in the entire program. Alternately, when it is placed in a local procedure, its effect is limited to the scope of the procedure. If the command is in both places, the local declaration takes precedence over the SETUP declaration.

In addition to declaring variables, the command allows the default numeric type to be specified using the DEFAULT-NUMERIC setting as FLOAT, INTEGER, or DECIMAL. When dealing with money or where more precision is required, you can use the DECIMAL qualifier.

DECLARE-VARIABLE, the -DNT command-line flag, and the DEFAULT-NUMERIC setting in SQR.INI affects the way numeric literals are typed. If V30 is specified, then all numeric literals are FLOAT (just as in pre-version 4.0 releases); otherwise, the use or lack of a decimal point determines the type of the literal as either FLOAT or INTEGER, respectively. Finally, not specifying DECLARE-VARIABLE, the -DNT command-line flag, and the DEFAULT-NUMERIC setting in SQR.INI is the same as specifying V30.

Note:

In Production Reporting DDO, list variables should not be declared using this construct.

Example

```
begin-setup
  declare-variable
    default-numeric=float
    decimal #decimal(10)
    integer #counter
    date $date
  end-declare
end-setup
.
.
let $date = strtodate('Jan 01 1995', 'Mon DD YYYY')
print $date (1,1)
position (+2,1)

let #counter = 0
while #counter < 10
  let #decimal = sqrt(#counter)
  add 1 to counter
  print #decimal (+1,1) 9.999999999
end-while

do sub1($date, 'day', 10)

do sub2

.
.
begin-procedure sub1(:$dvar, $units, #uval)
  declare-variable
    date $dvar
```



```

        integer #uval
    end-declare
    let $dvar = dateadd($dvar, $units, #uval)
    print $dvar (+1,1)
    position (+2,1)
end-procedure
.
.
begin-procedure sub2 LOCAL
    declare-variable
        date $mydate
    end-declare
    let $mydate = dateadd($_date, 'year', 5)
    print $mydate (+1,1)
    position (+2,1)
end-procedure
.
.

```

See Also

- The `-DNT` command-line flag, described in [Chapter 1, “Introduction.”](#)
- The [Default-Settings] section of SQR.INI described in [Chapter 6, “SQR.INI.”](#)

#DEFINE

Function

Declares a value for a substitution variable within the body of the report (rather than using ASK).

Syntax

```
#DEFINE substitution_variable value
```

Arguments

substitution_variable

Variable to use as the substitution variable. The substitution variable is used to substitute any command, argument, or part of a SQL statement at compile time.

value

Value to substitute.

Description

#DEFINE is useful for specifying constants such as column locations, printer fonts, or any number or string that is used in several locations in the program. When the value of the number or string must be changed, you need only change your #DEFINE command. All references to that variable change automatically, which makes modifying programs much simpler.

If `ASK` is used to obtain the value of a substitution variable that has already been defined, `ASK` uses the previous value and the user is not prompted. This gives you the flexibility of being able to predefine some variables and not others. When the report runs, `ASK` requests values for only those variables that have not had a value assigned.

You can use `#DEFINE` commands inside an include file. This is a method of gathering commonly used declarations into one place, and reusing them for more than one report.

The *value* in the `#DEFINE` command can have embedded spaces, and needs no enclosing quotes. The entire string is used as is.

The `#DEFINE` command cannot be broken across program lines.

Examples

The following code defines several constants:

```
#define page_width 8.5
#define page_depth 11
#define light LS^10027
#define bold LS^03112
#define col1 1
#define col2 27
#define col3 54
#define order_by state, county, city, co_name
```

The following excerpt from a report uses the preceding definitions:

```
begin-setup
  declare-printer contacts
    type=hp
    paper-size=({page_width}, {page_depth})
  end-declare
end-setup

begin-heading 5
  print 'Company Contacts' (1,1) center
  print 'Sort: {order_by}' (2,1) center
  print 'Company' (4,{col1})
  print 'Contact' (4,{col2})
  print 'Phone' (4,{col3})
end-heading

begin-procedure main
  begin-select
    company (1,{col1})
    print '{bold}' (0,{col2}) ! Print contact in boldface.
    contact ()
    print '{light}' () ! Back to lightface.
    phone (0,{col3}) ! Note:There must be enough
    next-listing ! space between col2
    from customers ! and col3 for both
    order by {order_by} ! font changes and the
  end-select ! contact field.
end-procedure
```

See Also

[ASK](#)

DISPLAY

Function

Displays the specified column, variable, or literal.

Syntax

```
DISPLAY {any_lit|_var|_col}  
[[:$]edit_mask|NUMBER|MONEY|DATE] [NOLINE]
```

Arguments

any_lit|*_var*|*_col*

Text, number, or date to display.

edit_mask

Edits the field before displaying it. (see [“Edit Masks” on page 247](#))

NUMBER

Formats *any_lit*|*_var*|*_col* with the NUMBER-EDIT-MASK of the current locale. This option is not legal with date variables.

MONEY

Formats *any_lit*|*_var*|*_col* with the MONEY-EDIT-MASK of the current locale. This option is not legal with date variables.

DATE

Formats *any_lit*|*_var*|*_col* with the DATE-EDIT-MASK of the current locale. This option is not legal with numeric variables. If DATE-EDIT-MASK has not been specified, then the date is displayed using the default format for that database (see [Table 61 on page 251](#)).

NOLINE

Suppresses the carriage return after displaying the field.

Description

DISPLAY can display data to a terminal. The data is displayed to the current location on the screen. If you wish to display more than one field on the same line, use NOLINE on each display except the last.

Dates can be contained in a date variable or column, or a string literal, column, or variable. When a date variable or column is displayed without an edit mask, the date appears in the following manner:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by `SQR_DB_DATE_FORMAT`. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,” on page 251](#).
- For DATE columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If this is not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252](#).
- For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252](#).

When displaying a date in a string literal, column, or variable using `EDIT` or `DATE`, the string uses the format specified by `SQR_DB_DATE_FORMAT`, one of the database-dependent formats in [Table 61 on page 251](#), or the database-independent format `YYYYMMDD[HH24[MI[SS[NNNNNN]]]]`.

If you require more control over the display, use `SHOW`.

Examples

The following segments illustrate the various features of `DISPLAY`:

The following code:

```
!  
! Display a string using an edit mask  
!  
display '123456789' xxx-xx-xxxx
```

Produces the following output:

```
123-45-6789
```

The following code:

```
!  
! Display a number using an edit mask  
!  
display 1234567.89 999,999,999.99
```

Produces the following output:

```
1,234,567.89
```

The following code:

```
!  
! Display a number using the default edit mask (specified in SQR.INI)  
!  
display 123.78
```

Produces the following output:

123.780000

The following code:

```
!  
! Display a number using the locale default numeric edit mask  
!  
alter-locale number-edit-mask = '99,999,999.99'  
display 123456.78 number
```

Produces the following output:

123,456.78

The following code:

```
!  
! Display a number using the locale default money edit mask  
!  
alter-locale money-edit-mask = '$$,$$$,$$9.99'  
display 123456.78 money
```

Produces the following output:

\$123,456.78

The following code:

```
!  
! Display a date column using the locale default date edit mask  
!  
begin-select  
dcol  
  from tables  
end-select  
alter-locale date-edit-mask = 'DD-Mon-YYYY'  
display &dcol date
```

Produces the following output:

01-Jan-1999

The following code:

```
!  
! Display two values on the same line  
!  
display 'Hello' noline  
display ' World'
```

Produces the following output:

Hello World

The following code:

```
!  
! Display two values on the same line with editing of the values  
!
```

```
alter-locale money-edit-mask = '$$, $$$, $$9.99'  
let #taxes = 123456.78  
display 'You owe ' noline  
display #taxes money noline  
display ' in back taxes.'
```

Produces the following output:

```
You owe $123,456.78 in back taxes.
```

See Also

- [SHOW](#) for information on screen control
- [LET](#) for information on copying, editing, or converting fields
- The `EDIT` parameter of [PRINT](#) for edit mask descriptions
- [ALTER-LOCALE](#) for descriptions of `NUMBER-EDIT-MASK`, `MONEY-EDIT-MASK`, and `DATE-EDIT-MASK`

DIVIDE

Function

Divides one number into another.

Syntax

```
DIVIDE {src_num_lit|_var|_col} INTO dst_num_var  
[ON-ERROR={HIGH|ZERO}] [ROUND=nn]
```

Arguments

src_num_lit|*_var*|*_col*

Divided into the contents of *dst_num_var*.

dst_num_var

Result after execution.

ON-ERROR

Sets the result to the specified number when a division by zero is attempted. If `ON-ERROR` is omitted and a division by zero is attempted, Production Reporting halts with an error message.

ROUND

Rounds the result to the specified number of digits to the right of the decimal point. For float variables, this value can be from 0 to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate.

Description

The source field is divided into the destination field and the result is placed in the destination. The source is always first, the destination always second.

When dealing with money-related values (dollars and cents), use decimal variables rather than float variables. Float variables are stored as double precision floating point numbers, and small inaccuracies can appear when dividing many numbers in succession. These inaccuracies can appear due to the way different hardware and software implementations represent floating point numbers.

Examples

```
divide 37.5 into #price ! #price / 37.5
divide &rate into #tot on-error=high
divide #j into #subtot on-error=zero
```

Note:

High in the preceding example is the “Maximum Value,” while zero is the “Lowest Value.”

See Also

- [LET](#) for a discussion of complex arithmetic expressions
- [ADD](#)

DO

Function

Invokes the specified procedure.

Syntax

```
DO procedure_name [arg1 [, argi] ...]
```

Arguments

procedure_name

Name of the procedure to execute.

arg1 [, *argi*]

Arguments to pass to the procedure. Arguments can be any type of variable or constant value.

Description

When the procedure ends, processing continues with the command following DO. You can use arguments to send values to or receive values from a procedure.

Arguments passed by DO to a procedure must match in number:

- Database text columns, string variables, and literals can be passed to procedure string or date arguments.
- Database numeric columns, numeric variables, and numeric literals can be passed to procedure numeric arguments.
- Numeric variables (DECIMAL, INTEGER, FLOAT) can be passed to procedure numeric arguments without regard to the argument type of the procedure. Production Reporting automatically converts the numeric values upon entering and leaving the procedure as required.
- Date variables can be passed to procedure date or string arguments.

When a field in DO receives a value back from a procedure (a colon indicates it is a back value—that is, a value that’s being returned), it must be a string, numeric, or date variable, depending on the procedure argument; however, a date can be returned to a string variable and vice versa.

When a date is passed to a string, the date is converted to a string according to the following rules:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by SQR_DB_DATE_FORMAT. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,” on page 251](#).
- For DATE columns, Production Reporting uses the format specified by SQR_DB_DATE_ONLY_FORMAT. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252](#).
- For TIME columns, Production Reporting uses the format specified by SQR_DB_TIME_ONLY_FORMAT. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252](#).

Examples

```
do get_names
do add_to_list ($name)
do print_list ('A', #total, &co_name, $name)
```

See Also

[BEGIN-PROCEDURE](#) for information on passing arguments

DRAW

Function

Draws an object.

Syntax

```
DRAW {position}
TYPE={type_lit|_var|_col}
```



```
[HEIGHT={height_lit_|_var|_col}]
[WIDTH={width_lit_|_var|_col}]
[RULE={rule_lit_|_var|_col}]
[FILL-COLOR=({color_name_lit_|_var|_col}|{rgb})]
[LINE-COLOR= {color_name_lit_|_var|_col}|{rgb})]
[CAP={cap_lit_|_var|_col}]
[LINE-STYLE={line_style_lit_|_var|_col}]
[END-POINT=(row_lit_|_var|_col, column_lit_|_var|_col)]
```

Arguments

position

Starting position of the object.

TYPE

Type of object to draw. The value can be BOX, HORZ-LINE, VERT-LINE, LINE, or OVAL. (LINE and OVAL are not supported with -PRINTER:LP, -PRINTER:HP, or PRINTER:PS.)

HEIGHT

Object's vertical length. Applies to BOX, VERT-LINE, and OVAL.

WIDTH

Object's horizontal length. Applies to BOX, HORZ-LINE, and OVAL.

RULE

Line thickness. Applies to BOX, HORZ-LINE, VERT-LINE, LINE, and OVAL. This value is expressed in decipoints. (There are 720 decipoints per inch.) The default value is two decipoints.

FILL-COLOR

Color used to fill the object. Applies to BOX and OVAL. The default value is the FILL COLOR value in SET-COLOR.

LINE-COLOR

Color of the lines in the object. Applies to BOX, HORZ-LINE, VERT-LINE, LINE, and OVAL. The default value is the LINE COLOR value in SET-COLOR.

CAP

Type of cap used with an object. Applies to BOX, HORZ-LINE, VERT-LINE, and LINE. BOX CAP values can ROUND, BEVEL, or MITER. HORZ-LINE, VERT-LINE, and LINE CAP values can be ROUND, SQUARE, or FLAT.








Note:

CAP settings are ignored with -PRINTER:HT and -PRINTER:EH. When using -PRINTER:WP, MITER is substituted for BEVEL. CAP is ignored unless the LINE-SYTTLE is set to SOLID.

LINE-STYLE

Line style of the object. Applies to `HORZ-LINE`, `VERT-LINE`, `LINE`, `BOX`, and `OVAL`.

Table 39 LINE-STYLE Values

Line Style	Value
SOLID	
SQUARE-DOT	
DASH	
DASH-DOT	
LONG-DASH	
LONG-DASH-DOT	
LONG-DASH-DOT-DOT	

Note:

LINE STYLE settings are ignored with `-PRINTER:LP`, `-PRINTER:HP`, or `PRINTER:PS`. If you use one of these flags, the default value of `SOLID` is applied to the line.

END-POINT

The object's ending coordinate. Applies to `LINE`.

Description

Draws the specified object on the page. For horizontal lines (`TYPE=HORZ-LINE`), the line is drawn just below the base of the line. For vertical lines (`TYPE=VERT-LINE`), the line is drawn just below the base of the starting line position to just below the base of the ending line position. After `DRAW` executes, Production Reporting changes the current print location to the starting location of the object. (This is different from the way `PRINT` works.)

Examples

Draw a wide box around the page:

```
DRAW (1,1) TYPE='BOX' WIDTH=78 HEIGHT=66 RULE=20
```

Draw a five-line shaded box without a border:

```
DRAW (1,1) TYPE='BOX' WIDTH=66 HEIGHT=5 RULE=0 FILL COLOR=(204,204,204)
```

Draw a line under the page heading:

```
DRAW (1,1) TYPE='HORZ LINE' WIDTH=66 RULE=10
```

Redline the paragraph:

```
DRAW (+3,+2) TYPE='VERT LINE' HEIGHT=4 RULE=6
```

Draw a blue circle with a red border (assumes that the layout line height and character width are the same value)

```
DRAW (1,1) TYPE='OVAL' HEIGHT=10 WIDTH=10 RULE=20 FILL COLOR=('BLUE') LINE  
COLOR=('RED')
```

Draw a box using a dashed line

```
DRAW (1,1) TYPE='BOX' WIDTH=78 HEIGHT=66 RULE=20 LINE STYLE='DASH'
```

See Also

[GET-COLOR](#), [PRINT](#), and [SET-COLOR](#)

DUMP-TABLE

Function

Dumps table data into an SPF file.

Syntax

```
DUMP-TABLE  
NAME=table_name_var|_lit|_col  
[CONTINUATION=continuation_var|_lit|_col]
```

Arguments

NAME

Name of the table created by CREATE-TABLE.

CONTINUATION

Defines whether the table data is a continuation of a previous DUMP-TABLE command. Valid values are YES and NO. The default is NO.

Description

Use DUMP-TABLE in any section except BEGIN-SETUP, BEGIN-SQL, and BEGIN-DOCUMENT. The data dumped into the SPF file will be in a format that the Enhanced HTML or Generic Driver can use to provide coherent output such as XML, BQD, and CSV.

Example

```
dump-table  
  name='customers'  
  continuation='yes'
```

See Also

[ALTER-TABLE](#), [CREATE-TABLE](#), [DECLARE-TABLE](#), [FILL-TABLE](#), [PRINT-TABLE](#)

#ELSE

Function

Compiles the code following #ELSE when a preceding #IF, #IFDEF, or #IFNDEF is FALSE. (#ELSE is a compiler directive that works with the #IF, #IFDEF, and #IFNDEF compiler directives.)

Syntax

```
#ELSE
```

See Also

[#IF](#), [#IFDEF](#), and [#IFNDEF](#) for descriptions of compiler directives

ELSE

Function

An optional command in IF.

Syntax

```
ELSE
```

See Also

[IF](#) for a description and example.

ENCODE

Function

Assigns a non-display or display character to a string variable.

Syntax

```
ENCODE src_code_string_lit INTO dst_txt_var
```

Arguments

src_code_string_lit

String of characters to encode and place in *dst_txt_var*.

dst_txt_var

Result after execution.

Description

ENCODE can define nondisplay characters or escape sequences sent to an output device. These characters or sequences can perform complex output device manipulations. ENCODE also displays characters not in the keyboard. If your keyboard does not have the Euro symbol, use the Encode feature to create a string variable for it.

The encode characters can be included in a report at the appropriate location using PRINT or PRINT-DIRECT.

Only values <001> to <255> can be defined in ENCODE.

Examples

```
encode '<27>L11233' into $bold      ! Code sequence to turn bold on.  
print $bold () code-printer=lp
```

See Also

- The chr function described in [Table 52 on page 212](#) under LET
- [PRINT](#) and [PRINT-DIRECT](#)
- “Encode Variables” in Volume 1 of the *Production Reporting User's Guide*

END-DECLARE, END-DOCUMENT, END-EVALUATE, END-FOOTING, END-HEADING

Function

Completes a section or paragraph.

Syntax

```
END-DECLARE  
END-DOCUMENT  
END-EVALUATE  
END-FOOTING  
END-HEADING
```

Description

END-DECLARE completes a paragraph started with:

- DECLARE-CHART
- DECLARE-IMAGE
- DECLARE-LAYOUT
- DECLARE-PRINTER
- DECLARE-PROCEDURE
- DECLARE-REPORT

- DECLARE-VARIABLE

Other *END-section* commands complete the corresponding *BEGIN-section* command:

- BEGIN-DOCUMENT
- EVALUATE
- BEGIN-FOOTING
- BEGIN-HEADING

Each command must begin on its own line.

Examples

```
begin-footing 2
  print 'Company Confidential' (1) center
end-footing
```

See Also

- DECLARE-paragraph
- BEGIN-section

#END-IF, #ENDIF

Function

Ends an #IF, #IFDEF, or #IFNDEF command. (#END-IF is a compiler directive.)

Syntax

```
#END-IF
```

Description

#ENDIF (without the dash) is a synonym for #END-IF.

Examples

```
#ifdef debuga
  show 'DebugA: #j = ' #j edit 9999.99
  show 'Cust_num = ' &cust_num
#end-if
```

See Also

[#IF](#), [#IFDEF](#), and [#IFNDEF](#) for a descriptions of compiler directives

END-IF

Function

Ends an `IF` command.

Syntax

```
END-IF
```

See Also

[IF](#)

END-PROCEDURE, END-PROGRAM, END-SELECT, END-SETUP, END-SQL, END-WHILE, END-EXECUTE

Function

Completes the corresponding section or paragraph.

Syntax

```
END-PROCEDURE  
END-PROGRAM  
END-SELECT  
END-SETUP  
END-SQL  
END-WHILE  
END-EXECUTE
```

Description

Each `END-section` command completes the corresponding `BEGIN-section` command:

- `BEGIN-PROCEDURE`
- `BEGIN-PROGRAM`
- `BEGIN-SELECT`
- `BEGIN-SETUP`
- `BEGIN-SQL`
- `WHILE`
- `BEGIN-EXECUTE`

Each command must begin on its own line.

Note:

END-EXECUTE (and BEGIN-EXECUTE) is only required when additional information about the datasource or query is needed, such as 'Connection', 'Schema', 'Command', 'GetData', 'Procedure', or 'Parameters'.

Examples

```
begin-program
  do main
end-program
```

See Also

- [BEGIN-*section*](#)
- [WHILE](#)

EVALUATE

Function

Determines the value of a column, literal, or variable and takes action based on that value.

Syntax

```
EVALUATE {any_lit|_var|_col}
```

This command is equivalent to case/switch in C or Java. The general format of EVALUATE is:

```
EVALUATE {any_lit|_var|_col}
WHEN comparison_operator {any_lit|_var|_col}
SQR_Commands...
[BREAK]
[WHEN comparison_operator {any_lit|_var|_col}
SQR_Commands...
[BREAK] ]
[WHEN-OTHER SQR_Commands...
[BREAK] ]
END-EVALUATE
```

Arguments

any_lit|*_var*|*_col*

A text or numeric column; a text, numeric, or date variable; or a text or numeric literal to use in the evaluation. In short, an evaluation argument.

comparison_operator

Any valid comparison operator. See comparison operators in [Table 45 on page 194](#).

WHEN

Evaluation expression. The evaluation argument is compared with the argument, beginning from the first `WHEN`. If the expression is `TRUE`, Production Reporting processes the commands after the `WHEN`. If the expression is `FALSE`, Production Reporting processes the next `WHEN` expression. Each `WHEN` must be on its own line.

If more than one `WHEN` expression appears directly before a set of commands, any one of them, if `TRUE`, causes the commands to execute.

`BREAK`

Immediately exits `EVALUATE`. Use `BREAK` at the end of a set of commands.

`WHEN-OTHER`

Signifies the start of default commands to process if all other `WHEN` expressions are `FALSE`. `WHEN-OTHER` must appear after all other `WHEN` expressions.

Description

`EVALUATE` is useful for branching to different commands depending on the value of a specified variable or column.

`EVALUATE` commands can be nested.

Evaluating a date variable or column with a string results in a date comparison (chronological, not a byte by byte comparison as is done for strings). The string must be in the proper format as follows:

- For `DATETIME` columns and Production Reporting `DATE` variables, Production Reporting uses the format specified by `SQR_DB_DATE_FORMAT`, one of the database-dependent formats (see [Table 61, “Default Formats by Database,” on page 251](#)), or the database-independent format `'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]'`.
- For `DATE` columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252](#).
- For `TIME` columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252](#).

Examples

```
evaluate &code
  when = 'A'
    move 1 to #j
    break
  when = 'B'
  when = 'C'
    move 2 to #j ! Will happen if &code is B or C.
    break
  when > 'D'
    move 3 to #j ! Move 3 to #j and continue checking.
  when > 'H'
    add 1 to #j ! Add 1 to #j and continue checking.
```

```

when > 'W'
    add 2 to #j
    break
when-other
    if isnull (&code)
        do null_code
    else
        move 0 to #j ! Unknown code.
    end-if
    break
end-evaluate

```

See Also

[IF](#) and [LET](#)

EXECUTE

Function

Executes a stored procedure. EXECUTE is available with the DB2, ODBC, Oracle, and Sybase versions of Production Reporting. (For DB2, Production Reporting does not support overloaded stored procedures.)

Syntax

```

EXECUTE [-XC] [ON-ERROR=procedure[(arg1[,argi]...)]]
[DO=procedure[(arg1[,argi]...)]]
{[@#status_var=]stored_procedure_name} |
{[@$return_var=]stored_procedure_name}
[[@param=]{any_col|_var|_lit} [OUTPUT] [, ...]]
[INTO any_coldata_type[(length_int_lit)]
[, ...]] [WITH RECOMPILE]

```

The syntax of EXECUTE roughly follows that of the Sybase Transact-SQL EXECUTE command, with the exception of optional arguments and the INTO argument.

Arguments

-XC

(Sybase only) Specifies that EXECUTE shares the same connection as the DO= procedure it can invoke. This argument is required to share Sybase temporary tables.

ON-ERROR

Production Reporting procedure to execute if an error occurs. If ON-ERROR is omitted and an error occurs, Production Reporting halts with an error message. For severe errors (for example, passing too few arguments) Production Reporting halts, even if an error procedure is specified.

You can specify arguments to pass to the ON-ERROR procedure. Arguments can be any variable, column, or literal.

DO

Production Reporting procedure to execute for each row selected in the query. Processing continues until all rows are retrieved.

You can specify arguments to pass to the procedure. Arguments can be any variable, column, or literal.

@#status_var

The procedure's status in the specified *numeric* variable. The status is returned only after selected rows are retrieved.

@\$return_var

(Oracle only) The called stored function's return value into the specified variable. Oracle stored functions can return any column data type. No procedure status is returned for Oracle stored procedures.

stored_procedure_name

Stored procedure or function to execute.

For Oracle installations only, you can add schema and package information to the stored procedure name as follows: *[[schema.][package.]].stored_procedure_name*.

@param

Parameter passed to the stored procedure. Parameters can be passed with or without names. If used without names, they must be listed in the same sequence as defined in the stored procedure.

any_lit|_var|_col

Value passed to the stored procedure. It can be a string, numeric, or date variable, a previously selected column, a numeric literal, or a string literal.

OUTPUT

Indicates that the parameter receives a value from the stored procedure. The parameter must be a string, numeric, or date Production Reporting variable. Output parameters receive their values only after rows selected have been retrieved. If you specify multiple output parameters, they must be in the same sequence as defined in the stored procedure.

INTO

Where to store rows retrieved from the stored procedure's *SELECT* statement. The *INTO* argument contains the names of the columns with data types and lengths (if needed). You must specify the columns in the same sequence and match the data type used in the stored procedure's *SELECT* statement.

[Table 40](#) lists the valid data types for each database.

Table 40 Valid Data Types

Database	Valid Data Types
Oracle	CHAR[(n)] DATE DECIMAL[(p[,s])] FLOAT[(b)] INTEGER LONG NCHAR[(n)] NVARCHAR2[(n)] NUMBER[(p[,s])] NUMERIC[(p[,s])] REAL ROWID SMALLINT VARCHAR[(n)] VARCHAR2[(n)]
ODBC	BIT TINYINT SMALLINT INT CHAR[(n)] NCHAR[(n)] VARCHAR[(n)] NVARCHAR[(n)] NTEXT TEXT REAL FLOAT[(b)] IMAGE SMALLMONEY MONEY DECIMAL[(p[,s])] NUMERIC [(p[,s])] SYSNAME SMALLDATETIME DATETIME TIMESTAMP

Database	Valid Data Types
	BINARY VARBINARY
Sybase	BIT TINYINT SMALLINT INT CHAR[(n)] NCHAR[(n)] VARCHAR[(n)] NVARCHAR[(n)] TEXT REAL FLOAT[(b)] IMAGE SMALLMONEY MONEY DECIMAL[(p[,s])] NUMERIC [(p[,s])] SYSNAME SMALLDATETIME DATETIME TIMESTAMP BINARY VARBINARY UNICHAR[(n)] UNIVARCHAR[(n)]
DB2	CHAR[(n)] VARCHAR[(n)] DATE TIME TIMESTAMP FLOAT DOUBLE NUMERIC DECIMAL[(p[,s])] INTEGER GRAPHIC[(n)]

Database	Valid Data Types
	VARGRAPHIC[(n)]

If the stored procedure contains more than one result set, only the first query is described with the INTO argument. Rows from subsequent queries are ignored.

WITH RECOMPILE

(Sybase and ODBC only) Causes the query to recompile each time it executes rather than using the plan stored with the procedure. Normally, this is not required or recommended.

Description

If the stored procedure specified in *stored_procedure_name* contains a SELECT query, EXECUTE must specify an INTO argument in order to process the values from the query. If no INTO argument is specified, then the values from the query are ignored.

EXECUTE retrieves just the first row when the following instances are true:

- The DO procedure is not specified.
- The stored procedure, *stored_procedure_name* selects one or more rows.
- An INTO argument is specified.

This is useful for queries returning a single row.

Note:

Oracle stored functions can return any column data type. ODBC, Sybase and DB2 can only return a numeric status.

Note:

If you are using Oracle or DB2 keep in mind the following:

- Oracle and DB2 can return multiple Result Sets of data; however, Production Reporting only processes the *first* Result Set returned from a stored procedure or function. After processing the first Result Set, all other Result Sets are ignored.
- When Oracle or DB2 encounters an INTO clause, an *implied* Result-Set handle is created. The implied Result-Set handle processes an open cursor returned from a stored procedure or function. The procedure or function is “described” to ensure that the stored object returns a handle to a result set. The data returned from the “describe” is then used to validate the data types declared for each column contained in the INTO clause of the EXECUTE command.

Examples

The following example invokes the stored procedure `get_total` with two parameters: a string literal and a string variable. The result from the stored procedure is stored in the variable `#total`.

```
execute get_total 'S. Q. Reporter' $State #Total Output
```

The following example invokes the stored procedure `get_products` with two parameters. The stored procedure selects data into five column variables. The Production Reporting procedure `print_products` is called for each row retrieved. The return status from the stored procedure is placed in the variable `#proc_return_status`.

```
execute do=print_products
  @#proc_return_status=
  get_products
  @prodcode=&code,@max=#maximum
  INTO &prod_code int,
      &description char(45),
      &discount float,
      &restock char,
      &expire_date datetime
begin-procedure print_products
  print &prod_code(+1,1)
  print &description(+5,45)
  print &discount(+5) edit 99.99
  print &restock(+5) match Y 0 5 Yes N 0 5 No
  print &expire_date(+5,) edit 'Month dd, yyyy'
end-procedure
```

EXIT-SELECT

Function

Exits a `SELECT` paragraph immediately.

Syntax

```
EXIT-SELECT
```

Description

Jumps to the command immediately following `END-SELECT`.

Use `EXIT-SELECT` when you need to end a query before all rows are retrieved.

Examples

```
begin-select
cust_num, co_name, contact, city, state, zip, employees
  add &employees to #tot_emps
  if #tot_emps >= 5000
    exit-select ! Have reached required total emps.
  end-if
  do print_company
from customers order by employees desc
end-select
```

See Also

[BEGIN-SELECT](#)

EXTRACT

Function

Copies a portion of a string into a string variable.

Syntax

```
EXTRACT {dst_txt_var|date_var} FROM  
{{src_txt_lit|_var|_col}|{src_date_var|_col}}  
{start_num_lit|_var}{length_num_lit|_var}
```

Arguments

dst_txt_var|*date_var*

Text or date variable into which the extracted string is placed.

{src_txt_lit|_var|_col}|*{src_date_var|_col}*

Text or date variable, column, or literal from which to extract the string.

start_num_lit|_var

Starting location of the string.

length_num_lit|_var

Length of the string.

Description

You must specify the starting location of the string as an offset from the beginning of the string and its length. An offset of zero (0) begins at the left-most character; an offset of 1 begins one character beyond that, and so on.

If the source is a date variable or column, it is converted to a string before the extraction according to the following rules:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses `SQR_DB_DATE_FORMAT`. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,” on page 251](#).
- For DATE columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252](#).
- For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252](#).

If the destination is a date variable, the string extracted from the source must be in one of the following formats:

- The format specified by `SQR_DB_DATE_FORMAT`.
- One of the database-dependent formats (see [Table 61, “Default Formats by Database,” on page 251](#))
- The database-independent format `'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'`.

Examples

```
extract $state from $record 45 2
extract $foo from "Oracle Rocks" 0 4 ! $foo='Oracle'
extract $zip_four from &zip 5 4
extract $rec from $tape_block #loc #rec_len
```

See Also

- The `substr` function described in [Table 52 on page 212](#) under `LET`
- [FIND](#)

FILL-TABLE

Function

Manipulates table attributes.

Syntax

```
FILL-TABLE
NAME=table_name_var|_lit|_col
VALUE=value_var|_lit|_col
LOCATION=(row_var|_lit|_col, column_var|_lit[,length_var|_lit])
[ATTRIBUTES=({keyword1}},{value1}, ..., {keywordn}},{valuen)]
```

Arguments

NAME

Name of the table created by `CREATE-TABLE`.

VALUE

Value to insert into the table.

LOCATION

Where to place the data in the table. The row and column elements define where to place the data. The length element, expressed in coordinate units, temporarily overrides the width of the column and allows data to span multiple columns.

ATTRIBUTES

Attributes to apply to the specified location. If an attribute is not specified, the current value as defined by CREATE-TABLE and ALTER-TABLE is used.

Table 41 FILL-TABLE Attributes

Attribute	Description
ANNOTATION	Arbitrary text string associated with the specified location. SQR passes the data to the backend printer driver—it does not validate the data.
BACKGROUND	Background color name or RGB triplet
BOLD	YES NO
CENTER	YES NO
COLUMN-LEADING	Expressed in decipoints
COLUMN-LINE-COLOR	Color name or RGB triplet
COLUMN-LINE-STYLE	SOLID SQUARE-DOT DASH DASH-DOT LONG-DASH LONG-DASH-DOT LONG-DASH-DOT-DOT
COLUMN-LINE-THICKNESS	Expressed in decipoints
EDIT-MASK	Edit mask or keyword to use
FILL-COLOR	Fill color name or RGB triplet. Default=NONE
FONT	Font number
FOREGROUND	Foreground color name or RGB triplet
FORMULAE	Arbitrary formula associated with the specified location. SQR passes the data to the backend printer driver – it does not validate the data.
ITALIC	YES NO
POINT-SIZE	Point size of the font
ROW-LINE-COLOR	Color name or RGB triplet
ROW-LINE-STYLE	SOLID SQUARE-DOT DASH DASH-DOT LONG-DASH LONG-DASH-DOT LONG-DASH-DOT-DOT
ROW-LINE-THICKNESS	Expressed in decipoints
UNDERLINE	YES NO
URL	Specifies the hypertext link for the specified location. SQR does not validate the address.
URL-TARGET	Specifies the target within the URL. SQR does not validate the target.
WRAP	YES NO maximum number of lines
WRAP-HEIGHT	Expressed as the number of lines between each wrapped line.

Attribute	Description
WRAP-ON	Characters on which to force a <code>WRAP</code>
WRAP-STRIP	Characters to change to a space before the <code>WRAP</code> is done

Description

Use `FILL-TABLE` in any section except `BEGIN-SETUP`, `BEGIN-SQL`, and `BEGIN-DOCUMENT` to manipulate table attributes.

Example

```
fill-table
  name='tab2'
  value=$column1
  location=(3,2)
  attributes=('column-line-color',('green'),'row-line-color',
    ('black'),'font',5)
```

See Also

[ALTER-TABLE](#), [CREATE-TABLE](#), [DECLARE-TABLE](#), [DUMP-TABLE](#), [PRINT-TABLE](#)

FIND

Function

Determines the location of a character sequence within a string.

Syntax

```
FIND {{obj_txt_lit|_var|_col}|{date_var|_col}} IN
{{src_txt_var|_col}|{date_var|_col}}
{start_int_lit|_var} dst_location_int_var
```

Arguments

{obj_txt_lit|_var|_col}|{date_var|_col}

Text variable, column, or literal in *src_txt_var|_col*.

{src_txt_var|_col}|{date_var|_col}

Text variable or column to search.

start_int_lit|_var

Starting location of the search.

dst_location_int_var

Returned starting location of the left-most character of the matching text in `{src_txt_var|_col|date_var|_col}`.

Description

FIND searches the specified string for a character sequence and, if the string is found, returns its location as an offset from the beginning of the specified string. If the sequence is not found, FIND returns -1 in `dst_location_int_var`.

You must specify an offset from which to begin the search and supply a numeric variable for the return of the location.

If the source or search object is a date variable or column, it is converted to a string before the search according to the following rules:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by `SQR_DB_DATE_FORMAT`. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,” on page 251](#).
- For DATE columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252](#).
- For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252](#).

Examples

```
find 'aw.2' in &code5 0 #loc
find ',' in &name 0 #comma_loc
if #comma_loc = -1
    ...comma not found...
```

See Also

- The `instr` function described in [Table 52 on page 212](#) under LET
- [EXTRACT](#)

GET

Function

Retrieves data from an array and places it into a date, string, or numeric variable.

Syntax

```
GET dst_any_var...FROM src_array_name(element)
[field(occurs)]...
```

Arguments

dst_any_var

Dates, strings, or numeric variables (not database columns) can be destination variables. Numeric variables (decimal, float, integer) are copied from number fields. String variables are copied from char, text, or date fields. Date variables are copied from char, text, or date fields.

When a date field is copied to a string variable, Production Reporting converts the date to a string in the format specified by `SQR_DB_DATE_FORMAT`. If not set, Production Reporting uses the first database-dependent format in [Table 61 on page 251](#).

If the destination is a date variable, the string extracted from the source must be in the format specified by `SQR_DB_DATE_FORMAT`, one of the database-dependent formats (see [Table 61 on page 251](#)), or the database-independent format `'SYYYYMMDD [HH24 [MI [SS [NNNNNNN]]]]'`.

src_array_name (element)

If the array's field names are listed, Production Reporting takes the values from the fields and occurrences specified. If the array's field names are not listed, the values are taken from consecutively defined fields in the array.

field[(occurs)]

Array element and field occurrence numbers can be numeric literals (such as 123) or numeric variables (such as `#j`). If no field occurrence is stated, occurrence *zero* is used.

Examples

The following example copies `$name`, `$start_date`, and `#salary` from the first three fields in the `#j`'th element of the `emps` array.

```
get $name $start_date #salary from emps(#j)
```

The following example copies `#city_tot` and `#county_tot` from the fields `cities` and `counties` in the `#j`'th element of the `states` array.

```
get #city_tot #county_tot from states(#j) cities counties
```

The following example copies `$code` from the `#j`'th occurrence of the `code` field in the `#n`'th element of the `codes` array.

```
get $code from codes(#n) code(#j)
```

See Also

- [LET](#) for information on assigning the value of an expression
- [PUT](#) for information on moving data into an array

GET-COLOR

Function

Retrieves the current colors.

Syntax

```
GET-COLOR
[PRINT-TEXT-FOREGROUND=({color_name_var})]
[PRINT-TEXT-BACKGROUND=({color_name_var})]
[PRINT-PAGE-BACKGROUND=({color_name_var})]
[LINE-COLOR=({color_name_var})]
[FILL-COLOR=({color_name_var})]
```

Arguments

PRINT-TEXT-FOREGROUND

Color in which the text prints.

PRINT-TEXT-BACKGROUND

Background color behind the text.

PRINT-PAGE-BACKGROUND

Page background color.

LINE-COLOR

Line color used in [DRAW](#) and [PRINT BOX](#).

FILL-COLOR

Fill color used in [DRAW](#) (TYPE=BOX) and [PRINT BOX](#).

{color_name_var}

Text variable that receives the name of the specified color.

Description

GET-COLOR is allowed wherever PRINT or DRAW is allowed. It is used to retrieve certain attributes of PRINT and DRAW. If the requested attribute does not map to a defined color name, then the name is returned as *RGBredgreenblue*, where each component is a three digit number. For example, *RGB127133033*. You can use this format wherever you use a color name. The color name 'none' is returned if no color is associated with the requested attribute.

```
begin-setup
  lighter_unknown = (93,122,129)
  set-color print-text-foreground=('lighter_unknown')
  get-color print-text-foreground=($print-foreground)
  print $print-foreground (+2,7)
end-setup
```

Examples

```
begin-program
  declare-color-map
    light_blue = (193, 222, 229)
  end-declare
```

```

end-program

begin-program
  alter-color-map name = 'light_blue' value = (193, 233, 230)
  print 'Yellow Submarine' ()
    foreground = ('yellow')
    background = ('light_blue')
  get-color print-text-foreground = ($print-foreground)
  set-color print-text-foreground = ('purple')
  print 'Barney' (+1,1)
  set-color print-text-foreground = ($print-foreground)
end-program

begin-program
  get-color line-color=($line-color)
  set-color line-color=('purple')
  draw (5,5) type='horz-line' width=10
  set-color line-color=($line-color)
end-program

begin-program
  get-color fill-color=($fill-color)
  set-color fill-color=('light grey')
  draw (5,5) type='box' width=10 height=10
  set-color fill-color=($fill-color)
end-program

```

See Also

[DECLARE-COLOR-MAP](#), [ALTER-COLOR-MAP](#), and [SET-COLOR](#)

GOTO

Function

Skips to the specified label.

Syntax

```
GOTO label
```

Arguments

label

A label in the same section or paragraph.

Description

Labels must end with a colon (:) and can appear anywhere within the same section or paragraph as GOTO.

Examples

```
begin-select
```

```

price
  if &price < #old_price
    goto next
  end-if
print &price (2,13,0) edit 999,999.99
...
next:
  add 1 to #count
  from products
end-select

```

#IF

Function

Indicates that the commands following are to be compiled when the expression is TRUE. (#IF is a compiler directive.)

Syntax

```
#IF {txt_lit|num_lit}comparison_operator
{txt_lit|num_lit}
```

Arguments

txt_lit|*num_lit*

Any text or numeric literal.

comparison_operator

Any of the following comparison operators:

Operator	Description
=	Equal
!=	Not Equal
<>	Not Equal
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal

Description

Production Reporting has five compiler directives that allow different pieces of Production Reporting code to be compiled, depending on the existence or value of substitution variables (not program variables, such as, string, numeric, or date).

Substitution variables defined automatically for each `-DEBUGxxx` letter can also be used with the `#IF`, `#IFDEF`, and `#IFNDEF` directives. They can turn entire sections of an Production Reporting program on or off from the command line, depending on the `-DEBUGxxx` flag.

You can nest `#IF`, `#IFDEF`, or `#IFNDEF` directives to a maximum of 10 levels.

The `#IF`, `#IFDEF`, or `#IFNDEF` directives cannot be broken across program lines.

[Table 42](#) lists the compiler directives.

Table 42 Production Reporting Compiler Directives

Directive	Example	Description
#IF	<code>#IF {option}='A'</code>	Compiles the commands following the <code>#IF</code> directive if the substitution variable option is equal to 'A'. The test is case-insensitive. Only one simple expression is allowed per <code>#IF</code> command.
#ELSE	<code>#ELSE</code>	Compiles the commands following the <code>#ELSE</code> directive when the <code>#IF</code> expression is FALSE.
#ENDIF	<code>#ENDIF</code>	Ends the <code>#IF</code> directive. <code>#ENDIF</code> can also be typed <code>#END-IF</code> (with a hyphen).
#IFDEF	<code>#IFDEF option</code>	Compiles the commands following the <code>#IFDEF</code> directive if the substitution variable option is defined.
#IFNDEF	<code>#IFNDEF option</code>	Compiles the command following the <code>#IFNDEF</code> directive if the substitution variable option is not defined.

Examples

```
begin-setup
  ask type 'Use Male, Female or Both (M,F,B) '
end-setup

begin-procedure Main
#if {type} = 'M'
  ...code for M here
#else
#if {type} = 'F'
  ...code for F here
#else
#if {type} = 'B'
  ...code for B here
#else
  show 'M, F or B not selected. Report not created.'
  stop
#endif ! for B
#endif ! for F
#endif ! for M

#ifdef debug
  show 'DEBUG: Cust_num = ' &cust_num edit 099999
#endif

#ifdef debugB ! DebugB turned on with -DEBUGB on
  do test_procedure! Production Reporting command line.
#endif
```

See Also

[#DEBUG](#) for information on the `-DEBUG` command-line flag

IF

Function

Executes commands depending on the value of a condition.

Syntax

```
IF logical_expression
```

IF commands have the following structure:

```
IF logical_expression  
  sqr_commands...  
[ELSE  
  sqr_commands...]  
END-IF
```

Arguments

logical_expression

Any valid logical expression. See [LET](#) for a description of logical expressions.

Operators

See [“Bit-Wise Operators” on page 195](#) for information on the bit-wise operators supported by IF.

Description

The expression is evaluated as a logical TRUE or FALSE. A value or expression that evaluates to nonzero is TRUE.

Each IF must have a matching END-IF.

IF commands can be nested.

Comparing a date variable or column with a string, results in a date comparison (chronological, not a byte by byte comparison as is done for strings). The string must be in the proper format as follows:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by `SQR_DB_DATE_FORMAT`, one of the database-dependent formats (see [Table 61, “Default Formats by Database,” on page 251](#)), or the database-independent format `'SYYYYMMDD [HH24 [MI [SS [NNNNNN]]]]'`.
- For DATE columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252](#).

- For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63](#), “TIME Column Formats,” on page 252.

Examples

```

if &price > &old_price and instr(&code, 'M', 1) > 0
  add 1 to #price_count
  if #price_count > 50
    show 'More than 50 prices found.' noline
    input $x 'Continue? (Y/N)'
    if upper($x) = 'N'
      stop
    end-if
  end-if
else
  add 1 to #old_price_count
end-if

if #rows      ! Will be TRUE if #rows is non-zero.
  do print-it
end-if

if $date1 > 'Apr 21 1996 23:59'
  do past_due
end-if

```

See Also

- [LET](#) for a description of logical expressions
- [EVALUATE](#)

#IFDEF

Function

Indicates that the following commands are to be compiled when the substitution variable has been declared by an `ASK` or `#DEFINE` command, or by the `-DEBUG` flag on the Production Reporting command line. (`#IFDEF` is a compiler directive.)

Syntax

```
#IFDEF substitution_variable
```

Arguments

substitution_variable

Variable used as the substitution variable.

See Also

[#IF](#) for a description of each compiler directive

#IFDEF

Function

Indicates that the following commands are to be compiled when the substitution variable has not been declared by an `ASK` or `#DEFINE` command, or by the `-DEBUG` flag on the Production Reporting command line. (`#IFDEF` is a compiler directive.)

Syntax

```
#IFDEF substitution_variable
```

Arguments

substitution_variable

Variable used as the substitution variable.

See Also

[#IF](#) for descriptions of compiler directives

#INCLUDE

Function

Includes an external source file into the Production Reporting report specification.

Syntax

```
#INCLUDE filename_lit
```

Arguments

filename_lit

A valid filename for the platform on which this application is compiled.

Description

You may want to keep commonly used routines in a single file and reference or “include” that file in programs that use the routine. For example, you might have a set of `#DEFINE` commands for different printers to control initialization, font changes, and page size declarations. You can reference the appropriate include file depending on which printer you want to use.

`INCLUDE` files can be nested up to four levels.

Variable substitution scanning takes place before the `#INCLUDE` command is processed. This allows you to substitute all or part of the `INCLUDE` file name at run time, adding flexibility to controlling which file is included for the run.

Examples

```
#include 'gethours.dat'      ! Common procedure.
#include 'XYZheader.dat'    ! Common report heading for XYZ Company.
#include 'printer{num}.dat' ! Include printer definitions for
                           ! printer {num}, which is passed
                           ! on the command line:
                           ! SQR REP1A SAM/JOE 18
                           ! where 18 is the arbitrary
                           ! number assigned your printer
                           ! definition file, 'printer18.dat'.
                           ! The report would contain the
                           ! command: ASK num
                           ! in the SETUP section, preceding
                           ! this #include statement.
```

INPUT

Function

Accepts data entered by the user at a terminal.

Syntax

```
INPUT input_var[MAXLEN=nn] [prompt]
[TYPE={CHAR|TEXT|NUMBER|INTEGER|DATE}]
[STATUS=num_var] [NOPROMPT] [BATCH-MODE]
[FORMAT={txt_lit|_var|_col}]
```

Arguments

input_var

Text, numeric, or date variable for the input data.

MAXLEN

Maximum length for the data.

prompt

Prompt (literal not variable) displayed to the user.

TYPE

Datatype required for the input.

STATUS

Numeric variable for a return status code.

NOPROMPT

Prevents the prompt from displaying before **INPUT** is processed.

BATCH-MODE

If `BATCH-MODE` is specified and no more arguments are in the command line, a value of 3 is returned in the `STATUS` variable and the user is not prompted for input.

FORMAT

Format for entering a date (see [Table 57, “Date Edit Format Characters,”](#) on page 245).

Description

Use `MAXLEN` to prevent entering data that is too long. If `INSERT` or `UPDATE` references a variable whose length is greater than that defined in the database, the SQL is rejected and Production Reporting halts. If the maximum length is exceeded, the terminal beeps (on some systems, this may cause the screen to flash instead).

If `prompt` is omitted, Production Reporting uses the default prompt, `Enter [$|#] var: .` In any case, a colon (`:`) and two spaces are added to the prompt.

Specifying `TYPE` causes data type checking to occur. If the string entered is not the type specified, the terminal beeps and an error message is displayed. `INPUT` is then re-executed. If `TYPE=DATE` is specified, then `input_var` can be a date or text variable; however, `TYPE=DATE` is optional if `input_var` is a date variable. If a numeric variable is used, it is validated as a numeric variable. `CHAR`, `TEXT`, and `DATE` are invalid types.

Table 43 Data Types Supported by `INPUT`

Datatype	Description
CHAR, TEXT	Any character. This is the default datatype.
NUMBER	A floating point number in the format [+ -]9999.999[E[+ -]99]
INTEGER	An integer in the format [+ -]999999
DATE	A date in one of the following formats:
	MM/DD/YYYY [BC AD] [HH:MI[:SS[.NNNNNN]] [AM PM]]
	MM-DD-YYYY [BC AD] [HH:MI[:SS[.NNNNNN]] [AM PM]]
	MM.DD.YYYY [BC AD] [HH:MI[:SS[.NNNNNN]] [AM PM]]
	YYYYMMDD[HH24[MI[SS[NNNNNN]]]]

Specifying `STATUS` causes `INPUT` to complete regardless of what the user enters. No error message is displayed. A nonzero error code is stored in the indicated numeric variable if the length or datatype entered is incorrect.

Table 44 Values of the `STATUS` Argument of the `INPUT` Command

Status Value	Indicates
0	Successful.

Status Value	Indicates
1	Bad type (did not match the datatype of TYPE).
2	Too long (longer than MAXLEN or the input for an INTEGER variable is < -2147483648 or > +2147483647).
3	No arguments remain on the command line. The command was ignored.

By using `NOPROMPT` and `STATUS` with `SHOW`, you can write a sophisticated data entry routine.

`FORMAT` can only be used with dates. It can be a date edit mask or the keyword `DATE`. Use the keyword `DATE` if the date must be in the format as specified with `INPUT-DATE-EDIT-MASK` for the current locale. If `FORMAT` has not been set, use a database-independent format for the data in [Table 43 on page 190](#).

Examples

The following example shows several `INPUT` commands:

```
input $state maxlen=2 'Please enter state abbreviation'
input #age 'Enter lower age boundary' type=integer
input $start_date 'Enter starting date for report' type=date
input $date_in format='Mon dd yyyy'
input $date format=date
```

The following example shows another `INPUT` command:

```
show clear-screen (5,32) reverse 'CUSTOMER SUMMARY' normal
Try_again:
show (12,20) 'Enter Start Date: ' clear-line
input $start-date noprompt status=#istat type=date
if #istat != 0
    show (24,1) 'Please enter date in format DD-MON-YY' beep
    goto try_again
end-if
show (24,1) clear-line! Clear error message line.
```

The following example illustrates the use of `BATCH-MODE`:

```
begin-program
    while (1)
        input $A status=#stat batch-mode
        if #stat = 3
            break
        else
            do procedure ($a)
        end-if
    end-while
end-program
```

See Also

- [ALTER-LOCALE](#)
- The `INPUT-DATE-EDIT-MASK` setting in [Chapter 6, “SQR.INI.”](#)

LAST-PAGE

Function

Places the last page number on each page, as in “page *n* of *m*”.

Syntax

```
LAST-PAGE position [pre_txt_lit[post_txt_lit]]
```

Arguments

position

Position for printing the last page number.

pre_txt_lit

Text string printed before the last page number.

post_txt_lit

Text string printed after the last page number.

Description

The text strings specified in *pre_txt_lit* and *post_txt_lit* are printed immediately before and after the number.

Using `LAST-PAGE` causes the `SQR` and `SQRT` executables to delay printing until the last page has been processed so that the number of the last page is known.

Examples

```
begin-footing 1
  page-number(1,37) 'Page ' Will appear as
  last-page () ' of ' .' ! "Page 12 of 25."
end-footing
```

See Also

[PAGE-NUMBER](#), [BEGIN-HEADING](#), and [BEGIN-FOOTING](#)

LET

Function

Assigns the value of an expression to a string, numeric, date, or list (DDO only) variable.

Syntax

```
LET dst_var=expression
```


Arguments

dst_var

String, numeric, date, or list (DDO only) variable or array field to which the result of the expression is assigned.

expression

Expression to evaluate.

Description

Valid expressions are formed as a combination of:

- [Operands](#)
- [Operators](#)
- [Functions](#)

String, numeric, date, and array field operands can be used in an expression as well as embedded functions. Production Reporting supports a standardized set of mathematical operators and logical comparison operators working within a carefully defined set of precedence rules.

Production Reporting also provides the user with a rich set of numeric, string, date, unicode, and file manipulation functions along with a number of special purpose utility functions. All combined, the Production Reporting expression provides the user with a very powerful tool that can be tailored to suit any information processing need. (Note that all string indices are one-based, not zero-based.)

Examples

The following examples show some complex expressions:

```
let #j = ((#a + #b) * #c) ^ 2
if #j > 2 and sqrt(#j) < 20 or #i + 2 > 17.4
while upper(substr(&descrip,1,#j+2)) != 'XXXX'
and not isnull(&price)
let #len = length(&fname || &initial || &lname) + 2
let $s = edit(&price * &rate, '99999.99')
let summary.total(#j) = summary.total(#j) + (&price * &rate)
if summary.total(#j) > 1000000
let store.total (#store_id, #dept)
    = store.total (#store_id, #dept) + #total
let #diff = datediff(datenow(), strtodate('1995','YYYY'),'day')
let $newdate = dateadd(datenow(), 'month', 50)
let $date1 = datetostr(strtodate(&sale_date), 'Day Month DD, YYYY')
```

Production Reporting analyzes LET, IF, and WHILE expressions when it compiles code and saves the result in an internal format so that repetitive execution is at maximum speed.

Operands

Operands form the backbone of an Production Reporting expression. Operands do not have to be the same type. You can combine string, numeric and array field operands to form a valid

expression. Production Reporting performs a sequence of automatic operand conversions as it evaluates expressions that contain dissimilar operand types. As the expression is evaluated, operands of lower precision are converted to match the operand of higher precision. Consider the following example:

```
let #answer = #float * #decimal / #integer
```

Since the *multiply* and *divide* operators are equal in precedence, the expression is evaluated as $(\#float * \#decimal) / \#integer$. Working from the inside out, the *#float* variable is converted to a decimal type where a multiply is performed yielding the simplified expression, $(\#decimal) / \#integer$. Production Reporting now converts the *#integer* operand to a decimal type before performing the final divide. When finished with the expression evaluation, Production Reporting converts the result to match the type of the *#answer* variable.

Converting operands of lower precision to operands of higher precision preserves the number of significant digits. The number of significant digits is not lost when an integer is converted to float or decimal. In a similar manner, the number of significant digits is preserved when floating point operands are converted to the decimal type. The number of significant digits is only sacrificed when the final result is converted to match the type of the *#answer* variable and this variable is less precise than the highest of the operands being evaluated. In the example, precision is not lost if the *#answer* is declared as a decimal type. Production Reporting considers integer variables as the lowest in the precision hierarchy, followed by float and then decimal.

Here are a few simple expression examples:

```
let #discount=round (&price * #rate / 100, 2)
let $name=$first_name || ' ' || $last_name
let customer.total (#customer_id) =
  customer.total (#customer_id) + #invoice_total
  if not range(upper($code), 'A', 'G')
  ...processing when out of range...
let store.total (#store_id, #qtr) =
  store.total (#store_id, #qtr) + #invoice_total
let $date1 = strtodate ('Apr 10 1996', 'MON DD YYYY')
```

The following sections list operators and functions supported in expressions.

Operators

[Table 45](#) lists operators in descending order of precedence. Operators listed in the same row within the table have the same precedence (the operators **,/,%* are equal in precedence).

Operators of the same precedence are processed in the sequence they appear in the expression, from left to right. Use parentheses to override the normal precedence rules. All numeric types (decimal, float, integer) are supported for all operators.

Table 45 Operators

Operator	Explanation
	Concatenate two strings or dates

Operator	Explanation
+ , -	Sign prefix (positive or negative)
^	Exponent
* , / , %	Multiply, divide, remainder: $a \% b = \text{mod}(a,b)$ for integers
+ , -	Plus, minus Note: Production Reporting distinguishes between a sign prefix and arithmetic operation by the context of the expression.
> , < , >= , <= , <> , != , =	Comparison operators: greater than, less than, greater or equal to, less than or equal to, not equal (!= or <>), equal
not	Logical NOT
and	Logical AND
or , xor	Logical OR, XOR (exclusive OR)

Bit-Wise Operators

Bit-Wise operators allow you to utilize bitmasks within your program.

Bit-Wise operators act just like their logical counterparts (**AND**, **OR**, **XOR**) except that instead of returning **TRUE** or **FALSE**, they return the actual result.

To facilitate the use of these operators, **LET** recognizes the hexadecimal notation of **0X?** for expressing a numerical constant. The **?** character is from 1 to 8 hexadecimal characters (0-F).

Table 46 Bit-Wise Operators

Operator	Explanation
BitAND	Acts just like Logical AND except returns the actual result instead of TRUE or FALSE
BitOR	Acts just like Logical OR except returns the actual result instead of TRUE or FALSE
BitXOR	Acts just like Logical XOR except returns the actual result instead of TRUE or FALSE

Note:

In addition to **BitAND**, **BitOR**, and **BitXOR**, you can use the **HEX** function. The **HEX** function takes a numerical argument and returns a string, in the form of **0X?**, which is the hexadecimal representation of the argument.

To prevent any loss of precision, declare the arguments to the Bit-Wise operators and the **HEX** function as an **INTEGER**.

Sample program:

```
!
! Validation test for Bit-Wise LET operators
```

```

!
begin-setup
  declare-variable
    integer #mask
  end-declare
end-setup

begin-program
  let #mask = 0x1000
  if not (#mask BitAND 0x1000)
    let $mask = hex(#mask)
    show 'impossible ' $mask ' BitAND 0x1000 failed'
  end-if
  if (#mask BitOR 0x1000) <> 0x1000
    let $mask = hex(#mask)
    show 'impossible ' $mask ' BitOR 0x1000 failed'
  end-if
  if (#mask BitXOR 0x1000)
    let $mask = hex(#mask)
    show 'impossible ' $mask ' BitXOR 0x1000 failed'
  end-if
  let #Mask = 0
  if (#mask BitXOR 0x1000) <> 0x1000
    let $mask = hex(#Mask)
    show 'impossible ' $mask ' BitXOR 0x1000 failed'
  end-if
  let $mask = hex(0xffffffff)
  if $mask <> '0xffffffff'
    show 'impossible ' $mask ' <> 0xffffffff'
  end-if
  let $mask = hex(0x12345678)
  If $Mask <> '0x12345678'
    show 'impossible ' $mask ' <> 0x12345678'
  end-if
  let $mask = hex(0xabcdef12)
  if $mask <> '0xabcdef12'
    show 'impossible ' $mask ' <> 0xabcdef12'
  end-if
end-program

```

Functions

Production Reporting functions include:

- [Numeric Functions](#)
- [File-Related Functions](#)
- [String Functions](#)
- [Date Functions](#)
- [Unicode Functions](#)
- [Miscellaneous Functions](#)

Function arguments are enclosed in parentheses and can be nested. Arguments referenced as x, y, or z indicate the first, second, or third argument of a function. Otherwise, functions take a single argument or no arguments. All arguments are evaluated before a function is evaluated.

Not all functions support all numeric types (decimal, float, integer). Certain functions do not support the decimal type directly, but convert input decimal operand(s) to the float type before the function is evaluated. [Table 47](#) annotates the functions that directly support the decimal type and which ones do not.

Use parentheses to override the normal precedence rules.

Note:

In functions where a string argument is expected and a date variable, column, or expression is entered, Production Reporting converts the date to a string according to the following rules:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by `SQR_DB_DATE_FORMAT`. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,”](#) on page 251.
- For DATE columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,”](#) on page 252.
- For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,”](#) on page 252.

On the other hand, except where noted in an individual function, if a string variable, column, or expression is entered where a date argument is expected, then the string must be in the format specified by `SQR_DB_DATE_FORMAT`, one of the database-dependent formats in [Table 61, “Default Formats by Database,”](#) on page 251, or the database-independent format `'SYYYMMDD [HH24 [MI [SS [NNNNNN]]]]'`

Numeric Functions

Table 47 Numeric Functions

Function	Description
abs	<p>Returns the absolute value of <i>num_value</i>.</p> <p>Value type: Same as <i>num_value</i></p> <p>Syntax: <i>dst_var</i> = abs(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let#dabsvar = abs(#dvar)</p>

Function	Description
acos	<p>Returns the arccosine of <i>num_value</i> in the range of 0 to π radians. The value of <i>num_value</i> must be between -1 and 1.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = acos(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #facosvar = acos(#fvar)</p>
asin	<p>Returns the arcsine of <i>num_value</i> in the range of $-\pi/2$ to $\pi/2$ radians. The value of <i>num_value</i> must be between -1 and 1.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = asin(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fasinvar = asin(#fvar)</p>
atan	<p>Returns the arctangent of <i>num_value</i> in the range of $-\pi/2$ to $\pi/2$ radians. The value of <i>num_value</i> must be between -1 and 1.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = atan(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fatanvar = atan(#fvar)</p>
ceil	<p>Returns a value representing the smallest integer that is greater than or equal to <i>num_value</i>.</p> <p>Value type: Same as <i>num_value</i>.</p> <p>Syntax: <i>dst_var</i> = ceil(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fceilvar = ceil(#fvar)</p>
cos	<p>Returns the cosine of <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = cos(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fcosvar = cos(#fvar)</p>
cosh	<p>Returns the hyperbolic cosine of <i>num_value</i>. This function returns a float value.</p>

Function	Description
	<p>Syntax: <i>dst_var</i> = cosh(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fcoshvar = cosh(#fvar)</p>
deg	<p>Returns a value expressed in degrees of <i>num_value</i> which is expressed in radians.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = deg(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fdegvar = deg(#fvar)</p>
e10	<p>Returns the value of 10 raised to <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = e10(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fe10var = e10(#fvar)</p>
exp	<p>Returns the value of e raised to <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = exp(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fexpvar = exp(#fvar)</p>
floor	<p>Returns a value representing the largest integer that is less than or equal to <i>num_value</i>.</p> <p>Value type: Same as <i>num_value</i></p> <p>Syntax: <i>dst_var</i> = floor(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #ffloorvar = floor(#fvar)</p>
hex	<p>Returns a string, in the form of 0x?, which is the hexadecimal representation of the argument</p> <p>Syntax: <i>dst_var</i> = hex(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to an integer. ● <i>dst_var</i> = string variable. <p>Example: let \$hexvar = hex(#fvar)</p>

Function	Description
log	<p>Returns the natural logarithm of <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = log(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #flogvar = log(#fvar)</p>
log10	<p>Returns the base-10 logarithm of <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = log10(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #flog10var = log10(#fvar)</p>
mod	<p>Returns the fractional remainder, <i>f</i>, of <i>x_value</i>/<i>y_value</i> such that $x_value = i * y_value + f$, where <i>i</i> is an integer, <i>f</i> has the same sign as <i>x_value</i>, and the absolute value of <i>f</i> is less than the absolute value of <i>y_value</i>. The arguments are promoted to the type of the greatest precision and the function returns a value of that type.</p> <p>Syntax: <i>dst_var</i> = mod(<i>x_value</i>, <i>y_value</i>)</p> <ul style="list-style-type: none"> • <i>x_value</i> = decimal, float, or integer literal, column, variable, or expression. • <i>y_value</i> = decimal, float, or integer literal, column, variable, or expression. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fmodvar = mod(#fxvar, #fyvar)</p>
power	<p>Returns the value of <i>x_value</i> raised to the power of <i>y_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = power(<i>x_value</i>, <i>y_value</i>)</p> <ul style="list-style-type: none"> • <i>x_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>y_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fpowervar = power(#fxvar, #fyvar)</p>
rad	<p>Returns a value expressed in radians of <i>num_value</i> which is expressed in degrees.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = rad(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>place_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.

Function	Description
	<ul style="list-style-type: none"> ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fradvar = rad(#fvar)</p>
round	<p>Returns a value that is <i>num_value</i> rounded to <i>place_value</i> digits after the decimal separator.</p> <p>Value type: Same as <i>num_value</i></p> <p>Syntax: <i>dst_var</i> = round(<i>num_value</i>, <i>place_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. ● <i>place_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #frndvar = round(#fvar, #fplace)</p>
sign	<p>Returns a -1, 0, or +1 depending on the sign of <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = sign(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fsignvar = sign(#fvar)</p>
sin	<p>Returns the sine of <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = sin(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fsinvar = sin(#fvar)</p>
sinh	<p>Returns the hyperbolic sine of <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = sinh(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fsinhvar = sinh(#fvar)</p>
sqrt	<p>Returns the square root of <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = sqrt(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #fsqrtvar = sqrt(#fvar)</p>

Function	Description
tan	<p>Returns the tangent of <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = tan(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #ftanvar = tan(#fvar)</p>
tanh	<p>Returns the hyperbolic tangent of <i>num_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = tanh(<i>num_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #ftanhvar = tanh(#fvar)</p>
trunc	<p>Returns a value that is <i>num_value</i> truncated to <i>place_value</i> digits after the decimal separator.</p> <p>Value type: Same as <i>num_value</i></p> <p>Syntax: <i>dst_var</i> = trunc(<i>num_value</i>, <i>place_value</i>)</p> <ul style="list-style-type: none"> • <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. • <i>place_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. • <i>dst_var</i> = decimal, float, or integer variable. <p>Example: let #ftruncvar = trunc(#fvar, #fplace)</p>

The transcendental functions `sin`, `cos`, `tan`, `sinh`, `cosh`, and `tanh` take their arguments in radians. The functions `asin`, `acos`, and `atan` return radian values. To convert from radians to degrees or degrees to radians, use the `rad` or `deg` functions as follows:

```
let #x = sin(rad(45)) ! Sine of 45 degrees.
let #y = deg(asin(#x))! Convert back to degrees.
```

If arguments or intermediate results passed to a numeric function are invalid for that function, Production Reporting halts with an error message.

For example, passing a negative number to the `sqrt` function causes an error. Use the `cond` function described in [Table 52](#) to prevent division by zero or other invalid function or operator argument values.

File-Related Functions

Note:

File-related functions return zero (0) if successful; otherwise, they return the system error code.

Table 48 File-Related Functions

Function	Description
delete	<p>Deletes <i>filename</i>.</p> <p>Syntax: <i>stat_var</i> = delete(<i>filename</i>)</p> <ul style="list-style-type: none"> ● <i>filename</i> = text literal, column, variable, or expression. ● <i>stat_var</i> = decimal, float, or integer variable. <p>Example: let #fstatus = delete(\$filename)</p>
exists	<p>Determines if <i>filename</i> exists.</p> <p>Syntax: <i>stat_var</i> = exists(<i>filename</i>)</p> <ul style="list-style-type: none"> ● <i>filename</i> = text literal, column, variable, or expression. ● <i>stat_var</i> = decimal, float, or integer variable. <p>Example: let #fstatus = exists(\$filename)</p>
filesize	<p>Accepts the name of an external file and returns the number of bytes it contains. If the file size cannot be determined, -1 is returned.</p> <p>Syntax: <i>dst_var</i> = filesize(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #size = filesize(\$file)</p>
rename	<p>Renames <i>old_filename</i> to <i>new_filename</i>.</p> <p>Syntax: <i>stat_var</i> = rename(<i>old_filename</i>, <i>new_filename</i>)</p> <ul style="list-style-type: none"> ● <i>old_filename</i> = text literal, column, variable, or expression ● <i>new_filename</i> = text literal, column, variable, or expression ● <i>stat_var</i> = decimal, float, or integer variable <p>Example: let #fstatus = rename(\$old_filename, \$new_filename)</p>

String Functions

Table 49 String Functions

Function	Description
ascii	<p>Returns the ASCII value for the first character in <i>str_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>ascii_var</i> = ascii(<i>str_value</i>)</p> <ul style="list-style-type: none"> ● <i>str_value</i> = date or text literal, column, variable, or expression ● <i>ascii_var</i> = decimal, float, or integer variable <p>Example: let #fascii = ascii(\$filename)</p>
asciic	<p>Returns the numeric value for the first character (rather than byte) of the specified string.</p> <p>Syntax: <i>ascii_var</i> = asciic(<i>str_value</i>)</p>

Function	Description
	<ul style="list-style-type: none"> ● <i>str_value</i> = date or text literal, column, variable, or expression ● <i>ascii_var</i> = decimal, float, or integer variable <p>Example: let #fascii = asciic(\$filename)</p>
chr	<p>Returns a string composed of a character with the ASCII value of <i>num_value</i>.</p> <p>Syntax: <i>dst_var</i> = chr(<i>num_value</i>)</p> <ul style="list-style-type: none"> ● <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>dst_var</i> = text variable <p>Example: let \$svar = chr(#num)</p>
edit	<p>Formats <i>source_value</i> according to <i>edit_mask</i> and returns a string containing the result.</p> <p>Syntax: <i>dst_var</i> = edit(<i>source_value</i>, <i>edit_mask</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = Any literal, column, variable, or expression ● <i>edit_mask</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$phone = edit(&phone, '(xxx) xxx-xxxx') let \$price = edit(#price, '999.99') let \$today = edit(\$date, 'DD/MM/YYYY')</p>
fromhex	<p>Accepts a TEXT variable that contains a string of hexadecimal characters (case insensitive) and returns a BINARY variable. Each byte of BINARY data consists of two hexadecimal characters.</p> <p>Syntax: <i>dst_var</i> = fromhex(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = binary variable <p>Example: let \$image = fromhex(\$hexchars)</p>
instr	<p>Returns the numeric position of <i>sub_value</i> in <i>source_value</i> or zero (0) if not found. The search begins at offset <i>offset_value</i>.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = instr(<i>source_value</i>, <i>sub_value</i>, <i>offset_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>sub_value</i> = text literal, column, variable, or expression ● <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #offset = instr(&description, 'auto', 10)</p>
instrb	<p>Performs the same functionality as the <code>instr</code> function except that the starting point and returned value are expressed in bytes rather than characters.</p> <p>Syntax: <i>dst_var</i> = instrb(<i>source_value</i>, <i>sub_value</i>, <i>offset_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>sub_value</i> = text literal, column, variable, or expression ● <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.

Function	Description
	<ul style="list-style-type: none"> ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #offset = instrb(&description, 'auto', 10)</p> <p>Note: <code>instrb</code> does <i>not</i> allow you to specify the target encoding. If you are using Unicode internally, specify the target encoding with <code>lengthp</code>, <code>lengtht</code>, <code>substrp</code>, <code>substrt</code>, or <code>transform</code>.</p>
isblank	<p>Returns a value of one (1) if <i>source_val</i> is an empty string, null string, or composed entirely of whitespace characters; otherwise, returns a value of zero (0).</p> <p>Syntax: <i>dst_var</i> = isblank(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression (character data type columns only, no numeric data type columns) ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #blank = isblank(&description)</p> <p>Note: <code>isblank</code> can only be used for character data type columns.</p>
length	<p>Returns the number of characters in <i>source_value</i>.</p> <p>Syntax: <i>dst_var</i> = length(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #length = length(&description)</p>
lengthb	<p>Same functionality as <code>length</code> except that the return value is expressed in bytes, rather than characters.</p> <p>Syntax: <i>dst_var</i> = lengthb(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #length = lengthb(&description)</p> <p>Note: <code>lengthb</code> does <i>not</i> allow you to specify the target encoding. If you are using Unicode internally, specify the target encoding with <code>lengthp</code>, <code>lengtht</code>, <code>substrp</code>, <code>substrt</code>, or <code>transform</code>.</p>
lower	<p>Converts the contents of <i>source_value</i> to lowercase and returns the result.</p> <p>Syntax: <i>dst_var</i> = lower(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$lower = lower(&description)</p>
lpad	<p>Pads the <i>source_value</i> on the left to a length of <i>length_value</i> using <i>pad_value</i> and returns the result.</p> <p>Syntax: <i>dst_var</i> = lpad(<i>source_value</i>, <i>length_value</i>, <i>pad_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer ● <i>pad_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable

Function	Description
	Example: let \$lpad = lpad(\$notice, 25, '!')
ltrim	<p>Trims characters in <i>source_value</i> from the left until a character is not in <i>set_value</i> and returns the result.</p> <p>Syntax: <i>dst_var</i> = ltrim(<i>source_value</i>, <i>set_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>set_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$ltrim = ltrim(&description, '!')</p>
replace	<p>Inspects the contents of <i>source_value</i> and replaces all occurrences of <i>from_string</i> with <i>to_string</i> and returns the modified string.</p> <p>Syntax: <i>dst_var</i> = replace(<i>source_value</i>, <i>from_string</i>, <i>to_string</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>from_string</i> = text literal, column, variable, or expression ● <i>to_string</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$replaced = replace(\$paragraph, 'good', 'excellent')</p>
rpadd	<p>Pads the <i>source_value</i> on the right to a length of <i>length_value</i> using <i>pad_value</i> and returns the result.</p> <p>Syntax: <i>dst_var</i> = rpadd(<i>source_value</i>, <i>length_value</i>, <i>pad_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. ● <i>pad_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$rpadd = rpadd(\$notice, 25, '!')</p>
rtrim	<p>Trims characters in <i>source_value</i> from the right until a character is not in <i>set_value</i> and returns the result.</p> <p>Syntax: <i>dst_var</i> = rtrim(<i>source_value</i>, <i>set_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date, or text literal, column, variable, or expression ● <i>set_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$rtrim = rtrim(&description, '!')</p>
substr	<p>Extracts the specified portion <i>source_value</i>. The extraction begins at <i>offset_value</i> (origin is 1) for a length of <i>length_value</i> characters.</p> <p>Syntax: <i>dst_var</i> = substr(<i>source_value</i>, <i>offset_value</i>, <i>length_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.

Function	Description
	<ul style="list-style-type: none"> ● <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. ● <i>dst_var</i> = text variable <p>Example: let \$piece = substr(&record, 10, #len)</p>
substrb	<p>Has the same functionality as <code>substr</code> except that the starting point and length are expressed in bytes, rather than in characters.</p> <p>Syntax: <i>dst_var</i> = substrb(<i>source_value</i>, <i>offset_value</i>, <i>length_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. ● <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. ● <i>dst_var</i> = text variable <p>Example: let \$piece = substrb(&record, 10, #len)</p> <p>Note: <code>substrb</code> does <i>not</i> allow you to specify the target encoding. If you are using Unicode internally, specify the target encoding with <code>lengthp</code>, <code>lengtht</code>, <code>substrp</code>, <code>substrt</code>, or <code>transform</code>.</p>
to_char	<p>Converts <i>source_value</i> to a string, using maximum precision.</p> <p>Syntax: <i>dst_var</i> = to_char(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = decimal, float, or integer literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$string = to_char(#number)</p>
tohex	<p>Accepts a BINARY variable and returns a string composed of uppercase hexadecimal characters that represents the data. Each byte of BINARY data consists of two hexadecimal characters.</p> <p>Syntax: <i>dst_var</i> = tohex(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = binary literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$hexchars = tohex(\$vargraphic)</p>
to_multi_byte	<p>Converts the specified string as follows: any occurrence of a single-byte character that also has a multi-byte representation (numerals, punctuation, roman characters, and katakana) is converted.</p> <p>Syntax: <i>dst_var</i> = to_multi_byte (<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$multi = to_multi_byte (&text)</p>
to_number	<p>Converts <i>source_value</i> to a number.</p> <p>Value type: float</p> <p>Syntax: <i>dst_var</i> = to_number(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = decimal, float, or integer literal, column, variable, or expression ● <i>dst_var</i> = decimal, float, or integer variable

Function	Description
	Example: let #value = to_number(\$number)
to_single_byte	<p>Converts the specified string as follows: any occurrence of a multi-byte character that also has a single-byte representation (numerals, punctuation, roman characters, and katakana) is converted.</p> <p>This function also converts a sequence of kana characters followed by certain grammatical marks into a single-byte character that combines the two elements. For all other encodings, the string is not modified.</p> <p>Note: If you are running Production Reporting without the use of Unicode (UseUnicodeInternal=FALSE in SQR.INI), this conversion only occurs when the database encoding (ENCODING-DATABASE setting in SQR.INI) is set to SJIS, EBCDIK290, and EBCDIK1027.</p> <p>Syntax: <i>dst_var</i> = to_single_byte (<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$single = to_single_byte (&text)</p>
translate	<p>Inspects the contents of <i>source_value</i> and converts characters that match those in <i>from_set</i> to the corresponding character in <i>to_set</i> and returns the translated string.</p> <p>If <i>to_set</i> does not contain a matching translation character in the corresponding <i>from_set</i>, then the original is left unchanged with regard to that character. If the translation string in <i>to_set</i> is empty, then all characters specified in the <i>from_set</i> string are removed.</p> <p>Syntax: <i>dst_var</i> = translate(<i>source_value</i>, <i>from_set</i>, <i>to_set</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>from_set</i> = text literal, column, variable, or expression ● <i>to_set</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$translated = translate(edit(&price, '999,999.99'), ',.', ',.')</p>
upper	<p>Converts the contents of <i>source_value</i> to uppercase and returns the result.</p> <p>Syntax: <i>dst_var</i> = upper(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$upper = upper(&description)</p>

Date Functions

Table 50 Date Functions

Function	Description
dateadd	<p>Returns a date after adding (or subtracting) the specified units to the <i>date_value</i>.</p> <p>Syntax: <i>date_var</i> = dateadd(<i>date_value</i>, <i>units_value</i>, <i>quantity_value</i>)</p> <ul style="list-style-type: none"> ● <i>date_value</i> = date variable or expression ● <i>units_value</i> = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second'

Function	Description
	<ul style="list-style-type: none"> ● <i>quantity_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>date_var</i> = date variable <p>Example: let \$date = dateadd(\$startdate, 'day', 7.5)</p>
datediff	<p>Returns the difference between the specified dates expressed in <i>units_value</i>. The function returns a float value. The result can be negative if the first date is earlier than the second date.</p> <p>Syntax: <i>dst_var</i> = datediff(<i>date1_value</i>, <i>date2_value</i>, <i>units_value</i>)</p> <ul style="list-style-type: none"> ● <i>date1_value</i> = date variable or expression ● <i>date2_value</i> = date variable or expression ● <i>units_value</i> = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second' ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #diff = datediff(\$date1, \$date2, 'hour')</p>
datetimeow	<p>Returns the current local date and time from the client machine.</p> <p>Syntax: <i>dst_var</i> = datetimeow()</p> <ul style="list-style-type: none"> ● <i>dst_var</i> = date variable <p>Example: let \$date = datetimeow()</p>
datetostr	<p>Converts the date <i>date_value</i> to a string in the format <i>format_mask</i>.</p> <p>Syntax: <i>dst_var</i> = datetostr(<i>date_value</i>, [<i>format_mask</i>])</p> <ul style="list-style-type: none"> ● <i>date_value</i> = date variable or expression ● <i>format_mask</i> = text literal, column, variable, or expression. DATE can be used to specify DATE-EDIT-MASK from the current locale. If this argument is not specified, then the format specified by SQR_DB_DATE_FORMAT is used. If this has not been set, then the first database-dependent format in Table 61, “Default Formats by Database,” on page 251 is used. ● <i>dst_var</i> = text variable <p>Example: let \$formdate = datetostr(\$date, 'Day Mon DD, YYYY') let \$localedate = datetostr(\$date, DATE)</p>
strtodate	<p>Converts the string <i>source_value</i> in the format <i>format_mask</i> to a date type.</p> <p>Syntax: <i>dst_var</i> = strtodate(<i>source_value</i> [, <i>format_mask</i>])</p> <ul style="list-style-type: none"> ● <i>source_value</i> = text literal, column, variable, or expression ● <i>format_mask</i> = text literal, column, variable, or expression that describes the exact format of the <i>source_value</i>. DATE can specify the DATE-EDIT-MASK setting from the current locale. If not specified, then <i>source_value</i> must be in the format specified by SQR_DB_DATE_FORMAT, one of the database-dependent formats (see Table 61, “Default Formats by Database,” on page 251), or the database-independent format 'SYYYYMDD[HH24[MI[SS[NNNNN]]]]'. Valid format codes are specified in Table 57 on page 245. See “PRINT” on page 239 for information regarding the default date-time components as a result of converting an incomplete date. ● <i>dst_var</i> = date variable <p>Example: let \$date = strtodate(\$str_date, 'Mon DD, YYYY') let \$date = strtodate(\$str_date, DATE)</p>

Unicode Functions

Note:

Unicode functions are only allowed when converting to Unicode internally.

Table 51 Unicode Functions

Function	Description
lengthp	<p>Returns the string length in print position. Half-width characters take one print position, full-width characters take two, and combining characters take zero.</p> <p>Syntax: <i>dst_var</i> = lengthp(<i>source_value</i>)</p> <ul style="list-style-type: none">● <i>source_value</i> = date or text literal, column, variable, or expression● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #printLen = lengthp(\$string)</p>
lengtht	<p>Returns the string length in bytes when converted (transformed) to a specified encoding. Encoding names are the same as those allowed in OPEN or in SQR.INI. String and column variables can be used in place of the literal encoding name.</p> <p>Syntax: <i>dst_var</i> = lengtht(<i>source_value</i>, <i>encoding_value</i>)</p> <ul style="list-style-type: none">● <i>source_value</i> = date or text literal, column, variable, or expression● <i>encoding_value</i> = text literal, column, variable, or expression● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #sjisLen = lengtht(\$string, 'shift-jis')</p>
substrp	<p>(Returns a substring of a given string starting at a specified print position into the string and of a specified print length. When #printPos is in the middle of a full-width character, Production Reporting “rounds up” to the next character. When #printLen ends in a partial character, Production Reporting “rounds down” to the previous character.</p> <p>Syntax: <i>dst_var</i> = substrp(<i>source_value</i>, <i>offset_value</i>, <i>length_value</i>)</p> <ul style="list-style-type: none">● <i>source_value</i> = date or text literal, column, variable, or expression.● <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.● <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.● <i>dst_var</i> = text variable <p>Example: let \$sub = substrp(&string, #printPos, #printlen)</p>
substrt	<p>Returns a Unicode string equivalent to a byte level substring of a given string after converting (transforming) the given string to a given encoding. If the substring of the converted string yields a partial character, that character will be truncated.</p> <p>Syntax: <i>dst_var</i> = substrt(<i>source_value</i>, <i>offset_value</i>, <i>length_value</i>, <i>encoding_value</i>)</p> <ul style="list-style-type: none">● <i>source_value</i> = date or text literal, column, variable, or expression● <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.

Function	Description
	<ul style="list-style-type: none"> ● <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. ● <i>encoding_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$sjisPrep = SUBSTR (\$string, 1, 10, 'Shift-JIS')</p>
transform	<p>Returns a Unicode string which is specified transform of a given string.</p> <p>Syntax: <i>dst_var</i> = transform (<i>source_value</i>, <i>transform_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable or expression ● <i>transform_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> - text variable <p>Example: let \$hiragana = transform (&string, 'ToHiragana')</p> <p>Production Reporting supports the following transforms:</p> <p>(<i>***Source: Rosette API Reference</i>)</p> <ul style="list-style-type: none"> ● ToLowercase—Transforms all uppercase Latin letters to lowercase (this includes both "half-width" and "full-width" Latin characters). ● ToUppercase—Transforms all lowercase Latin letters to uppercase (this includes both "half-width" and "full-width" Latin characters). ● ToFullwidth—Transforms all half-width characters that also have a full-width representation to their full-width form. <p>Characters with full-width representations are: Roman alphabet characters (A-z), digits (0-9), Japanese <i>katakana</i> characters, and the most commonly used punctuation characters (including Space).</p> <ul style="list-style-type: none"> ● ToHalfwidth —ransforms all full-width characters that also have a half-width representation to their half-width form. <p>Characters with half-width representations are: Roman alphabet characters (A-z), digits (0-9), Japanese <i>katakana</i> characters, and the most commonly used punctuation characters (including Space).</p> <ul style="list-style-type: none"> ● ToHiragana—Transforms all full-width <i>katakana</i> characters to <i>hiragana</i>. <p>To convert half-width <i>katakana</i> characters to <i>hiragana</i>, you must first convert the characters to full-width using the FullWidth transform.</p> <ul style="list-style-type: none"> ● ToParagraphSeparator—Standardizes the line/paragraph separators in the text according to the following standards: <p>Standard Code Point Line/Paragraph Separator</p> <p>Windows 0x0D0A 0x0D0A</p> <p>Macintosh 0x0D ToCR</p> <p>UNIX 0x0A ToLF</p> <p>Unicode U+2028 ToLineSeparator</p> <p>Unicode U+2029 ToParagraphSeparator</p> <p>EBCDIC 0x15 ToEBCDICNewLine</p> <ul style="list-style-type: none"> ● HankakuKatakanaToZenkaku—Converts half-width (<i>hankaku</i>) Japanese <i>katakana</i> characters to the full-width (<i>zenkaku</i>) form.

Function	Description
	<p>This conversion is almost identical to ToFullwidth, except that it automatically composes and combines katakana "accent" marks (<i>dakuten</i> and <i>handakuten</i>) appropriately, whereas ToFullwidth does not provide any special treatment for these marks.</p> <ul style="list-style-type: none"> ● ZenkakuKatakanaToHankaku—Converts full-width (<i>zenkaku</i>) Japanese katakana characters to the half-width (<i>hankaku</i>) form. <p>This conversion is almost identical to ToHalfwidth, except that it automatically decomposes and separates katakana "accent" marks (<i>dakuten</i> and <i>handakuten</i>) appropriately, whereas ToHalfwidth does not provide any special treatment for these marks.</p>
unicode	<p>Returns a Unicode string from the string of hexadecimal values provided. The syntax of the literal for UNICODE is</p> <pre>' [whitespace U+ \u]XXXX...'</pre> <p>where <i>x</i> is a valid hexadecimal digit: 0-9, a-f, or A-F. The hexadecimal value will always be in big-endian form.</p> <p>Syntax: <i>dst_var</i> = unicode(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = text literal, column, variable or expression ● <i>dst_var</i> = text variable <p>Example: let \$uniStr = unicode ('U+5E73 U+2294')</p>

Miscellaneous Functions

Note:

Miscellaneous functions return a string value unless otherwise indicated.

Table 52 Miscellaneous Functions

Function	Explanation
array	<p>Returns a pointer to the starting address of the specified array field. The value returned can only be used by a user-defined function. See <i>printarray</i> in UFUNC.C.</p> <p>Syntax: <i>array_var</i> = array(<i>array_name</i>, <i>field_name</i>)</p> <ul style="list-style-type: none"> ● <i>array_name</i> = text literal, column, variable, or expression ● <i>field_name</i> = text literal, column, variable, or expression ● <i>array_var</i> = text variable <p>Example: let #fstatus = printarray(array('products', 'name'), 10, 2, 'c')</p>
command_line	<p>Returns command line arguments passed to SQR (or SQRRT).</p> <p>Syntax: <i>dst_var</i> = command_line()</p> <ul style="list-style-type: none"> ● <i>dst_var</i> = text variable <p>Example: let \$cmdline = command_line()</p>
cond	<p>Returns <i>y_value</i> if <i>x_value</i> is nonzero (0); otherwise, returns <i>z_value</i>. If <i>y_value</i> is numeric, then <i>z_value</i> must also be numeric; otherwise, date and textual arguments are compatible. If either</p>

Function	Explanation
	<p><i>y_value</i> or <i>z_value</i> is a date variable, column, or expression, a date is returned. The return value of the function depends on which value is returned.</p> <p>Syntax: <i>dst_var</i> = cond(<i>x_value</i>, <i>y_value</i>, <i>z_value</i>)</p> <ul style="list-style-type: none"> ● <i>x_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ● <i>y_value</i> = Any literal, column, variable, or expression ● <i>z_value</i> = Any literal, column, variable, or expression ● <i>dst_var</i> = Any variable <p>Example: let #avg = #total / cond(&rate != 0, &rate, 1)</p>
getenv	<p>Returns the value of the environment variable. If the environment variable does not exist, an empty string is returned.</p> <p>Syntax: <i>dst_var</i> = getenv(<i>env_value</i>)</p> <ul style="list-style-type: none"> ● <i>env_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$myuser = getenv('USER')</p>
getfilemapname	<p>Returns the mapped filename. In Oracle Enterprise Performance Management Workspace, Fusion Edition, if the filename has a mapped equivalent the mapped filename is returned; otherwise, the filename is returned unchanged. Outside of EPM Workspace, the filename is returned unchanged.</p> <p>Syntax: <i>dst_var</i> = getfilemapname(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = text variable <p>Example: let \$realfile = getfilemapname('data.fil') ! get real filename is run under EPM Workspace let #Status = System('cp ' \$RealFile '/tmp') ! Copy to temp directory</p>
isnull	<p>Returns one (1) if <i>source_val</i> is null; otherwise, returns zero (0).</p> <p>Syntax: <i>dst_var</i> = isnull(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = date or text literal, column, variable, or expression ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #null = isnull(\$date)</p>
isnumber	<p>Returns one (1) if <i>source_value</i> is a number; otherwise, returns zero (0). A number is defined to be of the form: [Sign] [Digits] [.Digits] [E] e [Sign] Digits]. Leading and trailing blanks are ignored.</p> <p>Syntax: <i>dst_var</i> = isnumber(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = text literal, column, variable, or expression ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #isnumber = isnumber(\$string)</p>
nvl	<p>Returns <i>y_value</i> if the <i>x_value</i> is null; otherwise, returns <i>x_value</i>. If <i>x_value</i> is numeric, <i>y_value</i> must also be numeric; otherwise, date and textual arguments are compatible. In any case, the <i>x_value</i> determines the type of expression returned. The return value of the function depends on which value is returned.</p> <p>Syntax: <i>dst_var</i> = nvl(<i>x_value</i>, <i>y_value</i>)</p> <ul style="list-style-type: none"> ● <i>x_value</i> = Any literal, column, variable, or expression

Function	Explanation
	<ul style="list-style-type: none"> ● <i>y_value</i> = Any literal, column, variable, or expression ● <i>dst_var</i> = Any variable <p>Example: let \$city = nvl(&city, '-- not city --')</p> <p>If <i>x_value</i> is a date and <i>y_value</i> is textual, then <i>y_value</i> is validated according to the following rules:</p> <ul style="list-style-type: none"> ● For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by SQR_DB_DATE_FORMAT, one of the database-dependent formats (see Table 61, “Default Formats by Database,” on page 251), or the database-independent format 'SYYYYMDD[HH24[MI[SS[NNNNNN]]]]'. ● For DATE columns, Production Reporting uses the format specified by SQR_DB_DATE_ONLY_FORMAT, or the format in Table 62, “DATE Column Formats,” on page 252. ● For TIME columns, Production Reporting uses the format specified by SQR_DB_TIME_ONLY_FORMAT, or the format in Table 63, “TIME Column Formats,” on page 252.
range	<p>Returns one (1) if <i>x_value</i> is between <i>y_value</i> and <i>z_value</i>; otherwise, returns zero (0). If the first argument is text or numeric, the other arguments must be of the same type. If the first argument is a date, the remaining arguments can be dates and/or text. It is also possible to perform a date comparison on a mix of date and text arguments, for example, where <i>x_value</i> is a date and <i>y_value</i> and <i>z_value</i> are text arguments. In a comparison of this sort, <i>y_value</i> must represent a date that is earlier than that of <i>z_value</i>.</p> <p>Syntax: <i>dst_var</i> = range(<i>x_value</i>, <i>y_value</i>, <i>z_value</i>)</p> <ul style="list-style-type: none"> ● <i>x_value</i> = Any literal, column, variable, or expression ● <i>y_value</i> = Any literal, column, variable, or expression ● <i>z_value</i> = Any literal, column, variable, or expression ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #inrange = range(&grade, 'A', 'D') let #inrange = range(\$date, \$startdate, \$enddate) let #inrange = range(\$date, \$startdate, '15-Apr-97') let #inrange = range(#price, #low, #high)</p> <p>If <i>x_value</i> is a date and <i>y_value</i> and/or <i>z_value</i> is textual, then <i>y_value</i> and/or <i>z_value</i> is validated according to the following rules:</p> <ul style="list-style-type: none"> ● For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by SQR_DB_DATE_FORMAT, one of the database-dependent formats (see Table 61, “Default Formats by Database,” on page 251), or the database-independent format 'SYYYYMDD[HH24[MI[SS[NNNNNN]]]]'. ● For DATE columns, Production Reporting uses the format specified by SQR_DB_DATE_ONLY_FORMAT, or the format in Table 62, “DATE Column Formats,” on page 252. ● For TIME columns, Production Reporting uses the format specified by SQR_DB_TIME_ONLY_FORMAT, or the format in Table 63, “TIME Column Formats,” on page 252.
roman	<p>Returns a string that is the character representation of <i>source_value</i> expressed in lower case roman numerals.</p> <p>Syntax: <i>dst_var</i> = roman(<i>source_value</i>)</p> <ul style="list-style-type: none"> ● <i>source_value</i> = text literal, column, variable, or expression. ● <i>dst_var</i> = text variable

Function	Explanation
	Example: <code>let \$roman = roman(#page-count)</code>
wrapdepth	<p>Returns the number of print lines required by <i>source_value</i>. See the WRAP argument in PRINT for detailed descriptions of the parameters to this function. This function returns a float value.</p> <p>Syntax: <code>dst_var = wrapdepth(source_value, wrap_width, line_height, on, strip)</code></p> <ul style="list-style-type: none"> ● <i>source_value</i> = text literal, column, variable, or expression ● <i>wrap_width</i> = decimal, float, or integer literal, column, variable, or expression ● <i>line_height</i> = decimal, float, or integer literal, column, variable, or expression ● <i>on</i> = text literal, column, variable, or expression. ● <i>strip</i> = text literal, column, variable, or expression ● <i>dst_var</i> = decimal, float, or integer variable <p>Example: <code>let #depth = wrapdepth(&description,40,1,'<13>')</code></p>

Writing Custom Functions

In addition to using Production Reporting's built-in functions, you can write your own functions in C using the supplied source file UFUNC.C (or EXTUFUNC.C on Windows). If the C routine accesses a C++ routine, rename the file to UFUNC.CPP. (This enables the C++ compiler to recognize that it is a C++ source file.) The supplied UFUNC.C source file is compatible with both C and C++ compilers.

You can pass any number of arguments to your function. Values can be returned by the function or passed back in variables. Arguments and return values can either be numeric, single byte character strings, or UTF-8 encoded character strings. The specifics on how to specify the various argument and return types are explained in the UFUNC.C module. When using UTF-8 encoded strings, set `UseUnicodeInternal=TRUE` in SQR.INI.

After editing and recompiling the UFUNC.C module, you must recreate the Production Reporting executables. For UNIX, execute the `sqrmake` script located in the `lib` directory. For Windows, use the `sqrext.mak` make file located in the `lib` directory to recreate the DLL module.

When adding a new user-defined function to UFUNC.C, follow these rules:

- For routines that return a string value, define the routine to take the following arguments:
 - (int) Number of arguments
 - (char *) or (double *) Array of argument pointers, to either `char[]` or `double`.
 - (char *) Address of the result string. If unchanged, the function returns a NULL string.
 - (int) Maximum length of the result string, in bytes.
- For routines that return a numerical value, define the routine to take the following arguments:
 - (int) Number of arguments
 - (char *) or (double *) Array of argument pointers, to either `char[]` or `double`.
 - (double *) Address of the result value. If unchanged, the function returns a zero.

Following is an example of how to add a user-defined function to Production Reporting so that it can be invoked using LET, IF, or WHILE. The example adds a new function called rand, which returns a random number. The function accepts a single parameter used as the seed to start a new sequence of numbers. If the seed is zero, the same sequence is used.

To add the rand function to the UFUNC.C module, make the following modifications:

1. Add the prototype for the function.

```
LINKAGE void sqr_ufunc_rand CC_ARGS((int, double *[], double *));
```

2. Add the function name to the declaration table.

- The name in the table must be in lower case; however, you can reference it in either upper case or lower case in your Production Reporting program.
- The name of the function called from Production Reporting is rand.
- The return type is n for numeric, the number of arguments is “1”, and the argument type is n for numeric.
- The function name in the UFUNC.C module is sqr_ufunc_rand.
- You must enter the characters “PVR” before the function name.

Name	Return_Type	Number of Arguments	Arg_Types	Function
{(char *)"max",	'n',	0,	(char *)"n",	PVR sqr_ufunc_max},
{(char *)"split",	'n',	0,	(char *)"C",	PVR sqr_ufunc_split},
{(char *)"printarray",	'n',	4,	(char *)"cnnc",	PVR sqr_ufunc_printarray},
{(char *)"system",	'n',	1,	(char *)"c",	PVR sqr_ufunc_sys},
{(char *)"sleep",	'n',	1,	(char *)"n",	PVR sqr_ufunc_sleep},
{(char *)"rand",	'n',	1,	(char *)"n",	PVR sqr_ufunc_rand},
{(char *)"unittest",	'u',	2,	(char *)"uU",	PVR sqr_ufunc_unittest},
/* Last entry must be NULL -- do not change */				
{(char *)"", '\0', 0, (char *)"", 0}				

3. At the end of the UFUNC.C module add the sqr_ufunc_rand routine.

```
/*
 * RandNumb function -- Get random number and optionally set seed
 *
 * Usage: LET #Number = rand(#Seed)
 *
 */
LINKAGE void sqr_ufunc_rand CC_ARGL((argc, argv, result))
CC_ARG(int, argc) /* Number of actual arguments */
CC_ARG(double *, argv[]) /* Pointers to arguments */
```



```

CC_LARG(double *, result)    /* Where to store result */
{
#if defined(UNIX)
    if (*argv[0] > 0)        /* If seed > 0 then set it */
        srand48((unsigned int)*argv[0]);
        *result = drand48(); /* Get random number */
#else
    if (*argv[0] > 0)
        srand((unsigned int)*argv[0]);
        *result = (double)rand()/(double)(RAND_MAX);
#endif
    return;
}

```

4. After you make these modifications, compile the UFUNC.C module and recreate the Production Reporting executables.

The following is an example of a simple Production Reporting program that uses the newly added function:

```

BEGIN-PROGRAM
    DO Get-Random-Number
    DO Process-Calculations
END-PROGRAM

BEGIN-PROCEDURE Get-Random-Number
    LET #Seed = 44
    LET #Ran = RAND(#Seed)
END-PROCEDURE

BEGIN-PROCEDURE Process-Calculations
    .
    .
    .
END-PROCEDURE

```

LOAD-LOOKUP

Function

Loads an internal table with columns from the database. Allows for quick search using LOOKUP.

Syntax

In the SETUP section:

```

LOAD-LOOKUP
NAME=lookup_table_name
TABLE=database_table_name
KEY=key_column_name
RETURN_VALUE=return_column_name
[ROWS=initial_row_estimate_int_lit]
[EXTENT=size_to_grow_by_int_lit]
[WHERE=where_clause_txt_lit]

```

```

[SORT=sort_mode]
[QUIET]
[SCHEMA=schema_txt_lit]
[
  PROCEDURE=proc_txt_lit
  [PARAMETERS=({arg1 [IN|INOUT]}|NULL) [[, argi [IN|INOUT]] |
  NULL] ... )]
  (or)
  COMMAND=command_txt_lit
  (or)
  GETDATA=getdata_txt_lit
]
[{FROM-ROWSETS=({m|m-n|m-|-n} [, ...])|{ALL}}]|
[FROM-PARAMETER=parameter_txt_lit}]

```

In the body of the report:

```

LOAD-LOOKUP
NAME=lookup_table_name
TABLE=database_table_name
KEY=key_column_name
RETURN_VALUE=return_column_name
[ROWS=initial_row_estimate_lit|_var|_col]
[EXTENT=size_to_grow_by_lit|_var|_col]
[WHERE=where_clause_txt_lit|_var|_col]
[SORT=sort_mode]
[QUIET]
[SCHEMA={txt_lit|_var}]
[
  PROCEDURE={txt_lit|_var}
  [PARAMETERS=({arg1 [IN|INOUT]}|NULL) [[, argi [IN|INOUT]] |
  NULL] ... )]
  (or)
  COMMAND={txt_lit|_var}
  (or)
  GETDATA={txt_lit|_var}
]
[{FROM-ROWSETS=({m|m-n|m-|-n} [, ...])|{ALL}}]|
[FROM-PARAMETER={txt_lit|_var}}]

```

Note:

The following LOAD-LOOKUP elements are specific to Production Reporting DDO:

- SCHEMA
- PROCEDURE
- COMMAND
- GETDATA
- FROM ROWSETS
- FROM PARAMETER

The following LOAD-LOOKUP elements are *not* supported (processed but not used) in Production Reporting DDO:

- TABLE
- WHERE
- SORT-DC
- SORT-DI

Arguments

NAME

Name of the lookup table referenced in LOOKUP.

TABLE

Name of the table in the database, where the KEY and RETURN_VALUE columns or expressions are stored (not supported for DDO).

KEY

Name of the column used as the key in the array that is used for looking up the information. Keys can be character, date, or numeric data types. If numeric, Production Reporting permits only integers 12 digits or less for the KEY column. Keys can be any database-supported expression. See the RETURN_VALUE argument.

RETURN_VALUE

Name of the column (expression) returned for each corresponding key.

You can combine several columns into an expression if you need several fields returned for each lookup. You can do this by concatenating columns. (This is not supported for DDO.)

The following example is for ORACLE. See your database manual for the correct syntax.

```
RETURN_VALUE='name || '-' || country || '-' || population'
```

ROWS

(Optional) Initial size of the lookup table. If not specified, a value of 100 is used.

EXTENT

(Optional) Amount to increase the array when it becomes full. If not specified, a value of 25% of the ROWS value is used.

WHERE

WHERE clause used to select a subset of all the rows in the table. If specified, the selection begins after the word WHERE. The WHERE clause is limited to 255 characters (not supported for DDO).

SORT

Sorting method.

- DC—Database sorts data, case-sensitive sort (not supported for DDO)
- DI—Database sorts data, case-insensitive sort (not supported for DDO)

- SC—Production Reporting sorts data, case-sensitive sort
- SI—Production Reporting sorts data, case-insensitive sort

The default is SC or the method specified by the `-LL` command-line flag. The DI method is applicable only to databases that provide this feature and have been installed in that manner.

QUIET

Suppresses the message *Loading lookup array...* when the command executes. The warning message stating the number of duplicate keys found is also suppressed.

SCHEMA (DDO only)

Identifies the location in the datasource of the object being queried. You can enter the following options under SCHEMA:

- PROCEDURE—Name of datasource-stored procedure to execute. If the datasource is SAP R/3, this procedure is a BAPI. The name may include spaces.
- PARAMETERS—Scalar and/or list variables of the form `list_var|num_lit|txt_lit|txt_var|um_var|any_col`. If you do not specify the keywords IN or INOUT, IN is the default. Specify all parameters in order; leaving any parameters unnamed causes a syntax error. To ignore a parameter, fill its position with the keyword NULL. This results in a Null value for that parameter position.
- COMMAND—Text string passed to the datasource without modification by Production Reporting. This string can include embedded Production Reporting variables.
- GETDATA—Supports the Java (DDO) GetData paradigm for data access.

FROM-ROWSETS (DDO only)

Special case addition to the LOAD-LOOKUP syntax. Available for use with all datasource types, including SAP R/3 and JDBC. Names the rowset(s) from which to retrieve the column variables. For multiple row sets, use identical column name/type signatures. Row set numbers must be sequential from left-to-right within the parentheses, and must not overlap as in this example: (1-3, 2-4). Numeric literals or #variables are allowed.

In the FROM ROWSETS argument, “m” and “n” are integer values (1, 2, 3, 4, 5). “m-n” is 3-5 (rowsets 3, 4, 5). “m-” is 4- (rowsets 4, 5). “-n” is -3 (rowset 1, 2, 3).

FROM-PARAMETER (DDO only)

Special case addition to the LOAD-LOOKUP syntax. Available only for SAP R/3 datasources. Use only in conjunction with the PROCEDURE keyword. This argument names an output parameter containing one or more rows from which the column variables are retrieved.

Note:

This is a similar concept to the PARAMETERS = statement in DECLARE-CONNECTION and ALTER-CONNECTION, except that the properties specified here alter the flow of returned information, as opposed to simply setting login properties. Can be used in conjunction with any data-access model (Procedure, Command, Getdata). An application of this statement would be

in the MDB setting, where it might be used to specify such things as Level, Generation, or Include-Column. For example, PROPERTIES = ('SetColumn' = 5)

Description

Use `LOAD-LOOKUP` with one or more `LOOKUP` commands.

`LOAD-LOOKUP` retrieves two columns from the database, the `KEY` field and the `RETURN_VALUE` field. Rows are ordered by `KEY` and stored in an array.

`LOAD-LOOKUP` commands specified in the `SETUP` section are always loaded and cannot reference variables for the `ROWS`, `EXTENT`, and `WHERE` arguments.

When you use `LOOKUP`, Production Reporting searches the array (with a “binary” search) to find the `RETURN_VALUE` corresponding to the `KEY` referenced in the lookup.

Usually this type of lookup can be done with a database join, but joins take substantially longer. However, if your report is small and the number of rows joined is small, using a lookup table can be slower, since the entire table has to be loaded and sorted for each report run.

By default, Production Reporting lets the database sort the data. This works fine if the database and Production Reporting both use the same character set and collating sequence. The `SORT` argument allows you to specify the sorting method if this is not true. Additionally, if the machine that Production Reporting is running on is faster than the machine the database is running on, letting Production Reporting perform the sort could decrease the execution time of the report.

The only limit to the size of a lookup table is the amount of memory your computer has available. You could conceivably load an array with many thousands of rows. The binary search is performed quickly regardless of how many rows are loaded.

Except for the amount of available memory, there is no limit to the number of lookup tables that can be defined.

Examples

The following command loads the array `states` with the columns `abbr` and `name` from the database table `stateabbrs` where `country` is “USA.”

```
load-lookup
name=states
rows=50
table=stateabbrs
key=abbr
return_value=name
where=country='USA'
```

The preceding array is used in the example for `LOOKUP` to retrieve the full text of a state name from the abbreviation.

The following example uses `LOOKUP` to validate data entered by a user using `INPUT`:

```
get_state:
input $state 'Enter state abbreviation'
uppercase $state
lookup states $state $name
if $name = '' ! Lookup didn't find a match
```

```

    show 'No such state.'
    goto get_state
end-if

```

Surround any command argument with embedded spaces by single quotes, as shown here:

```
where='country='USA'' and region = 'NE''
```

The entire `WHERE` clause is surrounded by quotes. The two single quotes around `USA` and `NE` are translated to one single quote in the SQL statement.

The following example uses joins in `LOAD-LOOKUP` by including two tables in `TABLE` and the join in `WHERE`:

```

load-lookup
name=states
rows=50
sort=sc
table='stateabbrs s, regions r'
key=abbr
return_value=name
where='s.abbr = r.abbr and r.location = 'ne''

```

The following example uses multiple columns as the `KEY` for `LOAD-LOOKUP`:

```

begin-program
  load-lookup
    name=emp
    table=emp
    key='ename||','||job_title'
    return_value=comm
  do main
end-program

begin-procedure main
  lookup emp 'Martin,Salesperson' $comm
  print $comm (+1,1)
end-procedure

```

See Also

[LOOKUP](#)

LOOKUP

Function

Searches a lookup table (an array) for a key value and returns the corresponding text string.

Syntax

```
LOOKUP lookup_table_name {key_any_lit|_var|_col}
      {ret_txt_var|_date|_var}
```

Arguments

lookup_table_name

Name of the lookup table. This table must be previously loaded with `LOAD-LOOKUP`.

key_any_lit|_var|_col

Key used for the lookup.

ret_txt_var|_date|_var

String variable into which to return the corresponding value.

Description

Speeds up processing for long reports. For example, to print the entire state name rather than the abbreviation, use `LOAD-LOOKUP` followed by `LOOKUP`.

Examples

The following example works in conjunction with the example for `LOAD-LOOKUP`:

```
lookup states &state_abbrev $state_name
```

This example searches the *states* lookup table for a matching *&state_abbrev* value; if found, it returns the corresponding state name in *\$state_name*. If not found, a null is placed in *\$state_name*.

See Also

[LOAD-LOOKUP](#)

LOWERCASE

Function

Converts a text variable to lowercase.

Syntax

```
LOWERCASE txt_var
```

Arguments

txt_var

Text variable to convert to lowercase.

Description

Converts the contents of a text variable to lowercase.

Examples

```
input $answer 'Type EXIT to stop'
lowercase $answer ! Allows user to enter upper or lowercase.
if $answer = 'exit'
    ...etc...
```

See Also

The *lower* function listed in [Table 52, “Miscellaneous Functions,”](#) on page 212.

MBTOSBS

Function

Converts a double-byte string to its single-byte equivalent.

Syntax

```
MBTOSBS {txt_var}
```

Arguments

txt_var

String to convert.

Description

Converts the specified string as follows: Any occurrence of a double-byte character that also has a single-byte representation (numerals, punctuation, roman characters, and katakana) is converted.

See Also

The `TO_SINGLE_BYTE` function of [LET](#)

MOVE

Function

Moves one field to another field and optionally edits the field.

Syntax

```
MOVE {src_any_lit|_var|_col} TO dst_any_var
[[:$] format_mask|NUMBER|MONEY|DATE]
```

Arguments

src_any_lit|*_var*|*_col*

Specifies any source column, variable, or literal.

Note:

A date can be stored in a date variable or column, or a string literal, column, or variable. When using a date *format_mask* or the keyword DATE with MOVE, the source, if a string literal, column, or variable, must be in the format specified by SQR_DB_DATE_FORMAT, one of the database-dependent formats in [Table 61, “Default Formats by Database,” on page 251](#), or the database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.

When numerical precision is important, use LET. When no edit mask is specified, MOVE uses the 6 digit precision default mask.

dst_any_var

A destination variable.

format_mask

Optional format mask. (see [“Edit Masks” on page 247](#))

NUMBER

Formats *src_any_lit|_var|_col* with the NUMBER-EDIT-MASK from the current locale. Not legal with date variables. (see [“Edit Masks” on page 247](#))

MONEY

Formats *src_any_lit|_var|_col* with the MONEY-EDIT-MASK from the current locale. Not legal with date variables. (see [“Edit Masks” on page 247](#))

DATE

Formats *src_any_lit|_var|_col* with the DATE-EDIT-MASK from the current locale. Not legal with numeric variables. (see [“Edit Masks” on page 247](#))

Description

Moves the source field to the destination field. Optionally, you can reformat the field using the *format_mask* argument. Source and destination fields can be different types, numeric, text, or date. MOVE is also useful for converting from one type to another; however, date and numeric variables are incompatible.

When a date variable or column is moved to a string variable, the date is converted according to the following rules:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by SQR_DB_DATE_FORMAT. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,” on page 251](#).

- For DATE columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252.](#)
- For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252.](#)

Finally, as this example shows, the edit mask can be contained in a string variable.

Examples

This example illustrates the various features of `MOVE`:

The following code:

```
!
! Convert a string in place
!
move '123456789' to $ssn
move $ssn to $ssn xxx-xx-xxxx
show '$SSN = ' $ssn
```

Produces the following output:

```
$SSN = 123-45-6789
```

The following code:

```
!
! Convert a number to a string using an edit mask
!
move 1234567.89 to #value
move #value to $value 999,999,999.99
show '$Value = ' $value
```

Produces the following output:

```
$Value = 1,234,567.89
```

The following code:

```
!
! Convert a number to a string using a variable edit mask
!
move 123 to #counter
move '099999' to $mask
move #counter to $counter :$mask
show '$Counter = ' $counter
```

Produces the following output:

```
$Counter = 000123
```

The following code:

```
!
! Convert a number to a string using the default edit mask
```

```
!  
! Production Reporting, by default, outputs six digits of precision.  
! If you require more or less precision, specify an edit mask.  
!  
move 123.78 to #defvar  
move #defvar to $defvar  
show '$DefVar = ' $defvar
```

Produces the following output:

```
$DefVar = 123.780000
```

The following code:

```
!  
! Convert the number to a string using the locale default  
! numeric edit mask  
!  
alter-locale number-edit-mask = '99,999,999.99'  
move 123456.78 to #nvar  
move #nvar to $nvar number  
show '$NVar = ' $nvar
```

Produces the following output:

```
$NVar = 123,456.78
```

The following code:

```
!  
! Convert the money value to a string using the locale default  
! money edit mask  
!  
alter-locale money-edit-mask = '$9,999,999.99'  
move 123456.78 to #mvar  
move #mvar to $mvar money  
show '$MVar = ' $mvar
```

Produces the following output:

```
$MVar = $ 123,456.78
```

The following code:

```
!  
! Convert the date column to a string using the locale default  
! date edit mask  
!  
begin-select  
dcol  
  from tables  
end-select  
alter-locale date-edit-mask = 'Mon-DD-YYYY'  
move &dcol to $dvar date  
show '$DVar = ' $dvar
```

Produces the following output:

```
$DVar = Jan-01-1999
```

The following code:

```
!  
! Reset date to first day of the month  
! ($date1 and $date2 have been defined as date variables)  
!  
let $date1 = datenow()  
move $date1 to $date2 'MMYYYY'  
show '$Date2 = ' $date2 edit 'MM/DD/YY HH:MI'
```

Produces the following output if the report was run in October of 1995.

```
$Date2 = 10/01/95 00:00
```

The following code:

```
!  
! Convert date to a string  
! ($date1 has been defined as a date variable)  
!  
move $date1 to $str_date 'DD-MON-YYYY'  
show '$Str_Date = ' $str_date
```

Produces the following output.

```
$Str_Date = 01-DEC-1995
```

The following code:

```
!  
! Convert string (in partial format of SYYYYMMDDHHMISSNNN) to a  
! date  
!  
move '19951129' to $date1  
show '$Date1 = ' $date1 edit 'Mon DD YYYY HH:MI'
```

Produces the following output.

```
$Date1 = Nov 29 1995 00:00
```

See Also

- [LET](#) for information on copying, editing, or converting fields
- The `EDIT` parameter of [PRINT](#) for edit mask descriptions
- [ALTER-LOCALE](#) for descriptions of `NUMBER-EDIT-MASK`, `MONEY-EDIT-MASK`, and `DATE-EDIT-MASK`
- [PRINT](#) for the default date-time components as a result of moving an incomplete date to a date variable

MULTIPLY

Function

Multiplies one number by another.

Syntax

```
MULTIPLY {src_num_lit|_var|_col} TIMES dst_num_var  
[ROUND=nn]
```

Arguments

src_num_lit|*_var*|*_col*

Numeric source column, variable, or literal.

dst_num_var

Destination numeric variable.

ROUND

Rounds the result to the specified number of digits to the right of the decimal point. For float variables, this value can be from 0 to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate.

Description

MULTIPLY multiplies the first field by the second and places the result into the second field.

When dealing with money-related values (dollars and cents), use decimal variables rather than float variables. Float variables are stored as double precision floating point numbers, and small inaccuracies can appear when multiplying many numbers in succession. These inaccuracies can appear due to the way different hardware and software implementations represent floating point numbers.

Examples

```
multiply &quantity times #cost  
multiply 1.5 times #result
```

See Also

- [ADD](#)
- [LET](#) for a discussion of complex arithmetic expressions

NEW-PAGE

Function

Writes the current page and begins a new one.

Syntax

```
NEW-PAGE [erase_from_line_num_lit|_var|_col]
```

Arguments

erase_from_line_num_lit|_var|_col

Numeric column, variable, or literal for line printers.

Description

For line printers, `NEW-PAGE` can optionally erase the old page starting at a specified line. After this action is performed, the location on the page is unchanged—that is, the value of `#CURRENT-LINE` is the same. The default action is to erase the entire page and reset `#CURRENT-LINE` to its initial value for the page.

In reports where an overflow page is needed, sometimes it is useful to retain information from the first page on succeeding pages.

Each `NEW-PAGE` occurrence adds a form feed character to the output file unless you specify `FORMFEED=NO` in the `DECLARE-LAYOUT` for this program in the `SETUP` section.

Note:

A `NEW-PAGE` automatically occurs if page overflow is detected. Tabular reports do not require explicit `NEW-PAGE` commands; use `NEXT-LISTING` instead.

Examples

```
new-page 5
    ! Write current page, then erase it beginning at line 5.
```

NEW-REPORT

Function

Closes the current report output file and opens a new one with the specified filename.

Syntax

```
NEW-REPORT {report_filename_txt_lit|_var|_col}
```

Arguments

report_filename_txt_lit|_var|_col

A new file name.

Description

`NEW-REPORT` is normally used with single reports. When used with multiple report declarations, `NEW-REPORT` affects the *current* report only.

The internal page counter is reset to 1 when `NEW-REPORT` is executed.

Note:

Production Reporting does not actually create a report output file until the first page is completed. It is possible that NEW-REPORT will not create a new file, for example, if no data is selected and nothing is printed on the page.

Examples

The following example shows the NEW-REPORT command:

```
new-report 'rep2a.lis'
new-report $next-file
```

You can assign the report filename within an Production Reporting report by issuing NEW-REPORT before printing. You might even prompt for the filename to use, as shown here:

```
begin-report
  input $file 'Enter report filename'
  new-report $file
  ...
```

After NEW-REPORT executes, the reserved variable *\$sqr-report* updates to reflect the new report name.

See Also

- [DECLARE-REPORT](#) and [USE-REPORT](#)
- The `-F` command-line flag

NEXT-COLUMN

Function

Sets the current position on the page to the next column defined with [COLUMNS](#).

Syntax

```
NEXT-COLUMN [AT-END={NEWLINE|NEWPAGE}]
[GOTO-TOP={num_lit|_var|_col}]
[ERASE-PAGE={num_lit|_var|_col}]
```

Arguments

AT-END

Takes effect if the current column is the last one defined when NEXT-COLUMN is invoked.

GOTO-TOP

Causes the current line in the next column to be *num_lit|_var|_col*. This argument is useful when printing columns down the page.

ERASE-PAGE

Where to begin erasing the page when an AT-END=NEWPAGE occurs.

Examples

The following example prints columns across the page:

```
columns 10 50! Define two columns
  begin-select
  name   (0,1,20)
  phone  (0,+3,0) edit (xxx)bxxx-xxxx
  next-column at-end=newline ! Print names across the page
  order by name              ! from phonelist within two columns.
end-select
```

The following example prints columns down the page:

```
columns 10 50
move 55 to #bottom_line
begin-select
  name   (0,1,20)
  phone  (0,+3,0) edit (xxx)bxxx-xxxx
  if #current-line >= #bottom_line
    next-column goto-top=1 at-end=newpage
  else
    position (+1,1)
  end-if
  from phonelist
  order by name
end-select
```

See Also

[COLUMNS](#) and [USE-COLUMN](#)

NEXT-LISTING

Function

Ends the current set of detail lines and begins another.

Syntax

```
NEXT-LISTING [NO-ADVANCE]
[SKIPLINES={num_lit|_var|_col}]
[NEED={num_lit|_var|_col}]
```

Arguments

NO-ADVANCE

Suppresses any line movement when no printing has occurred since the previous NEXT-LISTING or NEW-PAGE. The default increments the line position even when nothing was printed.

SKIPLINES

Number of lines to skip before setting up the new offset.

NEED

Minimum number of lines needed to begin a new listing or set of detail lines. If this number of lines does not exist, a new page is started. You can use NEED to prevent a group of detail lines from being broken across two pages.

Description

Used in tabular reports, NEXT-LISTING causes a new vertical offset in the page.

After NEXT-LISTING executes, line 1 is reset one line below the deepest line previously printed in the page body. That is, if you then write PRINT (1, 5), the string is printed on the next available line starting in column 5. Note that the Production Reporting reserved variable #current-line still reflects the actual line number within the page body.

SKIPLINES must be a nonnegative integer. If it is less than 0, then 0 is assumed.

NEED must be an integer greater than 0. If it is less than or equal to 0, then 1 is assumed.

Examples

```
begin-select
  cust_num (1,1)edit 099999          ! Each detail group prints
  city(+3)                          ! starting on line 1 since
  name(2,10,30)                     ! NEXT-LISTING keeps
  address(+2)                       ! moving line 1 down the
  next-listing skiplines=1 need=2 ! page. NEED=2 keeps 2
  from customers order by cust_num  ! line detail groups from
end-select                          ! breaking across
                                     ! pages.
```

Note:

NEXT-LISTING automatically issues a Use-Column 1 command if columns are active.

OPEN

Function

Opens an operating system file for reading or writing.

Syntax

```
OPEN {filename_lit|_var|_col} AS
{filenum_num_lit|_var|_col}
{FOR-READING|FOR-WRITING|FOR-APPEND}
{RECORD=length_num_lit|_var|_col[:FIXED[:FIXED_NOLF[:VARY: BINARY]]]}
[STATUS=num_var]
[ENCODING={_var|_col|ASCII|ANSI|SJIS|JEUC|EBCDIC|EBCDIK290|EBCDIK1027|
UCS-2|UTF-8|others...}]
```

Note:

The ENCODING directive is only allowed when converting to Unicode internally.

Arguments

filename_lit|_var|_col

The file name. The file name can be literal, variable, or column. This makes it easy to prompt for a file name at run time.

filenum_num_lit|_var|_col

Number that identifies the file in the application. All file commands use the file number to reference the file. File numbers can be numeric variables as well as literals. The number can be any positive integer less than 64,000.

FOR-READING

When a file is opened for reading, Production Reporting procures all data sequentially. Production Reporting does not allow for random access of information.

FOR-WRITING

When a file is opened for writing, a new file is created. If a file of the same name already exists, it can be overwritten (this depends on the operating system).

FOR-APPEND

When a file is opened in append mode, the current file contents are preserved. All data written is placed at the end of the file. Production Reporting creates the file if one does not already exist. For existing files, make sure the attributes used are the same as those used when the file was created. Failure to do this can produce unpredictable results.

RECORD

For the VARY file type, this is the maximum size for a record. For the FIXED file type, this is the size of each record without the line terminator. For the FIXED_NOLF file type, this is the size of each record.

FIXED

Defines that all records contained within the file are the same length. Terminate each record by a line terminator (system dependent). You can use this file type when writing or reading binary data.

FIXED_NOLF

Defines that all records contained within the file are the same length with no line terminators. When writing records, Production Reporting pads short records with blank characters to ensure each record is the same length. This file type can be used when writing or reading binary data.

VARY

Defines that the records can be of varying length. Each record is terminated by a line terminator (system-dependent). Only records containing display characters (no binary data) can be used safely. When reading records, any data beyond the maximum length specified is ignored. This is the default file type.

STATUS

Sets the numeric variable to zero if `OPEN` succeeds and to -1 if it fails. Without the `STATUS` argument, a failure on `OPEN` causes Production Reporting to halt. By using a `STATUS` variable, you can control what processing should occur when a file cannot be opened.

ENCODING

Allows differently encoded files to be managed in a single run of Production Reporting. When no encoding is specified, Production Reporting uses the file input or output encoding specified in the INI file unless the file is UCS-2 encoded and auto-detection of UCS-2 files is enabled. Encoding is only allowed when converting to Unicode internally.

Description

After a file is opened, it remains open until explicitly closed by the `CLOSE` command. A maximum of 256 files can be opened at one time.

Examples

```
open 'stocks.dat' as 1 for-reading record=100
open 'log.dat'    as 5 for-writing  record=70
open $filename   as #j for-append   record=80:fixed
open $filename   as 2 for-reading  record=80:fixed_nolf
open $filename   as 6 for-reading  record=132:vary status=#filestat
if #filestat != 0
    ... error processing ...
end-if
```

See Also

[READ](#), [WRITE](#), and [CLOSE](#) for information about using files

OPEN-RS

Function

Opens a row set.

Syntax

```
OPEN-RS
NAME=row_set_name_var|_lit|_col
FILENAME=file_name_var|_lit|_col
COLUMN=({name_var|_lit|_col},{type_var|_lit|_col})
```

Arguments

NAME

Name of the row set.

FILENAME

Name of the external file used to hold the row set.

COLUMN

Column name and data type. Can be repeated as many times as needed. The column name is case-sensitive.

Description

OPEN-RS is used to instantiate the specified row set. When executed, the specified external file is created. If it already exists, the current contents are replaced. OPEN-RS can reside in any section except [BEGIN-SETUP](#), [BEGIN-SQL](#), and [BEGIN-DOCUMENT](#). Validation rules include:

- The row set specified by *row_set_name* must not be active, or an exception is thrown.
- The external file specified by *file_name* must be writable. An exception is thrown if the file cannot be created or written to.
- Column names must be unique within the row set.
- The data type must be Integer, Double, Decimal, String, Date, Time, DateTime, or Binary.
- If both the name and data type for a column are empty strings, then the corresponding COLUMN entry is ignored.
- You must define at least one active COLUMN entry, or an exception is thrown.

The row set file is an XML file. You can define whether to create the XML file in a BI Publisher (BIP) format or an SQR format in the `FormatForRowsetXML` entry in the [Default-Settings] section of SQR.INI.

Example

```
Begin-Report
  Open-RS Name='customer' FileName='customer.xml'
    Column = ('cust_num', 'integer')
    Column = ('name', 'string')
    Column = ('addr1', 'string')
    Column = ('addr2', 'string')
    Column = ('city', 'string')
    Column = ('state', 'string')
    Column = ('zip', 'string')
    Column = ('phone', 'string')
    Column = ('tot', 'integer')
  Begin-Select
  cust_num
  name
  addr1
  addr2
  city
```

```

state
zip
phone
tot
Write-RS Name='customer'
  Value = ('cust_num', &cust_num)
  Value = ('name', &name)
  Value = ('addr1', &addr2)
  Value = ('addr2', &addr2)
  Value = ('city', &city)
  Value = ('state', &state)
  Value = ('zip', &zip)
  Value = ('phone', &phone)
  Value = ('tot', &tot)
from customers
order by cust_num
End-Select
Close-RS Name='customer'
End-Report

```

See Also

[CLOSE-RS](#), [WRITE-RS](#)

PAGE-NUMBER

Function

Places the current page number on the page.

Syntax

```
PAGE-NUMBER position [pre_txt_lit [post_txt_lit]]
```

Arguments

position

Position of the page number.

pre_txt_lit

Text string to print before the page number.

post_txt_lit

Text string to print after the page number.

Description

The text specified in *pre_txt_lit* and *post_txt_lit* are printed immediately before and after the number.

Examples

```
begin-footing 1
  page-number(1,37) 'Page '!' Will appear as
  last-page () ' of ' '!' "Page 12 of 25."
end-footing
```

See Also

[LAST-PAGE](#)

POSITION

Function

Sets the current position on a page.

Syntax

```
POSITION position
[@document_marker [COLUMNS{num_lit|_var|_col}
[num_lit|_var|_col]...]]
```

Arguments

@document_marker

A location defined in a DOCUMENT paragraph. In this case, the position used is the location of that marker in the text of the document.

COLUMNS

The columns beginning at the location of the document marker. The columns defined are relative to the position of the document marker.

When COLUMNS is used, the entire command cannot be broken across more than one program line.

Examples

```
position (12,5)! Set current position to line 12, column 5.
position (+2,25)! Set position 2 lines down, at 25th column.
position () @total_location! Set position to document
print #total () edit 999,999,999! marker @total_location.
position () @name_loc columns 1 30
print name ()! Columns are defined at @name_loc and
next-column! 29 characters to the right of @name_loc
print title ()
```

See Also

- [COLUMNS](#)

- The examples with the description of a DOCUMENT paragraph in “Creating Form Letters” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*

PRINT

Function

Prints data on the page at a specified position.

Syntax

```
PRINT {any_lit|_var|_col} position[format_command[format_cmd_params]...]
```

Arguments

any_lit|_var|_col

Data to print.

position

Position where the data is printed. (For additional information, see “Changing Fonts” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*.)

format_command[*format_cmd_params*]

Optional formatting commands and parameters.

Note:

Dates can be contained in a date column or variable, or in a string literal, column, or variable. When using EDIT or DATE with PRINT, a date in a string literal, column, or variable must be in an acceptable format. See the description for “EDIT” on page 244 for further details.

Note:

Production Reporting DDO does not support printing of List variables.

Description

PRINT has the following format commands:

BACKGROUND	EDIT	POINT-SIZE
BOLD	FILL	SHADE
BOX	FONT	UNDERLINE
BOX-FILL-COLOR	FOREGROUND	URL
BOX-LINE-COLOR	ITALIC	URL-TARGET
CENTER	MATCH	WRAP

CENTER-WITHIN MONEY
 CODE-PRINTER NOP
 DATE NUMBER
 DELAY ON-BREAK

Some format commands can be used with others and some are mutually exclusive. In Table 53, “•” indicates which format commands can be used together.

Table 53 Valid Print Format Command Combinations

	BOLD	BOX	BOX-FILL-COLOR	BOX-LINE-COLOR	CENTER	CENTER-WITHIN	CODE-PRINTER	DELAY	EDIT NUMBER MONEY DATE	FILL	FONT	F/B	ITALIC	MATCH	NOP
BOLD		•	•	•	•	•		•	•	•	•	•	•	•	•
BOX	•		•	•	•	•		•	•	•		•	•	•	•
BOX-FILL-COLOR	•	•		•	•	•		•	•	•	•		•	•	•
BOX-LINE-COLOR	•	•	•		•	•		•	•	•	•	•	•	•	•
CENTER	•	•	•	•					•	•		•	•	•	•
CENTER-WITHIN	•	•	•	•					•	•		•	•	•	•
CODE-PRINTER															•
DELAY	•	•	•	•					•	•		•	•		•
EDIT NUMBER MONEY DATE	•	•	•	•	•	•			•			•	•		•
FILL	•	•	•	•	•	•		•				•	•		•
FONT	•	•	•	•	•	•			•	•		•	•	•	•
F/B	•			•	•	•		•	•	•			•	•	•
ITALIC	•	•	•	•	•	•		•	•	•	•			•	•
MATCH	•	•	•	•	•	•						•	•		•
NOP	•	•	•	•	•	•	•	•	•	•		•	•	•	

	BOLD	BOX	BOX-FILL-COLOR	BOX-LINE-COLOR	CENTER	CENTER-WITHIN	CODE-PRINTER	DELAY	EDIT NUMBER MONEY DATE	FILL	FONT	F/B	ITALIC	MATCH
ON-BREAK	•	•	•	•	•	•			•	•		•	•	•
POINT-SIZE	•	•	•	•	•	•			•	•	•	•	•	•
SHADE	•	•		•	•	•		•	•	•		•	•	•
UNDER-LINE	•				•	•	•		•	•	•	•	•	•
URL URL-TARGET	•	•	•	•	•	•	•		•	•	•	•	•	•
WRAP	•	•	•	•						•		•	•	

Note:

In the above table, F/B stands for FOREGROUND/BACKGROUND.

The following topics describe these format commands.

BOLD

Causes the string or number to print in **bold** type.

For HP LaserJets, the appropriate boldface font must be loaded in the printer.

For PostScript printers, the appropriate boldface must be defined in the PostScript startup file, POSTSCRI.STR. See [Table 32, “DECLARE-PRINTER Command Arguments,” on page 138](#) for information on which fonts you can bold.

For line printers, when DECLARE-PRINTER uses the BEFORE-BOLD and AFTER-BOLD arguments, the specified strings are added before and after the data to bold. If BEFORE-BOLD and AFTER-BOLD are not specified, then BOLD has no effect.

For example:

```
print &name (+1, 20) bold
print 'Your account is in arrears' (1,1) bold
```

BOX

Draws a one-line deep graphical box around printed data. BOX has no effect for line printers.

For example:

```
print &grand_total (+5, 20) box
print 'Happy Birthday !!' (1,1) box
```

Note:

For HP LaserJets using proportional fonts, BOX and SHADE cannot determine the correct length of the box since it varies with the width of the characters printed. BOX and SHADE work well with fixed-pitch fonts and with all PostScript fonts.

BOX-FILL-COLOR

The fill color used when BOX is specified. The default value is the FILL-COLOR value from the SET-COLOR command. You cannot use BOX-FILL-COLOR with SHADE.

```
print 'Hello World' (5,5) box box-fill-color=('green')
```

BOX-LINE-COLOR

The line color used when BOX is specified. The default value is the LINE-COLOR value from the SET-COLOR command.

```
print 'Hello World' (5,5) box box-line-color=('red')
```

CENTER

Centers the field on a line. The position qualifier for column is ignored. For example:

```
print 'Quarterly Sales' (1) center
```

CENTER-WITHIN

Centers the field within the specified number of characters. The centered text is relative to the column value in the position qualifier. For example:

```
print 'Hello World' (+5,10) center-within=40
```

CODE-PRINTER

Adds nondisplay characters to the program for sending a sequence to the printer.

Syntax

```
CODE-PRINTER printer_type
```

Valid values for *printer_type* are HT, HP, PS, LP, HTML, HPLASERJET, POSTSCRIPT, and LINEPRINTER.

CODE-PRINTER places the string “behind” the page buffer, rather than within it, so alignment of printed data is not thrown off by the white space consumed by the nondisplay characters. Only strings can be printed using CODE-PRINTER.

Since the report might be printed on different types of printers, you should specify for which type this data is used. The report is ignored if printed to a different type. If necessary, you can send a different sequence to another type with a second PRINT statement.

For example:

```
encode '<27>[5U' into $big_font
encode '<27>[6U' into $normal_font
...
print $big_font (0, +2) code-printer=lp
print &phone () edit '(xxx) xxx-xxxx'
print $normal_font () code-printer=lp
```

In the previous example, the two CODE-PRINTER arguments put the *\$big_font* and *\$normal_font* sequences into the output, without overwriting any data in the page buffer. Sequences printed with the CODE-PRINTER argument are positioned using the regular line and column positioning. However, unlike PRINT, the current print location after execution is the beginning location where the CODE-PRINTER string was placed. Multiple coded strings printed using the same line and column location appear in the output in the same sequence in which they were printed.

DATE

Formats the field using the DATE-EDIT-MASK from the current locale. (See “ALTER-LOCALE” on page 41.) If not defined, the date prints according to the rules in Table 54.

Table 54 Date Formats if Column Type Not Set

Column Type	Default Mask	If not set
DATETIME	SQR_DB_DATE_FORMAT	See Table 61, “Default Formats by Database,” on page 251 for the format.
DATE	SQR_DB_DATE_ONLY_FORMAT	See Table 61, “Default Formats by Database,” on page 251 for the format.
TIME	SQR_DB_TIME_ONLY_FORMAT	See Table 63, “TIME Column Formats,” on page 252 for the format.

You cannot use DATE with numeric columns or variables.

DELAY

Delays the printing of the data until a SET-DELAY-PRINT command is issued against the variable used in PRINT. For example:

```
PRINT $Last_User (1,10) Delay
.
```

SET-DELAY-PRINT \$Last_User with &Username

EDIT

Syntax

EDIT=*edit_format*

Edits each field before printing it. The three types of edits are:

- Text edit (see [Table 55 on page 244](#))
- Numeric edit (see [Table 56 on page 244](#))
- Date edit (see [Table 57 on page 245](#))

Text Edit Format Characters

Table 55 Text Edit Format Characters

Character	Description
X	Use character in field.
B	Insert blank.
~ (tilde)	Skip character in field.
R[n]	Reverse sequence of string, for languages such as Hebrew. The optional number indicates right justification within length indicated.

Any other character (for example, punctuation) in a text edit mask is treated as a constant and is included in the edited field.

The characters 8, 9, 0, V, and \$ are illegal in a text edit mask because they are used to indicate that the mask is for a numeric edit.

Numeric Edit Format Characters

Table 56 Numeric Edit Format Characters

Character	Description
8	Digit, zero fill to the right of the decimal point, trim leading blanks (left justify the number).
9	Digit, zero fill to the right of the decimal point, space fill to the left.
0	Digit, zero fill to the left.
\$	Dollar sign, optionally floats to the right.

Character	Description
B	Treated as a “9”, but if a value is zero, the field is converted to blanks.
C	Entered at the end of the mask, causes the comma and period characters to be transposed when the edit occurs. This is to support monetary values where periods delimit thousands and commas delimit decimals. (Example: 1.234,56).
E	Scientific format. The number of 9s after the decimal point determines the number of significant digits displayed. The “E” can be upper or lower case; the display follows the case of the mask.
V	Implied decimal point.
MI	Entered at the end of the mask, causes a minus to display at the right of the number.
PR	Entered at the end of the mask, causes angle brackets (< >) to display around the number if the number is negative.
PS	Entered at the end of the mask, causes parentheses to display around the number if the number is negative.
PF	Entered at the end of the mask, causes floating parentheses to display around the number if the number is negative.
NA	Entered at the end of the mask, causes “N/A” to display if the numeric column variable is null. The case of N/A follows that of the mask.
NU	Entered at the end of the mask, causes blanks to display if the numeric column variable is null.
.	Decimal point.
,	Comma.

Note:

Characters other than those listed in [Table 56](#) are illegal for numeric edit masks and cause errors during processing.

Date Edit Format Characters

Table 57 Date Edit Format Characters

Character	Description
YYY YY Y	Last 3, 2, or 1 digit(s) of year. On input, for calculating the 4-digit year, the current century and/or decade are used. For example, a ‘9’ using the ‘Y’ mask would result in 1999 as the year if the current year is in the 1990s.
YYYY SYYYY	4 digit year, “S” prefixes BC dates with “-”.
RR	Last 2 digits of year; for years in other centuries. See Table 58 on page 247 .
CC or SCC	Century; “S” prefixes BC dates with “-”.
BC AD	BC/AD indicator.

Character	Description
Q	Quarter of year (1,2,3,4; JAN-MAR=1).
RM	Roman numeral month (I-XII; JAN=I).
WW	Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year.
W	Week of the month (1-5) where week 1 starts on the first day of the month and ends on the seventh.
DDD	Day of year (1-366).
DD	Day of month (1 - 31).
D	Day of week (1-7). Sunday is first day of week.
DAY	Name of day.
DY	Abbreviated name of day.
ER	Japanese Imperial Era. Returns the name of the of the Japanese Imperial Era in the appropriate kanji ('Heisei' is the current era).
EY	Year of the Japanese Imperial Era. Returns the current year within the Japanese Imperial Era. Note: The common Japanese date format is: 'YYYY<nen>MM<gatsu>DD<nichi>' where <nen>, <gatsu>, and <nichi> are the kanji strings for year, month, and day respectively.
J	Julian day; the number of days since Jan 1, 4713 BC. Numbers specified with 'J' must be integers.
AM PM	Meridian indicator.
HH	Assumes 24 hour clock unless meridian indicator specified.
HH12	Hour of day (1-12).
HH24	Hour of day (0-23).
SSSSS	Seconds past midnight (0-86399).
N NN NNN NNNN NNNNN NNNNNN	Fractions of a second. Precise to microseconds; however, for most hardware and databases, this much accuracy will not be attainable.
MONTH	Name of month.
MON	Abbreviated name of month.
MM	Month (01-12; JAN=01).
MI	Minute (0-59).
SS	Second (0-59).
 	Used to concatenate different masks.

Table 58 Date Edit Format Code-RR

Last 2 digits of current year	2-digit year is 00 - 49	2-digit year is 50 - 99
00 - 49	The return date is in the current century.	The return date is in the century before the current one.
50 - 99	The return date is in the century after the current one.	The return date is in the current century.

Edit Masks

As you work with edit masks, keep in mind the following:

- When using text, date, and numeric scientific edit masks with `PRINT`, the specified width value of `PRINT` sets the length allocated for the data displayed. For all other numeric edit masks, the edit mask sets the allocated length.
- All masks can be used by the `strtodate` function except for `CC`, `SCC`, `Q`, `W`, and `WW`.
- A backslash forces the next character into the output from the mask. For example, a mask of “*The cu\rrre\nt \mo\nth is Month*” results in the output string of “*The current month is January*”. Without the backslashes the output string would be “*The cu95e7t january is January*”.
- You can use a vertical bar as a delimiter between format codes; however, in most cases the bar is not necessary. For example, the mask 'YYYY|MM|DD' is the same as 'YYYYMMDD'.
- Any other character (for example, punctuation) in a date edit mask is treated as a constant and is included in the edited field. If the edit mask contains spaces, you must enclose it in single quotes (').
- The masks `MON`, `MONTH`, `DAY`, `DY`, `AM`, `PM`, `BC`, `AD`, and `RM` are case-sensitive and follow the case of the mask entered. For example, if the month is January, the mask `Mon` yields “Jan” and `MON` yields “JAN”. All other masks are case-insensitive and can be entered in either uppercase or lowercase.
- National Language Support is provided for the following masks: `MON`, `MONTH`, `DAY`, `DY`, `AM`, `PM`, `BC`, and `AD`. See “[ALTER-LOCALE](#)” on page 41 or in [Chapter 7](#), “[Production Reporting Samples](#)” for additional information.

If the value of the date field being edited is “Mar 14 1996 9:35”, the edit masks produce the results in [Table 59](#).

Table 59 Sample Date Edit Masks

Edit Mask	Result
<code>dd/mm/yy</code>	14/03/96
<code>DD-MON-YYYY</code>	14-MAR-1996
<code>'Month dd, YYYY'</code>	March 14, 1996
<code>MONTH-YYYY</code>	MARCH-1996

Edit Mask	Result
HH:MI	09:35
'HH:MI PM'	09:35 AM
YYYYMMDD	19960314
MM.DD.YYYY	03.14.1996
Mon	Mar
DD D DDD	143073

- In addition to the `EDIT` argument, you can use edit masks with `MOVE`, `CONCAT`, `DISPLAY`, and `SHOW`, and with the `edit` function of `LET`. Edit the field using the supplied mask before storing or displaying it.
- When a date with missing date and/or time components displays or prints, the defaults are as follows:
 - The default year is the current year.
 - The default month is the current month.
 - The default day is one.
 - The default time is zero (00:00:00.000000).

For example, assuming today is September 7, 1996, the following assignment would produce an equivalent date-time of September 1, 1996 13:21:00.000000:

```
let $date1 = strtodate('13:21', 'HH:MI')
```

- You can dynamically change edit masks by storing them in a string variable and referencing the variable name preceded by a colon (:).

For example:

```
move '$999,999.99' to $mask
print #total (5,10) edit :$mask
show #total edit :$mask
```

- When a date stored in a string literal, column, or variable prints with an edit mask, it must be in one of the following formats:
 - The format specified by `SQR_DB_DATE_FORMAT`, or the corresponding setting in `SQR.INI`.
 - One of the database-dependent formats in [Table 61, “Default Formats by Database,” on page 251](#).
 - The database-independent format, 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.
- When a date column or variable prints without an edit mask, the date prints in the format specified by `SQR_DB_DATE_FORMAT` or the corresponding setting in `SQR.INI`. If this is not set, the date prints in the primary database format (the first entry) in [Table 61](#).

This applies to `DISPLAY`, `MOVE`, and `SHOW` as well as `PRINT`.

Sample Edit Masks

Table 60 Sample Edit Masks and Resulting Fields

Mask	Value	Display
999.99	34.568	34.57
9,999,999V9999	123,456.7890	123,4567890
8,888,888.888	123,456.789	123,456.789
9,999	1234	1,234
9,999	123	123
09999	1234	01234
9999	-123	-123
9999	-1234	****
9999	12345	****
9999mi	-123	123-
9999pr	-123	< 123>
999999ps	-123	(123)
999999pf	-123	(123)
9999na	(null)	n/a
9999nu	(null)	(blank)
\$\$9,999.99c	1234.56	\$1.234,56
\$\$9,999.99	1234.56	\$1,234.56
\$\$9,999.99	12.34	\$ 12.34
\$\$\$,\$\$9.99	12.34	\$12.34
9.999e	123456	1.235e+05
B9,999	0	(blank)
B9,999	12345	12,345
(xxx)bxxx-xxxx	2169910551	(216) 991-0551
xxx-xx-xxxx	123456789	123-45-6789
~~xx~xx	ABCDEFGHIJ	CDFG
r10	ABCDEFGF	GFEDCBA

Uses of Edit Masks

The following example shows some uses of edit masks:

```
print #total (7,55,0) edit $999,999.99 ! $ 12,345.67
print #total (7,55,0) edit $$$9,999.99 ! $12,345.67
print #total (7,55,0) edit 999,999.99pr! < 12,345.67>(if
! neg)
print #comm (7,55,0) edit b99,999.99! Blank if zero
print #cnum (16,1,0) edit 099999! 001234
print #cat (5,10,0) edit 9.999E! 1.235E+04
print #phone (16,60,0) edit (xxx)bxxx-xxxx ! (216) 397-0551
print #total (7,55,0) edit fff9,999.99! Dollar-Symbol £
```

Edit Masks with Specified Width Value

The following examples show some uses of edit masks with a specified width value.

- **Text Edit Masks** – Print width sets the length allocated.

Print Statement	Display	Current Position
Print 'ABCDEFGHJ' (1,1,5) Edit xxxxxxxx	ABCDE	(1,1,6)
Print 'ABCDEFGHJ' (1,1,5) Edit xxxx	ABCDE	(1,1,6)
Print 'ABCDE' (1,1,10) Edit xxxxxxxx	ABCDE	(1,1,11)
Print 'ABCDE' (1,1,10) Edit xxxx	ABCDE	(1,1,11)

- **Date Edit Masks** – Print width sets the length allocated.

Print Statement	Display	Current Position
Print \$current-date (1,1,5) Edit DD/MM/YYYY	16/05	(1,1,6)
Print \$current-date (1,1,5) Edit DD/MM	16/05	(1,1,6)
Print \$current-date (1,1,10) Edit DD/MM/YYYY	16/05/2003	(1,1,11)
Print \$current-date (1,1,10) Edit DD/MM	16/05	(1,1,11)

- **Numeric Scientific Edit Masks** – Print width sets the length allocated.

Print Statement	Display	Current Position
Print 1234567890 (1,1,5) Edit 9e999999	1.23	(1,1,6)
Print 1234567890 (1,1,10) Edit 9e999999	1.234568e	(1,1,11)
Print 1234567890 (1,1,20) Edit 9e999999	1.234568e+009	(1,1,21)

- All other Numeric Edit Masks – Edit mask sets the length allocated.

Print Statement	Display	Current Position
Print 1234567890 (1,1,5) Edit 9999999999	12345	(1,1,11)
Print 1234567890 (1,1,5) Edit 99999	*****	(1,1,6)
Print 12345 (1,1,10) Edit 9999999999	____12345	(1,1,11)
Print 12345 (1,1,10) Edit 99999	12345	(1,1,6)
Print 1234567890 (1,1,5) Edit 8888888888	12345	(1,1,11)
Print 1234567890 (1,1,5) Edit 88888	12345	(1,1,6)
Print 12345 (1,1,10) Edit 8888888888	12345	(1,1,6)
Print 12345 (1,1,10) Edit 88888	12345	(1,1,6)

Default Formats

Review the following tables for information on default formats by database, date column formats, and time column formats.

Table 61 Default Formats by Database

Database	Default Formats
DB2	YYYY-MM-DD-HH:MI:SS.NNNNNN YYYY-MM-DD

Database	Default Formats
Informix	YYYY-MM-DD HH:MI:SS.NNN MM/DD/YYYY MM-DD-YYYY MM.DD.YYYY
ODBC	'MON DD YYYY HH:MIPM'
Oracle	DD-MON-YY
Sybase	MON DD YYYY HH:MIPM MON DD YYYY [HH:MI[:SS[:NNN]]][PM] MON DD YYYY [HH:MI[:SS[:.NNN]]][PM] YYYYMMDD [HH:MI[:SS[:NNN]]PM] YYYYMMDD [HH:MI[:SS[:.NNN]]PM]

Table 62 DATE Column Formats

Database	DATE Column Formats
DB2	YYYY-MM-DD
Informix	MM/DD/YYYY
ODBC	DD-MON-YYYY

Table 63 TIME Column Formats

Database	TIME Column Formats
DB2	HH24.MI.SS
ODBC	HH24:MI:SS

FILL

Fills the page with the specified character or string as indicated by the print position and length.

The following example prints a line of stars and then a line of dashes followed by stars:

```
print '*' (1,1,79) fill ! Fill line with *'s
print '-*' (+1,20,40) fill ! Fill with '-*' characters.
```

Note:

When using the Text, Numeric, and Date edit masks with PRINT, the specified width value of PRINT determines the length allocated for the displayed data. For all other "Numeric" edit masks, the edit mask sets the allocated length.

FONT

Prints the string in the specified font. For example:

```
print 'Hello world' (3,3) font=5
```

FOREGROUND/BACKGROUND

When you specify a color in `PRINT`, it has the same scope as `PRINT`. If you do not define the specified color name, then the setting for “default” is used. Use the color name “none” to turn off color for the specified area.

Syntax

```
PRINT {any_lit|_var|_col}

[FOREGROUND=({color_name_lit|_var|_col} | {rgb})]

[BACKGROUND=({color_name_lit|_var|_col} | {rgb})]
```

Note:

See the example in “[ALTER-COLOR-MAP](#)” on page 37 to better understand the `FOREGROUND` and `BACKGROUND` commands.

ITALIC

Prints a string or number in *italic* type. For example:

```
print &name (+1, 20) italic
print 'Your account is in arrears' (1,1) bold
```

Note:

Italic is not applicable for Line and PostScript printers.

MATCH

Compares a field to a list of key values and if a match is found, prints the corresponding string at the specified line and column.

If the *match_text* contains white space, it must be enclosed in single quotes (').

Any number of match text(s) can be tested, but each must have its own line, column, and *print_text*.

If a match is not found, the unmatched field is printed at the position specified in the parentheses.

Line and column positions for each matched string are treated as fixed or relative positions depending on the type of positioning used in the position qualifier for the `PRINT` command.

Syntax

```
MATCH match_text {line_num_lit|_var|_col}

{column_num_lit|_var|_col} print_text...
```

For example:

```
print &type_buyer (20,12) match
```

```
A 20 12 Casual
B 20 22 Impulsive
C 21 12 Informed
D 21 22 Choosey
```

To use relative line and fixed column positioning, you could enter:

```
print $state (0,25) match
  OH 0 25 Ohio
  MI 0 37 Michigan
  NY 0 25 'New York'
```

The column positions are treated as fixed locations due to the fixed “25” position declared in parentheses.

MONEY

Formats the column or variable using the `MONEY-EDIT-MASK` from the current locale. (See [“ALTER-LOCALE” on page 41.](#)) `MONEY` can only be used with a numeric column or variable.

NOP

Suppresses the print command, causing “no operation” to execute. `NOP` is useful for temporarily preventing a field from printing.

For example:

```
print &ssn (1,1) nop ! Hide the social security number.
```

NUMBER

Formats the column or variable using the `NUMBER-EDIT-MASK` from the current locale. (See [“ALTER-LOCALE” on page 41.](#)) `NUMBER` can only be used with a numeric column or variable.

ON-BREAK

Causes the specified action in a tabular report when the value of a field changes (a break occurs). The default action prints the field only when its value changes (`PRINT=CHANGE`).

Syntax

```
ON-BREAK [PRINT={ALWAYS|CHANGE|CHANGE/TOP-PAGE|NEVER}]
[SKIPLINES={num_lit|_var|_col}]
[PROCEDURE=procedure_name[(arg1[ ,argi]...)]]
[AFTER=procedure_name[(arg1[ ,argi]...)]]
[BEFORE=procedure_name[(arg1[ ,argi]...)]]
[SAVE=txt_var]
[LEVEL=nn]
[SET=nn]
```

ON-BREAK has the following qualifiers:

- **PRINT**—Specifies when the break field prints.
 - ALWAYS duplicates the break field for each detail group.
 - CHANGE prints the value only when it changes. This is the default.
 - CHANGE/TOP-PAGE prints the value both when it changes and at the top of each new page.
 - NEVER suppresses printing.
- **SKIPLINES**—Specifies how many lines to skip when the value changes.
- **PROCEDURE**—Specifies the procedure to invoke when the value changes. This qualifier cannot be used with either the AFTER or BEFORE qualifiers.
- **AFTER/BEFORE**—Specifies procedures to invoke either after or before the value changes. If no rows are fetched, neither procedure executes. You can only use AFTER and BEFORE within a SELECT paragraph.

Following is the sequence of events:

- **SAVE**—Indicates a string variable where the previous value of a break field is stored.
- **LEVEL**—Specifies the level of the break for reports containing multiple breaks. For example, a report sorted by state, county, and city might have three break levels: state is level 1 (the most major), and city is level 3 (the most minor). When a break occurs, other breaks with equal or higher level numbers are cleared. The level number also affects the sequence in which AFTER and BEFORE procedures are processed.
- **SET**—Assigns a number to the set of leveled breaks in reports with more than one set of independent breaks.

The sequence of events for a query containing ON-BREAK fields is:

1. Any BEFORE procedures are processed in ascending LEVEL sequence before the first row of the query is retrieved.
2. When a break occurs in the query, the following happens:
 - a. AFTER procedures are processed in descending sequence from the highest level to the level of the current break field.
 - b. SAVE variables are set with the new value.
 - c. BEFORE procedures are processed in ascending sequence from the current level to the highest level break.
 - d. Any breaks with the same or higher level numbers are cleared so they do not break on the next value.
 - e. If a PROCEDURE has been declared, the procedure is invoked.
 - f. If SKIPLINES was specified, the current line position is advanced.
 - g. The value is printed (unless PRINT=NEVER was specified).
3. After the query finishes (at END-SELECT) any AFTER procedures are processed in descending level sequence.

For example:

```
begin-select
state (+1,1,2) on-break level=1 after=state-tot skiplines=2
county (+2,14) on-break level=2 after=county-tot skiplines=1
city (+2,14) on-break level=3 after=city-tot
...
end-select
```

Breaks are processed as follows:

- When city breaks, the `city-tot` procedure is executed.
- When county breaks, first the `city-tot` procedure is executed, then the `county-tot` procedure is executed.
- When state breaks, the `city-tot`, `county-tot`, and `state-tot` procedures are processed in that sequence.

If any `BEFORE` breaks were indicated, they are processed automatically, after all of the `AFTER` breaks and in sequence from lower to higher level numbers.

For example:

```
begin-select
state (+1,1,2) on-break level=1 before=bef-state after=state-tot
county (+2,14) on-break level=2 before=bef-cnty after=cnty-tot
city (+2,14) on-break level=3 before=bef-city after=city-tot
...
end-select
```

Now when state breaks, the sequence of procedures executed is as follows:

1. City-tot
2. Cnty-tot
3. State-tot
4. Bef-state
5. Bef-cnty
6. Bef-city

Upon entering the query at `BEGIN-SELECT`, the three `BEFORE` procedures are executed in sequence:

1. Bef-state
2. Bef-cnty
3. Bef-city

After the last row is retrieved, at `END-SELECT`, the three `AFTER` procedures are executed in sequence:

1. City-tot
2. Cnty-tot
3. State-tot

The **SAVE** qualifier saves the previous break value in the specified string variable for use in an **AFTER** procedure. You may want to print the previous break field with a summary line:

```
print &state (+1,1) on-break after=state-tot save=$old-state
```

The **SET** qualifier allows you to have sub-reports with leveled breaks. By separating the **ON-BREAKs** into sets, the associated leveled breaks in each set will not interfere with each other.

```
begin-select  
state (+1,1,2) on-break set=1 after=state-tot level=1
```

SET=1 associates this leveled break with other breaks having the same set number.

POINT-SIZE

Prints the string in the specified point size. For example:

```
print 'This is large text' (5,5) point-size=36
```

SHADE

Draws a one-line deep, shaded graphical box around printed data. For line printers this argument has no effect.

```
print 'Company Confidential' (1,1) shade  
print &state (+2, 40) shade
```

Note:

For HP LaserJets using proportional fonts, **BOX** and **SHADE** are not able to determine the correct length of the box since it varies with the width of the characters printed. **BOX** and **SHADE** work well with fixed pitch fonts and with all PostScript fonts.

UNDERLINE

Prints the specified data with underlined characters. For line printers, **UNDERLINE** causes backspace and underscore characters to output, which emulates underlining.

For example:

```
print &name (+1, 45) underline  
print 'Your account is in arrears' (1,1) underline
```

URL

Creates a hypertext link to the specified address. (Production Reporting does not validate the address.) For example:

```
Print "My web page" (40,10) URL="http://www.somewebhost.com/~myusername/index.htm"
```

Creates a link to the following URL in your report:

<http://www.somewebhost.com/~myusername/index.htm>

When you click on the "My web page" your browser is directed to the page.

URL-TARGET

The target within the specified URL. (Production Reporting does not validate the target.) For example:

```
Print $URL (40,10)
  Point-Size=#Font
  Font=8
  URL=$URL
  URL-TARGET=$Target
  Background = (255, 0, 0)
  Foreground = ('yellow')
  Underline
```

WRAP

Wraps text at word spaces and moves additional text to a new line.

Syntax

```
WRAP {line_length_lit|_var|_col}
{max_lines_lit|_var|_col} [KEEP-TOP]
[STRIP=strip_chars] [ON=break_chars] [R]
[LINE-HEIGHT={line_height_lit|_var|_col}]

line_length_lit|_var|_col
```

The maximum paragraph width in characters.

Note:

After a string wraps, the current position is one character to the right of the last character in the column. When a string ends on the last position of a line, an implicit line feed causes the new current position to be the first character of the following line. In the SETUP section, use DECLARE-LAYOUT to make the page width one character wider than the right edge of the wrapped text to avoid generating an implicit line feed.

For example:

```
print  &comment  (48,20,0) wrap 50 3
print  &note1    (1,20,30) wrap 30 4
print  &note2    (1,+2,30) wrap 30 4
print  &note3    (1,+2,30) wrap 30 4
```

In this example, the paragraph is 50 characters wide with a maximum depth of 3 lines..The line position is 1 for each of the three wrapped fields: *note1*, *note2*, and *note3*. The current print position after a wrap occurs at the bottom right edge of the wrapped paragraph. To continue printing on the same line, you must use a fixed line number for the next field.

`max_lines_lit|_var|_col`

Specifies the maximum paragraph depth in lines. Usually, the line length and maximum lines are indicated with numeric literals. However, `WRAP` can also reference numeric variables or columns. This is useful when you want to change the width or depth of a wrapped paragraph during report processing. The numeric variable can optionally be preceded by a colon (:).

For example:

```
print $comments (1,30) wrap #wrap_width 6
print $message (5,45) wrap #msg_wid #msg_lines
```

`KEEP-TOP` retains the current line position except if a page break occurs, in which case, line 1 is used as the current line position. The default action is to set the next print position at the bottom of the wrapped data.

In the following example, the column `&resolution` prints on the same line as the first line of the column `&instructions`:

```
print &phone (+1,10) edit '(xxx) xxx-xxxx'
print &instructions (+1,10,30) wrap 6 10 keep-top
print &resolution (0,+3,25)
```

The `STRIP` and `ON` arguments affect which characters are to convert before wrapping, and which characters force a wrap to occur.

- Characters in the `STRIP` string argument are converted to spaces before the wrap occurs.
- Characters in the `ON` string argument cause a wrap at each `ON` character found. The `ON` character is not printed.

Both arguments accept regular characters and nondisplay characters whose ASCII values are surrounded by angled brackets, `<nn>`.

For example, to print a long data type that contains embedded carriage returns, the setup would be:

```
print &long_field (5,20) wrap 42 30 on=<13>
```

The paragraph wraps at each carriage return, rather than at the usual word boundaries. If the `ON` character is not found within the width specified for the paragraph, the wrap occurs at a word space.

The following example converts the `STRIP` characters to spaces before wrapping on either a line feed `<10>` or a space (the default):

```
print &description (20,10) wrap 50 22 strip=/\^\@<13> on=<10>
```

`WRAP` can also be used to print reversed characters, for support of languages such as Hebrew. An `R` after the length and `max_lines` arguments causes the field to be reversed before the wrap takes place. In addition, the entire paragraph is right-justified within the length indicated.

```
! Reverse wrap, in 30 character field.
print &comment (2,35) wrap 30 5 r
print $notes (1,50) wrap 50 7 r
```

`LINE-HEIGHT` specifies the number of lines to skip between each line of the wrapped data. By default a value of 1 (single space) is assumed.

The following example prints the *comment* column with one blank line between each printed line for a maximum of four printed lines:

```
print &comment (1,1) wrap 40 4 line-height = 2
```

See Also

- [LET](#) for information on copying, editing, or converting fields
- [ALTER-LOCALE](#) for a description of NUMBER-EDIT-MASK, MONEY-EDIT-MASK, and DATE-EDIT-MASK
- [DISPLAY](#) and [SHOW](#)

PRINT-BAR-CODE

Function

Prints bar codes.

Syntax

```
PRINT-BAR-CODE position  
{TYPE={bar_code_type_num_lit|_var|_col}}  
{HEIGHT={bar_code_height_num_lit|_var|_col}}  
{TEXT={bar_code_txt_lit|_var|_col}}  
[CAPTION={bar_code_caption_txt_lit|_var|_col}]  
[CHECKSUM={bar_code_checksum_txt_lit|_var|_col}]
```

Arguments

position

Position of the upper left corner. Position parameters can be relative. See [POSITION](#) for examples of relative positioning. Document markers are not allowed. After execution, the current position is returned to this location; however, the next listing line is the next line below the bottom of the bar code. (This is different than the way [PRINT](#) works.)

TYPE

Type of bar code to print. (See [Table 64, “Bar Code Types,” on page 261.](#))

HEIGHT

Height of the bar code in inches. The height must be between 0.1 and 2 inches. The code prints to the nearest one-tenth of an inch. For Zip+4 Postnet, the height of the bar code is fixed. The height should be between 0.2 and 2.0 for Zip+4 Postnet. If it is less than 0.2, the bar code extends above the position specified.

TEXT

Text to encode and print. The number and type of text characters permitted or required depends on the bar code type. See [Table 64, “Bar Code Types,” on page 261](#) for specifications.

CAPTION

Optional text to print under the bar code in the current font. Production Reporting attempts to center the caption under the bar code; however, for proportional fonts this may vary slightly.

CAPTION is not valid for Zip+4 Postnet. If specified, it is ignored.

CHECKSUM

Optional check sum to compute and print in the bar code. Valid values are YES and NO, where NO is the default.

Note:

Some bar code types ignore the CHECKSUM qualifier. See [Table 64](#) for those bar code types for which CHECKSUM is relevant.

Description

PRINT-BAR-CODE prints industry standard bar codes. Production Reporting supports the bar code types listed in [Table 64](#).

Table 64 Bar Code Types

Type	Description	Text Length	Text Type*	CHECKSUM RECOGNIZED
1	UPC-A	11, 13, or 16	9	
2	UPC-E	11, 13, or 16	9	
3	EAN/JAN-13	12, 14, or 17	9	
4	EAN/JAN-8	7, 9, or 12	9	
5	3 of 9 (Code 39)	1 to 100	9, X, p	y
6	Extended 3 of 9	1 to 100	9, X, x, p, c	y
7	Interleaved 2 of 5	2 to 100	9	y
8	Code 128	1 to 100	9, X, x, p, c	
9	Codabar	1 to 100	9, p	y
10	Zip+4 Postnet	5, 9, or 11	9	
11	MSI Plessey	1 to 100	9	y
12	Code 93	1 to 100	9, X, p	y
13	Extended 93	1 to 100	9, X, x, p	y
14	UCC-128	19	9	
15	HIBC	1 to 100	9	y

*9- Numbers (0-9) X- Upper Case Letters (A-Z) x- Lower Case Letters (a-z) p- Punctuation c- Control Characters

Note:

Production Reporting does not check bar code syntax. (For example, with bar code type 9, Codabar, you must add your own start/stop character to the text argument.) See your bar code documentation for the proper formatting of certain bar codes.

Examples

This example shows how to use PRINT-BAR-CODE to create a UPC-A bar code.

```
begin-program
  print-bar-code (3,1)
  type=1          ! UPC-A
  height=0.3
  text='01234567890'
  caption='0 12345 67890'
end-program
```



This example shows how to use PRINT-BAR-CODE to create a ZIP+4 Postnet code.

```
begin-program
  print 'John Q. Public'      (3,1)
  print '1234 Main Street'   (4,1)
  print 'AnyTown, USA 12345-6789' (5,1)
  print-bar-code              (7,1)
  type=10
  height=0.2
  text='12345678934'
end-program
```

```
John Q. Public
1234 Main Street
AnyTown, USA 12345-6789
```



This example references the last page value from within a bar code. When the report runs, the meta sequence %LAST-PAGE% will be replaced with the value of the last page of the report. This functionality is not available with bar code types 1,2,3,4,10 and 14.

```
begin-report
let $Caption = 'Page ' || Edit(#Page-Count, '88888') || ' of %LAST-PAGE%'
let $Text = Edit(#Page-Count, '88888') || '%LAST-PAGE%'
print-bar-code (,30)
  type=5
  height=0.3
  text=$Text
  caption=$Caption
  checksum=Yes
end-report
```

PRINT-CHART

Function

Prints a chart. Only PostScript printers or HP printers that support HPGL (generally, this is HPLaserJet 3 and higher) render chart output.

Syntax

```
PRINT-CHART [chart_name] position
[TYPE={chart_type_txt_lit|_var|_col}]
[CHART-SIZE=(chart_width_num_lit|_var|_col, chart_depth_num_lit|_var|_col)]
[TITLE={title_txt_lit|_var|_col}]
[SUB-TITLE={subtitle_txt_lit|_var|_col}]
[FILL={fill_txt_lit|_var|_col}]
[3D-EFFECTS={3d_effects_txt_lit|_var|_col}]
[BORDER={border_txt_lit|_var|_col}]
[COLOR-PALETTE=color_palette_lit|_var|_col]
[POINT-MARKERS={point_markers_txt_lit|_var|_col}]
[ATTRIBUTES={selector_lit|_var|_col|
  LIST:{selector_list_name_lit|_var|_col|
    (selector_lit|_var|_col,...)}, {decl_key_lit|_var|_col,
    {decl_value_lit|_var|_col|
      LIST:{decl_val_list_name_lit|_var|_col|
        (decl_val_lit|_var|_col,...)}}|
      PALETTE:{color_palette_lit|_var|_col}}},...}}]
[DATA-ARRAY=array_name]
[DATA-ARRAY-ROW-COUNT={x_num_lit|_var|_col}]
[DATA-ARRAY-COLUMN-COUNT={x_num_lit|_var|_col}]
[DATA-ARRAY-COLUMN-LABELS={NONE|array_name|({txt_lit|
_var|_col},...)}]
[DATA-LABELS={data_labels_txt_lit|_var|_col}]
[FOOTER-TEXT=NONE|text_lit|_var|_lit]
[SUB-FOOTER-TEXT=NONE|text_lit|_var|_col]
[ITEM-COLOR=(item_color_keyword|_lit|_var|_col,
{color_txt_lit|_var|_col}|(r,g,b))]
[ITEM-SIZE=(item_size_keyword_lit|_var|_col,
item_size_num_lit|_var|_col)]
[LEGEND={legend_txt_lit|_var|_col}]
[LEGEND-TITLE={legend_title_txt_lit|_var|_col}]
[LEGEND-PLACEMENT={legend_placement_txt_lit|_var|_col}]
[LEGEND-PRESENTATION={legend_presentation_txt_lit|_var|_col}]
[PIE-SEGMENT-QUANTITY-DISPLAY={pie_segment_quantity_
display_txt_lit|_var|_col}]
[PIE-SEGMENT-PERCENT-DISPLAY={pie_segment_percent_
display_txt_lit|_var|_col}]
[PIE-SEGMENT-EXPLODE={pie_segment_explode_txt_lit|_var|_col}]
[X-AXIS-GRID={x_axis_grid_txt_lit|_var|_col}]
[X-AXIS-LABEL={x_axis_label_txt_lit|_var|_col}]
[X-AXIS-MIN-VALUE={x_axis_min_value_num_lit|_var|_col}]
[X-AXIS-MAX-VALUE={x_axis_max_value_num_lit|_var|_col}]
[X-AXIS-MAJOR-INCREMENT={x_axis_major_increment_num_lit|_var|_col}]
[X-AXIS-MINOR-INCREMENT={x_axis_minor_increment_num_lit|_var|_col}]
[X-AXIS-MAJOR-TICK-MARKS={x_axis_major_tick_marks_txt_lit|_var|_col}]
[X-AXIS-MINOR-TICK-MARKS={x_axis_minor_tick_marks_txt_lit|_var|_col}]
```

```

[X-AXIS-TICK-MARK-PLACEMENT={x_axis_tick_mark_placement
_txt_lit|_var|_col}]
[X-AXIS-ROTATE={x_num_lit|_var|_col}]
[X-AXIS-SCALE={x_axis_scale_txt_lit|_var|_col}]
[Y-AXIS-GRID={y_axis_grid_txt_lit|_var|_col}]
[Y-AXIS-LABEL={y_axis_label_txt_lit|_var|_col}]
[Y-AXIS-MASK={mask_txt_lit|_var|_col}]
[Y-AXIS-MIN-VALUE={y_axis_min_value_num_lit|_var|_col}]
[Y-AXIS-MAX-VALUE={y_axis_max_value_num_lit|_var|_col}]
[Y-AXIS-MAJOR-INCREMENT={y_axis_major_increment_num_lit|_var|_col}]
[Y-AXIS-MINOR-INCREMENT={y_axis_minor_increment_num_lit|_var|_col}]
[Y-AXIS-MAJOR-TICK-MARKS={y_axis_major_tick_marks_txt_lit|_var|_col}]
[Y-AXIS-MINOR-TICK-MARKS={y_axis_minor_tick_marks_txt_lit|_var|_col}]
[Y-AXIS-TICK-MARK-PLACEMENT={y_axis_tick_mark_placement
_txt_lit|_var|_col}]
[Y-AXIS-SCALE={y_axis_scale_txt_lit|_var|_col}]
[Y2-AXIS-LABEL={y2_axis_label_txt_lit|_var|_col}]
[Y2-AXIS-MASK={mask_txt_lit|_var|_col}]
[Y2-AXIS-MIN-VALUE={y2_axis_min_value_num_lit|_var|_col}]
[Y2-AXIS-MAX-VALUE={y2_axis_max_value_num_lit|_num_lit|_var|_col}]
[Y2-AXIS-MAJOR-INCREMENT={y2_axis_major_increment_num_lit|_var|_col}]
[Y2-AXIS-MINOR-INCREMENT={y2_axis_minor_increment_num_lit|_var|_col}]
[Y2-AXIS-MAJOR-TICK-MARKS={y2_axis_major_tick_marks
_txt_lit|_var|_col}]
[Y2-AXIS-MINOR-TICK-MARKS={y2_axis_minor_tick_marks
_txt_lit|_var|_col}]
[Y2-AXIS-SCALE={y2_axis_scale_txt_lit|_var|_col}]
[Y2-COLOR-PALETTE=color_palette_lit|_var|_col]
[Y2-DATA-ARRAY=array_name]
[Y2-DATA-ARRAY-ROW-COUNT={x_num_lit|_var|_col}]
[Y2-DATA-ARRAY-COLUMN-COUNT={x_num_lit|_var|_col}]
[Y2-DATA-ARRAY-COLUMN-LABELS={NONE|array_name|
({txt_lit|_var|_col},...)}]
[Y2-TYPE={chart_type_txt_lit|_var|_col}]

```

Note:

If you do not define CHART-SIZE with this command, you must define it with DECLARE-CHART.

Arguments

chart_name

Name of the chart defined in DECLARE-CHART. This name is not necessary if you specify the CHART-SIZE and all other pertinent attributes in PRINT-CHART.

position

(row, column) Position of the upper left corner. Position parameters can be relative. See POSITION for examples of relative positioning. Document markers are not allowed. After execution, the current position is returned to this location; however, the next listing line is the next line below the bottom of the chart area. (This is different than the way the PRINT command works.)

Note:

For definitions of the other arguments in PRINT-CHART, see “[DECLARE-CHART](#)” on page 100. For information on NewGraphics, see NewGraphics under “[\[Default-Settings\] Section](#)” on page 328.

Description

PRINT-CHART directs Production Reporting to output a chart according to the named chart, if any, and the overridden attributes, if any.

As you use PRINT-CHART, keep in mind the following:

- All the arguments defined for DECLARE-CHART are valid for PRINT-CHART. (See [Table 20 on page 102](#) for argument descriptions.) The only exception is the *position* argument, which specifies the position of the chart. This argument is required and is only valid for PRINT-CHART.
- The data that supports the charts is defined in the following arguments: (Y2 arguments are for combination charts.)
 - DATA-ARRAY
 - DATA-ARRAY-ROW-COUNT
 - DATA-ARRAY-COLUMN-COUNT
 - DATA-ARRAY-COLUMN-LABELS
 - Y2-DATA-ARRAY
 - Y2-DATA-ARRAY-ROW-COUNT
 - Y2-DATA-ARRAY-COLUMN-COUNT
 - Y2-DATA-ARRAY-COLUMN-LABELS
- The following arguments must be specified in either DECLARE-CHART or PRINT-CHART:
 - DATA-ARRAY
 - DATA-ARRAY-ROW-COUNT
 - DATA-ARRAY-COLUMN-COUNT
- The following arguments are required for combination charts and must be specified in either DECLARE-CHART or PRINT-CHART:
 - Y2-DATA-ARRAY
 - Y2-DATA-ARRAY-ROW-COUNT
 - Y2-DATA-ARRAY-COLUMN-COUNT
 - Y2-TYPE
- PRINT-CHART can be used without referencing a named chart if all required attributes for the DECLARE-CHART are supplied in addition to all its required parameters.

- `PRINT-CHART` directs Production Reporting to display the chart on the current page using the attribute values at the moment the command is executed. Manipulation of chart attribute values has no effect on the appearance of the chart after `PRINT-CHART` is executed.

For example, if you execute a `PRINT-CHART` with `TITLE=$ttl` and `$ttl='Encouraging Results'`, and then change the value of `$ttl` to `'Discouraging Results'` immediately afterward, then the chart is printed with first value, `'Encouraging Results'`.

- `PRINT-CHART` expects the `DATA-ARRAY` to be organized in a particular way. See [Table 65 on page 266](#) for details.
- `PRINT-CHART` fills the area defined by `CHART-SIZE` as much as possible while maintaining an aesthetically pleasing ratio of height to width. In cases where the display area is not well suited to the chart display, the chart is centered within the specified region, and the dimensions are scaled to accommodate the region. Therefore, do not be alarmed if the chart does not fit exactly inside the box you have specified. It simply means that Production Reporting has accommodated the shape of the region to provide the best looking chart possible.
- Chart commands used to send output to a line printer are ignored. Only PostScript printers or HP printers that support Hewlett Packard's HPGL (generally, this is HP LaserJet model 3 and higher) render chart output. If you attempt to print a chart to a LASERJET printer that does not support HPGL, the HPGL command output will likely become part of your output, leaving one or more lines of meaningless data across your report.
- If the first field in the array designated by `DATA-ARRAY` is of type `CHAR`, then the value on the x-axis is the contents of that column. If the first field is not of type `CHAR`, then the value of the x-axis is the row number of the array designated by `DATA-ARRAY`, beginning with 1. Pie charts show the character value in the legend area. Histograms show the character value on the y-axis. XY-Scatter charts do not use the character value and none is needed in the array.
- If a `PIE` chart contains many small slices, the user must set the `PIE-SEGMENT-QUANTITY-DISPLAY` and/or `PIE-SEGMENT-PERCENT-DISPLAY` arguments to `NO` to prevent the values from one slice overwriting the values of another slice.

As was mentioned earlier, each chart type meets a specific organizational requirement.

[Table 65](#) and [Table 66](#) describe these requirements.

Table 65 Chart Array Field Types (fewer than four fields)

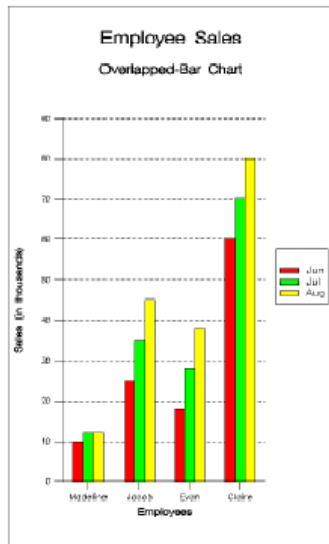
Chart Type	Field 0	Field 1	Field 2	Field 3
BUBBLE	Type=num X-Axis values	Type=num Y-Axis values	Type=num radius of bubble at (x,y)	
PIE	Type=char Pie segment labels, the names associated with each segment	Type=num The value associated with each pie segment	(Optional) Type=char Pie segment explode flag setting, 'Y' or 'N'	
LINE BAR	Type=char	Type=num	(Optional)	(Optional)

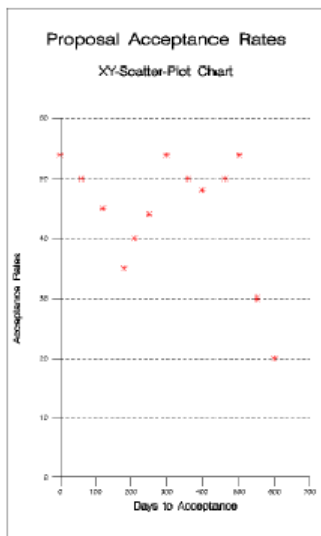
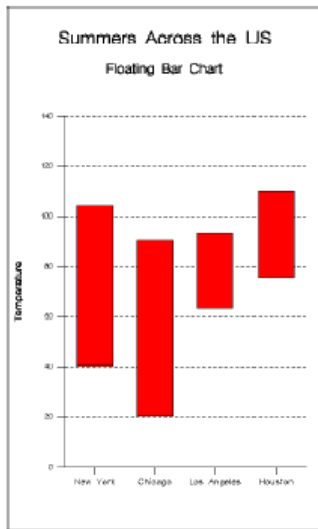
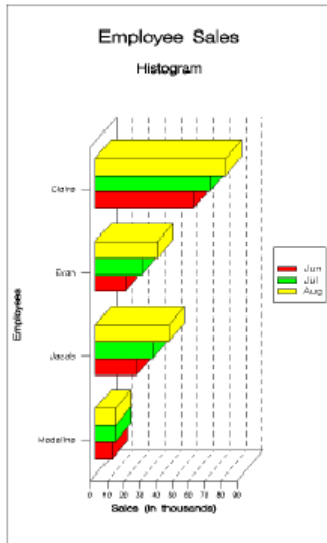
Chart Type	Field 0	Field 1	Field 2	Field 3
STACKED-BAR 100%-BAR OVERLAPPED-BAR HISTOGRAM AREA STACKED-AREA 100%-AREA	X-Axis values	Series 1 Y-Axis values	Type=num Series 2 Y-Axis values	Type=num Series 3... Y-Axis values
XY-SCATTER-PLOT	Type=num Series 1 X-Axis values	Type=num Series 1 Y-Axis values	(Optional) Type=num Series 2 X-Axis values	(Optional) Type=num Series 2 ... Y-Axis values
FLOATING-BAR	Type=char X-Axis values	Type=num Series 1 Y-Axis offset	Type=num Series 1 Y-Axis duration	(Optional) Type=Num Series 2 ... Y-Axis offset

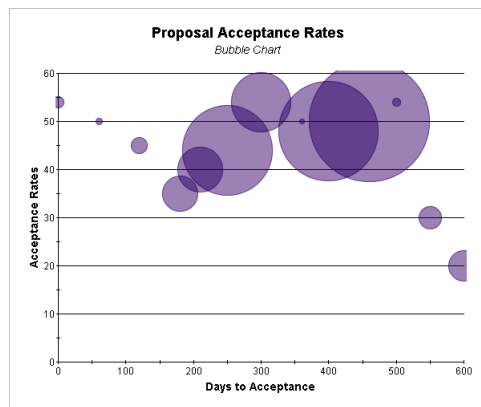
Table 66 Chart Array Field Types for HIGH-LOW-CLOSE

Chart Type	Field 0	Field 1	Field 2	Field 3	Field 4
HIGH-LOW-CLOSE	Type=char X-Axis values	Type=num High value	Type=num Low value	Type=num Closing value	(Optional) Type=num Opening value

Examples







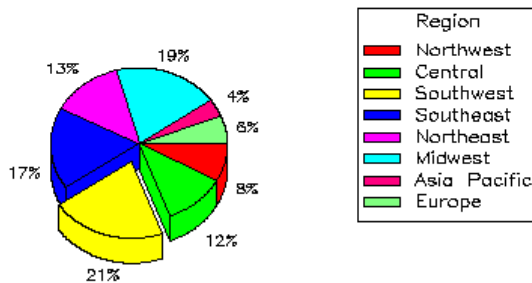
Note:

See “Creating Bubble Charts” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide* for information on creating a Bubble Chart.

See”Use ATTRIBUTES in DECLARE-CHART and PRINT-CHART” in Volume 1 of the *Production Reporting Developer's Guide* for information on combination charts created using the Y2-Axis syntax in DECLARE-CHART and PRINT-CHART.

In the “Sales by Region for the Year” example below, a pie chart is printed without explicit reference to a chart declared with DECLARE-CHART.

Sales by Region for the Year



You must supply all necessary arguments in PRINT-CHART as shown in the following code:

```

.
.
create-array
  name=q_four
  size=8
  field=name:char
  field=num:number=0.
.
print-chart (,1)
  title = 'Sales by Region for the Year'
  sub-title = NONE

```

```
chart-size = (40, 20)
type = pie
3d-effects = yes
legend-title = 'Region'
legend = yes
border = no
pie-segment-quantity-display = no
pie-segment-explode = max
data-array = q_four
data-array-column-count = 3
data-array-row-count = 8
```

See Also

[DECLARE-CHART](#)

PRINT-DIRECT

Function

Writes directly to the print output file without using the Production Reporting page buffer.

Syntax

```
PRINT-DIRECT
[NOLF]
[PRINTER={LINEPRINTER|POSTSCRIPT|HPLASERJET|HTML|LP|PS|HP|HT}]
{txt_lit|_var|_col}...
```

Arguments

NOLF

Defines that no carriage return and line feed is to print. By default, printed text is followed by a carriage return and line feed character.

PRINTER

Type of printer to which this text applies.

txt_lit|*_var*|*_col*

Text to print.

Description

PRINT-DIRECT can be used for special applications that cannot be accomplished directly with PRINT commands, such as initializing a page with graphics or other special sequences. Since this text is often printer-dependent and since the report can be printed on different types of printers that require different control characters, you can use the PRINTER qualifier to specify the printer type. If no PRINTER qualifier is specified, the command applies to all printer types.

When using PRINT-DIRECT with PRINT, the Production Reporting page buffer is copied to the output file only when each page is full or when a NEW-PAGE command is issued. One approach

is to use PRINT-DIRECT commands inside a BEFORE-PAGE or AFTER-PAGE procedure (declared with the DECLARE-PROCEDURE command), so they are coordinated with the information coming out of the page buffer.

Examples

```
print-direct printer=ps  '%%Page: ' $page-number
print-direct nolf printer=lp  reset
```

PRINT-IMAGE

Function

Prints an image.

Syntax

```
PRINT-IMAGE[ image_name] position
[TYPE={ image_type_lit | _var | _col}]
[IMAGE-SIZE=(width_num_lit | _var | _col, height_num_lit | _var | _col)]
[SOURCE={ file_name_lit | _var | _col}]
[[FOR-PRINTER=( {POSTSCRIPT|HPLASERJET|HTML|PDF|WINDOWS|PS|HP|HT|PD|WP|
printer_type_lit | _var | _col}, {image_type_lit | _var | _col}, {file_name_lit
_var | _col})]...]
```

Note:

DECLARE-IMAGE and PRINT-IMAGE work together to identify information about the image. The IMAGE-SIZE argument is required and must be defined in either DECLARE-IMAGE or PRINT-IMAGE. The SOURCE and TYPE arguments are optional; however, if you define one you must define the other.

Arguments

image_name

Name of an image specified by a DECLARE-IMAGE.

position

(row, column) Position of the upper left corner. Position parameters can be relative. See the POSITION command for examples of relative positioning. Document markers are not allowed. After execution, the current position is returned to this location; however, the next listing line is the next line below the bottom of the image area. (This is different from the way PRINT works.)

TYPE

Image type. Types can be EPS-FILE, HPGL-FILE, GIF-FILE, JPEG-FILE, BMP-FILE., PNG-FILE, or AUTO-DETECT.

IMAGE-SIZE

Width and height of the image in Production Reporting coordinates.

SOURCE

Name of a file containing the image. The file must be in the SQRDIR directory, or you must specify the full path.

FOR-PRINTER

Specific image file for each report output type.

Tip:

The TYPE and SOURCE arguments contain the default values. You can override these defaults for a specific printer by using the FOR-PRINTER argument. (See the second example under DECLARE-IMAGE.)

Description

PRINT-IMAGE can be placed in any section of a report except the SETUP section. The image file pointed to can be any file of the proper format.

PRINT-IMAGE can be used without referencing a named image if all required attributes for DECLARE-IMAGE are supplied in addition to all its required parameters.

If an image has not been declared, or if the image type is not supported for a particular report output type, or if the image file has incomplete header information, then a box (either shaded for HP printers or with a diagonal line through it for Postscript printers) appears where the image is expected. [Table 28, “Valid Images Types,” on page 131](#) illustrates the valid relationships between image type and report output type.

Examples

For PostScript:

```
print-image office-signature (50, 20)
print-image (50, 20)
  type = eps-file
  source = 'sherman.eps'
  image-size = (10, 3)
```

For Windows:

```
print-image company-logo (+21, 25)
  type=bmp-file
  source='m:\logos\gustavs.bmp'
  image-size=(75,50)
```

```
print-image (1,1)
  type='auto-detect'
  image-size=(30,15)
  source=$Binary_Variable
```


Note:

For an example of the FOR-PRINTER argument used with DECLARE-IMAGE and PRINT-IMAGE, see the example under “[DECLARE-IMAGE](#)” on page 130.

See Also

- [DECLARE-IMAGE](#)
- “Adding Graphics” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*

PRINT-TABLE

Function

Prints a table.

Syntax

```
PRINT-TABLE [{position}]  
NAME=table_name_var|_ lit|_col  
[CONTINUATION=continuation_var|_ lit|_col]
```

Arguments

NAME

Name of the table created with CREATE-TABLE.

CONTINUATION

Whether the table is a continuation of a previous PRINT-TABLE command. Valid Values are YES and NO. The default is NO.

Description

Use PRINT-TABLE in any section except BEGIN-SETUP, BEGIN-SQL, and BEGIN-DOCUMENT to print a table at the specified location. PRINT-TABLE also performs the functionality of fill.

Example

```
print-table (+1,10)  
  name=$table-name  
  continuation='yes'
```

See Also

[ALTER-TABLE](#), [CREATE-TABLE](#), [DECLARE-TABLE](#), [DUMP-TABLE](#), [FILL-TABLE](#)

PUT

Function

Moves data into an array.

Syntax

```
PUT {src_any_lit|_var|_col}...  
INTO dst_array_name(element) [field[(occurs)] ]...
```

Arguments

src_any_lit|*_var*|*_col*

Source variable or literal to move into the array. Numeric variables, literals, and database columns can be put into number (decimal, float, integer) fields. String variables, literals, and database columns can be put into *char*, *text*, or *date* fields. Date variables can be put into *date*, *char*, or *text* fields.

When a date variable or column is moved into a text or char array field, the date is converted to a string according to the following rules:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by SQR_DB_DATE_FORMAT. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,” on page 251](#).
- For DATE columns, Production Reporting uses the format specified by SQR_DB_DATE_ONLY_FORMAT. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252](#).
- For TIME columns, Production Reporting uses the format specified by SQR_DB_TIME_ONLY_FORMAT. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252](#).

When a string variable, column, or literal is moved to a date array field, the string must be in the format specified by SQR_DB_DATE_FORMAT, one of the database-dependent formats in [Table 61, “Default Formats by Database,” on page 251](#), or the database-independent format 'SYYYYYMMDD [HH24 [MI [SS [NNNNNN]]]]'.

dst_array_name(*element*)

If array fields are listed, data is placed into each field in the sequence it is listed, in the occurrence specified of that field.

If array fields are not listed, data is placed into consecutive fields in the order they were defined in CREATE-ARRAY; data is copied into occurrence zero of each field of the element specified in the array.

field[(*occurs*)]

Array element and field occurrence numbers can be numeric literals (123) or numeric variables (#j).

If no occurrence is specified, occurrence zero is used.

Description

Columns retrieved from the database and Production Reporting variables or literals can be moved into an array. The array must have been created previously using `CREATE-ARRAY`.

Examples

In the following example, the four variables `&name`, `#count`, `$date1`, and `$code` is placed into the first four fields defined in the `names` array. The data is put into the `#j`'th element of the array.

```
put &name #count $date1 $code into names(#j)
```

The following command places `#j2`, `#j3`, and `#j4` into the zero through 2nd occurrences of the `tot` field in the `#j`'th element of the `totals` array.

```
put #j2 #j3 #j4 into totals(#j) tot(0) tot(1) tot(2)
```

The following command copies `#count` into the `#j2`'th occurrence of the `count` field in the `#j`'th element of the `states` array.

```
put #count into states(#j) count(#j2)
```

READ

Function

Reads the next record of a file into the specified variables.

Syntax

```
READ {filenum_lit|_var|_col} INTO {any_var:length_int_lit|_var|_col}...  
[STATUS=status_num_var]
```

Arguments

filenum_lit|*_var*|*_col*

Number assigned in [OPEN](#) to the file to be read.

any_var:length_int_lit|*_var*|*_col*

One or more variables into which data from the record read are to be put. *length_int_lit*|*_var*|*_col* specifies the length of each field of data.

STATUS

Optional variable into which a read status is returned.

Description

Text and binary data is parsed according to the following:

- Text data is any string of characters. The length of the variable name indicates how many characters to place into the variable. After being transferred, trailing blanks in the variable are omitted.
- If the field was written as a date variable, then it may be read into a date variable or text variable. When reading a date into a date variable, it must be in the format specified by `SQR_DB_DATE_FORMAT`, one of the database-dependent formats in [Table 61 on page 251](#) or the database-independent format `'SYYYYMMDD [HH24 [MI [SS [NNNNNN]]']`.
- Binary numbers, may be 1, 2, or 4 bytes in length. They must be read into numeric variables. Note that the bytes making up the binary number must be in the standard sequence expected by your operating system.
- When reading binary data the file must be opened with the `FIXED` or `FIXED-NOLF` qualifier.
- Only the integer portion of the number is represented with binary numbers. To maintain the decimal portion of the number convert the number to a string variable.
- If you use binary numbers, the file is not portable across platforms. This is because different hardware represents binary numbers differently.

The total length indicated for the variables must be less than or equal to the length of the record being read.

If there are no more records to read, the `#end-file` reserved variable is set to 1; otherwise, it is set to 0 (zero). Your program should check this variable after each `READ` command.

If `STATUS` is specified, Production Reporting returns 0 if the read is successful; otherwise, it returns the value of `errno`, which is system-dependent.

Examples

The following example shows several `READ` commands:

```
read 1 into $name:30 $addr:30 $city:20 $state:2 $zip:5
read 3 into $type:2 #amount:2 #rate:1 $code:5 $date:11
read #j into #sequence:2 $name:20 $title:15
```

The following example shows a `READ` command that reads two dates. One is loaded into a date variable; the other is loaded into a string variable, which is then converted to a date using the `strtodate` function.

```
.
.
.
declare-variable
  date $date1 $date2
  text $text
end-declare
.
.
.
read 4 into $date1:18 $text1:18
let $date2 = strtodate($text1, 'SYYYYMMDDHHMISSNNN')
or
let $date2 = strtodate($text1)
```

The following example shows a READ command with an INSERT loop:

```
begin-sql
  begin transaction
end-sql

while 1      ! Infinite loop, exited by BREAK, below.
  read 10 into $company:40 $parent:30 $location:50
  if #end-file
    break      ! End of file reached.
  end-if
  begin-sql
    insert into comps (name, parent, location)
      values ($company, $parent, $location)
  end-sql
  add 1 to #inserts
  if #inserts >= 100
    begin-sql
      end transaction;
      begin transaction
    end-sql
    move 0 to #inserts
  end-if
end-while

begin-sql
  end transaction
end-sql
```

See Also

[OPEN](#), [CLOSE](#), and [WRITE](#) for information on files

ROLLBACK

Function

Causes a database rollback to the last commit.

Syntax

```
ROLLBACK
```

Description

An automatic rollback is performed whenever Production Reporting aborts due to program errors. ROLLBACK is useful in testing or in certain error conditions.

ROLLBACK is an Production Reporting command and should not be used inside an SQL paragraph.

Note:

ROLLBACK can be used with DB2, ODBC, DDO, Teradata, and Oracle. For Sybase, use BEGIN TRANSACTION and ROLLBACK TRANSACTION within SQL paragraphs as in the following example. See the COMMIT command for an example of ROLLBACK.

Examples

```
if #error-status = 1
  rollback
  stop
end-if
```

See Also

[COMMIT](#)

SBTOMBS

Function

Converts a single-byte character into a multi-byte equivalent.

Syntax

```
SBTOMBS {txt_var}
```

Arguments

txt_var

String to convert.

Description

Converts the specified string as follows: Any occurrence of a single-byte character that also has a multi-byte representation (numerals, punctuation, roman characters and katakana) is converted. SBTOMBS also converts a sequence of a kana character followed by certain grammatical marks into a single multi-byte character that combines the two elements.

See Also

The TO_MULTI_BYTE function of [LET](#)

SECURITY

Function

Marks sections of a report for security purposes.

Syntax

```
SECURITY  
[SET=(sid [,sid]...)]  
[APPEND=(sid [,sid]...)]  
[REMOVE=(sid [,sid]...)]  
[MODE=mode]
```

Arguments

SET

List of security IDs for subsequent commands. The previous list of security IDs is replaced by the specified security IDs. This argument is optional and can only be used once.

sid

Any string literal, column, or variable. The value is case sensitive.

APPEND

Appends the specified security IDS to the current list. This argument is optional and can be used multiple times.

REMOVE

Removes the specified security IDS from the current list. This argument is optional and can be used multiple times.

MODE

Turns on (reactivates) or turns off (suspends) the security feature for the current report. This argument is optional and can only be used once.

mode

Any string literal, column, or variable. The value is *not* case sensitive and can be either ON or OFF.

Description

SECURITY can be repeated as many times as desired for the current report. After SECURITY is executed, all subsequent commands for the current report are constrained by the designated Security IDS (SIDs) until the report ends or another SECURITY command executes.

You can use multiple SECURITY commands with the SET, APPEND, and REMOVE options. When a SECURITY command with MODE=ON is processed, the resultant access control list (as built by the previous and current command) is used.

Note:

SECURITY is useful only when used in conjunction with EPM Workspace. The Security IDs refer to EPM Workspace groups. To have the Security ID refer to a specific user, prefix it with U#. For example:

sales, marketing, u#King

refers to the sales group, the marketing group, and the user *King*.

You can use SECURITY wherever you use PRINT.

Examples

```
Begin-Report
  Security Mode='On' Set=('Directors', 'Vice-Presidents')
  .
  . ! Only Directors and VPS can see this
  .
  Security Mode='On' Remove=('Directors')
  .
  . ! Only VPS can see this
  .
  Security Mode='Off'
  .
  . ! Anybody can see this
  .
  Security Mode='On' Append=('Managers')
  .
  . ! Only VPs and Managers can see this
  .
  Security Mode='On' Append=('Engineers')
  .
  . ! Only VPs, Managers, and Engineers can see this
  .
End-report
```

SET-COLOR

Function

Defines default colors.

Syntax

```
SET-COLOR
[PRINT-TEXT-FOREGROUND=({color_name_lit|_var|_col}|{rgb})]
[PRINT-TEXT-BACKGROUND=({color_name_lit|_var|_col}|{rgb})]
[PRINT-PAGE-BACKGROUND=({color_name_lit|_var|_col}|{rgb})]
[LINE-COLOR=({color_name_lit|_var|_col}|{rgb})]
[FILL-COLOR=({color_name_lit|_var|_col}|{rgb})]
```

Arguments

PRINT-TEXT-FOREGROUND

Color in which the text is printed.

PRINT-TEXT-BACKGROUND

Background color behind the text.

PRINT-PAGE-BACKGROUND

Page background color.

LINE-COLOR

Line color used in **DRAW** and **PRINT BOX**. If not specified, the default is **BLACK**.

FILL-COLOR

Fill color used in **DRAW** (**TYPE=BOX**) and **PRINT BOX**. If not specified, the default is **NONE**.

{color_name_lit|_var|_col}

A *color_name* is composed of the alphanumeric characters (A-Z, 0-9), the underscore (`_`) character, and the dash (`-`) character. It must start with an alpha (A-Z) character. It is case insensitive. The name 'none' is reserved and cannot be assigned a value. A name in the format (RGBredgreenblue) cannot be assigned a value. The name 'default' is reserved and may be assigned a value. 'Default' is used during execution when a referenced color is not defined in the runtime environment.

{rgb}

red_lit_var|_col, green_lit|_var|_col, blue_lit|_var|_col where each component is a value in the range of 000 to 255. In the **BEGIN-SETUP** section, only literal values are allowed.

The default colors implicitly installed with Production Reporting include:

black = (0,0,0)

white=(255,255,255)

gray=(128,128,128)

silver=(192,192,192)

red=(255,0,0)

green=(0,255,0)

blue=(0,0,255)

yellow=(255,255,0)

purple=(128,0,128)

olive=(128,128,0)

navy=(0,0,128)

aqua=(0,255,255)

lime=(0,128,0)

maroon=(128,0,0)

teal=(0,128,128)

fuchsia=(255,0,255)

Description

SET-COLOR is allowed wherever PRINT or DRAW is allowed. It is used to set certain attributes of PRINT and DRAW. If the specified color name is not defined, then the setting for the color name 'default' is used. Use the color name 'none' to turn off color for the specified attribute.

Examples

```
begin-setup
  declare-color-map
    light_blue = (193, 222, 229)
  end-declare
end-setup

begin-program
  alter-color-map name = 'light_blue' value = (193, 233, 230)
  print 'Yellow Submarine' ()
    foreground = ('yellow')
    background = ('light_blue')
  get-color print-text-foreground = ($print-foreground)
  set-color print-text-foreground = ('purple')
  print 'Barney' (+1,1)
  set-color print-text-foreground = ($print-foreground)
end-program

begin-program
  get-color line-color=($line-color)
  set-color line-color=('purple')
  draw (5,5) type='horz-line' width=10
  set-color line-color=($line-color)
end-program

begin-program
  get-color fill-color=($fill-color)
  set-color fill-color=('light grey')
  draw (5,5) type='box' width=10 height=10
  set-color fill-color=($fill-color)
end-program
```

See Also

[DECLARE-COLOR-MAP](#), [ALTER-COLOR-MAP](#), and [GET-COLOR](#)

SET-DELAY-PRINT

Function

Sets the values of a DELAY variable.

Syntax

```
SET-DELAY-PRINT delay_var WITH {src_lit|_var|_col}
```

Arguments

delay_var

Affected delay variable.

{src_lit|_var|_col}

Source variable.

Description

Replaces each reference of *delay_var* with the specified value. The data is formatted according to the `PRINT` command parameters.

Examples

```
print $Last_User (1,10) Delay
.
.
.
set-delay-print $Last_User with &Username
```

See Also

The `DELAY` parameter under [PRINT](#)

SHOW

Function

Displays one or more variables or literals on the screen. Cursor control is supported for ANSI terminals.

Syntax

```
SHOW[cursor_position]
[CLEAR-SCREEN|CS|CLEAR-LINE|CL] [any_lit|_var|_col]
[EDIT edit_mask|NUMBER|MONEY|DATE] [BOLD] [BLINK]
[UNDERLINE] [REVERSE] [NORMAL] [BEEP] [NOLINE] ...
```

Arguments

cursor_position

Position to begin the display.

`CLEAR-SCREEN` or `CS`

Clears the screen and sets the cursor position to (1,1).

`CLEAR-LINE` or `CL`

Clears a line from the current cursor position to the end of the line.

any_lit|_var|_col

Information to display.

EDIT

Variables under an edit mask. If the mask contains spaces, enclose it in single quotes. For additional information regarding edit masks, see `PRINT`.

NUMBER

Formats *any_lit|_var|_col* with the `NUMBER-EDIT-MASK` from the current locale. (See [ALTER-LOCALE](#).) Not legal for date variables.

MONEY

Formats *any_lit|_var|_col* with the `MONEY-EDIT-MASK` from the current locale. (See [ALTER-LOCALE](#).) Not legal for date variables.

DATE

Formats *any_lit|_var|_col* with the `DATE-EDIT-MASK` from the current locale. (See [ALTER-LOCALE](#).) Not legal for numeric variables. If `DATE-EDIT-MASK` is not specified, the date is displayed using the default format for that database (see [Table 61 on page 251](#)).

BOLD, **BLINK**, **UNDERLINE**, and **REVERSE**

Changes the display of characters. Some terminals support two or more characteristics at the same time for the same text. To turn all special display characteristics off, use `NORMAL`.

NORMAL

Turns off all special display characteristics set with `BOLD`, `BLINK`, `UNDERLINE`, and `REVERSE`.

BEEP

Causes the terminal to beep.

NOLINE

Inhibits a line advance.

Description

Any number of variables and screen positions can be used in a single command. Each one is processed in sequence.

Screen locations can be indicated by either fixed or relative positions in the format (A,B), where A is the line and B is the column on the screen. A and B can also be numeric variables. Relative positions depend on where the previous `SHOW` command ended. If the line was advanced, the screen cursor is usually immediately to the right of the previously displayed value and one line down.

Fixed or relative cursor positioning can be used only within the boundaries of the terminal screen. Scrolling off the screen using relative positioning, for example (+1,1), is not supported.

Instead, use `SHOW` without any cursor position when you want to scroll. You cannot mix `SHOW` and `DISPLAY` while referencing relative cursor positions.

`SHOW` does not advance to the next line if a cursor location (...), `CLEAR-SCREEN`, `CLEAR-LINE`, or `BEEP` is used. (`SHOW` without any of these arguments automatically advances the line.) To add a line advance, add `(+1,1)` to the end of the line or use an extra empty `SHOW` command.

Only ANSI terminals are supported for cursor control, screen blanking, line blanking, and display characteristics.

Dates can be contained in a date variable or column, or a string literal, column, or variable. When displaying a date variable or column, without an edit mask, the date is displayed according to the following rules:

- For `DATETIME` columns and Production Reporting `DATE` variables, Production Reporting uses the format specified by `SQR_DB_DATE_FORMAT`. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,” on page 251](#).
- For `DATE` columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,” on page 252](#).
- For `TIME` columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,” on page 252](#).

When displaying a date in a string literal, column, or variable using `EDIT` or `DATE`, the string must be in the format specified by `SQR_DB_DATE_FORMAT`, one of the database-dependent formats in [Table 61 on page 251](#), or the database-independent format `'SYYYYMMDD [HH24 [MI [SS [NNNNNN]]]]'`.

Examples

The following code:

```
!  
! Show a string using an edit mask  
!  
let $ssn = '123456789'  
show $ssn edit xxx-xx-xxxx
```

Produces the following output:

```
123-45-6789
```

The following code:

```
!  
! Show a number using an edit mask  
!  
show 1234567.89 edit 999,999,999.99
```

Produces the following output:

```
1,234,567.89
```

The following code:

```
!  
! Show a number using the default edit mask  
!  
show 123.78
```

Produces the following output:

```
123.780000
```

The following code:

```
!  
! Show a number using the locale default numeric edit mask  
!  
alter-locale number-edit-mask = '99,999,999.99'  
show 123456.78 number
```

Produces the following output:

```
123,456.78
```

The following code:

```
!  
! Show a number using the locale default money edit mask  
!  
alter-locale money-edit-mask = '$$,$$$,$$8.99'  
show 123456.78 money
```

Produces the following output:

```
$123,456.78
```

The following code:

```
!  
! Show a date column using the locale default date edit mask  
!  
begin-select  
dcol  
  from tables  
end-select  
alter-locale date-edit-mask = 'DD-Mon-YYYY'  
show &dcol date
```

Produces the following output:

```
01-Jan-1999
```

The following code:

```
!  
! Show two values on the same line  
!  
show 'Hello' ' World'
```

Produces the following output:

Hello World

The following code:

```
!  
! Show two values on the same line with editing of the values  
!  
let #taxes = 123456.78  
show 'You owe ' #taxes money ' in back taxes.'
```

Produces the following output:

```
You owe $123,456.78 in back taxes.
```

The following program illustrates the usage of additional options of `SHOW`. Only terminals that support the ANSI escape characters can use the cursor control, screen blanking, line blanking and display attributes.

```
begin-program  
!  
! Produces a menu for the user to select from  
!  
show clear-screen  
  (3,30) bold 'Accounting Reports for XYZ Company' normal  
  (+2,10) '1. Monthly Details of Accounts'  
  (+1,10) '2. Monthly Summary'  
  (+1,10) '3. Quarterly Details of Accounts'  
  (+1,10) '4. Quarterly Summary'  
!  
! Show a line of text and numerics combined  
!  
show (+2,1)  
  'The price is ' #price edit 999.99  
  'Total = ' #total edit 99999.99  
!  
! Put an error message on a particular line  
!  
show (24,1) clear-line 'Error in SQL. Please try again.' beep  
end-program
```

See Also

- [LET](#) for information on copying, editing, or converting fields
- The `EDIT` parameter of [PRINT](#) for edit mask descriptions
- [ALTER-LOCALE](#) for a description of `NUMBER-EDIT-MASK`, `MONEY-EDIT-MASK`, and `DATE-EDIT-MASK`
- [DISPLAY](#)

STOP

Function

Halts Production Reporting.

Syntax

```
STOP [QUIET]
```

Arguments

QUIET

Completes the report with the “Production Reporting: End Of Run” message, instead of aborting with an error message.

Description

STOP halts Production Reporting and executes a ROLLBACK command (not in Sybase, ODBC, or Informix). All report page buffers are flushed if they contain data; however, no headers or footers are printed and the AFTER-PAGE and AFTER-REPORT procedures are not executed.

STOP is useful in testing.

Examples

```
if #error-status = 1
  rollback
  stop
else
  commit
  stop quiet
end-if
```

STRING

Function

Concatenates a list of variables, columns, or literals into a single text variable. Each member of the list is separated by the specified delimiter string.

Syntax

```
STRING {src_any_lit|_var|_col}...BY {delim_txt_lit|_var|_col}
INTO dst_txt_var
```

Arguments

src_any_lit|_var|_col

One or more fields to concatenate, separated by the *delim_txt_lit|_var|_col* character or characters, and placed into the *dst_txt_var* variable.

If the source is a date variable or column, it is converted to a string according to the following rules:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by SQR_DB_DATE_FORMAT. If not set, Production Reporting uses

the first database-dependent format in [Table 61, “Default Formats by Database,”](#) on page 251.

- For DATE columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,”](#) on page 252.
- For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,”](#) on page 252.

delim_txt_lit|_var|_col

Characters used as separators between source fields.

dst_txt_var

The destination field for the concatenated result.

Description

Do not include the destination string in the list of source strings.

Examples

```
string &name &city &state &zip by ' - ' into $show-info
! Result: Sam Mann - New York - NY - 11287
```

```
string &cust_num &entry-date &total by ',' into $cust-data
! Result: 100014,12-MAR-89,127
! Use null delimiter.
```

```
string &code1 &code2 &code3 by '' into $codes123
! Result: AGL
```

See Also

- [UNSTRING](#)
- The “||” concatenation operator in [Table 45 on page 194](#) under `LET`

SUBTRACT

Function

Subtracts one value from another.

Syntax

```
SUBTRACT {src_num_lit|_var|_col} FROM dst_num_var[ROUND=nn]
```

Arguments

src_num_lit|_var|_col

Subtracted from the contents of *dst_num_var*.

dst_num_var

The result after execution.

ROUND

Rounds the result to the specified number of digits to the right of the decimal point. For float variables this value can be from 0 to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate.

Description

Subtracts the first value from the second and moves the result into the second field.

When dealing with money-related values (dollars and cents), use decimal variables rather than float variables. Float variables are stored as double precision floating point numbers, and small inaccuracies can appear when subtracting many numbers in succession. These inaccuracies can appear due to the way floating point numbers are represented by different hardware and software implementations.

Examples

```
subtract 1 from #total      ! #total - 1
subtract &discount from #price ! #price - &discount
```

See Also

- [ADD](#)
- [LET](#) for information on complex arithmetic expressions

TOC-ENTRY

Function

Places an entry into the Table of Contents.

Syntax

```
TOC-ENTRY
TEXT={src_txt_lit|_var|_col}
[LEVEL={level_num_lit|_var|_col}]
```

Arguments

TEXT

Text to place in the Table of Contents.

LEVEL

Level at which to place the text. If this argument is not specified, the value of the previous level is used.

Description

Enter the text in the Table of Contents at the desired level.

Examples

```
toc-entry text = &heading  
toc-entry text = &caption level=2
```

See Also

[DECLARE-TOC](#)

UNSTRING

Function

Copies portions of a string into one or more text variables.

Syntax

```
UNSTRING {{src_txt_lit|_var|_col} | {src_date_var|_col}}  
BY {delim_txt_lit|_var|_col}  
INTO dst_txt_var...
```

Arguments

{*src_txt_lit*|_var|_col} | {*src_date_var*|_col}

Source field to parse.

delim_txt_lit|_var|_col

Characters used to delimit the fields in {*src_txt_lit*|_var|_col} | {*src_date_var*|_col}

dst_txt_var

Destination fields to receive the results.

Description

Each substring is located using the specified delimiter. The source string must not be included in the list of destination strings.

If more destination strings than substrings are found in the source strings, the extra destination strings are each set to an empty string.

If more substrings are found in the source string than in the destination strings, the extra substrings are not processed. It is up to the programmer to ensure that enough destination strings are specified.

If the source is a date variable or column, it is converted to a string according to the following rules:

- For DATETIME columns and Production Reporting DATE variables, Production Reporting uses the format specified by `SQR_DB_DATE_FORMAT`. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,”](#) on page 251.
- For DATE columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,”](#) on page 252.
- For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,”](#) on page 252.

Examples

```
unstring $show-info by ' - ' into $name $city $state $zip
unstring $cust-data by ', ' into $cust_num $entry-date $total
```

See Also

- [STRING](#) and [EXTRACT](#)
- The `substr` and `instr` functions in [Table 52, “Miscellaneous Functions,”](#) on page 212 under [LET](#)

UPPERCASE

Function

Converts a string variable to uppercase.

Syntax

```
UPPERCASE txt_var
```

Arguments

txt_var

The field to convert.

Examples

```
input $state 'Enter state abbreviation'
uppercase $state ! Force uppercase.
```

See Also

The `upper` function in [Table 52, “Miscellaneous Functions,”](#) on page 212 under [LET](#)

USE

Function

Uses the named database, rather than the default database associated with your user name. (Sybase and ODBC only)

Syntax

```
USE database
```

Arguments

database

The name of the database to use.

Description

Use `USE` in the `SETUP` section only. When used, it must appear at the top of your report, before any queries are defined.

To reference more than one database in a program, specify secondary databases explicitly. For example:

```
from sqdb.sqr.customers
```

You cannot issue the Sybase or ODBC `USE` command from within an SQL paragraph.

Examples

```
begin-setup  
  use pubs  
end-setup
```

See Also

The `-DB` command-line flag described in [“Production Reporting Command-line Flags”](#) on page 21.

USE-COLUMN

Function

Sets the current column.

Syntax

```
USE-COLUMN {column_number_int_lit|_var|_col}
```

Arguments

column_number_int_lit|*_var*|*_col*

Number of the defined column (not the location on the page). For example, if five columns are defined, then the `column_number_int_lit|_var|_col` can be 1 to 5.

Description

The column must be previously defined with the `COLUMNS`.

To stop printing within columns, use a column number of 0 (zero). Printing returns to normal; however, the columns remain defined for subsequent `NEXT-COLUMN` or `USE-COLUMN` commands.

Examples

```
use-column 3      ! Print total in 3rd column.
print #total  ()  999,999
use-column 0      ! End of column printing.
```

USE-PRINTER-TYPE

Function

Sets the printer type to use for the current report.

Syntax

```
USE-PRINTER-TYPE printer-type
```

Arguments

printer-type

Printer type to use for the current report. See `DECLARE-PRINTER` for valid types.

Description

Sets or alters the printer type used for the current report. `USE-PRINTER-TYPE` must appear before the first output is written to that report. If output has already been written to the report file, `USE-PRINTER-TYPE` is ignored.

Examples

```
use-report customer_orders
use-printer-type PostScript
print (1, 1) 'Customer Name: '
print () $customer_name
```

See Also

[DECLARE-PRINTER](#), [DECLARE-REPORT](#), and [USE-REPORT](#)

USE-PROCEDURE

Function

Changes the procedure usage.

Syntax

```
USE-PROCEDURE  
[FOR-REPORTS=(report_name1[,report_namei]...)]  
[BEFORE-REPORT=procedure_name[(arg1[,argi]...)]]  
[AFTER-REPORT=procedure_name[(arg1[,argi]...)]]  
[BEFORE-PAGE=procedure_name[(arg1[,argi]...)]]  
[AFTER-PAGE=procedure_name[(arg1[,argi]...)]]
```

Arguments

FOR-REPORTS

Reports that use the procedures. This argument is required only for a program with multiple reports. If you are writing a program that produces a single report, you can ignore this argument.

BEFORE-REPORT

Procedure to execute when the first command execute, which causes output to be generated. For example, you can use the command to create a report heading.

AFTER-REPORT

Procedure to execute just before the report file is closed at the end of the report. This argument can be used to print totals or other closing summary information. If no report was generated, the procedure does not execute.

BEFORE-PAGE

Procedure to execute at the beginning of every page, just before the first output command for the page. It can be used, for example, to set up page totals.

AFTER-PAGE

Procedure to execute just before each page is written to the file. This argument can be used, for example, to display page totals.

You can also specify arguments to pass to the procedure. Arguments can be any variable, column, or literal.

Description

USE-PROCEDURE must be issued in the PROGRAM or PROCEDURE sections of an Production Reporting program. USE-PROCEDURE is a run-time command; its compile-time equivalent is DECLARE-PROCEDURE. You can use the command as often as required to change to the necessary procedures required by the reports. If you issue multiple USE-PROCEDURE commands, each remains in effect for that report until altered by another USE-PROCEDURE command for that report. In this way, you can use one to change common procedures for ALL reports and others

to change unique procedures for individual reports. The referenced procedures can accept arguments.

If no `FOR-REPORTS` is specified, `ALL` is assumed. Initially, the default for each of the four procedure types is `NONE`. If a procedure is defined in one `DECLARE-PROCEDURE` for a report, that procedure is used unless `NONE` is specified.

You can change `BEFORE-REPORT` only before the first output is written to that report, since that causes the `BEFORE-REPORT` procedure to execute.

Examples

```
use-procedure           ! These procedures will
for-reports=(all)       ! be used by all reports.
before-report=report_heading
after-report=report_footing

use_procedure           ! These procedures will

for-reports=(customer) ! be used by the customer report.

before-page=page_setup

after-page=page_total

use-procedure           ! The after-report procedure will be
for-reports=(summary)  ! disabled for the summary report.
after-report=none
```

See Also

[DECLARE-PROCEDURE](#)

USE-REPORT

Function

For programs with multiple reports, allows the user to switch between reports.

Syntax

```
USE-REPORT {report_name_lit|_var|_col}
```

Arguments

```
report_name_lit|_var|_col
```

The report to become the “*current*” report. All subsequent `PRINT` and `PRINT-DIRECT` statements are written to this report until the next `USE-REPORT` is encountered.

Description

Defines to which report file(s) the subsequent report output is to be written. An application can contain several `USE-REPORT` statements to control several reports.

You must specify the report name and report characteristics in a `DECLARE-REPORT` paragraph and in the associated `DECLARE-LAYOUT` and `DECLARE-PRINTER` paragraphs.

Examples

```
use-report customer_orders
use-printer-type PostScript
print (1, 1) 'Customer Name: '
print () $customer_name
```

See Also

[DECLARE-REPORT](#), [DECLARE-LAYOUT](#), [DECLARE-PRINTER](#), and [USE-PRINTER-TYPE](#)

WHILE

Function

Begins a `WHILE ... END-WHILE` loop.

Syntax

```
WHILE logical_expression
```

The general format of `WHILE` is as follows:

```
WHILE logical_expression
SQR_commands...
[BREAK]
[CONTINUE]
SQR_commands...
END-WHILE
```

Arguments

```
logical_expression
```

A valid logical expression. See [LET](#) for a description of logical expressions.

Operators

See “[Bit-Wise Operators](#)” on page 195 for information on the bit-wise operators supported by `WHILE`.

Description

The `WHILE` loop continues until the condition being tested is `FALSE`.

An expression returning 0 (zero) is considered `FALSE`; an expression returning nonzero is `TRUE`.

`BREAK` causes an immediate exit of the `WHILE` loop; Production Reporting continues with the command immediately following `END-WHILE`.

`CONTINUE` ends the current iteration of a loop. Program control passes from the `CONTINUE` parameter to the end of the `WHILE` loop body.

WHILE commands can be nested to any level and can include or be included within IF and EVALUATE commands.

Examples

The following example shows an IF nested within a WHILE:

```
while #count < 50
  do get_statistics
  if #stat_count = 100
    break      ! Exit WHILE loop.
  end-if
  add 1 to #count
end-while
```

You can use single numeric variables in your expression to make your program more readable, for example when using flags.

```
move 1 to #have_data
...
while #have_data
  ...processing...
end-while
```

The following example sets up an infinite loop:

```
while 1
  ...processing...
  if ...
    break      ! Exit loop
  end-if
end-while
```

You can use any complex expression in WHILE as shown in the following example:

```
while #count < 100 and (not #end-file or isnnull(&state))
  ...
end-while
```

The following example shows the use of CONTINUE in a WHILE loop:

```
while #count < 50
  if #count = 10
    continue
  end-if
  do get-statistics(#count)
  add 1 to #count
end-while
```

See Also

[LET](#) for a description of expressions

WRITE

Function

Writes a record to a file from data stored in variables, columns, or literals.

Syntax

```
WRITE {filenum_lit|_var|_col} FROM  
{{txt_lit|_var|_col}|{date_var|_col}|num_col}  
[:len_int_lit|_var|_col}]|{num_lit|_var:len_int_lit|_var|_col}}...  
[STATUS=status_num_var]
```

Arguments

filenum_lit|*_var*|*_col*

Number assigned in [OPEN](#) to the file to write.

{txt_lit|*_var*|*_col*}|*{date_var*|*_col*}|*num_col*} [:*len_int_lit*|*_var*|*_col*}]|*{num_lit*|*_var*:*len_int_lit*|*_var*|*_col*}

Specifies one or more variables to write. *len_int_lit*|*_var*|*_col* specifies the length of each field of data.

STATUS

An optional variable into which a write status is returned.

Description

The file must already be opened for writing.

If length is specified, the variable is either truncated at that length or padded with spaces to that length. If length is not specified (for string variables or database columns), the current length of the variable is used.

When writing numeric variables, the length argument is required. Only 1, 2, or 4 byte binary integers are written. Floating point values are not supported directly in `WRITE`. However, you can first convert floating point numbers to strings and then write the string.

When writing binary data the file must be open using the `FIXED` or `FIXED-NOLF` qualifiers. The file is not portable across platforms since binary numbers are represented differently.

When writing a date variable or column, the date is converted to a string according to the following rules:

- For `DATETIME` columns and Production Reporting `DATE` variables, Production Reporting uses the format specified by `SQR_DB_DATE_FORMAT`. If not set, Production Reporting uses the first database-dependent format in [Table 61, “Default Formats by Database,”](#) on page 251.
- For `DATE` columns, Production Reporting uses the format specified by `SQR_DB_DATE_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 62, “DATE Column Formats,”](#) on page 252.

- For TIME columns, Production Reporting uses the format specified by `SQR_DB_TIME_ONLY_FORMAT`. If not set, Production Reporting uses the format in [Table 63, “TIME Column Formats,”](#) on page 252.

Text literals take the length of the literal.

Files opened for writing are treated as having variable-length records. If you need a fixed-length record, specify a length for each variable written to the file.

The total length of the variables and literals being written must not be greater (but can be less) than the record length specified when the file was opened. Records are not padded, but are written with the total length of all variables in `WRITE`.

If `STATUS` is specified, Production Reporting returns 0 if the write is successful; otherwise, it returns the value of `errno`, which is system-dependent.

Examples

```
write 5 from $name:20 $city:15 $state:2
write 17 from $company ' - ' $city ' - ' $state ' ' $zip
write #j2 from #rate:2 #amount:4 #quantity:1
move #total to $tot 99999.99      ! Convert floating point to
                                ! string.

write 1 from $tot
let $date1 = datenow()           ! Put the current date and time
                                ! into DATE variable

write 3 from $date1:20
```

See Also

[OPEN](#), [CLOSE](#), and [READ](#)

WRITE-RS

Function

Writes values to the specified row set.

Syntax

```
WRITE-RS
NAME=row_set_name_var|_lit|_col
VALUE=({name_var|_lit|_col},{data_var|_lit|_col})
```

Arguments

NAME

Name of the row set.

VALUE

Column name and value. Can be repeated as many times as needed to satisfy the row set definition.

Description

WRITE-RS can reside in any section except [BEGIN-SETUP](#), [BEGIN-SQL](#), and [BEGIN-DOCUMENT](#). Validation rules include:

- The row set specified by *row_set_name* must be active, or an exception is thrown.
- If NAME is an empty string, then VALUE is ignored.
- If NAME is not an empty string, then it must be defined in OPEN-RS.
- The data type should match the type specified for the column. If needed, implicit conversions are performed according to SQR rules. If a required conversion cannot be done, an exception is thrown (for example, a numeric value specified for a DATE column).
- Based on the row set definition in OPEN-RS, if a column is not specified with a VALUE entry, it is assumed to be a NULL value.

The row set file is an XML file. You can define whether to create the XML file in a BI Publisher (BIP) format or an SQR format in the `FormatForRowsetXML` entry in the [Default-Settings] section of SQR.INI.

Example

```
Begin-Report
  Open-RS Name='customer' FileName='customer.xml'
  Column = ('cust_num', 'integer')
  Column = ('name', 'string')
  Column = ('addr1', 'string')
  Column = ('addr2', 'string')
  Column = ('city', 'string')
  Column = ('state', 'string')
  Column = ('zip', 'string')
  Column = ('phone', 'string')
  Column = ('tot', 'integer')
  Begin-Select
  cust_num
  name
  addr1
  addr2
  city
  state
  zip
  phone
  tot
  Write-RS Name='customer'
  Value = ('cust_num', $cust_num)
  Value = ('name', &name)
  Value = ('addr1', &addr1)
  Value = ('addr2', &addr2)
  Value = ('city', &addr3)
  Value = ('state', &addr4)
  Value = ('zip', &zip)
  Value = ('phone', &phone)
  Value = ('tot', $&tot)
  from customers
  order by cust_num
  End-Select
```

```
Close-RS Name='customer'  
End-Report
```

See Also

[OPEN-RS, CLOSE-RS](#)

4

HTML Procedures

In This Chapter

About HTML Procedures	303
HTML General Purpose Procedures.....	303
HTML Heading Procedures	305
HTML Highlighting Procedures	307
HTML Hypertext Link Procedures.....	309
HTML List Procedures.....	309
HTML Table Procedures.....	312

About HTML Procedures

HTML procedures enable Production Reporting to generate HTML output. For information on using HTML procedures, see “Working with HTML” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*.

HTML General Purpose Procedures

Table 67 HTML General Purpose Procedures

Procedure	Description
html_br	<p>Line break in a paragraph.</p> <p>Syntax: <code>html_br(number count, string attributes)</code></p> <ul style="list-style-type: none">● <code>count</code> = Number of
 tags● <code>attributes</code> = HTML attributes inside
 <p>Example:</p> <pre>print 'Here is some text' () do html_br(3, '') print 'Here is some three lines down' ()</pre>
html_center	<p>Start of centered text. (You can also use <code>PRINT</code> and specify <code>CENTER</code> in the code.)</p> <p>Syntax: <code>html_center(string attributes)</code></p> <ul style="list-style-type: none">● <code>attributes</code> = HTML attributes inside <CENTER> <p>Example:</p>

Procedure	Description
	<pre>do html_center('') print 'Here is some text' () do html_center_end</pre>
html_center_end	<p>End of centered text.</p> <p>Syntax: <code>html_center_end</code></p>
html_hr	<p>Horizontal divider between sections of text.</p> <p>Syntax: <code>html_hr(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <HR> <p>Example:</p> <pre>print 'Here is some text' () do html_hr('') print 'And some more text' ()</pre>
html_img	<p>Image. (You can also use <code>PRINT-IMAGE</code>; however, <code>html_img</code> allows you to define the full set of available HTML attributes.)</p> <p>Syntax: <code>html_img(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <p>Common attributes: src-URL of image (Example: <code>src=/images/abc.gif</code>) height-Image height in pixels (Example: <code>height=200</code>) width-Image width in pixels (Example: <code>width=400</code>)</p> <p>Example:</p> <pre>do html_img('src="/images/stop.gif"')</pre>
html_nobr	<p>Start of text that cannot be wrapped.</p> <p>Syntax: <code>html_nobr</code></p> <p>Example:</p> <pre>do html_nobr('') print 'Here's long text that should not wrap' () do html_nobr_end</pre>
html_nobr_end	<p>End of text that cannot be wrapped.</p> <p>Syntax: <code>html_nobr_end</code></p>
html_on	<p>Turns on HTML procedures. Called at the start of Production Reporting programs.</p> <p>Syntax: <code>html_on</code></p> <p>Example</p> <pre>do html_on</pre>
html_p	<p>Start of a paragraph.</p> <p>Syntax: <code>html_p(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <P> <p>Some common attributes: align = left right center-Alignment of the paragraph.</p>

Procedure	Description
	<p>Example:</p> <pre>do html_p('ALIGN=RIGHT') print 'Right aligned text' (1,1) do html_p_end print 'Normal text' (+1,1)</pre>
html_p_end	<p>End of a paragraph. Typically implied; however, can be defined for completeness.</p> <p>Syntax: html_p_end</p>
html_set_body_attributes	<p>Attributes inside <BODY>. Called at the start of Production Reporting programs.</p> <p>Syntax: html_set_body_attributes(<i>string attributes</i>)</p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <BODY> <p>Common attributes:</p> <p>background-Image displayed on the background of the Web page. (Example: background=/images/logo.gif)</p> <p>bgcolor=#rrggbb-Background color of the Web page. (Example: bgcolor=#80FFF)</p> <p>Example:</p> <pre>do html_set_body_attributes('BACKGROUND="/images/x.gif"')</pre>
html_set_head_tags	<p>Tags between <HEAD> and </HEAD>. (Empty by default.) Called at the start of Production Reporting programs.</p> <p>Syntax: html_set_head_tags(<i>string attributes</i>)</p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes between <HEAD> and </HEAD> <p>Example:</p> <pre>do html_set_head_tags('<TITLE>My Report</TITLE>')</pre>

HTML Heading Procedures

Table 68 HTML Heading Procedures

Procedure	Description
html_h1	<p>Start of heading level one text.</p> <p>Syntax: html_h1(<i>string attributes</i>)</p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <H1> <p>Example::</p> <pre>do html_h1('') print 'This is a heading' () do html_h1_end</pre>
html_h1_end	<p>End of heading level one text.</p> <p>Syntax: html_h1_end</p>

Procedure	Description
html_h2	<p>Start of heading level two text.</p> <p>Syntax: <code>html_h2(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <H2> <p>Example:</p> <pre>do html_h2('') print 'This is a heading' () do html_h2_end</pre>
html_h2_end	<p>End of heading level two text.</p> <p>Syntax: <code>html_h2_end</code></p>
html_h3	<p>Start of heading level three text. (This heading is the default.)</p> <p>Syntax: <code>html_h3(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <H3>
html_h3_end	<p>End of heading level three text.</p> <p>Syntax: <code>html_h3_end</code></p>
html_h4	<p>Start of heading level four text.</p> <p>Syntax: <code>html_h4(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <H4>
html_h4_end	<p>End of heading level four text.</p> <p>Syntax: <code>html_h4_end</code></p>
html_h5	<p>Start of heading level five text.</p> <p>Syntax: <code>html_h5(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <H5>
html_h5_end	<p>End of heading level five text.</p> <p>Syntax: <code>html_h5_end</code></p>
html_h6	<p>Start of heading level six text.</p> <p>Syntax: <code>html_h6(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <H6>
html_h6_end	<p>End of heading level six text.</p> <p>Syntax: <code>html_h6_end</code></p>

HTML Highlighting Procedures

Table 69 HTML Highlighting Procedures

Procedure	Description
html_blink	Start of blinking text. Syntax: <code>html_blink(string attributes)</code> <ul style="list-style-type: none">● <i>attributes</i> = HTML attributes inside <BLINK> Example: <pre>do html_blink('') print 'This is blinking' () do html_blink_end</pre>
html_blink_end	End of blinking text. Syntax: <code>html_blink_end</code>
html_cite	Start of citation text. Syntax: <code>html_cite(string attributes)</code> <ul style="list-style-type: none">● <i>attributes</i> = HTML attributes inside <CITE> Example: <pre>do html_cite('') print 'This is a citation' () do html_cite_end</pre>
html_cite_end	End of citation text. Syntax: <code>html_cite_end</code>
html_code	Start of code text. Syntax: <code>html_code(string attributes)</code> <ul style="list-style-type: none">● <i>attributes</i> = HTML attributes inside <CODE> Example: <pre>do html_code('') print 'Here is the code' () do html_code_end</pre>
html_code_end	End of code text. Syntax: <code>html_code_end</code>
html_kbd	Start of keyboard input text. Syntax: <code>html_kbd(string attributes)</code> <ul style="list-style-type: none">● <i>attributes</i> = HTML attributes inside <KBD> Example: <pre>do html_kbd('') print 'Here is keyboard' () do html_kbd_end</pre>

Procedure	Description
html_kbd_end	End of keyboard input text. Syntax: html_kbd_end
html_samp	Start of sample text. Syntax: html_samp(<i>string attributes</i>) <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <SAMP> Example: <pre>do html_samp('') print 'Here is sample' () do html_samp_end</pre>
html_samp_end	End of sample text. Syntax: html_samp_end
html_strike	Start of strike-through text. Syntax: html_strike(<i>string attributes</i>) <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <STRIKE> Example: <pre>do html_strike('') print 'Here is strike-through' () do html_strike_end</pre>
html_strike_end	End of strike-through text. Syntax: html_strike_end
html_sub	Start of subscript text. Syntax: html_sub(<i>string attributes</i>) <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <SUB> Example: <pre>print 'Here is' () do html_sub('') print 'subscript text' () do html_sub_end</pre>
html_sub_end	End of subscript text. Syntax: html_sub_end
html_sup	Start of superscript text. Syntax: html_sup(<i>string attributes</i>) <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <SUP> Example: <pre>print 'Here is' () do html_sup('') print 'superscript text' () do html_sup_end</pre>

Procedure	Description
<code>html_sup_end</code>	End of superscript text. Syntax: <code>html_sup_end</code>

HTML Hypertext Link Procedures

Table 70 HTML Hypertext Link Procedures

Procedure	Description
<code>html_a</code>	<p>Start of a hypertext link.</p> <p>Syntax: <code>html_a(string attributes)</code></p> <ul style="list-style-type: none"> <i>attributes</i> = HTML attributes inside <A>. At a minimum, define HREF, which specifies the URL of HTML documents. <p>Common attributes:</p> <p><code>href</code>—Where the hypertext link points. (Example: <code>href=home.html</code>)</p> <p><code>name</code>—Anchor to which a hypertext link can point. (Example: <code>name=marker1</code>)</p> <p>Example: Create an anchor with two hypertext links. Position the anchor at the top of the document. Point the first hypertext link to <i>otherdoc.html</i>. Point the second hypertext link to the anchor named <i>TOP</i>.</p> <pre>do html_a('NAME=TOP') do html_a_end print 'At the top of document' () do html_br(20, '') do html_a('HREF=otherdoc.html') print 'Goto other document' () do html_a_end do html_p('') do html_a('HREF=#TOP') print 'Goto top of document' () do html_a_end</pre>
<code>html_a_end</code>	<p>End of a hypertext link.</p> <p>Syntax: <code>html_a_end</code></p>

HTML List Procedures

Table 71 HTML List Procedures

Procedure	Description
<code>html_dd</code>	<p>Start of a definition in a definition list.</p> <p>Syntax: <code>html_dd(string attributes)</code></p> <ul style="list-style-type: none"> <i>attributes</i> = HTML attributes inside <DD>

Procedure	Description
html_dd_end	End of a definition in a definition list. Typically implied; however, can be defined for completeness. Syntax: <code>html_dd_end</code>
html_dir	Start of a directory list. Syntax: <code>html_dir(string attributes)</code> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <DIR> Example: <pre>do html_dir('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_dir_end</pre>
html_dir_end	End of a directory list. Syntax: <code>html_dir_end</code>
html_dl	Start of a definition list. Terms display above and to the left of definitions. <code>html_dt</code> displays terms. <code>html_dd</code> displays definitions. Syntax: <code>html_dl(string attributes)</code> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <DL> Example: Definition list with two terms and definitions: <pre>do html_dl('') do html_dt('') print 'A Daisy' () do html_dd('') print 'A sweet and innocent flower.' () do html_dt('') print 'A Rose' () do html_dd('') print 'A very passionate flower.' () do html_dl_end</pre>
html_dl_end	End of a definition list. Syntax: <code>html_dl_end</code>
html_dt	Start of a term in a definition list. Syntax: <code>html_dt(string attributes)</code> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <DT>
html_dt_end	End of a term in a definition list. Syntax: <code>html_dt_end</code>
html_li	Start of a list item. Syntax: <code>html_li(string attributes)</code>

Procedure	Description
	<ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside
html_li_end	<p>End of a list item. Typically implied; however, can be defined for completeness.</p> <p>Syntax: <code>html_li_end</code></p>
html_menu	<p>Start of a menu. <code>html_li</code> identifies menu items.</p> <p>Syntax: <code>html_menu(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <MENU> <p>Example:</p> <pre>do html_menu('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_menu_end</pre>
html_menu_end	<p>End of a menu.</p> <p>Syntax: <code>html_menu_end</code></p>
html_ol	<p>Start of an ordered list. List items typically display indented to the right with a number to the left. <code>html_li</code> identifies list items.</p> <p>Syntax: <code>html_ol(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <p>Example:</p> <pre>do html_ol('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_ol_end</pre>
html_ol_end	<p>End of an ordered list.</p> <p>Syntax: <code>html_ol_end</code></p>
html_ul	<p>Start of an unordered list. List items typically display indented to the right with a bullet to the left. <code>html_li</code> identifies list items.</p> <p>Syntax: <code>html_ul(string attributes)</code></p> <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <p>Example:</p> <pre>do html_ul('') do html_li('') print 'First item' () do html_li('') print 'Second item' ()</pre>

Procedure	Description
	<pre>do html_li('') print 'Last item' () do html_ul_end</pre>
html_ul_end	End of an unordered list. Syntax: html_ul_end

HTML Table Procedures

Table 72 HTML Table Procedures

Procedure	Description
html_caption	Start of a table caption. Syntax: html_caption(<i>string attributes</i>) <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <CAPTION>
html_caption_end	End of a table caption. Typically implied; however, can be defined for completeness. Syntax: html_caption_end
html_table	Start of a table. Syntax: html_table(<i>string attributes</i>) <ul style="list-style-type: none"> ● <i>attributes</i> = HTML attributes inside <TABLE> Common attributes: border—Displays a border around each table cell. width—Table width in pixels. cols—Number of table columns. (Example: COLS=4) Example: Database records in a tabular format. html_caption_end, html_tr_end, html_td_end, and html_th_end are used for completeness; however, they are typically implied. <pre>!start the table & display the column headings do html_table('border') do html_caption('') print 'Customer Records' (1,1) do html_caption_end do html_tr('') do html_th('') print 'Cust No' (+1,1) do html_th_end do html_th('') print 'Name" (,10) do html_th_end do html_tr_end ! display each record begin-select do html_tr('') do html_td('') cust_num (1,1,6) edit 099999</pre>

Procedure	Description
	<pre>do html_td_end do html_td('') name (1,10,25) do html_td_end do html_tr_end next-listing skiplines=1 need=1 from customers end-select ! end the table do html_table_end</pre>
html_table_end	<p>End of a table.</p> <p>Syntax: html_table_end</p>
html_td	<p>Start of a new column in a table row. The text that follows displays within the column.</p> <p>Syntax: html_td(string attributes)</p> <ul style="list-style-type: none"> ● attributes = HTML attributes inside <TD>
html_td_end	<p>End of a column in a table. Typically implied; however, can be defined for completeness.</p> <p>Syntax: html_td_end</p>
html_th	<p>Start of a new column header in a table row. The text that follows displays as the column header.</p> <p>Syntax: html_th(string attributes)</p> <ul style="list-style-type: none"> ● attributes = HTML attributes inside <TH>
html_th_end	<p>End of a column header in a table. Typically implied; however, can be defined for completeness.</p> <p>Syntax: html_th_end</p>
html_tr	<p>Start of a new table row.</p> <p>Syntax: html_tr(string attributes)</p> <ul style="list-style-type: none"> ● attributes = HTML attributes inside <TR>
html_tr_end	<p>End of a table row. Typically implied; however, can be defined for completeness.</p> <p>Syntax: html_tr_end</p>

5

Encoding in Production Reporting

In This Chapter

Encoding Methods	315
Encoding Keys in SQR.INI	315
Encodings Supported without Using Unicode Internally	318
Encodings Supported in Production Reporting	320

Encoding Methods

You can setup Production Reporting to:

- Read character streams into the system by "widening" them into 16-bit character strings.
- Use Unicode internally to normalize data.

The default method is to read character streams into the system. To override the default and use Unicode internally, set the following in SQL.INI:

```
UseUnicodeInternal=TRUE
```

When `UseUnicodeInternal=TRUE`, any combination of encodings is valid in a single Production Reporting run, including ASCII and EBCDIC.

Encoding Keys in SQR.INI

You can define encoding keys in the following sections of SQR.INI:

- [Default-Settings]
- [Environment]

Encoding Keys in the [Default-Settings] Section

Table 73 Encoding Keys in the [Default-Settings] Section

Encoding Key	Description
<code>UseUnicodeInternal</code>	TRUE uses Unicode internally.
<code>AutoDetectUnicodeFiles</code>	TRUE automatically detects UCS-2 encoded files.

Encoding Key	Description
Substitution-Character	Defines a substitution character on a character set by character set basis.

UseUnicodeInternal Key

Forces the use of Unicode internally. When `UseUnicodeInternal=TRUE`, any combination of encodings is valid in a single Production Reporting run, including ASCII and EBCDIC.

AutoDetectUnicodeFiles Key

By convention, all UCS-2 encoded files start with a Byte Order Mark (BOM). The BOM is the Unicode character `ZERO WIDTH NO-BREAK SPACE` that has a hexadecimal value of `0xFEFF`. The BOM serves two purposes:

- Indicates that the file is encoded as UCS-2 (two bytes per character)
- Indicates the order in which the individual bytes of each Unicode character are written to the file.

On little-endian architectures such as Intel, the high order byte is written first so the BOM is physically recorded in the file as `0xFFFE`. On big-endian architectures, the BOM is recorded as `0xFEFF`.

If auto-detection of UCS-2 encoded files is enabled, Production Reporting checks whether the first two bytes of each file that it opens equal either `0xFEFF` or `0xFFFE`. If so, the file reads as a UCS-2 encoded file.

If an `ENCODING` directive is specified on an `OPEN` statement, Production Reporting does not attempt to auto-detect. It uses the encoding specified.

The BOM is not considered part of the file when performing fixed field width file I/O. In other words, reading 2 bytes from a UCS-2 file after it is opened returns the first Unicode character after the BOM, not the BOM itself.

When creating a UCS-2 output file, Production Reporting writes a BOM to the file as the file's first two bytes.

Substitution-Character Key

Allows for the substitution character to be defined on a character set by character set basis. The substitution character is the character placed in the output when a Unicode character does not exist in the target encoding. For readability's sake and to avoid character conversion problems when moving INI files between platforms, specify the substitution character as a hexadecimal string.

The format of the entry is:

```
[Default-Settings]
SUBSTITUTION-CHARACTER=XX EncodingName1 [, XX EncodingName2...]
```

where XX is the complete hexadecimal representation of the substitution character and EncodingName is a valid encoding name. Additionally, you can use the encoding name *Default* to specify the substitution character for all the encodings not explicitly listed. A default substitution character is used whenever no substitution character is explicitly or implicitly specified in the *Default* settings.

Encoding Keys in the [Environment] Section

Table 74 describes the encoding keys in the [Environment] section of SQR.INI. All of these encoding settings will accept as valid values any of the encoding literals. Any encoding can be specified for any encoding setting entry.

Table 74 Encoding Keys in the [Environment] Section

Encoding Key	Description
Encoding	Default encoding.
Encoding-Console	Encoding used for console input and output.
Encoding-Database	Encoding used for interfacing with the database.
Encoding-File-Input	Default encoding used for “OPEN” for-reading files and argument files.
Encoding-File-Output	Default encoding used for “OPEN” for-writing files.
Encoding-Report-Output	Default encoding used for report output, such as, SPF, LIS, HTM, etc.
Encoding-SQR-Source	Encoding used for Production Reporting source and include files.

The following is an example of encoding settings in the [Environment] section:

```
[Environment:Common]
Encoding=ISO-8859-1
Encoding-File-Output=Greek
Encoding-File-Input=Shift-Jis
Encoding-Report-Output=UCS-2
Encoding-database=utf-8
Encoding-console=ascii
Encoding-SQR-Source=ucs-2
```

If these keys are not specified, encodings default to ASCII. The `Encoding` setting, which specifies the default encoding, can be overridden by the other encoding settings.

As with other [Environment] section settings, Production Reporting first checks the [Environment] section of its database type and then checks the [Common Environment] section. For example, an ODBC version of Production Reporting first checks the [Environment:ODBC] section of SQR.INI for a setting and, if not found, then checks the [Environment:Common] section.

To access these encoding settings within an Production Reporting program, use the following reserved variables.

- \$SQR-ENCODING-REPORT-OUTPUT
{SQR-ENCODING-REPORT-OUTPUT}
- \$SQR-ENCODING-FILE-INPUT
{SQR-ENCODING-FILE-INPUT}
- \$SQR-ENCODING-FILE-OUTPUT
{SQR-ENCODING-FILE-OUTPUT}
- \$SQR-ENCODING-CONSOLE
{SQR-ENCODING-CONSOLE}
- \$SQR-ENCODING-SOURCE
{SQR-ENCODING-SOURCE}
- \$SQR-ENCODING-DATABASE
{SQR-ENCODING-DATABASE}

Encodings Supported without Using Unicode Internally

When you do not use Unicode internally (`UseUnicodeInternal=FALSE`), Production Reporting supports the following encodings:

- ASCII
- EBCDIC
- Shift-JIS
- EUC-J
- EBCDIK290
- EBCDIK1027
- UTF-8
- UCS-2

When you do not use Unicode internally, Production Reporting does not perform character conversion. As a result, you can only mix encodings that are logical supersets or subsets of each other. For example, you *can* combine Shift-JIS and ASCII or EBCDIC and EBCDIK1027; however, you *cannot* combine Shift-JIS with EBCDIC or UTF-8.

[Table 75](#) identifies a valid set of encoding settings for an Production Reporting run. For simplicity, `ENCODING-SQR-SOURCE` and `ENCODING-CONSOLE` have not been specified and are assumed to be either ASCII or EBCDIC, depending on the platform.

Table 75 Compatible Encodings without Unicode

Encoding-File-Input	Encoding-Database	Encoding-File-Output	Encoding-Report-Output
ASCII	ASCII	ASCII	ASCII

Encoding-File-Input	Encoding-Database	Encoding-File-Output	Encoding-Report-Output
ASCII	ASCII	ASCII	Shift-JIS
ASCII	ASCII	ASCII	JEUC
ASCII	ASCII	ASCII	UTF-8/UCS-2
ASCII	Shift-JIS	ASCII	ASCII
ASCII	Shift-JIS	ASCII	Shift-JIS
ASCII	JEUC	ASCII	ASCII
ASCII	JEUC	ASCII	JEUC
ASCII	UTF-8/UCS-2	ASCII	ASCII
ASCII	UTF-8/UCS-2	ASCII	UTF-8/UCS-2
ASCII	Shift-JIS	Shift-JIS	Shift-JIS
ASCII	JEUC	JEUC	JEUC
ASCII	UTF-8/UCS-2	UTF-8/UCS-2	UTF-8/UCS-2
EBCDIC	EBCDIC	EBCDIC	EBCDIC
EBCDIC	EBCDIC	EBCDIC	EBCDIK290 or 1027
EBCDIC	EBCDIK290 or 1027	EBCDIC	EBCDIC
EBCDIC	EBCDIK290 or 1027	EBCDIC	EBCDIK290 or 1027
EBCDIC	EBCDIK290 or 1027	EBCDIK290 or 1027	EBCDIK290 or 1027
EBCDIK290 or 1027	EBCDIK290 or 1027	EBCDIC	EBCDIK290 or 1027
EBCDIK290 or 1027	EBCDIK290 or 1027	EBCDIK290 or 1027	EBCDIK290 or 1027
Shift-JIS	Shift-JIS	ASCII	Shift-JIS
Shift-JIS	Shift-JIS	Shift-JIS	Shift-JIS
JEUC	JEUC	ASCII	JEUC
JEUC	JEUC	JEUC	JEUC
UTF-8/UCS-2	UTF-8/UCS-2	ASCII	UTF-8/UCS-2
UTF-8/UCS-2	UTF-8/UCS-2	UTF-8/UCS-2	UTF-8/UCS-2

Note:

Production Reporting works differently on EBCDIC platforms than in ASCII. Specifically, the `UseUnicodeInternal` setting has no effect on EBCDIC platforms. Instead, the distribution

media contains two sets of executables (SQR, SQRT, and SQRP), where one set works with non-unicode processing and the other works for unicode processing.

Encodings Supported in Production Reporting

Table 76 Production Reporting-Supported Encodings

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable Production Reporting Names
CP10004	Arabic	Macintosh Arabic	Microsoft & IBM	CP10004
CP1256	Arabic		Microsoft & IBM	CP1256
CP20420	Arabic	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20420
CP28596	Arabic	Arabic Alphabet (ISO)	Microsoft & IBM	CP28596
CP720	Arabic	Transparent ASMO	Microsoft & IBM	CP720
CP864	Arabic		Microsoft & IBM	CP864
ISO 8859-6	Arabic	ISOLatinArabic	International or National Standard	Arabic
CP708	Arabic	ASMO708	Microsoft & IBM	CP708
CP1257	Baltic		Microsoft & IBM	CP1257
CP28594	Baltic	Baltic Alphabet (ISO)	Microsoft & IBM	CP28594
CP775	Baltic		Microsoft & IBM	CP775
ISO 8859-4	Baltic	Latin4	International or National Standard	Latin4, ISO-8859-4
ISO 8859-13	Baltic	Latin7	International or National Standard	Latin7, ISO-8859-13
ISO 8859-14	Celtic	Latin8	International or National Standard	ISO-8859-14
ISO 2022-CN	Chinese		International or National Standard	ISO-2022-CN
GB18030	Chinese		International or National Standard	GB18030
HKSCS	Chinese	Big5-HKSCS	International or National Standard	Big5-HKSCS
CP936	Chinese, Simplified	GBK	Microsoft & IBM	CP936
GB2312	Chinese, Simplified	EUC-CN, EUC-SC	International or National Standard	GB2312, EUC-CN

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable Production Reporting Names
HZ-GB-2312	Chinese, Simplified	HZ-GB-2312	International or National Standard	HZ
Big5	Chinese, Traditional		International or National Standard	Big5
BIG5+	Chinese, Traditional		International or National Standard	BIG5+
CNS-11643-1986	Chinese, Traditional	EUC-TW	International or National Standard	CNS-11643-1986
CNS-11643-1992	Chinese, Traditional	EUC-TW	International or National Standard	CNS-11643-1992
EUC-TW	Chinese, Traditional	CNS-11643-1986, CNS-11643-1992	UNIX	CNS-11643-1992
GB12345	Chinese, Traditional		International or National Standard	GB12345
CP10002	Chinese, Traditional	Macintosh Traditional Chinese	Microsoft & IBM	CP10002
CP950	Chinese, Traditional		Microsoft & IBM	CP950
CP10007	Cyrillic	Macintosh Cyrillic	Microsoft & IBM	CP10007
CP1251	Cyrillic	MS Windows Cyrillic (Slavic)	Microsoft & IBM	CP1251
CP20866	Cyrillic	Cyrillic Alphabet, KOI8-R	Microsoft & IBM	CP20866
CP20880	Cyrillic	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20880
CP21025	Cyrillic	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP21025
CP21866	Cyrillic	Ukrainian KOI8-RU	Microsoft & IBM	CP21866
CP28595	Cyrillic	Cyrillic Alphabet (ISO)	Microsoft & IBM	CP28595
CP855	Cyrillic	IBM Cyrillic	Microsoft & IBM	CP855
CP866	Cyrillic	MS DOS Russian	Microsoft & IBM	CP866
ISO 8859-5	Cyrillic	ISOLatinCyrillic	International or National Standard	ISOLatinCyrillic
CP10006	Greek	Macintosh Greek 1	Microsoft & IBM	CP10006
CP1253	Greek		Microsoft & IBM	CP1253
CP20423	Greek	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20423

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable Production Reporting Names
CP28597	Greek	Greek Alphabet (ISO)	Microsoft & IBM	CP28597
CP737	Greek		Microsoft & IBM	CP737
CP869	Greek	IBM Modern Greek	Microsoft & IBM	CP869
ISO 8859-7	Greek	ISOLatinGreek	International or National Standard	Greek
CP10010	Gurmukhi	Macintosh Gurmukhi	Microsoft & IBM	CP10010
CP10005	Hebrew	Macintosh Hebrew	Microsoft & IBM	CP10005
CP1255	Hebrew		Microsoft & IBM	CP1255
CP28598	Hebrew	Hebrew Alphabet (ISO)	Microsoft & IBM	CP28598
CP38598	Hebrew	ASCII + Hebrew and private use characters	Microsoft & IBM	CP38598
CP862	Hebrew		Microsoft & IBM	CP862
ISO 8859-8	Hebrew	ISOLatinHebrew	International or National Standard	Hebrew
CP10079	Icelandic	Macintosh Icelandic	Microsoft & IBM	CP10079
CP861	Icelandic	MS DOS Icelandic	Microsoft & IBM	CP861
CCSID 1027	Japanese	EBCDIK	Microsoft & IBM	CCSID-1027, EBCDIK1027
CCSID 290	Japanese	EBCDIK	Microsoft & IBM	CCSID-290, EBCDIK290
CCSID 942	Japanese		Microsoft & IBM	CCSID-942
CP10001	Japanese	Macintosh Japanese	Microsoft & IBM	CP10001
CP20290	Japanese	(full/half width Latin & halfwidth katakana)	Microsoft & IBM	CP20290
CP21027	Japanese	(halfwidth Latin, halfwidth katakana&private use)	Microsoft & IBM	CP21027
CP932	Japanese		Microsoft & IBM	CP932
EUC-JP	Japanese		UNIX	EUC-J, JEUC
EUC-JP-JISROMAN	Japanese		UNIX	EUC-JP-JISROMAN
ISO-2022-JP	Japanese		International or National Standard	ISO-2022-JP
JIS_X_0201	Japanese	HalfWidthKatakana	International or National Standard	JIS_X_0201, IBM897

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable Production Reporting Names
JIS_X_0208	Japanese		International or National Standard	JIS_X_0208
Shift-JIS	Japanese	MS_Kanji	Microsoft & IBM	Shift-JIS, SJIS
CP10003	Korean	Macintosh Korean	Microsoft & IBM	CP10003
CP1361	Korean	Korean Johab (based on KSC 5861-1992)	Microsoft & IBM	CP1361
CP949	Korean		Microsoft & IBM	CP949
EUC-KR	Korean	KS_C_5861-1992	UNIX	EUC-KR, EUC-K
ISO-2022-KR	Korean	KS_C_5601-1987	International or National Standard	ISO-2022-KR
Johab	Korean		International or National Standard	Johab
CP10000	Latin	Macintosh Roman	Microsoft & IBM	CP10000
CP10029	Latin	Macintosh Latin2	Microsoft & IBM	CP10029
CP10082	Latin	(with mathematical symbols)	Microsoft & IBM	CP10082
CCSID1047	Latin	EBCDIC (for IBM Open Systems platform)	Microsoft & IBM	CCSID1047
CP20261	Latin	(with private use characters)	Microsoft & IBM	CP20261
CP20269	Latin		Microsoft & IBM	CP20269
CP20273	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20273
CP20277	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20277
CP20278	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20278
CP20280	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20280
CP20284	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20284
CP20285	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20285
CP20297	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20297

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable Production Reporting Names
CP20833	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20833
CP20871	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20871
CP28591	Latin	ASCII + Latin accented vowels	Microsoft & IBM	CP28591
CP28593	Latin	Latin 3 Alphabet (ISO)	Microsoft & IBM	CP28593
CP850	Latin	MS DOS Multilingual, MS-DOS Latin1	Microsoft & IBM	CP850
CP870	Latin	(with fullwidth punctuation)	Microsoft & IBM	CP870
HP-ROMAN8	Latin	csHPRoman8, r8, roman8	HP	HP-ROMAN8
ISO 8859-1	Latin	Latin1	International or National Standard	Latin1, ISO-8859-1
ISO 8859-15	Latin	Latin1 + Euro symbol & accented characters	International or National Standard	ISO-8859-15
ISO 8859-2	Latin	Latin2	International or National Standard	Latin2, ISO-8859-2
UTF8-EBCDIC	Latin		Unicode	UTF8-EBCDIC
CP863	Latin, Canadian French	MS DOS Canadian French	Microsoft & IBM	CP863
CP28592	Latin, Central European	Central European Alphabet (ISO)	Microsoft & IBM	CP28592
CP1250	Latin, Eastern Europe		Microsoft & IBM	CP1250
CP20905	Latin, Esperanto	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20905
CP860	Latin, Portuguese	MS DOS Portuguese	Microsoft & IBM	CP860
ISO 8859-3	Latin, Southeast European	Latin3	International or National Standard	Latin3, ISO-8859-3
ASCII	Latin, US English	US-ASCII, CP367	International or National Standard	ASCII, ANSI
CP037	Latin, US English	EBCDIC	Microsoft & IBM	CP037
CP1026	Latin, US English	EBCDIC	Microsoft & IBM	CP1026
CP1252	Latin, US English	MS Windows Latin1 (ANSI)	Microsoft & IBM	CP1252
CP20105	Latin, US English	US ASCII	Microsoft & IBM	CP20105

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable Production Reporting Names
CP437	Latin, US English	MS-DOS Latin US	Microsoft & IBM	CP437
CP500	Latin, US English	EBCDIC	Microsoft & IBM	CP500
CP875	Latin, US English	EBCDIC	Microsoft & IBM	CP875
CP10017	Malayan	Macintosh Malayan	Microsoft & IBM	CP10017
CP865	Nordic	MS DOS Nordic	Microsoft & IBM	CP865
ISO 8859-10	Nordic	Latin6	International or National Standard	Latin6, ISO-8859-10
CP852	Slavic	MS DOS Slavic	Microsoft & IBM	CP852
Adobe-Symbol-Encoding	Symbol	(used in PS printers)	Adobe	Adobe-Symbol-Encoding
CP10008	Symbol	Macintosh RSymbol (Right-left symbol)	Microsoft & IBM	CP10008
CP20838	Thai	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20838
CP874	Thai	IBMThai	Microsoft & IBM	CP874
ISO 8859-11 (draft)	Thai	ISOLatinThai	International or National Standard	Thai
CP10081	Turkish	Macintosh Turkish	Microsoft & IBM	CP10081
CP1254	Turkish		Microsoft & IBM	CP1254
CP28599	Turkish	Turkish (ISO)	Microsoft & IBM	CP28599
CP857	Turkish	IBM Turkish	Microsoft & IBM	CP857
ISO 8859-9	Turkish	Latin5	International or National Standard	Latin5, ISO-8859-9
BMP	Unicode		Unicode	BMP
Java	Unicode	(way of representing Unicode chars in ASCII)	Sun	Java
UCS2	Unicode	ISO-10646-UCS2, UTF16	Unicode	UCS2
Unicode Big-endian	Unicode		Unicode	big-endian
Unicode Little-endian	Unicode		Unicode	little-endian
UTF7	Unicode		Unicode	UTF7

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable Production Reporting Names
UTF8	Unicode		Unicode	UTF8
UTF8-EBCDIC	Unicode		Unicode	UTF8-EBCDIC
CP1258	Vietnamese		Microsoft & IBM	CP1258

In This Chapter

Installation of SQR.INI	327
[Default-Settings] Section	328
[Environment: environment] Section	333
[SQR Extension] Section	335
[Locale:local-name] Section	335
[Fonts] Section	337
[PDF Fonts] Section	338
[PDF Settings] Section	340
[HTML Fonts] Section	341
[HTML:Images] Section	342
[Enhanced-HTML] Section	343
[Color Map] Section	344
[MAP-ODBC-DB] Section	345
[MAP-DDO-DB] Section	345
[SQR Remote] Section	345

Installation of SQR.INI

The installation process installs a default initialization file called SQR.INI. This file contains settings and parameters that Production Reporting uses during the compile and execution phases.

On Windows platforms, SQR.INI is placed in the main Windows directory. (On Windows XP, the default directory name is "WINDOWS." On Windows 2000, the default directory name is "WINNT.") On all other platforms, SQR.INI is placed in the same directory as the executable images (where SQRDIR points).

For Windows Platforms Only

Production Reporting looks for the initialization file in the following locations:

1. The file name specified by `-ZIF{file}`.
2. The directory where the executable image resides.

3. The Windows system directory.

Since the required environment variable *SQRDIR* is defined in the initialization file, Production Reporting produces an error message if it cannot find the file.

For All Other Platforms

Production Reporting looks for the initialization file in the following order:

1. The file name specified by `-ZIF{file}`.
2. The current working directory.
3. The directory specified with *SQRDIR*.

Since the required environment variable *SQRDIR* is defined at the operating system level, the initialization file does not need to be available.

You can make changes or additions to SQR.INI if desired.

The format of the file is as follows:

```
; Comments are lines which start with a semicolon. The semicolon
; must be the first character of the line and therefore cannot be
; part of another line.
;
; Leading and trailing space characters are ignored. To preserve
; the space characters you must surround the value with either
; single (') or double (") quote characters. Production Reporting will ;
; remove them when the entry is processed.
;
[Section_Name]
Entry = Value
.
.
[Another_Section_Name]
Entry = Value
.
.
```

[Default-Settings] Section

[Default-Settings] defines Production Reporting default actions.

Table 77 Entries in [Default-Settings]

Entry	Value	Description
AllowDateAsChar	TRUE FALSE Default = FALSE	By default, Production Reporting produces an error when a dynamic column specification does not match the column's database definition. That is, character equals character, date equals date, and numeric equals numeric. When set to TRUE,

Entry	Value	Description
		<p>Production Reporting allows characters to equal either character or date columns.</p> <p>When a date column is “type cast” as a character, Production Reporting creates the string according to the following rules:</p> <ul style="list-style-type: none"> ● For DATETIME columns, Production Reporting uses the first database-dependent format in Table 61 on page 251. ● For DATE columns, Production Reporting uses the format in Table 62 on page 252. ● For TIME columns, Production Reporting uses the format in Table 63 on page 252. <p>In the following example, AllowDateAsChar=True. This allows \$Col1 to be either date or text.</p> <pre> Begin-Select [\$Col1] &col1=Text [\$Col2] &col2=Date [\$Col3] &col3=Number from MyTable End-Select </pre>
AutoDetectUnicodeFiles	TRUE FALSE Default = FALSE	<p>When set to TRUE, Production Reporting auto-detects UCS-2 encoded files.</p> <p>See “AutoDetectUnicodeFiles Key” on page 316 for more information.</p>
CSVSeparator	Comma Semicolon Space Tab Default = Comma	<p>Defines the character used as a delimiter when creating CSV files.</p> <p>If the CSVSeparator setting is missing from SQR.INI, the default value of Comma is used.</p> <p>Note: Specifying the CSVSeparator as a semicolon or a space is only supported with the Production Reporting Server. Using this setting in Oracle Enterprise Performance Management Workspace, Fusion Edition is not recommended and may create corrupt files.</p>
DEFAULT-NUMERIC	INTEGER FLOAT DECIMAL[(p)] V30	<p>Specifies the default numeric type for variables. -DNT and DECLARE-VARIABLE override this setting. (See “DECLARE-VARIABLE” on page 150.)</p>
ExpirationWarningMessage	TRUE FALSE Default = TRUE	<p>Controls whether to print the license expiration warning message.</p>

Entry	Value	Description
FormatforRowsetXML	BIP SQR Default = SQR	Format in which to create the XML file defined with OPEN-RS , WRITE-RS , and CLOSE-RS .
ImageCompression	0 - 9 Default = 6	Defines the compression level when <code>PRINT-IMAGE</code> references a <code>BINARY</code> variable.
LOCALE	Name of a locale defined in <code>SQR.INI</code> or the name <code>SYSTEM</code> .	Defines the initial locale that Production Reporting loads when the program starts to execute. The value of <code>SYSTEM</code> is used to reference the default locale. (See “ALTER-LOCALE” on page 41.)
NewGraphics	TRUE FALSE Default = FALSE	<p>When set to <code>FALSE</code>, Production Reporting uses Graftsman chart package. When set to <code>TRUE</code>, Production Reporting uses Jchart chart package.</p> <p>Set <code>NewGraphics=TRUE</code> to use the following features:</p> <ul style="list-style-type: none"> ● <code>COLOR-PALETTE</code> (See “Specifying Chart Data Series Colors” in Volume 1 of the <i>Hyperion SQR Production Reporting Developer’s Guide</i>.) ● <code>ITEM-COLOR</code> (See “Specifying Chart Item Colors” in Volume 1 of the <i>Hyperion SQR Production Reporting Developer’s Guide</i>.) ● <code>ITEM-SIZE</code> (See “DECLARE-CHART” on page 100 and “PRINT-CHART” on page 263.) ● <code>Y-AXIS-MASK</code> and <code>Y2-AXIS-MASK</code> (See “DECLARE-CHART” on page 100 and “PRINT-CHART” on page 263.) ● <code>Y2 Syntax</code> (See “DECLARE-CHART” on page 100 and “PRINT-CHART” on page 263.) ● <code>Combination Charts</code> (See “Creating Combination Charts” in Volume 1 of the <i>Hyperion SQR Production Reporting Developer’s Guide</i>.) ● <code>Bubble Charts</code> (See “Creating Bubble Charts” in Volume 1 of the <i>Hyperion SQR Production Reporting Developer’s Guide</i>.)
ODBCExecuteRetry	TRUE FALSE Default = FALSE	<p>Describes whether to retry an <code>EXECUTE</code> command.</p> <p><code>FALSE</code>, retries <code>EXECUTE</code> and returns an error from the database.</p> <p><code>TRUE</code> does not attempt to retry.</p>

Entry	Value	Description
OracleWeakCursor	STOP WARN SKIP Default = WARN	<p>Describes what to do when a 'Weak' reference cursor is processed.</p> <p>STOP displays an error or warning message and terminates the Production Reporting procedure.</p> <p>WARN displays an error or warning message, and the Production Reporting procedure continues.</p> <p>SKIP does <i>not</i> display an error or warning message, and the Production Reporting procedure continues.</p>
OUTPUT-FILE-MODE	LONG SHORT Default = LONG	<p>Specifies the filename convention used for HTML output. SHORT specifies DOS style (8.3) and LONG specifies UNIX style (non 8.3). (Ignored on 16-bit platforms) <small>DECLARE-TOC and -Burst force Output-File-Mode = LONG.</small></p> <p>The following represent the file formats for UNIX, DOS, and Windows.</p> <p>SQR and SQR: {Program} is the name of the SQR/SQT file without the extension</p> <p>For Output-File-Mode = SHORT, SQR-generated filenames are limited to a DOS 8.3 format</p> <ul style="list-style-type: none"> ● Output file = {Program}.LIS for first, and {Program}.Lnn for multi-reports ● SFP file = {Program}.SFP for first, and {Program}.Snn for multi-reports ● PDF file = {Program}.PDF for first; and {Program}.Pnn for multi-reports ● HTM file = {Program}.HTM for "frame, and {Program}.Hbb for report bodies ● GIF file={Program}.Gxx for all reports <p>bb ranges from 00 to 99 and represents the report number.</p> <p>nn ranges from 01 to 99 and represents the report number.</p> <p>xx ranges from 00 to ZZ and represents the graphic number.</p> <p>For Output-File-Mode = LONG, SQR-generated filenames are not constrained to a DOS 8.3 format. {Output}={Program} of first report and {Program}_nn for multi-reports.</p> <ul style="list-style-type: none"> ● Output file = {Output}.LIS ● SPF file = {Output}.SPF ● PDF file = {Output}.PDF

Entry	Value	Description
		<ul style="list-style-type: none"> ● GIF file = {Output}_zz.SPF ● HTM files = {Output}.HTM, {Output}_bb.HTM, {Output}_frm.HTM, {Output}_toc.HTM, {Output}_nav.htm <p>bb ranges from 01 to ZHJOZI and represents the bursted page group number in radix 36.</p> <p>nn ranges from 01 to 99 and represents the report number.</p> <p>zz ranges from 01 to ZHJOZI and represents the graphic number in radix 36.</p>
		<p>SQRP: {Filename} is the name of the SPF file without the extension</p> <p>For Output-File-Mode = SHORT, SQR-generated filenames are limited to a DOS 8.3 format.</p> <ul style="list-style-type: none"> ● Output file = {Filename}.LIS ● GIF file = {Filename}.Gxx ● PDF file = {Filename}.PDF ● HTM file = .HTM and {Filename}.H00 <p>xx ranges from 00 to ZZ and represents the graphic number.</p> <p>For Output-File-Mode = LONG, SQR-generated filenames are not limited to a DOS 8.3 format.</p> <ul style="list-style-type: none"> ● Output file = {Filename}.LIS ● PDF file = {Filename}.PDF ● GIF file = {Filename}_zz.SPF ● HTM files = {Filename}.HTM, {Filename}_bb.HTM, {Filename}_frm.HTM, {Filename}_toc.HTM, {Filename}_nav.htm <p>bb ranges from 01 to ZHJOZI and represents the bursted page group number in radix 36.</p> <p>zz ranges from 01 to ZHJOZI and represents the graphic number in radix 36.</p>
OutputFormFeedWithDashD	TRUE FALSE Default = FALSE	When set to TRUE, -Dnn outputs the Form-Feed character that denotes a page break.
OutputTwoDigitYearWarningMsg	TRUE FALSE Default = TRUE	When set to TRUE, Production Reporting generates a warning message (sent to the warning file) when a YY or RR date edit mask is encountered during a program run. This affects only Production Reporting code that is processed.

Entry	Value	Description
PrinterHT	Standard Enhanced Default = Enhanced	Controls the HTML output produced by <code>-PRINTER:HT</code> . Standard produces version 2.0 HTML files with report content inside <code><PRE></PRE></code> tags. Enhanced maps <code>-PRINTER:HT</code> to <code>-PRINTER:EH</code> and produces content formatted with version 1.1 XHTML tags.
SUBSTITUTION-CHARACTER=XX EncodingName1 [,xx EncodingName2...]	XX is the hexadecimal representation of the substitution variable	Allows substitution characters to be defined on a character set by character set basis. See “Substitution-Character Key” on page 316 for more information.
TreatBinaryColumnAsText	TRUE FALSE Default = TRUE	Defines whether to treat BINARY columns as TEXT columns.
UseUnicodeInternal	TRUE FALSE Default = FALSE	By default, Production Reporting reads character streams into the system by “widening” them into 16-bit character strings. When set to TRUE, Production Reporting uses Unicode internally to normalize the data.
UseY2kCenturyAlgorithm	TRUE FALSE Default = FALSE	When set to TRUE, Production Reporting treats the YY date edit mask as though it is an RR edit mask.

Note:

Use the setting V30 to handle numbers in the same manner as in prior releases (before V4.0). Specifically, all numeric variables and literals are declared as FLOAT, including integer literals.

[Environment: environment] Section

[Environment:{ Common | DB2 | Informix | ODBC | Oracle | Sybase | DDO}] defines environment variables used by Production Reporting. An environment variable can be defined in multiple environment sections; however, a definition in a database-specific environment section takes precedence over an assignment in [Environment:Common].

The following environment variables can be set:

- **SQRDIR**—Default directory for all invocations of Production Reporting.
- **SQRFLAGS**—Default command-line flags for all invocations of Production Reporting.
- **DSQUERY** (Sybase only)—Default Sybase server to use.

On Windows systems, *SQRDIR* is required and is automatically defined in the appropriate database-specific environment section during the Production Reporting installation. The other environment variables are optional.

Using the Java Virtual Machine

When you use Production Reporting to produce reports with HTML files or charts, Production Reporting must make several calls to Java routines. The number of HTML reports and charts to produce can affect the runtime of Production Reporting programs. To reduce the total runtime of an Production Reporting program, embed the Java Virtual Machine (JVM) directly into Production Reporting.

Table 78 Java Virtual Machine Environment Variable

Entry	Value	Description
SQR_USEJVM	TRUE FALSE Default = TRUE	If TRUE, Production Reporting uses the JVM for Enhanced HTML and JClass charting applications (NewGraphics). The JVM is embedded directly into Production Reporting. If FALSE, Production Reporting invokes separate subprocesses for Enhanced HTML and JClass charting applications (NewGraphics). Note: Callable Production Reporting uses the JVM only when this entry equals TRUE.

DDO Variables

Table 79 DDO Variables in the [Environment] Section

DDO Variable	Description
SQR_DDO_JRE_CLASS	Classpath for DDO drivers and support files.
SQR_DDO_JRE_CLASSn	(Optional) Additional entries to the classpath.
SQR_DDO_JRE_PATH	Classpath information for the local JRE.
SQR_DDO_JRE_INIT_HEAP	Initial Java heap size.
SQR_DDO_JRE_MAX_HEAP	Maximum Java heap size.
SQR_DDO_JRE_NOCLASSGC	Disables class garbage collection.
SQR_DDO_JRE_NOJIT	Disables the JIT compiler (same as Java.compiler=NONE)
SQR_DDO_JRE_VERBOSE	Logs Java class loading and garbage collection events into the <code>vm.out</code> output file.

Each of these entries is automatically entered upon product installation (Windows only). You can specify additional classpath entries using up to nine *SQR_DDO_JRE_CLASSn* variables, where *n* is a number from 1-9. These additional variables are available to augment the normal 512-character line limit for entries in *SQR.INI*.

Encoding Keys

For detailed information about encoding keys in the [Environment] section SQR.INI, see “Encoding Keys in the [Environment] Section” on page 317.

[SQR Extension] Section

On Windows systems only, [SQR Extension] defines DLLs containing new user functions (ufunc) and user calls (ucall). Ufunc and ucall reside inside SQREXT.DLL and/or other DLLs.

When SQR.DLL and SQRT.DLL are loaded, they look for SQREXT.DLL in the same directory, and for any DLLs specified in [SQR Extension] such as:

```
[SQR Extension]
c:\sqrexts\sqrext1.dll=
c:\sqrexts\sqrext2.dll=
c:\sqrexts\sqrext3.dll=
```

Any new extension DLLs containing new user functions must be listed in [SQR Extension] in SQR.INI. For more information, see “Interoperability” in Volume 1 of the *Hyperion SQR Production Reporting Developer's Guide*.

For Windows/Oracle, Production Reporting uses dynamic binding of Oracle routines. When Production Reporting tries to access an Oracle database, it searches for the Oracle DLL as follows:

- The file described by the value of ORACLE_DLL in [Environment:Oracle].
- OCIW32.DLL (Oracle supplied)

[Locale:local-name] Section

[Locale:locale-name] defines the default settings for the locale identified by *locale-name* (which can consist of A-Z, 0-9, hyphen, or underscore). A number of locales are predefined in SQR.INI. Depending on your application, the settings for these locales may have to be altered or new locales may have to be added. A locale can be referenced or altered at run time using ALTER-LOCALE.

Note:

The SYSTEM locale is provided for your reference, but is commented out. The settings for the SYSTEM locale, if set, are ignored. Use ALTER-LOCALE to change the SYSTEM locale settings at run time.

Table 80 Entries in [Locale:locale-name]

Entry	Description
NUMBER-EDIT-MASK	Default numeric edit mask format when the keyword NUMBER accompanies DISPLAY , MOVE , PRINT , or SHOW .
MONEY-EDIT-MASK	Default numeric edit mask format when the keyword MONEY accompanies DISPLAY , MOVE , PRINT , or SHOW .
DATE-EDIT-MASK	Default date edit mask format when the keyword DATE accompanies DISPLAY , MOVE , PRINT , or SHOW , or the LET datetostr() or strtodate() functions.
INPUT-DATE-EDIT-MASK	Default date format to use with the INPUT command when TYPE=DATE is specified with the command or the input variable is a DATE variable. If this entry is not specified, then the date must be entered in one of the formats in Table 62 on page 252 .
MONEY-SIGN	Character(s) to replace the '\$' edit character.
MONEY-SIGN-LOCATION	MONEY-SIGN character(s) location. Valid values are LEFT and RIGHT.
THOUSAND-SEPARATOR	Character to replace the ',' edit character.
DECIMAL-SEPARATOR	Character to replace the '.' edit character.
DATE-SEPARATOR	Character to replace the '/' character.
TIME-SEPARATOR	Character to replace the ':' character.
EDIT-OPTION-NA	Character(s) to replace the 'na' option.
EDIT-OPTION-AM	Character(s) to replace 'AM'.
EDIT-OPTION-PM	Character(s) to replace 'PM'.
EDIT-OPTION-AD	Character(s) to replace 'AD'.
EDIT-OPTION-BC	Character(s) to replace 'BC'.
DAY-OF-WEEK-CASE	How the case for the DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT entries are affected when used with the format codes 'DAY' or 'DY'. Valid values are UPPER, LOWER, EDIT, and NO-CHANGE. UPPER and LOWER forces the output to either all uppercase or lowercase, ignoring the case of the format code in the edit mask. Use EDIT to follow the case as specified with the format code in the edit mask. Use NO-CHANGE to ignore the case of the format code and output the day of week as explicitly listed in the DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT entries.
DAY-OF-WEEK-FULL	Full names for the days of the week. Production Reporting considers the first day of the week to be Sunday. All seven days must be specified.
DAY-OF-WEEK-SHORT	Abbreviated names for the days of the week. Production Reporting considers the first day of the week to be Sunday. All seven abbreviations must be specified.
MONTHS-CASE	How the case for the MONTHS-FULL or MONTHS-SHORT entries is affected when used with the format codes 'MONTH' or 'MON'. Valid values are UPPER, LOWER, EDIT, and NO-CHANGE. UPPER and LOWER force the output to either all uppercase or lowercase, ignoring the case of the format code in the edit mask. Use EDIT to follow the case as specified with the format code in the edit mask. Use NO-CHANGE to ignore the case of the format code and output the month as explicitly listed in the MONTHS-FULL or MONTHS-SHORT entries.
MONTHS-FULL	Full names for the months of the year. Production Reporting considers the first month of the year to be January. All 12 months must be specified.

Entry	Description
MONTHS-SHORT	Abbreviated names for the months of the year. Production Reporting considers the first month of the year to be January. All 12 abbreviations must be specified.

Table 81 Date Column Formats

Database	DATE Column Formats
DB2	YYYY-MM-DD
INFORMIX	MM/DD/YYYY
ODBC	DD-MON-YYYY

Table 82 Time Column Formats

Database	TIME Column Formats
DB2	HH24.MI.SS
ODBC	HH24:MI:SS

[Fonts] Section

[*Fonts*] lists the fonts available to Production Reporting when printing on Windows printer devices (using `-PRINTER:WP`). This section does not apply to PostScript or HP LaserJet printer types. See `DECLARE-PRINTER` for available fonts for alternate printer types.

Adding [Fonts] Entries

[*Fonts*] includes several predefined font entries. Add entries by using the font numbers 900 through 999. Each entry includes a font name, a font style (fixed or proportional), and a bold indicator. For example:

```
4=Arial,proportional
or
300=Courier New, fixed, bold
```

Note:

Proportional is assumed if the second parameter starts with "P". Bold is assumed if a third parameter is supplied.

Use `ALTER-PRINTER` and `DECLARE-PRINTER` to reference a font style.

Specifying Character Sets in Windows

For Windows, use `CharacterSet` to determine the Windows default character set or to specify a character set. This allows you to print any standard character set to a Windows printer (`-PRINTER:WP`) or to view an SPF file displaying the appropriate character set.

Syntax

```
CharacterSet=DEFAULT|AUTO|character_set
```

Arguments

DEFAULT

Reflects current Production Reporting functionality.

AUTO

Automatically senses the default character set of the Windows installation and uses the default set when generating reports.

character_set

Specifies one of these keywords: ANSI, ARABIC, BALTIC, CHINESEBIG5, EASTEUROPE, GB2312, GREEK, HANGUL, HEBREW, JOHAB, MAC, OEM, RUSSIAN, SHIFTJIS, SYMBOL, THAI, TURKISH, VIETNAMESE.

[PDF Fonts] Section

[PDF Fonts] defines whether to embed fonts into PDF documents. In addition, it lists the available fonts for Production Reporting when printing using `-PRINTER:PD`. Fonts are case sensitive.

Embedding Fonts

Table 83 [PDF Fonts] Entry

Entry	Value	Description
Embed	All { <i>font numbers</i> }	Whether to embed fonts; and if so, what fonts to embed. Enter one of the following values: <ul style="list-style-type: none">● ALL For example: Embed=All● A list of fonts numbers to embed. (Font numbers must be separated by spaces.) For example: Embed=1 3 5 8 64 If you do not specify a value, Production Reporting does not embed any fonts in the PDF file. This is the default action.

Available Fonts

Each entry in [PDF Fonts] is defined to be:

`font_number=Roman_Typeface[,CJK_Typeface,Character_Map]`

The following describes each part of the entry:

`font_number`

Font number used within the Production Reporting program

`Roman_Typeface`

Name of the typeface (font) for non-Chinese/Japanese/Korean characters

`CJK_Typeface`

Name of the typeface (font) for Chinese/Japanese/Korean characters

`Character_Map`

Name of the character map

Table 84 Legal Chinese, Japanese, and Korean Typeface - Character Map Combinations

Local	CJK Typeface	Character Map	Encoding Supported
Simplified Chinese	STSong-Light	GB-EUC-H	GB2312
	STSongStd-Light-Acro	GBpc-EUC-H	GB2312
		GBK-EUC-H	CP936
		GBKp-EUC-H	CP936
		GBK2K-EUC-H	GB18030
		UniGB-UCS2-H	UCS-2
Traditional Chinese	MHei-Medium	B5pc-H	Big5
	MSung-Light	HKscs-B5-H	Big5-HKSCS
	MSung-Std-Light-Acro	Big5	Big5
		ETen-B5-H	EUC-TW
		ETenms-B5-H	UCS-2
		CNS-EUC-H	
		UniCNS-UCS2-H	
Korean	HYGoThic-Medium	KSC-EUC-H	EUC-KR
	HYSMyeongJo-Medium	KSCms-UHC-H	JOHAB
	HYSMyeongJoStd-Medium-Acro	KSCms-UHC-HW-H	JOHAB
		KSCpc-EUC-H	EUC-KR
		UniKS-UCS2-H	UCS-2
Japanese	HeiseiKakuGo-W5	83pv-RKSJ-H	Shift-JIS
	HeiseiMin-W3	90ms-RKSJ-H	Shift-JIS

Local	CJK Typeface	Character Map	Encoding Supported
	KozMinPro-Regular-Acro	90msp-RKSJ-H	Shift-JIS
		90pv-RKSJ-H	Shift-JIS
		Add-RKSJ-H	Shift-JIS
		EUC-H	JEUC
		Ext-RKSJ-H	Shift-JIS
		H	ISO-2022-JP
		UniJIS-UCS2-H	UCS-2
		UniJIS-UCS2-HW-H	UCS-2

Note:

You must install the required font packs (available from www.Adobe.com) to view PDF documents which reference Chinese/Japanese/Korean fonts on non-localized platforms.

[PDF Settings] Section

[PDF Settings] lists the available PDF settings for Production Reporting when producing PDF output.

Table 85 Entries in [PDF Settings]

Entry	Value	Description
Bookmarks	True False Default = False	True creates PDF bookmarks for Table of Contents entries. False disables bookmarks.
Compatibility	4 - 6 Default = 5	Minimum Acrobat version required to read PDF documents.
CompressionText	0 - 9 Default = 6	Amount of compression applied to text in PDF documents. 0 = no compression, 9 = maximum compression The default value of 6 is the best value for compression versus speed.
CompressionGraphics	0 - 9 Default = 6	Amount of compression applied to graphics in PDF documents 0 = no compression, 9 = maximum compression The default value of 6 is the best value for compression versus speed.
EmbedAction	Stop Warn Skip Default = Stop	Action to take when errors occur trying to embed fonts: <ul style="list-style-type: none"> ● Stop—An error message is issued, and program execution stops. ● Warn—A warning message is issued, and the font is used without being embedded. ● Skip—No message is issued and the font is used without being embedded.
EncodingNotInFontAction	Stop Warn Skip	Action to take when fonts do not support an encoding.

Entry	Value	Description
	Default = Warn	<ul style="list-style-type: none"> ● Stop—An error message is issued and program execution stops. ● Warn—A warning message is issued and CP1252 is used. ● Skip—No message is issued and CP1252 is used.
Subsetting	True False Default = True	True enables subsetting for all fonts where subsetting is possible. False disables subsetting.
SubsetLimit	1-100 Default = 100	If a document uses more than the specified percentage of glyphs in a font, then subsetting is disabled for that particular font, and the complete font is embedded.
SubsetMinSize	Numbers > zero Default = 100	If the original font file is smaller than the specified size (in KB), then font subsetting is disabled for that particular font.
UnsupportedEncodingAction	Stop Warn Skip Default = Skip	Action to take when an unsupported ENCODING-REPORT-OUTPUT value is set. (The ENCODING-REPORT-OUTPUT value is the value set as the encoding for non-CJK fonts in the [Environment] section.) <ul style="list-style-type: none"> ● Stop—An error message is issued and program execution stops. ● Warn—A warning message is issued and CP1252 is used. ● Skip—No message is issued and CP1252 is used.

Note:

Production Reporting Release 9.x and above supports interlaced GIF images when creating PDF files.

[HTML Fonts] Section

[HTML Fonts] lists available fonts for Production Reporting when printing using `-PRINTER:EH`. Fonts are case sensitive.

Each entry is defined to be:

```
font_number=CSS_Style
```

The following describes each part of the entry:

```
font_number
```

Font number used within the Production Reporting program

```
CSS_Style
```

A modified CSS style added to the reports style sheet that describes the corresponding font number. The CSS style is used to help browsers with font matching when rendering HTML. The modifications to the normal CSS rules for font specification are to disallow use of the `font-size` or `font-size-adjust` properties and to disallow any values of 'inherit' for properties that would normally allow the 'inherit' keyword. Production Reporting appends the `font-size` automatically when adding the font rule to the CSS file (see the `ALTER-PRINTER` command).

A CSS style is useful for a font that is normally considered italic or oblique since Production Reporting does not support the use of italic or oblique fonts through language constructs and cannot always generate a correct style entry automatically.

For example, assume that you define font 901 to be 'Arial Italic'. This is a valid font name; however, the CSS specification will not allow 'Arial Italic' as a font family. Instead, you need need to add an entry to the [HTML Fonts] section as follows:

```
901=Font-family: Arial, sans-serif; Font-style: Italic;
```

Note:

You must put quotes around values for a Font-family that contain white space. For example, "Times New Roman".

Note:

A Font-family entry contains a list of fonts that the browser searches from left to right until it finds a match. It is a good idea to end each list with a "generic family" such as sans-serif or monospace to ensure a more reasonable approximation is made during rendering if the specified font is not available to the browser.

For more information on CSS rules for Fonts, the allowed syntax of such rules, and the list of "generic family" fonts (and their descriptions) see:

<http://www.w3.org/TR/REC-CSS2/fonts.html#font-specification?>

[HTML:Images] Section

[HTML:Images] defines parameters that Production Reporting uses when generating HTML report output files.

Table 86 Entries in [HTML:Images]

Entry	Value	Default Value	Description
FIRST-PAGE	HEIGHT, WIDTH, NAME	60,60,firstpg.gif	NAME of the graphic image file that accesses the first page of the report. Specify HEIGHT and WIDTH values in pixels.
PREV-PAGE	HEIGHT, WIDTH, NAME	60,60,prevpg.gif	NAME of the graphic image file that accesses the previous page of the report. Specify HEIGHT and WIDTH values in pixels.
NEXT-PAGE	HEIGHT, WIDTH, NAME	60,60,nextpg.gif	NAME of the graphic image file that accesses the next page of the report. Specify HEIGHT and WIDTH values in pixels.
LAST-PAGE	HEIGHT, WIDTH, NAME	60,60,lastpg.gif	NAME of the graphic image file that accesses the last page of the report. Specify HEIGHT and WIDTH values in pixels.
WALLPAPER	NAME		NAME of the graphic image file used as the reports's background image.

Note:

Production Reporting does not perform any validation of the graphic image filenames provided. The user is responsible for ensuring that the graphic image files are in a location that the browser can access.

[Enhanced-HTML] Section

[Enhanced-HTML] defines default actions that Production Reporting takes when generating HTML output with -EH.

Table 87 Entries in [Enhanced-HTML]

Entry	Value	Description
Browser	BASIC IE NETSCAPE ALL Default = BASIC	Specifies the browser and generates the appropriate HTML. <ul style="list-style-type: none"> ● BASIC—Production Reporting generates HTML suitable for all browsers. ● IE—Production Reporting generates HTML designed for Internet Explorer. ● NETSCAPE—Production Reporting generates HTML designed for Netscape. ● ALL—If necessary, Production Reporting generates Basic, IE, and Netscape HTML files. <i>Report_frm.htm</i> contains Javascript to “sense” the browser on the user’s machine and displays the appropriate version. (In this case, the user’s machine is the machine of the person reading the report, not the person writing it.)
DefaultTemplate	Path to default template.	Specifies the location of the default template for Production Reporting HTML output. Production Reporting provides a <i>Template.xml</i> file that you can customize and use as the default for all Production Reporting HTML output. See “Customizing the HTML Navigation Bar” in Volume 1 of the <i>Hyperion SQR Production Reporting Developer’s Guide</i> for more details.
IncludePDFInZip	TRUE FALSE Default = FALSE	Defines whether to include the PDF file in the ZIP file Production Reporting creates when -PRINTER:EP is combined with -EH_ZIP. <ul style="list-style-type: none"> ● TRUE—Includes the PDF file in the ZIP file. ● FALSE—Does <i>not</i> include the PDF file in the ZIP file.
Language	English French German Portuguese Spanish Default = English	Sets the language used for the HTML navigation bar.
FullHTML	3.2	Specifies the level of HTML that the browser supports so appropriate Enhanced HTML code is generated. There is no default value. If you do not specify a value, XHTML 1.1 is produced. If you specify 3.2, HTML version 3.2 is produced.

Entry	Value	Description
		Note: For information on deprecated values for the FullHTML keyword see “Deprecated SQR.INI Entries” on page 436.

[Color Map] Section

[Color Map] defines the default colors in Production Reporting reports. Enter the default colors in the format of:

```
[Color Map]
color_name=({rgb})
color_name=({rgb})
.
.
.
color_name=({rgb})
```

The default colors implicitly installed are:

```
black = (0,0,0)
white=(255,255,255)
gray=(128,128,128)
silver=(192,192,192)
red=(255,0,0)
green=(0,255,0)
blue=(0,0,255)
yellow=(255,255,0)
purple=(128,0,128)
olive=(128,128,0)
navy=(0,0,128)
aqua=(0,255,255)
lime=(0,128,0)
maroon=(128,0,0)
teal=(0,128,128)
fuchsia=(255,0,255)
```


[MAP-ODBC-DB] Section

[MAP-ODBC-DB] maps unknown ODBC driver names (using *\$sqr-connected-db-name*) to a valid *\$sqr-connected-db* setting.

For example:

```
[MAP-ODBC-DB]
MyDriver=SYBASE
```

Note:

See [Table 3](#) for additional information on the Production Reporting reserved variables referenced in [MAP-ODBC-DB].

[MAP-DDO-DB] Section

[MAP-DDO-DB] maps unknown DDO driver names (using *\$sqr-connected-db-name*) to a valid *\$sqr-connected-db* class.

For example:

```
[MAP-DDO-DB]
NewOracleDriver=Oracle
```

Table 88 Entry in [MAP-DDO-DB]

Entry	Valid Values	Description
<i>driver_name</i>	CSV DB2 ESSBASE INFORMIX MSOLAP ORACLE SAP SQLSERVER SYBASE XML	Database class to which the unknown driver name is mapped.

Note:

See *\$sqr-connected {sqr-connected-db}* and *\$sqr-connected-db-name {sqr-connected-db-name}* in [Table 3](#), “[Production Reporting Reserved Variables](#),” on [page 16](#) for additional information on the Production Reporting reserved variables referenced from the [MAP-DDO-DB] section.

[SQR Remote] Section

[SQR Remote] defines default settings for running Production Reporting programs remotely.

Table 89 Entries in [SQR Remote]

Entry	Description
HostName	Default host name.
UserName	Default username used to log into the host.

Entry	Description
TimeOut	Timeout duration in seconds. Default = 30
Passive	Whether the FTP transfer is in (#0) passive or (0) non-passive mode. Default = 0

7

Production Reporting Samples

Production Reporting provides a library of sample Production Reporting programs you can use to customize and experiment with. If you installed Production Reporting in the default directory, the sample programs are in:

C:\Hyperion\products\biplus\docs\samples\Production Reporting

Modify these programs any way you like to create customized Production Reporting reports.

[Table 90](#) lists the sample Production Reporting programs and provides a brief description of each. Each program consists of a report specification and a sample of the output.

Table 90 Production Reporting Sample Reports

Name	Description
_____.DAT	Data files used by the LOADALL.Production Reporting programs
_____.MEM	Startup files to run tiny, medium, and big Production Reporting programs
APPEND.SQR	Demonstrates the append and fixed-nolf commands
APTDIARY.SQR	Demonstrates columns, text wrapping
AREA100.SQR	Demonstrates a 100% area chart
BAR100.SQR	Demonstrates a 100% bar chart
BARCODE.SQR	Demonstrates printing a bar code
CALENDAR.SQR	Demonstrates nondatabase formatting
COMP_FOR.SQR	Prints a graph of the forecasted and actual sales for a given employee
COMP_F_G.SQR	Prints a graph of the forecasted and actual sales for month or quarter
COMP_PLN.SQR	Prints a graph of the planned and actual sales for a given employee
COMP_P_G.SQR	Prints a graph of the planned and actual sales for month or quarter
COVLET02.SQR	Uses Production Reporting to input data from user, enter data in the database, and write a form letter using a DOCUMENT paragraph
CRUPSAL.SQR	(Oracle) Creates stored functions and procedures for Oracle Version 7
CUST.SQR	Prints a list of all of the customers bursted by page
CUSTLBLS.SQR	Demonstrates printing mailing labels within columns

Name	Description
CUSTOMER.SQR	Demonstrates multiple detail lines, NEXT-LISTING command
CUSTOMR2.SQR	Demonstrates the use of the ON-BREAK argument to the PRINT command
CUSTOMR3.SQR	Demonstrates the use of the INPUT command to change report output
CUSTOMR4.SQR	Demonstrates the use of arrays
CUSTOMR5.SQR	Demonstrates dynamic queries to allow user to qualify a report as it runs
CUST_SUM.SQR	Prints and charts on a bar chart information about each customer in the customer table
CUSTTAPE.SQR	Demonstrates the flat file output for magnetic tape or other post-processing
DATAA.DAT	Needed for <code>append.sqr</code>
DATAB.DAT	Needed for <code>append.sqr</code>
DROPALL.SQR	Drops all the Production Reporting sample tables created by the LOADALL program
DROPPROC.SQR	(Sybase) Deletes leftover temporary stored procedures belonging to the user
DYNAMCOL.SQR	Demonstrates use of dynamic columns, dynamic tables and variables passed to ON-ERROR procedure
EMP.SQR	Prints a list of all of the employees bursted by page
EMP_COMM.SQR	Calculates each employee's commission based on sales
EMP_P_Q.SQR	List all employee quotas for a given month or quarter
ENVELOPE.SQR	Demonstrates use of printing envelope with proper bar code
EXPORT.SQR	Creates two Production Reporting reports: one to export a database table, the second to import that table. Data from the table is stored in an external operating system file in compressed format, with trailing blanks removed.
FLATFILE.SQR	Creates a Production Reporting report to extract a database table and place it into a flat file
FLOATBAR.SQR	Demonstrates a floating bar chart
FOR_CUST.SQR	Sales forecast for given customer grouped by month or quarter
FOR_EMP.SQR	Sales forecast for given employee grouped by month or quarter
FOR_PROD.SQR	Sales forecast for given product grouped by month or quarter
FOR_REG.SQR	Sales forecast for given region grouped by month or quarter
FOR_SUM.SQR	Creates a table of projected product sales with links to more information
FORMLETR.SQR	Demonstrates form letters using a DOCUMENT paragraph
HILO.SQR	Demonstrates a high-low-close chart
HISTGRAM.SQR	Demonstrates a histogram chart

Name	Description
INQUIRY.SQR	Creates an Production Reporting program to display rows at your terminal selected from a database table you specify. The resulting Production Reporting program prompts you to qualify rows to be selected, display those rows, then repeat.
INVOICE.SQR	Demonstrates multiple reports, printing invoices, and printing envelopes
LOADALL.SQR	Creates and loads sample tables used in the above Production Reporting programs Note: The sample tables are in ASCII format. As a result, you must specify a valid ASCII-derived encoding value in SQR.INI. For more information on encoding values, see “Encoding Keys in the [Environment] Section” on page 317.
MAKEDATA.SQR	Creates a data file with fixed length and NOLF attributes
MAKEREPT.SQR	Helps you create Production Reporting reports more quickly
MITI1.EPS	Needed for <code>sqrlogo.sqr</code>
MULTIPLE.SQR	Demonstrates creating multiple reports
NESTREPT.SQR	Demonstrates nesting of procedures
ORDERS.SQR	Lists all the orders and the orderlines associated with them
ORD_MONG.SQR	List all orders for a given month and group them by employee number
ORD_M_Q.SQR	List all orders for a given month or quarter
ORD_PROD.SQR	List all orders for a given product
ORD_REGG.SQR	Creates a report of all orders from a given region grouped by month or grouped by quarter
ORD_SUM.SQR	Displays an order's summary by month
ORD_S_Q.SQR	Prints a graph of the percent of orders for each region (in a year) and four graphs of the percent of orders for each region (one for each quarter of that year)
OVERBAR.SQR	Demonstrates an overlapped bar chart
PHONELST.SQR	Demonstrates printing within columns, page headings, and page footings
PLN_EMP.SQR	Sales plan for given employee grouped by month or quarter
PLN_GEN.SQR	Sales plan grouped by month or quarter
PLN_REG.SQR	Sales plan for given region grouped by month or quarter
PRODUCT.SQR	List of products and their prices and a graph of orders of products
SALELEAD.SQR	Demonstrates DOCUMENT paragraphs
SALES.SQR	Demonstrates charting from stored data and printing several charts on one page
SCATTER.SQR	Demonstrates a scatter chart
SHOWPROC.SQR	(Sybase) Shows any leftover temporary stored procedures belonging to the user

Name	Description
STCKAREA.SQR	Demonstrates a stacked area chart
SQR3DBAR.SQR	Demonstrates a 3D bar chart
SQRLASER.SQR	Demonstrates graphic and file I/O commands
SQRLINE.SQR	Demonstrates a line chart
SQRLOGO.SQR	Demonstrates printing images
SQRPIE.SQR	Demonstrates a pie chart
TABREP.SQR	Creates a tabular Production Reporting report for a table you choose
UPDATE.SQR	Generates an Production Reporting program that allows you to query and update database tables. The created program uses the SHOW command to simulate a menu interface.
UPDSAL.SQR	A sample report that demonstrates use of stored functions and procedures in Oracle



Production Reporting Messages

In This Chapter

Unnumbered Messages	351
Numbered Messages	353

Unnumbered Messages

Note:

Two digits (*nn*) appear as replacement markers in the messages. Descriptions of these replacement markers are listed with the message. The messages contain the proper value when they appear on the screen.

Table 91 Unnumbered Messages

Error Message	Suggestion/Interpretation
Out of memory.	Occurs when a call to the C routine 'malloc()' fails. <ul style="list-style-type: none">● PC - Use the <i>-Mfile</i> to reduce memory requirements. Remove unneeded TSRs● UNIX - Increase the size of the system swap file.● VAX - Increase the amount of memory allowed for that user.
No cursors defined.	From the <i>-S</i> command line flag. The Production Reporting program did not contain any commands <i>that</i> required a database cursor.
Not processed due to report errors.	From the <i>-S</i> command line flag. Production Reporting cannot provide information about the cursor due to errors in the program.
Enter `01`02	Type the value to assign to the specified variable. `01 = First character of the variable name `02 = Rest of the variable name
NOPROMPT used - Enter value below	(Windows) Appears when an INPUT command is defined with the NOPROMPT argument.
Enter `01	Type the value to assign to the specified substitution variable. `01 = Name of the substitution variable

Error Message	Suggestion/Interpretation
Enter this run's parameters:	Enter the values for the parameters defined in the program.
Error on line `01: `02	Production Reporting detected an error while processing the report file. Correct the error and rerun. `01 = Source line number `02 = Source line
Error in include file "`01" on line `02: `03	Production Reporting detected an error while processing the report file. Correct the error and rerun. `01 = Name of the include file `02 = Source line number `03 = Source line
Warning on line `01: `02	Production Reporting detected a non-fatal error while processing the report file. `01 = Source line number `02 = Source line
Warning in include file "`01" on line `02: `03	Production Reporting detected a nonfatal error while processing the report file. `01 = Name of the include file `02 = Source line number `03 = Source line
Type RETURN for more, C to continue w/o display, X to exit run:	Informational message used with the -D command line flag.
Loading Oracle DLL Failed!!!	(Oracle) Title for the dialog box that informs the user that Production Reporting could not load the Oracle DLL.
Errors were found in the program file.	Correct the errors and rerun.
Errors were found during the program run.	Correct the errors and rerun.
`01: End of Run.	Informational message. `01 = Image name (for example, SQR)
Enter report name:	Enter the name of the report (.SQR or .SQT) to run.
Enter database name:	Enter the name of the database.
Enter Username:	Enter the user name to log onto the database.
Enter Password:	Enter the password. For security reasons, the password is not echoed.
Customer ID:	Text message
Press Enter to close...	Text message
Enter Subsystem Name:	Enter the subsystem name.
`01: Program Aborting	Informational message. `01 = Image name (for example, SQR)

Error Message	Suggestion/Interpretation
*** Internal Coding Error ***	Informational message.
SQL DataServer Message	(Windows) Title for the error message dialog box.
Operating-System error	(Windows) Title for the error message dialog box.
DB-Library error	(Windows) Title for the error message dialog box.
`01 is running. Click the Cancel button to interrupt it.	(Windows) This is the body of the -C cancel dialog box. The user can click the Cancel button to abort the program run.
Table of Contents	Text for HTML driver
Previous	Text for HTML driver
Next	Text for HTML driver
First Page	Text for HTML driver
Last Page	Text for HTML driver
PAGE	Text for HTML driver

Numbered Messages

Note:

Two digits (*nn*) appear as replacement markers in the messages. Descriptions of these replacement markers are listed with the message. The messages contain the proper value when they appear on the screen.

Table 92 Numbered Messages 000001 to 000999

Error Number	Error Message	Suggestion/Interpretation
000001	Error while opening the message file: ``01' (`02): `03	Try reloading <code>sqerrr.dat</code> from the release media. If the error persists, contact technical support. `01 = Name of the error message file `02 = System error code `03 = System error message
000002	Error while reading the message file. (`01): `02	Try reloading <code>sqerrr.dat</code> from the release media. If the error persists, contact technical support. `01 = Name of the error message file `02 = System error code `03 = System error message
000004	Error while seeking the message file. (`01): `02	Try reloading <code>sqerrr.dat</code> from the release media. If the error persists, contact technical support. `01 = Name of the error message file `02 = System error code `03 = System error message

Error Number	Error Message	Suggestion/Interpretation
000005	Corrupt message file: Invalid header information.	Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support.
000006	Corrupt message file: Invalid count (Got `01, Should be `02).	The header contains an invalid entry count. Ensure SQDIR points to the correct directory. Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. `01 = The value read from the header `02 = What the value should be
000010	Invalid SEMCode encountered: `01.	An invalid code was passed to the error message handler. Try reloading the files from the release media. If the error persists, contact technical support. `01 = Invalid code
000011	Unknown conversion type (`01) for code `02.	Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. `01 = Invalid type `02 = Internal error code
000012	Message `01 must be either Preload or BuiltIn.	The type error code is not correct. Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. `01 = Error code
000013	Cannot point to message `01.	The error handler cannot position to the desired error code. Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. `01 = Error code
000014	The required environment variable `01 has not been defined.	Define the named environment variable and restart Production Reporting. `01 = Environment variable name
000015	The Meta ESC characters do not match (Got '`01', Should be '`02').	The meta escape character defined in the header does not match what the error message handler expects. Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. `01 = What was found in the header `02 = What was expected to be found
000016	`01() called to process (`02) and the message file is not open.	The specified error routine was called but the error message file was not open. Try reloading the files from the release media. If the error persists, contact technical support. `01 = Name of the routine `02 = Error code
000017	Message `01 must be ReportParameters or CopyrightNotice.	Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. `01 = Error code
000028	Cannot access the initialization file: `01 (`02): `03	The initialization file specified by the -ZIF command line flag cannot be accessed. `01 = Name of the file `02 = System error code `03 = System error message
000029	Unknown encoding name: `01	The encoding name specified by the -ZEN command line flag is not valid `01 = Encoding name
000202	DPUT: Bad field number.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000203	DARRAY: Unknown command number.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000204	`01: Cannot find `02 command.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine `02 = Name of the command
000205	DDO: DO arguments do not match procedure's.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000206	SDO: Bad params for DO command.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000207	SDO: Bad params for BEGIN-PROCEDURE command.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000209	SGOTO: Bad goto function parameters.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000210	SGOTO: Could not find beginning of section or paragraph.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000211	SGOTO: Bad label: from parameters.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000212	COMPAR: Unknown relational (numeric) operator.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000213	COMPAR: Unknown relational (string) operator.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000214	DONBRK: Unknown case for putlin.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000216	GARRAY: Unknown command number.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000217	GCMDS: No Gfunc found.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000218	GDOC: Unknown document type.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000219	GLET: Bad operator.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000220	GLET: Stack incorrect for expression - arg `01.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Number of the argument
000221	GLET: Unknown operator type.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000222	GLET: Unknown operator in expression.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000223	GPARS: Column not SCOL, TCOL, or NCOL type.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000224	GPARS: Bad parameter format: `01 =`02=	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string `02 = Bad format field found
000225	GPARS: No end of required word in parfmt: `01	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string
000226	GPARS: Bad parfmt entry: `01	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string
000227	GPARS: Bad parameter string.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000228	GPARS: Repeat count bad: `01	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string
000229	GPARS: Only a,b,8,9 allowed for repeats: `01	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string
000230	GPARS: Missing required x: `01	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string
000231	GPARS: Bad type in 'ckvrpr()'. '	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000232	GPROC: No Gfunc found.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000233	GRDWRT: Unknown command number.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000234	GSHOW: Unknown SHOW option.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000235	PGMPARS: 'addvar()' passed maxlen but not column.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000238	PGMPARS: ``01' passed invalid parameter number: `02.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Routine name `02 = Invalid parameter number
000239	PGMPARS: 'fxclrf()' encountered bad column reference type: `01.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal variable type code
000240	PLCMNT: 'getplc()' passed invalid element number: `01.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Invalid element number
000241	RDPGM: Command array size exceeded (change COMDMAX to at least `01).	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Maximum internal command number supported.
000242	RDPGM: Bad match adding internal variable: `01	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal variable name
000243	RDPGM: No cmdget function found for BEGIN_S.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000244	Function `01 not included in run-time package.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the SQR routine
000245	SETSQL: Could not find variable ``01', in Run Time.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable name
000249	SPINIT: Bad parameters.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000251	DBFFIX: DBDATLEN returned out of range status.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000252	DPRPST: Error converting Sybase type for EXECUTE.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000254	SETSQL: Could not find variable entry in list.	(Oracle) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000255	DBDESC: SQLD not = number of select columns.	(DB2 and Informix) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000256	DBFETCH: Unknown variable dbtype encountered: `01 (`02)	(DB2 and Informix) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable name `02 = Unknown database type
000257	WRITE_SPF: Unknown code encountered: `01	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Unknown SPF code
000258	`01: Cannot find LOAD-LOOKUP table: `02	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine `02 = Name of the table
000259	PGMPARS: ``01' called with wrong variable ``02'	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine `02 = Name of the variable
000261	MODIFYVAR: Attempt to change variable which is not xVAR (`01).	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = name of the variable
000262	MODIFYVAR: Incompatible variable types (`01) and (`02).	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable type (from) `02 = Variable type (to)

Table 93 Numbered Messages 001000 to 001999

Error Number	Error Message	Suggestion/Interpretation
001201	Cannot open the argument file: ``01'. (`02): `03	Depends on the system error message. `01 = Name of the file `02 = System error code `03 = System error message
001202	Cannot close the argument file. (`01): `02	Depends on the system error message. `01 = System error code `02 = System error message
001203	Cannot open the -MFile: ``01'. (`02): `03	Depends on the system error message. `01 = Name of the file `02 = System error code `03 = System error message

Error Number	Error Message	Suggestion/Interpretation
001204	Minimum value for '&`01' in the -MFile is `02.	Correct the -Mfile entry. `01 = Keyword in question `02 = Minimum value allowed
001205	Maximum value for '&`01' in the -Mfile is `02.	Correct the -Mfile entry. `01 = Keyword in question `02 = Maximum value allowed
001206	Invalid -MFile entry: '&`01'.	Correct the -Mfile entry. `01 = The line from the -Mfile
001207	Cannot close the -MFile. (`01): `02	Depends on the system error message. `01 = System error code `02 = System error message
001209	The minimum value for '&`01' (`02) is `03.	Value out of range. `01 = Entry name `02 = Specified value `03 = Minimum value
001210	The maximum value for '&`01' (`02) is `03.	Value out of range. `01 = Entry name `02 = Specified value `03 = Maximum value
001211	The value for '&`01' (`02) is not an integer number.	Value must be a integer value. `01 = Entry name `02 = Specified value
001300	Bind list does not match query (do not use '@__p' string).	Production Reporting reserves the variable names that start with "@__p" for internal use. Edit the source code and use different variable names.
001301	Forward references not permitted in select list bind variables.	Within the body of BEGIN-SQL paragraphs, forward references to &column names are not permitted. Move the BEGIN-SQL paragraph after the &column definition.
001303	Error in SQL (perhaps missing &name after expression):	The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement.
001304	Check SELECT columns, expressions and 'where' clause for syntax.	The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement.
001305	CMPSQL: Unknown data type in database: `01.	Contact technical support with the version of the database you are connected to. `01 = Datatype in question
001307	CMPSQL: DBDEFN failed.	(ODBC, Oracle, Informix) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
001308	`01: Could not bind column `02.	(ODBC, Oracle, Informix) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the SQR routine `02 = Name of the column
001309	The type for '&`01' (`02) does not match the type from the database (`03).	Correct the source code. `01 = Name of the column/ expression pseudonym `02 = User specified type `03 = Database type
001400	Only numerics allowed for arithmetic.	Only #numeric variables, &columns, and literals are permitted in the arithmetic commands. Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
001401	Optional qualifier is ROUND=n (0-`01).	Correct the syntax. `01 = Maximum value for ROUND=
001402	Optional qualifiers for DIVIDE are ON-ERROR={HIGH ZERO} and ROUND=n.	Correct the syntax.
001403	Attempting division by zero.	Use the ON-ERROR = HIGH ZERO option to prevent this error from halting the program.
001404	Bad number of digits to ROUND or TRUNC (0-15).	Correct the syntax.
001405	WARNING: The ROUND or TRUNC qualifier is greater than the number's precision.	Correct the syntax.
001500	Array element out of range (`01) for array ``02' on line `03.	Correct the source logic. `01 = Element number passed `02 = Name of the array `03 = Program line number
001501	Field element out of range (`01) for array ``02', field ``03', on line `04.	Correct the source logic. `01 = Element number passed `02 = Name of the array `03 = Name of the field `04 = Program line number
001502	WARNING: Attempting division by zero on line `01. Array field ``02' unchanged. Run continuing...	The ARRAY-DIVIDE command has attempted division by zero. The division has been ignored; the result field is unchanged. Add logic to account for this possibility. `01 = Program line number `02 = Name of field
001601	'FILL' not appropriate for numeric data.	The FILL argument to the PRINT command may be used only for text fields. Move the #numeric variable to a \$string variable, and then print the string variable.
001700	Report ``01': Columns must be between 1 and the page width (`02).	The specified value is wider than the width of the page. Correct the source line. `01 = Name of the current report `02 = Page width
001702	Report ``01': GOTO-TOP=`02 must be between 0 and the page depth (`03).	The value specified on the GOTO-TOP argument of the NEXT-COLUMN command was either less than 1 or greater than the page depth. Correct the source line. `01 = Name of the current report `02 = Goto-Top value `03 = Page width
001703	Report ``01': ERASE-PAGE=`02 must be between 0 and the page depth (`03).	The line number specified on the ERASE-PAGE argument of the NEXT-COLUMN command is greater than the page depth. Correct the source line. `01 = Name of the current report `02 = Erase-Page value `03 = Page width
001704	Report ``01': The NEXT-COLUMN command is not legal in the `02 section with the qualifier AT-END=NEWPAGE.	Correct the source line. `01 = Name of the current report `02 = Name of the section
001705	Report ``01': Column number `02 is not defined.	The column number specified with the USE-COLUMN command is greater than the highest column defined in the COLUMNS command. Correct the source line. `01 = Name of the current report `02 = Column number
001800	Format for CONNECT: username/password [ON-ERROR=procedure[(arg1[,argi]...)]]	Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
001801	Cannot use CONNECT while SQL statements are active.	Correct the program logic to ensure that all BEGIN-SELECT paragraphs have completed before executing the CONNECT command.
001802	Logoff failed prior to CONNECT.	The database server returned an error while trying to log off from the database. SQR aborts the program run since it cannot continue.
001803	CONNECT failed. Perhaps username/password incorrect.	The specified connectivity information is incorrect or there might have been a network failure. Use the ON-ERROR flag to trap any errors during the program run; otherwise SQR aborts the program run.
001804	Sybase extensions SET and SETUSER not permitted in SQR.	Remove SET and SETUSER from the source.
001805	USE allowed once in SETUP section only, not in BEGIN-SQL. Elsewhere, specify db.[user].table...	Correct the source.
001807	The requested database connection (`01) is already active.	The -Cnn value specified is being used by another BEGIN-SELECT paragraph that is currently selecting data. Use another connection number. `01 = Connection number
001808	Cannot find inactive database cursor. Program too large.	Too many BEGIN-SELECT and BEGIN-SQL paragraphs are active at the same time. Reduce the complexity of the program.
001809	Database commit failed.	(DB2, ODBC, Oracle) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.
001810	Database rollback failed.	(DB2, ODBC, Oracle) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.
001811	Cannot open database cursor.	(ODBC, Oracle) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.
001901	Variable for date-time must begin with '&'.	Correct the syntax.
001913	Format code must be SYYY when specifying signed year.	Correct the edit mask.
001914	Bad input data (`01) for edit mask: ``02'.	Correct the input. `01 = Data being converted `02 = Edit mask
001915	Year cannot be zero.	Correct the date.
001916	Year must be between -4713 and 9999 inclusive.	Correct the date.
001917	Ambiguous date-time.	Correct the date.
001918	``01' is not a valid date part.	Correct the date part. `01 = Date part.
001919	Invalid day of week.	Correct the date.
001920	Format code cannot appear in date input format: ``01'.	Correct the edit mask. `01 = Improper format characters.

Error Number	Error Message	Suggestion/Interpretation
001921	Bad date mask starting at: ``01'.	Correct the edit mask. `01 = Improper format characters.
001922	Seconds past midnight must be between 0 and 86399.	Correct the date.
001923	Seconds must be between 0 and 59.	Correct the date.
001924	Minutes must be between 0 and 59.	Correct the date.
001925	Month must be between 1 and 12.	Correct the date.
001926	Day must be between 1 and `01.	Correct the date.
001927	Hour must be between 1 and 12.	Correct the date.
001928	Hour must be between 0 to 23.	Correct the date.
001929	HH24 precludes the use of meridian indicator.	Correct the edit mask.
001930	HH12 requires meridian indicator.	Correct the edit mask.
001931	Day of year must be between 1 and 365 (366 for leap year).	Correct the date.
001932	Date string too long.	Correct the date.
001933	The month (`01) is not valid for the current locale or database.	Correct the date. `01 = Name of the month.
001934	The format mask must be a literal when the date-time is not loaded into a variable.	Correct the format mask. The format mask must be a literal when the date-time is not loaded into a variable.
001935	Date-time format too long.	Correct the format mask.
001936	Bad date-time format.	Correct the format mask.
001937	Bad SQL for default date-time. (Table DUAL required for syntax.)	(Oracle) Possibly the format mask needs to be corrected; otherwise, there is a problem with the database server.
001937	Bad SQL for default date-time. (Table DUAL required for syntax.)	(DB2) Possibly the format mask needs to be corrected; otherwise, there is a problem with the database server.
001938	Cannot recompile sql.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.
001939	Problem executing cursor.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.
001940	Error fetching row.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.
001941	Cannot redefine variable addresses.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.

Error Number	Error Message	Suggestion/Interpretation
001942	The date ``01' is not in the format: SYYYYMMDD[HH24[MI[SS[NNNNNN]]]].	When specifying an SQR date, at a minimum, the date must be specified; the time is optional. `01 = The invalid date.
001943	The date ``01' is not in one of the accepted formats listed below: MM/DD/YYYY [BC AD] [HH:MI[:SS[.NNNNNN]] [AM PM]] MM-DD-YYYY [BC AD] [HH:MI[:SS[.NNNNNN]] [AM PM]] MM.DD.YYYY [BC AD] [HH:MI[:SS[.NNNNNN]] [AM PM]] SYYYYMMDD[HH24[MI[SS[NNNNNN]]]	The date specified with the INPUT command was not in one of the default formats. Please re-enter the date in a valid format. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: DD-MON-YY SYYYYMMDD[HH24[MI[SS[NNNNNN]]]	(Oracle) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: Mon DD YYYY [HH:MI[:SS[.NNN]]][AM PM]] Mon DD YYYY [HH:MI[:SS[:.NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[.NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[:.NNN]]][AM PM]] SYYYYMMDD[HH24[MI[SS[NNNNNN]]]	(Sybase) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: Mon DD YYYY [HH:MI[:SS[.NNN]]][AM PM]] Mon DD YYYY [HH:MI[:SS[:.NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[.NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[:.NNN]]][AM PM]] SYYYYMMDD[HH24[MI[SS[NNNNNN]]]	(ODBC) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: Mon DD YYYY [HH:MI[:SS[.NNN]]][AM PM]] Mon DD YYYY [HH:MI[:SS[:.NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[.NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[:.NNN]]][AM PM]] SYYYYMMDD[HH24[MI[SS[NNNNNN]]]	(DDO) The date was not in the one of the expected formats for this database. `01 = The invalid date.

Error Number	Error Message	Suggestion/Interpretation
001944	The date '`01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: YYYY-MM-DD HH:MI:SS.NNN SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]	(Informix) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date '`01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: YYY-MM-DD[-HH.MI.SS[.NNNNNN]] MM/DD/YYYY DD.MM.YYYY SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]	(DB2) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date '`01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: YYYY-MM-DD SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]	(Teradata) The date was not in the one of the expected formats for this database. `01 = The invalid date
001946	The date variables are incompatible with each other.	The SQR function references two date variables which cannot be logically be used together (for example, DateDiff of 'date-only' and 'time-only' dates).

Table 94 Numbered Messages 002000 to 002999

Error Number	Error Message	Suggestion/Interpretation
002000	Procedure name used more than once: '`01'.	Give the procedure a unique name. `01 = Procedure name
002001	Could not find procedure: '`01'.	Check for a misspelled procedure name. `01 = Procedure name
002002	DO arguments do not match procedure's.	The argument lists for the DO and BEGIN-PROCEDURE commands must match in both type and count. Correct the source line.
002003	DO argument must be \$string or #number to accept returned value.	Correct the syntax.
002100	Edit string too long.	The edit mask must be less than 255 characters. Reduce the length of the edit mask.
002101	Bad numeric 'edit' format: `01	The numeric edit mask contains an invalid character. See the PRINT command for the valid numeric edit mask characters. `01 = Invalid character
002103	DOLLAR-SYMBOL must be a single alphanumeric character or its decimal value enclosed in brackets: <nnn>.	Correct the syntax.
002104	DOLLAR-SYMBOL cannot be any of the following characters: `01	Correct the syntax. `01 = List of invalid characters

Error Number	Error Message	Suggestion/Interpretation
002106	MONEY-SYMBOL must be a single alphanumeric character or its decimal value enclosed in brackets: <nnn>.	Correct the syntax.
002107	MONEY-SYMBOL cannot be any of the following characters: `01	Correct the syntax.
002200	ENCODE string too large; maximum is `01.	Break up the ENCODE command. `01 = Maximum length of an ENCODE string supported by this version of SQR
002300	EXIT-SELECT failed.	The database command to cancel the query returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
002301	EXIT-SELECT valid only within SELECT paragraph.	Remove the EXIT-SELECT command.
002400	Duplicate label's - do not know which one to GOTO.	Labels must be unique within the section or paragraph where they are defined. Give each label a unique name.
002401	(Labels must be in same section or paragraph as GOTO.) Cannot find a matching label for GOTO command.	Check the source code.
002500	Error getting INPUT.	The C routine "fgets()" returned an error and SQR aborts the program run.
002501	Unknown INPUT datatype: type={char number integer date}	Correct the syntax.
002502	INPUT STATUS= must reference #variable.	Correct the syntax.
002503	Unknown qualifier for INPUT.	Correct the syntax.
002506	Too long. Maximum `01 characters.	The response to the INPUT statement was too long. Re-enter the data. `01 = Maximum characters allowed
002507	Incorrect. Format for floating point number: [+ -]99.99 [E99]	Invalid number was entered for an INPUT request. Re-enter the data.
002508	Incorrect. Format for integer: [+ -]999999	Invalid integer was entered for an INPUT request. Re-enter the data.
002510	A format mask can only be specified when TYPE=DATE is used.	Correct the syntax.
002511	The format mask cannot be stored in a date variable.	Correct the syntax.
002512	The input variable type does not match the TYPE qualifier.	Correct the syntax.
002513	Number too large for INTEGER. Valid range is -2147483648 to 2147483647.	The number was too large to be stored as an integer. Values are from -2147483648 to 2147483647. Re-enter the data.
002514	Enter a date in one of the following formats: MM/DD/YYYY [HH:MI[:SS[.NNNNNN]] [AM PM]] MM-DD-YYYY [HH:MI[:SS[.NNNNNN]] [AM PM]]	The date cannot be blank. Enter a date in one of the specified formats.

Error Number	Error Message	Suggestion/Interpretation
	MM.DD.YYYY [HH:MI[:SS[.NNNNNN]] [AM PM]] SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]	
002515	`01 required user interaction but user interaction was disabled by the -XI command line flag.	The specified command required user interaction, but user interaction was disabled by the -XI command line flag. `01 = Name of the command
002600	LOAD-LOOKUP table '`01' has not been defined.	Add a LOAD-LOOKUP command. `01 = Load lookup table name
002601	Missing value for `01= in LOAD-LOOKUP.	Correct the syntax. `01 = Name of missing required parameter
002602	Bad value for `01= in LOAD-LOOKUP.	Correct the syntax. `01 = Name of the parameter
002603	LOAD-LOOKUP `01= cannot reference a variable in the Setup section.	Either move the LOAD-LOOKUP command from the Setup section or remove the variable reference. `01 = Name of the parameter
002604	LOAD-LOOKUP names must be unique.	Give each LOAD-LOOKUP array a unique name.
002605	Cannot compile SQL for LOAD-LOOKUP table '`01'.	The database server returned an error while trying to compile the SQL statement needed to process the LOAD-LOOKUP command. Check the column and table names. Also check the WHERE= clause for errors. `01 = Load lookup table name
002606	Could not set up cursor for LOAD-LOOKUP table '`01'.	The database server returned an error while trying to compile the SQL statement needed to set up the LOAD-LOOKUP command. Check the column and table names. Also check the WHERE= clause for errors. `01 = Load lookup table name
002607	Problem executing the cursor for LOAD-LOOKUP table '`01'.	The database server returned an error while trying to execute the SQL statement needed to process the LOAD-LOOKUP command. `01 = Load lookup table name
002609	Integers only allowed in numeric lookup keys.	Correct the source line.
002610	Numeric lookup keys must be <= `01 digits.	Correct the source line. `01 = maximum length supported
002611	Bad return fetching row from database in LOAD-LOOKUP table '`01'.	The database server returned an error while fetching the data. `01 = Load lookup table name
002613	Loading '`01' lookup table ...	This message can be inhibited by using the QUIET argument on the LOAD-LOOKUP command. `01 = Name of the load lookup table `02 = Number of rows loaded
002615	Warning: `01 duplicate keys found in '`02' lookup table.	This message can be inhibited by using the QUIET argument on the LOAD-LOOKUP command. `01 = Number of duplicate keys `02 = Name of the load lookup table
002616	LOAD-LOOKUP `01= must reference a numeric variable or literal.	Correct the source line. `01 = Name of the parameter
002617	LOAD-LOOKUP `01= must reference a string variable or literal.	Correct the source line. `01 = Name of the parameter

Error Number	Error Message	Suggestion/Interpretation
002618	LOAD-LOOKUP `01= variable ``02' has not been defined.	Correct the source line. `01 = Name of the parameter `02 = Name of the undefined variable
002619	LOAD-LOOKUP cannot support `01 rows; maximum is `02.	Reduce the ROWS= value. `01 = ROWS= value `02 = Maximum value allowed
002620	`01 command not allowed with -XL option in effect.	Either use the #IF command to conditionally compile the program when -XL is being used or do not execute this SQR report with the -XL option. `01 = SQR command
002700	Line to stop erasing for 'NEW-PAGE' is larger than the page depth.	Correct the source line.
002800	'ON-BREAK' not appropriate for numeric data.	The ON-BREAK argument to the PRINT command may be used only for text fields. Move the #numeric variable to a \$string variable, and then print the \$string variable.
002801	SET= and LEVEL= must be >= zero when indicated.	Correct the source line.
002802	Cannot use old style PROCEDURE= with BEFORE= or AFTER=.	Correct the syntax.
002804	SET= must be same for all ON-BREAKs in Select.	All the ON-BREAKS in a query must belong to the same SET. Use SET= to differentiate between ON-BREAKs in different queries. Correct the source line.
002805	ON-BREAK with BEFORE or AFTER must be inside Select.	Correct the source line.
002806	SAVE= must be a \$string variable.	Correct the syntax.
002900	Record :types are FIXED, VARY, or FIXED_NOLF (default is VARY).	Correct the syntax.
002901	STATUS variable for `01 must be #Numeric.	Correct the syntax. `01 = SQR command affected
002902	OPEN missing required qualifiers: RECORD={rec_len} FOR-READING FOR-WRITING FOR-APPEND	Correct the syntax.
002903	Too many external files opened; maximum is `01.	Reduce the number of open external files needed by the program. `01 = Maximum number of open external files supported by this version of SQR
002904	File number already opened.	Check your program logic.
002905	Cannot open file ``01' AS `02.(`03): `04	SQR aborts. `01 = Filename `02 = File number `03 = System error code `04 = System error message
002906	Cannot close file `01 (`02): `03	SQR aborts. `01 = File number `02 = System error code `03 = System error message
002907	Problem closing user file(s) at the end of run.	This message may indicate system problems.
002908	Warning: Cannot CLOSE file `01 -- file not opened.	While not an error, this message indicates a problem with your SQR code. `01 = File number.
002909	Missing or invalid character set encoding: ``01'.	Specify a valid encoding: UCS-2, ASCII, EBCDIK1027, etc.

Error Number	Error Message	Suggestion/Interpretation
002910	This has been declared a UCS-2 file; however, the Byte-Order-Mark is missing or invalid.	A UCS-2 file must contain a Byte-Order-Mark.
002911	Cannot use `01 without setting UseUnicodeInternal to TRUE in SQR.INI.	The UseUnicodeInternal setting in the DEFAULT-SETTINGS section of the SQR.INI must be set to TRUE in order to use this functionality.
002912	The encoding `01 cannot be supported without the use of unicode. Set the UseUnicodeInternal setting in SQR.INI to TRUE.	The UseUnicodeInternal setting in the DEFAULT-SETTINGS section of SQR.INI must be set to TRUE in order to use this encoding.
002913	Invalid Unicode character representation used in the UNICODE Let function.	The string of hex characters representing a set of Unicode characters must be in the format '[whitespace U+ u]XXXX' for the UNICODE Let function.
002914	The encoding `01 is not compatible with `02 unless unicode is used internally. Set the UseUnicodeInternal setting in the INI file to TRUE and rerun.	Certain encodings are not compatible unless unicode is used. The UseUnicodeInternal setting in the DEFAULT-SETTINGS section of SQR.INI must be set to TRUE in order to use these encodings in the same run.
002915	The file encoding `01 is not compatible with `02 unless unicode is used internally. Set the UseUnicodeInternal setting in the INI file to TRUE and rerun.	Certain encodings are not compatible unless unicode is used. The UseUnicodeInternal setting in the DEFAULT-SETTINGS section of SQR.INI must be set to TRUE in order to use these encodings in the same run.
002916	The SQT encoding `01 is not compatible with `02. Set the UseUnicodeInternal setting in the INI file to TRUE and rerun.	Certain encodings are not compatible unless unicode is used. The SQT was not generated using unicode; therefore, the encodings specified are incompatible. Reset the encoding in conflict or regenerate the SQT with the UseUnicodeInternal setting in the DEFAULT-SETTINGS section of SQR.INI to TRUE in order to use these encodings in the same run.
002917	The SQT file was created using an SQR executable that does not support unicode. Please run the SQT using the appropriate executable.	(EBCDIC) The SQT file was created using SQR that does not support unicode. Therefore, it must be run with a non-unicode SQR or SQRT executable.
002918	An operation was detected that requires unicode; however, this executable cannot perform unicode processing. Relink or use the appropriate executable.	Unicode processing is required; however, the executable has not been linked with the Rosette Unicode library. Either relink the executable with the Rosette library or, for EBCDIC platforms, use the alternate executable that supports unicode processing.
002919	The field length `01 is invalid for I/O with UCS-2 data.	By definition, UCS-2 requires two bytes per character; however, the length specifier was odd. Make the length a multiple of two.
002920	:length must be a numeric literal, variable, or column.	Correct the syntax.
002921	Bad :length (`01) for OPEN command.	The length must be greater than zero. `01 = Length value

Table 95 Numbered Messages 003000 to 003999

Error Number	Error Message	Suggestion/Interpretation
003000	PAGE-NUMBER strings too long.	The pre-and post-PAGE-NUMBER strings must be less than 74 characters. Correct the source line.
003100	Cannot find document marker referenced in POSITION command.	Defines the specified @ marker in a BEGIN-DOCUMENT paragraph. Check for a misspelled @ marker name.
003101	Only 'COLUMNS nn...' allowed after document marker in POSITION command.	Correct the syntax.
003200	Specified file number not opened for reading.	Files must be opened for reading in order to use the READ command with them. Correct the program logic.
003201	Line `01: Error reading the file. (`02): `03	`01 = Program line number `02 = System error code `03 = System error message
003202	Specified file number not opened for writing.	Files must be opened for writing in order to use the WRITE command with them. Correct the program logic.
003203	Line `01: Error writing the file. (`02): `03	`01 = Program line number `02 = System error code `03 = System error message
003204	Length of variables exceeds record length.	The total of the lengths indicated in the command must be less than the RECORD= argument used on the OPEN command. Check for a typographical error or recalculate the RECORD= value.
003205	Numeric binary transfer allowed with FIXED or FIXED_NOLF records only.	By default, all files are opened in VARY (variable length) mode, thus prohibiting the transfer of numeric binary data. Add the:FIXED or FIXED_NOLF option to the RECORD= argument on the appropriate OPEN command.
003206	Command not complete.	Correct the syntax.
003207	File number must be a numeric literal, variable, or column.	Correct the syntax.
003208	Missing required :length in READ command.	Correct the syntax.
003209	Bad :length (`01) for READ or WRITE command.	The length must not be less than zero. `01 = Length value
003210	\$String or #numeric variables required for READ.	Correct the syntax.
003211	#Numeric variables and literals must have :length of 1, 2 or 4 bytes.	Correct the syntax.
003212	#Numeric variables and literals on CDC may only have :length of 1 or 3 bytes.	Correct the syntax.
003214	:length must be a numeric literal, variable, or column.	Correct the syntax.
003300	Unknown qualifier for STOP.	Correct the syntax.
003301	Program stopped by user request.	Informational message.

Error Number	Error Message	Suggestion/Interpretation
003400	Wrap not appropriate for numeric data.	The WRAP argument to the PRINT command may be used only for text fields. Move the #numeric variable to a \$string variable first, and then print the \$string variable.
003401	Max `01 chars/line for reverse WRAP.	Reduce the number of characters specified. `01 = Maximum number of characters supported by this version of SQR.
003402	Max `01 chars/line for WRAP with ON= or STRIP=	Reduce the number of characters specified. `01 = Maximum number of characters supported by this version of SQR
003403	Bad <number> in WRAP qualifier.	The number inside the angled brackets must be a valid ASCII number (1 - 255). Correct the source line.
003404	Missing '>' in WRAP qualifier.	A leading "<" in the ON= or STRIP= qualifier indicates that a numeric value is following, which must be ended by a closing ">". Correct the source line.
003405	The value for '`01' (`02) must be `03 0.	The value specified for the specified qualifier is invalid. Correct the program logic. `01 = Qualifier name `02 = Value encountered `03 = Relation to zero (<,<=,=,>=,>)
003500	PUT, GET or ARRAY-xxx command incomplete. Required word missing.	Correct the syntax.
003501	Did not find end of literal.	The ending quote character (') was not found at the end of the literal. Add the ending quote character.
003502	Literal too long.	Literal strings can be up to 256 characters long. Break up the literal into smaller pieces and combine using the LET command.
003503	Unknown variable type.	Variable names must begin with \$, #, or &. Correct the source line.
003504	Cannot find 'array_name (#element)'.	The element number was not specified. Correct the source line.
003505	'(#Element)' variable not found for array.	Each GET or PUT command must indicate the element or row number to access in the array. Correct the source line.
003506	Array specified not defined with CREATE-ARRAY.	Use the CREATE-ARRAY command to define each array before referencing that array in other commands. Check for a misspelled array name.
003507	Bad element reference for array (#variable 123).	The element number is larger than the number of rows defined in the CREATE-ARRAY command. Check program logic to make sure that the element number was not inadvertently changed.
003508	Did not find ending ')' for field.	The "occurs" number for an array field is missing a right parenthesis. Correct the source line.
003509	Field not defined in array: `01	Check for a misspelled field name against the CREATE-ARRAY command. `01 = Undefined field name

Error Number	Error Message	Suggestion/Interpretation
003510	More variables than fields specified in array command.	The ARRAY command must not have more variables listed to the left of the array name than there are matching fields defined for the array. Check against the CREATE-ARRAY command.
003511	More variables in command than fields in array.	The ARRAY command must not have more variables listed to the left of the array name than there are matching fields defined for the array. Check against the CREATE-ARRAY command.
003512	Only numeric variables and fields allowed with array arithmetic commands.	The ARRAY-ADD, ARRAY-SUBTRACT, ARRAY-MULTIPLY, and ARRAY-DIVIDE commands may have only numeric variables or literals as the source fields. Move the string data into a #numeric variable and then reference the #numeric variable.
003513	GET can only be used with \$string or #numeric variables.	You can move array fields only into \$string variables or #numeric variables. Correct the source line.
003514	PUT and GET variables must match array field types.	When moving data into or out of arrays, the source or destination variables must match the array fields in type. CHAR fields can be stored into/from strings, NUMBER fields into/from numeric variables. Check the CREATE-ARRAY command.
003515	More fields than variables found in array command.	The ARRAY command must not have more variables listed to the left of the array name than there are matching fields defined for the array. Check against the CREATE-ARRAY command.
003516	Too many arrays defined; maximum is `01.	Reduce the number of arrays needed by the program. `01 = Maximum number of arrays supported by this version of SQR
003517	Missing '=specifier' in qualifier: `01	Correct the syntax. 01 = Name of missing required parameter
003518	Duplicate array name: `01	Change the name of the array. `01 = Array name in question
003519	Too many fields defined; maximum is `01.	Reduce the number of fields. `01 = Maximum number of fields allowed per array
003520	Missing ':type' in CREATE-ARRAY FIELD= `01	Correct the syntax. `01 = The name of the field
003521	Duplicate FIELD name: `01	Change the name of one of the fields. `01 = The name of the field
003522	Optional :nn for FIELD must be between 1 and 64K.	Correct the source line.
003523	CREATE-ARRAY FIELDS :type must be one of the following: `01	Correct the syntax.
003525	Missing NAME= in CREATE-ARRAY.	Correct the syntax.
003526	Missing or incorrect SIZE= in CREATE-ARRAY.	Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
003527	Missing FIELD= statements in CREATE-ARRAY.	Correct the syntax.
003529	Missing or invalid initialization value for field `01.	Correct the syntax. 01 = Name of the field
003530	Invalid EXTENT= value in CREATE-ARRAY.	Correct the syntax.
003600	Missing 'ask' variable name.	Correct the syntax.
003603	WARNING: Substitution variables do not vary when saved with run-time.	Informational message.
003605	No substitution variable entered.	The C routine "fgets()" returned an error and SQR aborts the program run.
003700	Did not find end of paragraph: `01	Missing the END-paragraph command to match the specified paragraph. Correct the source file. `01 = BEGIN-paragraph in question
003701	Invalid command.	Check for a misspelled command.
003702	Command not allowed in this section: `01	Correct the syntax. `01 = Offending command name
003703	Paragraph not allowed inside procedure.	The BEGIN-paragraph command is not allowed here. Check your SQR code for a misplaced paragraph.
003704	Missing procedure name.	Correct the syntax.
003705	Extra argument found.	Correct the syntax.
003706	Missing Comma.	Correct the syntax.
003707	Bad Argument List.	The DO or BEGIN-PROCEDURE command has an error in its argument list, possibly extra characters after the final right parentheses. Correct the source line.
003708	Empty Argument.	The DO or BEGIN-PROCEDURE command has an error in its argument list, possibly two commas in a row inside the parentheses. Correct the source line.
003709	Only \$string and #number variables allowed for BEGIN-PROCEDURE parameters.	Correct the syntax.
003710	Unknown argument type.	An argument in a DO or BEGIN-PROCEDURE command is incorrect. Check for a misspelled variable type.
003711	Indicate :\$string or :#number returned values in BEGIN-PROCEDURE only.	Correct the syntax.
003712	Missing).	Correct the syntax.
003713	`01 paragraph not allowed with -XL option in effect.	Either use the #IF command to conditionally compile the program when -XL is being used or do not execute this SQR report with the -XL option. `01 = Name of the BEGIN-paragraph
003714	Bad database connection number.	The -Cnn value must be a non-zero value. Correct the source line.

Error Number	Error Message	Suggestion/Interpretation
003715	Did not find end of paragraph: `01 (No 'from...' clause found.)	Correct the source code. `01 = BEGIN-command in question
003716	Error in SQL statement.	The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement.
003717	Extra characters after expression continuation.	Remove the extra characters after the dash.
003718	Did not find end of expression.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003719	Columns names and expressions must be unique or be given unique pseudonyms (&name).	Columns retrieved from the database are assigned names by prepending an "&" to the beginning of the name. You are trying to select the same &column name more than once. Change the assigned &column name by using an alias after the name.
003720	Bad number specified for 'LOOPS=' on 'BEGIN-SELECT; Maximum is 32767'.	If your program logic requires that you stop processing after more than 32767 rows have been retrieved, you could count the rows manually and use the EXIT-SELECT command to break out of the SELECT loop.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [-Bnn] [LOOPS=nn] [ON-ERROR=procedure{(arg1[,argi]...)]	(DB2) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [LOOPS=nn] [ON-ERROR=procedure{(arg1[,argi]...)]	(Informix) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [LOOPS=nn] [ON-ERROR=procedure{(arg1[,argi]...)] [-DB=database]	(ODBC) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [LOOPS=nn] [ON-ERROR=procedure{(arg1 [,argi]...)] [BEFORE=procedure{(arg1[,argi]...)] [AFTER=procedure{(arg1[,argi]...)]	(DDO) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [-Bnn] [LOOPS=nn] [ON-ERROR=procedure{(arg1[,argi]...)]	(Oracle) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [-XP] [LOOPS=nnn] [ON-ERROR=procedure{(arg1[,argi]...)]	(Sybase) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: -SELECT [DISTINCT] [-Cnn] [-Bnn] [LOOPS=nn] [ON-ERROR=procedure{(arg1[,argi]...)]	(Teradata) Correct the syntax.
003722	Could not set up cursor.	An error occurred while trying to compile the SQL statement. Look closely at any \$string variable references. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.

Error Number	Error Message	Suggestion/Interpretation
003723	Problem executing cursor.	An error occurred while trying to execute the SQL statement. Look closely at any \$string variable references. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003724	Could not exit query loop.	The database command to cancel the query returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
003725	Bad return fetching row from database.	The database returned an error status for the last row that was fetched, commonly due to the buffer not being large enough. If selecting expressions, make sure that the length of the first expression will be adequate for all rows selected.
003726	Literal in SQL expression missing closing quote.	Literals must be surrounded by single quotes ('). To embed a quote within a literal use two single quotes in sequence ("). Correct the source line.
003727	SQL expression not ended, perhaps parentheses not balanced.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003728	SQL expression not ended, perhaps missing &name.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003729	SQL expression is missing &name or has unbalanced parentheses.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(DB2) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Informix) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [-NR] [ON-ERROR=procedure(arg1[,argi]...)] [-DB=]	(ODBC) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Oracle) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)] [-Connection=]	(DDO) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [-XP] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Sybase) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Teradata) Correct the syntax.
003731	Did not find 'END-SQL' after 'BEGIN-SQL'.	Correct the source file.

Error Number	Error Message	Suggestion/Interpretation
003732	ON-ERROR= for 'BEGIN-SQL' in SETUP section must be STOP, WARN or SKIP.	Correct the syntax.
003733	Could not create procedure for SQL.	(Sybase) SQR could not create a stored procedure for the SQL statement. The most likely cause for failure is that the user name you are using to run the report under does not have the proper privileges. Either grant the user CREATE PROCEDURE privilege or use the -XP command line option to inhibit SQR from creating temporary stored procedures for SQL statements.
003734	Could not compile SQL.	Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003735	Could not execute SQL.	An error occurred while trying to compile the SQL statement. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003736	Please use BEGIN-SELECT - END-SELECT section for SELECT statements.	(Informix, ODBC, Oracle, DDO) Correct the source code.
003737	Bad fetch buffer count.	(Oracle, Sybase) The -B flag specifies an illegal value. Correct the source code.
003738	Report interrupted by request.	Informational message.
003741	Dynamic column must be \$string variable.	Correct the syntax.
003742	Dynamic column missing '^01'.	Correct the syntax. ^01 = Missing character
003743	Dynamic columns must have a &pseudonym.	Correct the syntax.
003745	Only a variable name may be between the '^01' and '^02' characters.	Correct the syntax. ^01 = Leading character ^02 = Trailing character
003746	When dynamic columns are used all non-dynamic columns and expressions must be defined with &name=type.	Add &name=type to all expressions and non-dynamic columns.
003747	When the table name is dynamic each column and expression must be defined with &name=type.	Add &name=type to all expressions and non-dynamic columns.
003748	When selecting multiple rowsets, they must have the same columns (order, type, width).	Non-contiguous rowsets selected in ROWSETS=().
003749	The highest numbered rowset named in ROWSET=() must be less than 10.	Rowset upper bound exceeded in ROWSETS=().
003750	Correct syntax is ROWSETS=(-n,m,n-m,i-) or (all).	Correct the syntax in ROWSETS=().
003751	Cannot select more than 25 Rowsets in a single BEGIN-SELECT.	Correct the syntax in ROWSETS=().
003752	One of the selected fields ('^01') was not found in the specified row set(s). The available fields are: '^02'.	Field not found in BEGIN-SELECT.
003755	Rowsets must be listed in ascending order, and may not include duplicates.	Bad rowset numbering sequence in BEGIN-SELECT.

Error Number	Error Message	Suggestion/Interpretation
003800	Too many document paragraphs; maximum is `01.	There are too many BEGIN-DOCUMENT paragraphs. Reduce the number of DOCUMENT paragraphs needed by the program. `01 = Maximum number supported by this version of SQR
003801	Too many document markers; maximum is `01.	There are too many BEGIN-DOCUMENT paragraphs. Reduce the number of DOCUMENT paragraphs needed by the program. `01 = Maximum number supported by this version of SQR
003802	Duplicate document marker.	Give the document marker a unique name.
003803	Did not find 'END-DOCUMENT' after 'BEGIN-DOCUMENT'.	The BEGIN-DOCUMENT paragraph must end with END-DOCUMENT. Correct the source code.
003900	EXECUTE command is incomplete.	Correct the syntax.
003901	Bad -Cnn connection number for EXECUTE.	The -Cnn value must be a nonzero value. Correct the source line.
003902	@#Return_status must be #numeric (missing #).	(Sybase) Correct the source line.
003902	@#Return_status must be #numeric (missing #).	(ODBC) Correct the source line.
003902	@(# or \$)Return_status must be #numeric or \$variable.	(Oracle) Correct the source line.
003902	@#Return_status must be #numeric (missing #).	(DB2) Correct the source line.
003903	Missing '=' after `01.	Correct the source line. `01 = The parameter in question
003904	Unknown variable type.	Variable names must begin with \$, #, or &. Correct the source line.
003905	OUT[PUT] variables for EXECUTE may only be \$variable or #variable.	Correct the syntax.
003906	The only EXECUTE option is WITH RECOMPILE.	Correct the syntax.
003907	You must EXECUTE ... INTO &columns.	Correct the syntax.
003908	Unknown datatype for EXECUTE...INTO &columns.	Check for a misspelled data type. If the data type is correct, then contact customer technical support so SQR can be updated.
003909	EXECUTE...INTO &columns must be unique.	The &column name assigned to the column must be unique throughout the report. Give the column a unique name.
003910	Missing (length) for datatype in EXECUTE.	Correct the source line.
003911	Datatype should not have (length) in EXECUTE.	Correct the source line.
003912	DO= in EXECUTE requires INTO... variables.	Correct the syntax.
003913	Could not EXECUTE stored procedure.	(Sybase) Record the database error message displayed with this message. If needed, contact your system administrator.

Error Number	Error Message	Suggestion/Interpretation
003913	Could not EXECUTE stored procedure.	(ODBC) Record the database error message displayed with this message. If needed, contact your system administrator.
003913	Could not EXECUTE stored procedure or function.	(Oracle) Record the database error message displayed with this message. If needed, contact your system administrator.
003913	Could not EXECUTE stored procedure or function.	(DB2) Record the database error message displayed with this message. If needed, contact your system administrator.
003914	Bad return fetching row from database.	Record the database error message displayed with this message. If needed, contact your system administrator.
003915	Could not set up EXECUTE cursor.	The database server returned an error while trying to compile the SQL statement needed to set up the EXECUTE command.
003918	Missing Stored Procedure or Function - ``01'.	(Oracle) A describe of the store procedure or function could not be performed. `01 = Procedure or function name
003918	Missing Stored Procedure - ``01'.	(DB2) A describe of the store procedure or function could not be performed. `01 = Procedure or function name
003920	The EXECUTE command had an erroneous return status data type of ``02' declared. The Stored Function expects a return status of ``01'.	(Oracle) The wrong data type has been specified for the return status of a Stored Function. `01 = Expected Data Type `02 = Return Data Type
003921	The EXECUTE command detected an erroneous data type of ``02' declared for the IN/OUT parameter - name/position: `03/`04. The Stored Function or Procedure expects a parameter data type of ``01'.	(Oracle) The wrong data type has been specified for the return variable of a Stored Procedure. `01 = Expected Data Type `02 = Return Data Type `03 = Name of Parameter `04 = Position of Parameter
003921	The EXECUTE command detected an erroneous data type of ``02' declared for the IN/OUT parameter - name/position: `03/`04. The Stored Procedure expects a parameter data type of ``01'.	(DB2) The wrong data type has been specified for the return variable of a Stored Procedure. `01 = Expected Data Type `02 = Return Data Type `03 = Name of Parameter `04 = Position of Parameter
003922	The maximum number of allowable IN/OUT parameters was reached. No more than ``01' can be processed by the EXECUTE command.	(DB2) The maximum number of allowable IN/OUT parameters was reached. `01 = Maximum Number of Parameters

Error Number	Error Message	Suggestion/Interpretation
003923	The procedure was found in multiple schemas and will not be processed. The ``01.``02' procedure was also found in the ``03' schema.	(DB2) The request procedure was found in multiple schemas and can not be processed. `01 = Schema Name `02 = Stored Procedure Name `03 = Alternate Schema Name
003924	No match was found for the named parameter - ``01'.	(Oracle, DB2) No match was found for one of the input or output parameters with those returned from the describe of the stored procedure or function. Check the names of the stored procedure's or function's input or output parameters and make the necessary corrections. `01 = Parameter Name
003925	The EXECUTE command detected a data type of ``02' declared for the INTO parameter - name/position: ``03 / ``04'. The Stored Function or Procedure expects a parameter data type of ``01'.	(Oracle) The wrong data type has been specified for the INTO variable of a Stored Procedure. `01 = Expected Data Type `02 = Return Data Type `03 = Parameter Name `04 = Parameter Position
003926	The EXECUTE command detected that ``02' INTO parameters were requested. No more than ``01' can be processed by this Stored Function or Procedure.	(Oracle) The maximum number of allowable INTO parameters was exceeded. `01 = Number of Allowable Parameters `02 = Number of Parameters Entered
003927	The EXECUTE command detected a weak reference cursor.	(Oracle) A weak reference cursor has been detected. No validation on the data types for the INTO parameters is made.
003928	The wrong number of IN/INOUT parameters were found.	(DB2) The wrong number of IN/INOUT parameters were found. This Stored Function or Procedure requires that ``01' parameters be entered. `01 = Required Number of Parameters

Table 96 Numbered Messages 004000 to 004999

Error Number	Error Message	Suggestion/Interpretation
004000	Result #variabe or \$variable or '=' missing in expression.	The LET command is not properly formatted. Correct the source line.
004001	Expression too complex.	The expression is either too long or is too deeply nested. Break the expression into smaller expressions.
004002	Parentheses unbalanced in expression.	A left or right parenthesis is missing. Correct the source line.
004003	Too many variables; maximum is `01.	Break the expression into smaller expressions. `01 = Maximum number supported by this version of SQR

Error Number	Error Message	Suggestion/Interpretation
004004	Empty expression.	The expression is invalid. Correct the source line.
004005	Extra comma in expression.	An argument is missing after a comma in the expression. Correct the source line.
004006	Unknown operator '`01'. Do you mean `02 ?	The concatenation operator is . Correct the source line.
004007	Too many &column forward references in expression; maximum is `01.	The expression contains too many forward references. Break the expression into smaller expressions. `01 = Maximum number supported by this version of SQR
004008	Unknown function or variable in expression: `01	The specified function is not an SQR built-in function nor does it exist in the user-modifiable file UFUNC.C. Check for a misspelled function name. `01 = Function name
004009	Function '`01' missing parentheses.	All functions in an expression must be followed by their arguments enclosed in parentheses. Correct the source line.
004010	Empty parentheses or expression.	A pair of parentheses were found with nothing inside them. Remove the () in question from the source line.
004011	User function '`01' has incorrect number of arguments.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004012	Function '`01' has incorrect number of arguments.	Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name
004013	Missing operator in expression.	Correct the source line.
004014	Operator '`01' missing argument.	Correct the syntax of the function. Functions are described under the LET command. `01 = Operator
004015	Function '`01' missing argument.	Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name
004016	Function or operator '`01' missing arguments.	Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name
004017	User function '`01' requires character argument.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004018	User function '`01' requires numeric argument.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004019	User function '`01' requires \$string variable.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004020	User function '`01' requires #numeric variable.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name

Error Number	Error Message	Suggestion/Interpretation
004021	User function '`01' has incorrect argument type list. Must be of: c,n,C,N	The UFUNC.C file has a bad definition for the specified function. Correct the UFUNC.C program file; recompile UFUNC.C; and recreate the SQR executable. `01 = User function name
004022	User function '`01' missing arguments.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004023	User function '`01' has incorrect return type. Must be c or n.	The UFUNC.C file has a bad definition for the specified function. Correct the UFUNC.C program file; recompile UFUNC.C; and recreate the SQR executable. `01 = User function name
004024	'isnull' requires a &column, \$string, or \$date argument.	#numeric variables cannot be NULL. Correct the source line.
004025	'nvl' requires a &column, \$string, or \$date as its first argument.	#numeric variables cannot be NULL. Correct the source line.
004026	Function or operator '`01' requires character argument.	Correct the source line. `01 = Function or operator
004027	Function or operator '`01' requires numeric argument.	Correct the source line. `01 = Function or operator
004028	IF or WHILE expression must return logical result.	The expression used must evaluate a statement that will be TRUE or FALSE. Correct the source line.
004029	Attempting division by zero in expression.	The expression tried to divide a number by zero. Use the COND() function to check if the divisor is zero; then divide by something else (for example, 1).
004030	Attempting division by zero with '%'	An attempt was made to divide a number using the "%" operator. Use the COND() function to check if the divisor is zero; then divide by something else (for example, 1).
004031	The number used with '%' (`01) is out of range.	The "%" operator works with integers only. Correct the program logic. `01 = Maximum value allowed
004032	User function has unknown return type -- expecting n or c -- need to recompile Run-Time file?	SQR detected an error while processing a user defined function. If you are running an .sqt file, it probably needs to be recompiled because the user function has changed its definition. If you are running an .sqr file, then you need to correct the UFUNC.C program file; recompile UFUNC.C, and recreate the SQR executable.
004033	In user function use C type with allocated string to change \$variable.	SQR detected an error while processing a user defined function. Correct the UFUNC.C program file recompile UFUNC.C and recreate the SQR executable.
004034	Could not find array '`01' in ARRAY function.	Check for a misspelled array name. `01 = Array name
004035	Could not find array field '`01' in ARRAY function.	Check for a misspelled array field name. `01 = Array field name
004036	Math error in expression (usually over- or under-flow).	Most of the SQR mathematical built-in functions have a corresponding C library routine. One returned an error.

Error Number	Error Message	Suggestion/Interpretation
		Break the expression into discrete expressions in order to identify the function that caused the error.
004037	Error executing expression.	Record the steps leading up to the error and contact technical support.
004038	Out of space while processing expression; Use -Mfile to increase EXPRESSIONSPACE.	The expression requires more temporary string storage than is currently allocated. Use the -Mfile flag on the command line to specify a file that contains an entry that increases by a greater value than is currently defined.
004039	'`01' assumed to be a variable name, not an expression.	Warning message. `01 = Expression in question
004040	The array '`01' has not been defined.	Define the array using the CREATE-ARRAY command. `01 = Array name
004041	The field '`01' is not valid for array '`02'.	Correct the source code. `01 = Field name `02 = Array name
004042	The array reference '`01' has an incorrect number of parameters specified.	Correct the source code. `01 = Array name
004043	The array reference '`01' requires numeric parameters for the element and <i>occurs</i> arguments.	Correct the source code. `01 = Array name.
004045	Function or operator '`01' requires date argument.	Correct the source code. `01 = Array name
004046	Incompatible types between expression and variable.	Correct the source code.
004047	The field '`01' is must be 'char' or 'float'.	Correct the source code. `01 = Field name
004048	Function or operator '`01' must be a string or date argument.	Correct the source line. `01 = Function or operator
004049	Unknown transform value '`01' in TRANSFORMATION function.	Check for a misspelled transform value. `01 = Transform value
004100	Use 'print' command to format data outside SELECT query.	You must precede PRINT command arguments (WRAP, ON-BREAK.) with an explicit PRINT command when outside of a BEGIN-SELECT paragraph. Correct the source line.
004101	Cannot find required parameter.	Correct the syntax.
004102	Bad number found.	A command expecting a numeric literal or :#numeric variable reference found an illegal number definition or a reference to a string variable or column. Correct the source line.
004103	Cannot find required numeric parameter.	Correct the syntax.
004104	Cannot find placement parameters.	The position qualifier "(Row,Col,Len)" was not found. Check for a missing parentheses.
004105	Placement parameter incorrect.	The "Row", "Column" or "Length" fields are invalid or ill-formed. Correct the source line.

Error Number	Error Message	Suggestion/Interpretation
004106	Invalid second function on line.	An SQR command used as a qualifier for a primary command (for example, PRINT) is incorrect. Correct the source line.
004107	Second function must be FORMAT type.	The PRINT command may have format command qualifiers such as WRAP, CENTER, or FILL. Other qualifier commands are not permitted.
004108	Missing operator =, <, >, ...	Correct the source line.
004109	Invalid operator.	Correct the source line.
004110	Missing variable.	Correct the syntax.
004111	Please give this expression a &pseudonym.	Expressions selected in BEGIN-SELECT should be given an &Name or be followed by a print position "(Row,Col,Len)". Correct the source line.
004112	Wrong variable type.	Correct the syntax.
004113	Command incomplete, expected '`01'.	Correct the syntax. `01 = What was expected
004114	Expecting '`01', found '`02'.	Correct the syntax. `01 = What was expected `02 = What was encountered
004115	Unknown command or extra parameters found (missing quotes?).	Correct the syntax.
004116	Duplicate references to parameter '`01'.	Correct the syntax. `01 = Duplicated parameter
004117	Unexpected equal sign found with '`01'.	Correct the syntax. `01 = Parameter name
004118	Qualifier '`01' cannot be used with the following qualifiers:	Correct the syntax. `01 = Qualifier name
004119	Expecting numeric column, found string column.	Correct the syntax.
004120	Date variables (`01) cannot be used with this command.	Correct the syntax. `01 = Parameter name
004200	Page width and depth must be > 0 and < 32767.	The values specified with PAGE-SIZE are out of bounds. Specify legal values.
004300	Missing end of placement (...) in SHOW.	The placement parameter is ill-formed. Correct the source line.
004301	Bad (...) location in SHOW.	Screen positions must be valid numbers. Correct the source line.
004302	Missing literal or variable name to EDIT in SHOW.	The literal or variable name must immediately precede the EDIT, NUMBER, MONEY, or DATE keywords.
004303	Missing edit mask in SHOW.	The word EDIT must be followed by a valid edit mask. Correct the source line.
004304	Only string variable allowed for dynamic edit mask.	Dynamic edit masks may only be stored in \$Variables. Correct the line.

Error Number	Error Message	Suggestion/Interpretation
004305	Unknown option for SHOW.	Correct the syntax.
004406	Number `01 not allowed.	Use a different value. `01 = Internal number
004407	Referenced variables not defined:	References were made to column variables (&var) that were not defined in the program. The list of variable names follows this message.
004501	Use '+' and negate variable for reverse relative placement.	The use of "-#Variable" is not legal here. Negate the #Variable value and use "+#Variable".
004503	Fixed line placement #variable must be > 0. Use relative positioning, (+#line,10,0).	Correct the source line as indicated.
004504	Fixed column placement #variable must be > 0. Use relative positioning, (5,+#col,0).	Correct the source line as indicated.
004505	Length placement #variable must be >= 0.	The length field cannot be a negative value. Correct the source line.
004600	CODE not appropriate for numeric data.	The CODE qualifie rin PRINT may only be used for text fields. Move "#Variable" to "\$Variable" first and then print "\$Variable".
004601	Unknown option for GRAPHIC command: BOX, HORZ-LINE, VERT-LINE or FONT	Correct the syntax.
004602	GRAPHIC BOX out of bounds. Row: `01, Column: `02, Width: `03, Depth: `04	SQR aborts the program run. `01 = Row `02 = Column `03 = Width `04 = Depth
004603	GRAPHIC VERT-LINE out of bounds. Row: `01, Column: `02, Length: `03	SQR aborts the program run. `01 = Row `02 = Column `03 = Length
004604	GRAPHIC HORZ-LINE out of bounds. Row: `01, Column: `02, Length: `03	SQR aborts the program run. `01 = Row `02 = Column `03 = Length
004605	Cannot draw the box; values are out of bounds. Row: `01, Column: `02, Width: `03, Height: `04	SQR aborts the program run `01 = Row `02 = Column `03 = Width `04 = Height
004606	Cannot draw the vertical line; values are out of bounds. Row: `01, Column: `02, Height: `03	SQR aborts the program run `01 = Row `02 = Column `03 = Height
004607	Cannot draw the horizontal line; values are out of bounds. Row: `01, Column: `02, Width: `03	SQR aborts the program run `01 = Row `02 = Column `03 = Width
004700	Cannot open the program file: ``01' (`02): `03	Depends on the system error message. `01 = Name of the program file `02 = System error code `03 = System error message
004701	Cannot logon to the database.	Connectivity information is either incorrect or the database server is unavailable. Check connectivity information and the server availability.
004702	Line found outside paragraph.	All commands must be within BEGIN-... END statements. Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
004703	Cannot close the program file. (`01): `02	Depends on the system error message. `01 = System error code `02 = System error message
004704	#ENDIF not found for #IF.	Missing an #ENDIF to complete conditional compilation. Correct the source code.
004705	Program line too long; maximum is `01.	Break the program line into smaller lines. `01 = Maximum line length supported by this version of SQR
004706	Substitution variable {`01} would cause this line to exceed the maximum line length of `02 characters.	The substitution variable value would cause this line to exceed the maximum line size. Break the program line into smaller lines. `01 = Name of the substitution variable `02 = Maximum line length supported by this version of SQR
004707	No value found for substitution variable: {`01}	An empty value was found for the substitution variable. Check for a misspelled name. `01 = Name of the substitution variable
004708	#ELSE without preceding #IF.	Missing an #IF or #IFDEF or #IFDEF to begin conditional compilation. Correct the source code.
004709	#ENDIF without preceding #IF.	Missing an #IF or #IFDEF or #IFDEF to begin conditional compilation. Correct the source code.
004710	#IF's nested too deeply; maximum is `01.	Reduce the number of nested #IF directives. `01 = The maximum depth supported by this version of SQR
004711	#INCLUDE files nested too deeply; maximum is `01.	Reduce the number of nested #INCLUDE directives. `01 = The maximum depth supported by this version of SQR
004712	Include file name too long; Modify -I flag.	The combined -I directory name with the #INCLUDE file name exceeds the maximum length permitted for a complete pathname. Check the spelling of both the -I command flag and the #INCLUDE filename.
004713	Cannot open the #INCLUDE file: '`01' (`02): `03	`01 = Include file name `02 = System error code `03 = System error message
004714	Cannot close the #INCLUDE file: '`01' (`02): `03	`01 = Include file name `02 = System error code `03 = System error message
004716	'BEGIN-PROGRAM' command not found in program.	This section is required for all reports. Correct the source code.
004717	Cannot open the report output file: '`01' (`02): `03	`01 = Output file name `02 = System error code `03 = System error message
004719	Cannot logoff the database.	The database server returned an error while trying to log off from the database. SQR aborts the program run.
004720	Cannot open the run-time file: '`01'. (`02): `03	SQR aborts the program run. `01 = Run-Time file name `02 = System error code `03 = System error message
004721	Cannot close the run-time file. (`01): `02	SQR aborts the program run. `01 = System error code `02 = System error message

Error Number	Error Message	Suggestion/Interpretation
004722	Error reading the run-time file. (`01): `02	SQR aborts the program run. `01 = System error code `02 = System error message
004723	Run time file must be recreated for this version of SQR.	The run-time file was created by a earlier version of SQR and is incompatible with the current version. Recreate the .sqt (run-time) file.
004724	The -XL option cannot be specified with this run-time file because access to the database is required.	Do not use the -XL option.
004725	Cannot open cursor.	The database server returned an error indicating that a new database cursor or logon could not be completed. See the error message from the database server.
004726	Cannot create procedure for SQL statement.	(Sybase) SQR could not create a stored procedure for the SQL statement. The most likely cause for failure is that the user name you are running the report under does not have the proper privileges. Either grant the user CREATE PROCEDURE privilege or use the -XP command line option to inhibit SQR from creating temporary stored procedures for SQL statements.
004727	Error writing the run-time file. (`01): `02	`01 = System error code `02 = System error message
004729	Cannot find inactive database cursor. Program too large.	(DB2, Oracle) The program has too many concurrent database cursors. Reduce the complexity of the program.
004730	Run-time saved in file: `01	Informational message. `01 = Name of the .sqt file created
004735	Unknown variable type encountered in run-time file: `01	SQR aborts loading the run-time file. `01 = Variable type
004736	Unexpected End-Of-File while processing the run-time file.	SQR aborts loading the run-time file.
004737	Cannot load the run-time file because it was built for the `01database and `02 is built for the `03 database.	SQR aborts loading the run-time file. `01 = Database name from run-time file `02 = SQR image name `03 = Database that SQR is built for
004738	'END-REPORT' not paired with 'BEGIN-REPORT'.	Correct the source code.
004739	'END-PROGRAM' not paired with 'BEGIN-PROGRAM'.	Correct the source code.
004743	#INCLUDE filename must be enclosed in quotation marks.	Correct the syntax.
004744	#INCLUDE command format is: #Include 'filename'.	Correct the syntax.
004747	The SQT file is corrupted and cannot be processed.	SQR aborts loading the run-time file.
004748	The user function '`01' needs to be defined as entry `02 in the user function table. It requires a definition of: Return Type = '`03' Arg Count = `04 Arg Types = "`05"	The SQT file requires that the specified user function be defined. `01 = User function name `02 = Entry in the user function table `03 = Return type `04 = Argument count `05 = Argument types
004749	An attempt was made to move `01 characters into '`02'. The maximum allowed is `03 characters.	An attempt was made to move too much data into an SQR string variable. `01 = Number of characters to be moved `02 = Variable name `03 = Maximum characters allowed

Error Number	Error Message	Suggestion/Interpretation
004750	SQR has reached the architectural limit for `01' (`02).	While attempting to increase an internal table, SQR reached its architectural limit for that table. Processing will stop as SQR cannot continue. `01 = Internal table classification `02 = Architectural limit
004802	PRINTER TYPE must be HTML, HPLASERJET, POSTSCRIPT, or LINEPRINTER.	Correct the syntax.
004805	Both BEFORE-BOLD and AFTER-BOLD must be specified.	Correct the syntax.
004807	Unknown DECLARE qualifier.	Correct the syntax.
004901	Date variables (`01) cannot be used in BEGIN-SQL or BEGIN-SELECT paragraphs.	Correct the source code. `01 = Variable name

Table 97 Numbered Messages 005000 to 005999

Error Number	Error Message	Suggestion/Interpretation
005000	Report `01' heading section size exceeds the page depth.	Reduce the size of the heading or increase the page depth.
005001	Report `01' footing location must be less than the page depth.	Reduce the size of the footing or increase the page depth.
005002	Check 'BEGIN-HEADING' commands: Discovered 2nd page-initialization while heading in progress.	The BEGIN-HEADING procedure either caused an overflow of the current page or it issued a command that caused a page eject to occur. Check any procedure invoked by the BEGIN-HEADING section to ensure that the commands do not overflow the page or cause a page eject.
005003	Check 'BEGIN-FOOTING' commands; perhaps number of footing lines is too small. Discovered 2nd page-write while footing in progress.	The BEGIN-FOOTING procedure either caused an overflow of the current page or it issued a command that caused a page eject to occur. Check any procedure invoked by the BEGIN-FOOTING section to ensure that the commands do not overflow the page or cause a page eject.
005004	Attempt to execute the `01 command while processing the `02 section.	Change the SQR program logic to prevent the command from executing while the specified section is active. `01 = Command name `02 = Section name
005005	Report `01' already has been assigned a `02 section.	Correct the source code. `01 = Report name `02 = Duplicated section name
005006	You cannot define more than one default `01' section.	Correct the source code. `01 = Duplicated section name
005007	Report `01' has overlapping heading and footing sections.	Correct the source code. `01 = Report name
005008	TOC `01' already has been assigned a `02 section.	Correct the source code. `01 = Table of Contents name `02 = Duplicated section name
005009	The name can only contain characters [0-9 A-Z _ -].	Correct the source code.
005010	The name cannot be the reserved names 'none' or 'default'.	Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
005011	This name has already be used.	Correct the source code.
005012	The specified `01 (`02) does not exist.	Correct the source code. `01 = Heading or Footing `02 = Heading/Footing name
005013	FOR-REPORTS and FOR-TOC cannot be specified when NAME= is used.	Correct the source code.
005014	TOC (`01) has already been defined as the default.	Correct the source code. `01 = Default TOC name
005100	'IF', 'WHILE', 'EVALUATE' commands nested too deeply; maximum is `01.	Reduce the nested commands. `01 = Maximum depth allowed by this version of SQR
005101	'BREAK' found outside 'WHILE' or 'EVALUATE' statement.	The BREAK command is valid only in the context of a WHILE or EVALUATE statement. Correct the source code.
005103	END-WHILE found without matching 'WHILE'.	Correct the source code.
005104	'IF' or 'EVALUATE' command not completed before 'END-WHILE'.	Correct the syntax.
005105	'ELSE' found without matching 'IF'.	ELSE can be used only within the context of an IF command. Correct the source code.
005106	Single 'ELSE' found inside 'WHILE' or 'EVALUATE' statement.	ELSE can be used only within the context of an IF command. Correct the source code.
005107	Only one 'ELSE' allowed per 'IF'.	Rewrite the source code to use nested IF statements.
005108	Found 'END-IF' without matching 'IF'.	Each IF command must have a matching END-IF command. Correct the source code.
005109	'WHILE' or 'EVALUATE' command not completed before 'END-IF'.	You are missing a closing END-WHILE or END-EVALUATE command before END-IF. IF, WHILE, and EVALUATE statements can be nested, but they cannot cross each other's boundaries. Each inner statement must be complete before a closing statement is ended. Correct the source code.
005110	EVALUATE statements nested too deep; maximum is `01.	Reduce the number of nested statements. `01 = Maximum depth supported by this version of SQR
005111	'WHEN' found outside 'EVALUATE' clause.	WHEN may be used only in the context of an EVALUATE clause. Correct the source code.
005112	'IF' or 'WHILE' not completed before 'WHEN' statement.	Correct the syntax.
005114	Incorrect types for comparison. Both must be of the same type (string, numeric or date).	Correct the source line.
005115	'When-other' found outside 'Evaluate' statement.	WHEN can be used only in the context of an EVALUATE statement. Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
005116	'IF' or 'WHILE' not ended before 'WHEN-OTHER' command.	Correct the syntax.
005117	Only one 'WHEN-OTHER' allowed per 'EVALUATE'.	Correct the syntax.
005118	Found 'END-EVALUATE' without matching 'EVALUATE'.	Each EVALUATE command must have a matching END-EVALUATE command. Correct the source code.
005119	'IF' or 'WHILE' command not completed before 'END-EVALUATE'.	Correct the syntax.
005120	'WHEN-OTHER' must be after all 'WHEN's.	Correct the syntax.
005121	No 'WHEN's found inside 'EVALUATE' statement.	Correct the syntax.
005122	'IF', 'EVALUATE' and 'WHILE' statements cannot cross sections or paragraphs.	These commands must be contained within a single section or paragraph. Correct the source code.
005123	'CONTINUE' found outside 'WHILE' statement.	The CONTINUE command is valid only in the context of a WHILE statement. Correct the source code.
005200	Did not find '>' after <....	A leading left angled bracket "<" indicates that you are beginning an ASCII value, which must be ended by a right angled bracket ">". Correct the source line.
005201	Bad ascii character in <...>.	Numbers in angled brackets <> must be between 1 and 255. Correct the source line.
005202	Bad ascii number in <...>.	Numbers in angled brackets <> must be between 1 and 255. Correct the source line.
005203	<...> string is too long; maximum is `01 characters.	Reduce the length of the string. If this is not possible, use PRINT-DIRECT in BEGIN-REPORT or END-REPORT . `01 = Maximum number of characters supported by this version of SQR
005300	Did not find '=' after qualifier: `01	Correct the syntax. `01 = Qualifier name
005301	Qualifier '`01' requires a numeric value.	Correct the syntax. `01 = Qualifier name
005302	Incorrect value for qualifier '`01'. Valid values are:	Correct the source line. `01 = Qualifier name
005303	Invalid qualifier '`01'. Valid qualifiers are:	Correct the source line. `01 = Qualifier name
005304	Qualifier '`01' requires a numeric literal, variable, or column.	Correct the source line. `01 = Qualifier name
005305	Qualifier '`01' references a numeric variable that has not been defined.	Correct the source line. `01 = Qualifier name
005306	Qualifier '`01' requires a string literal, variable, or column.	Correct the source line. `01 = Qualifier name
005307	List not terminated.	Correct the syntax.
005308	Missing comma in list.	Correct the syntax.
005309	Required argument '`01' was not specified.	Correct the source line. `01 = Qualifier name

Error Number	Error Message	Suggestion/Interpretation
005310	Qualifier '`01' has already been specified.	Correct the source line. `01 = Qualifier name
005311	Qualifier '`01' requires a string literal.	Correct the source line. `01 = Qualifier name
005312	Qualifier '`01' requires a list of values: (val [,val]...).	Correct the source line. `01 = Qualifier name
005313	Qualifier '`01' requires a integer value.	Correct the source line. `01 = Qualifier name
005314	Invalid character in variable name '`01'.	Correct the source line. `01 = Invalid character
005315	Qualifier '`01' references a string variable that has not been defined.	Correct the source line. `01 = Qualifier name
005316	Qualifier '`01' uses an invalid Unit-Of-Measure suffix. Valid suffixes are: dp pt mm cm in	Correct the source line. `01 = Qualifier name
005317	Qualifier '`01' can only reference string literals or variables.	Correct the source line. `01 = Qualifier name
005318	Qualifier '`01' can only reference string or numeric literals.	Correct the source line. `01 = Qualifier name
005319	Qualifier '`01' requires a valid position value.	Correct the source line. `01 = Qualifier name
005320	Qualifier '`01' is not allowed.	Correct the source line. `01 = Qualifier name
005400	Second page write attempted while writing current page. Check BEFORE-PAGE, AFTER-PAGE procedures.	Check any procedure invoked by BEFORE-PAGE or AFTER-PAGE to ensure that the commands do not overflow the page or cause a page eject.
005402	String cannot be placed on page: `01 -- placement specified is out of range. (`02,`03,`04)	Ensure the values are within the page limits. `01 = Text value `02 = Row `03 = Column `04 = Length
005403	Error writing the output file. (`01): `02	`01 = System error code `02 = System error message
005404	Cannot open the Postscript startup file: `01 (`02): `03	`01 = Name of the file `02 = System error code `03 = System error message
005405	SQR trial copy exiting after `01 pages.	`01 = Number of pages.
005406	Exiting after requested number of test pages (`01).	`01 = Number of pages
005408	Program stopped by user request.	Informational message.
005500	Cannot set parse_only option.	(Sybase) The DB-Library routine dbsetopt() returned an error. This should never happen. Contact technical support.
005501	Cannot reset parse_only option.	(Sybase) The DB-Library routine dbcropt() returned an error. This should never happen. Contact technical support.
005502	Cannot drop SQR generated stored procedure: `01.	(Sybase) See the database server error message that was also output. This should never happen. Contact technical support. `01 = Stored procedure name
005503	Cannot use `01 datatype as bind variable.	(Sybase) Use another database column. `01 = The database datatype.

Error Number	Error Message	Suggestion/Interpretation
005504	Unknown datatype for bind variable: `01 Cannot create stored procedure.	(Sybase) Please contact technical support. `01 = Unknown database datatype
005505	SQL too large to create stored procedure.	(Sybase) The size of the SQL text needed to create the stored procedure is too large for SQR to handle. Add <code>-XP</code> to <code>BEGIN-SQL</code> or <code>BEGIN-SELECT</code> .
005506	SQR's EXECUTE command not available for this version of Sybase.	(Sybase) Some early versions of Sybase SQL Server or Microsoft SQL Server do not support Remote Procedure Calls (RPCs). Update your database server.
005507	Could not add param to remote procedure call.	(Sybase) A DB-Library routine returned an unexpected error. See the error message from the database.
005508	The number of EXECUTE...INTO &columns does not match the procedure.	(Sybase) Check the definition for the stored procedure you are referencing.
005509	Incorrect number of INTO &columns defined in EXECUTE.	(Sybase) Check the definition for the stored procedure you are referencing.
005510	Error converting OUTPUT Sybase type for EXECUTE.	(Sybase) The DB-Library routine <code>dbconvert()</code> failed to convert the data from the stored procedure. Contact technical support.
005511	Number of OUTPUT parameters from EXECUTE is incorrect.	(Sybase) Check the definition for the stored procedure you are referencing.
005512	Missing default database name for USE.	(Sybase) Correct the syntax.
005512	Missing default database name for USE.	(ODBC) Could not connect to the specified datasource.
005513	You may only specify 'USE db' once, before any SQL statements are executed.	(Sybase) Only one USE command is allowed in a report. Place the SETUP section at the beginning of the SQR report.
005515	Undefined variable referenced in -DB flag: `01	(ODBC) Check for a misspelling. `01 = Variable name
005523	Database commit failed.	The database command to perform a commit returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
005524	Cannot close database cursor.	The database command to close the database cursor returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
005528	DB2 SQL `01 error `02 in cursor `03:	(DB2) `01 = Routine name `02 = Error code `03 = SQR cursor number
	INFORMIX SQL `01 error `02 (ISAM: `03) in cursor `04: `05	(Informix) `01 = Routine name `02 = Error code `03 = ISAM code `04 = SQR cursor number `05 = Error message from database
	ODBC SQL `01 error `02 in cursor `03: `04	(ODBC) `01 = Routine name `02 = Error code `03 = SQR cursor number `04 = Error message from database
	ODBC SQL `01 error `02 in cursor `03: `04	(DDO)

Error Number	Error Message	Suggestion/Interpretation
		`01 = Routine name `02 = Error code `03 = SQR cursor number `04 = Error message from database
	ORACLE `01 error `02 in cursor `03: `04	(Oracle) `01 = Routine name `02 = Error code `03 = SQR cursor number `04 = Error message from database
	Sybase `01 error in cursor `02: `03	(Sybase) `01 = Routine name `02 = SQR cursor number `03 = Error message from database
	Teradata SQL `01 error `02 in cursor `03:	(Teradata) `01 = Routine name `02 = Error code `03 = SQR cursor number
005532	System 10 files are missing.	(Sybase) Contact your system administrator.
005533	Not a System 10 SQL Server.	(Sybase) The CT-Library version of SQR can only connect to a System 10 server. Use the DB-Library version of SQR to connect to a pre-System 10 server.
005534	SQL too long for PREPARE/DECLARE; maximum `01 characters.	(Teradata) The SQL statement is too large. `01 = Maximum number of characters supported by this version of SQR
005536	Unknown error message number: `01.	(DB2) `01 = Error message number
005537	Empty error message returned from system for error number: `01.	(DB2) `01 = Error message number
005538	Invalid SELECT statement; COMPUTE clauses are not supported.	(Sybase) The select statement contains a COMPUTE clause that is not supported.
005539	Could not connect to datasource specified in -db variable: ``01'.	(ODBC) Could not connect to the specified datasource.
005540	Not connected to a database, database access is not allowed.	The SQR program is no longer connected to a database. Commands that access the database can no longer be used. This situation can occur if the CONNECT fails and the ON-ERROR option was used.
005543	Specify the Oracle DLL name in SQR.INI in [Environment:Oracle] section for ORACLE_DLL entry, such as ORACLE_DLL=orant71.dll	(Oracle) SQR was unable to load the Oracle DLL. By default, SQR looks first for "ociw32.dll" or the DLL specified by the ORACLE_DLL entry in the [Environment:Oracle] section of SQR.INI. If that DLL could not be loaded, then SQR attempts to load 'orant71.dll'.
005600	GETWRD: Word too long; maximum is `01.	Reduce the length of the "word". `01 = Maximum size of a "word" supported by this version of SQR
005700	Cannot call SQR recursively.	SQR cannot be called recursively. This error can only occur if a User Function from either UFUNC.C or UCALL.C calls

Error Number	Error Message	Suggestion/Interpretation
		the sqr() routine. Do not call sqr() from a UFUNC.C or UCALL.C routine.
005701	Too many SQR command line arguments; maximum is `01	To pass more than this number of arguments, use a @file argument file containing one argument per line. `01 = Maximum number supported by this version of SQR.
005702	Log file name specified is too long.	Reduce the length of the log file name.
005703	Error opening the SQR log file: `01' (`02): `03	`01 = Name of the file `02 = System error code `03 = System error message
005704	Missing program name.	The name of the program file was not found on the command line. The program name must be the first parameter on the command line.
005705	Program file name specified is too long.	Reduce the length of the program file name.
005707	Error opening the -E error file: `01' (`02): `03	`01 = Name of the file `02 = System error code `03 = System error message
005708	Cannot find `01 in SQRDIR, PATH or \SQR.	The specified file cannot be located in any of the directories pointed to by the mentioned environment variables or default directories. Make sure the "file" is present in one of the locations searched. `01 = File name
005709	`01 environment variable is not defined.	As of version 2.5, the environment variable SQRDIR must be defined. `01 = Name of the environment variable
005710	`01 path too long.	The length of the directory path plus the length of the file name to be opened is too long for SQR to handle. Reduce the length of the directory path. `01 = Environment variable name
005711	Bad number in -T test flag.	The number specified must be > zero. Correct the value.
005712	-G option requires arguments.	(VAX) The command line option is ill-formed. Correct the syntax.
005713	Too many arguments to -G option; maximum is `01.	(VAX) The command line option is ill-formed. Correct the syntax. `01 = Maximum number of arguments supported by this version of SQR
005714	-G attribute too long; maximum is `01.	(VAX) The command line option is ill-formed. Correct the syntax. `01 = Maximum number of each attribute supported by this version of SQR
005716	Unknown flag on command line: `01	Correct the syntax. `01 = Unknown command line flag
005717	Cannot open channel to TT; status = `01	(VAX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System status
005718	Cannot read from TT; status = `01	(VAX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System status

Error Number	Error Message	Suggestion/Interpretation
005719	Cannot close channel to TT; status = `01	(VAX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System status
005720	Error opening tty. (`01): `02	(DG, UNIX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005721	Error with 'ioctl()'. (`01): `02	(DG, UNIX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005722	Error reading tty. (`01): `02	(DG, UNIX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005723	Error closing tty. (`01): `02	(DG, UNIX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005724	Bad number in -B flag.	(Oracle, Sybase) The number specified must be > zero. Correct the value.
005734	No program name given.	The report name must be the first command line argument.
005737	Unknown printer type specified with -PRINTER: switch.	The printer type can be EH, HT, LP, HP, PS, or WP. WP is valid only with PC/Windows.
005738	Database name needs to be included with -DB switch.	(ODBC) Could not connect to the specified datasource.
005738	Database name needs to be included with -DB switch.	(Sybase) Supply the database name.
005739	Too many -F switches; maximum is `01.	Reduce the number of -F switches. `01 = Maximum number allowed
005742	Attempt to invoke viewer (using WinExec) failed; error code = `01.	(Windows) `01 = System error code
005743	Unknown numeric type specified with -DNT: switch.	Correct the command line.
005744	-DNT:Decimal precision (`01) is out of range (`02 - `03).	Correct the command line. `01 = Specified precision `02 = Minimum allowed `03 = Maximum allowed
005745	The specified default numeric type '`01 = `02' is invalid.	Correct the SQR.INI file entry. `01 = Entry `02 = Value
005746	The decimal precision '`01 = `02' is out of range (`03 - `04).	Correct the SQR.INI file entry. `01 = Entry `02 = Value `03 = Minimum allowed `04 = Maximum allowed
005747	The following error(s) occurred while processing the [`01] section from SQR.INI.	See the error message(s) that follow. `01 = Name of the section
005750	The -Burst switch is not properly formatted.	The "Burst" command line flag is not properly formatted.

Error Number	Error Message	Suggestion/Interpretation
005751	The -Burst switch cannot be used with the -NOLIS switch.	The "Burst" command line flag cannot be specified when the -NOLIS command line flag is also specified.
005752	The -Burst switch requires either the -Printer:HT or -Printer:EH switch to be specified.	The "Burst" command line flag is applicable only when HTML code is produced. You must specify either the -PRINTER:HT or -PRINTER:EH switch.
005754	The -Burst switch caused no output to be generated.	The "Burst" command line flag was specified with a set of page ranges that prevented any output to be created. Change the page ranges.
005755	The -Printer:HT switch does not support UTF-8 encoded data. Use the -Printer:EH switch instead.	Spf_ht.c can't handle UTF-8
005756	The -EH_FullHTML switch support the following values: 30, 32, and 40.	The 'EH_FullHTML' command line flag is not properly formatted.
005757	The -EH_Browser switch can be specified with one of the following values: Basic, Netscape, IE, or ALL.	The 'EH_Browser' command line flag is not properly formatted.
005758	The -EH_Language switch can be specified with one of the following values: English, French, German, Portuguese, Spanish, SChinese, TChinese, or Japanese.	The 'EH_Language' command line flag is not properly formatted.
005781	An ASCII-based encoding (ASCII, CP1252, etc) must be specified in order to generate barcodes for HPLaserJet output.	An ASCII-based encoding must be specified for the ENCODING-REPORT-OUTPUT setting in the INI file when generating barcodes for HPLaserJet (-printer:hp)
005900	Bad number in -`01	(Windows) Specify a valid number. `01 = Command line option
005901	Bad filename in -`01	(Windows) Specify a valid file name. `01 = Command line option
005902	Bad directory in -`01	(Windows) Specify a valid directory path. `01 = Command line option
005903	Cannot access the @ parameter file (`01): `02	(Windows) Depends on the system error message. `01 = System error code `02 = System error message
005904	The argument list is too long; maximum is `01.	(Windows) To pass more than this number of arguments, use a @file argument file containing one argument per line. `01 = Maximum number supported by this version of SQR.
005905	Cannot open the report file (`01): `02	(Windows) Depends on the system error message. `01 = System error code `02 = System error message

Table 98 Numbered Messages 006000 to 006999

Error Number	Error Message	Suggestion/Interpretation
006000	Error writing the printer file. (`01): `02	Can occur during normal operations due to the system environment (for example, file locking, permissions). Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message

Error Number	Error Message	Suggestion/Interpretation
006001	Error reading the printer file. (`01): `02	Can occur during normal operations due to the system environment (for example, file locking, permissions). Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
006002	Cannot open the printer file: `01 (`02): `03	Can occur during normal operations due to the system environment (for example, file locking, permissions). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message
006003	Unexpected End-Of-File while processing the printer file.	Possibly the file got corrupted. Try to recreate the .spf file. If the error persists, contact technical support.
006004	Encountered unknown SPF code (`01) while reading the printer file.	Possibly the file got corrupted. Try to recreate the .spf file. If the error persists, contact technical support. `01 = Unknown SPF code
006100	Duplicate chart (`01).	Each chart must be given a unique name. `01 = Chart name
006101	Unknown chart (`01).	Chart could not be found. `01 = Chart name
006104	Too many pie segments (`01). Max is `02.	Correct the source code. `01 = Number of segments `02 = Maximum allowed segments
006105	Chart module is not initialized.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006106	XY charts may have only numeric columns.	Correct the syntax.
006107	The 3rd column in the data array must be a character column to specify USE-3RD-DATA-COLUMN.	Correct the syntax.
006120	INTERNAL: Bad chart index from stack (`01).	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Chart index
006122	INTERNAL: Unsupported Graftman chart type (`01).	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Chart type
006123	INTERNAL: Unsupported pie-explode setting (`01).	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Setting value
006124	INTERNAL: Unsupported tick-mark placement (`01).	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Placement value
006125	Graftman interface message (`01) not supported.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Message code
006126	Unrecognized return code (`01) from Graftman command message (`02).	Should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Return code `02 = Message code

Error Number	Error Message	Suggestion/Interpretation
006127	Cannot fit Chart/Image into the current page. Position: (`01, `02) Size: (`03, `04)	Correct the source code. Production Reporting aborts the program run. `01 = Row `02 = Column `03 = Width `04 = Depth
006140	Duplicate image (`01).	Images must be given unique names. `01 = Image name
006141	Unknown image (`01).	Image name could not be found. `01 = Image name
006142	Cannot open image file (`01). (`02): `03	`01 = Name of the file `02 = System error code `03 = System error message
006143	Unknown or missing image type (`01).	Enter a valid image type. `01 = Image type
006144	Unknown or missing printer type (`01).	Enter a valid Printer type. `01 = Printer type
006145	Duplicate FOR-PRINTER entries for printer (`01).	Only a single FOR-PRINTER can be specified for a printer type . `01 = Printer type
006146	The image type (`01) is not supported by printer type (`02).	The image, based on its type is invalid for the printer specified. For example, an EPS image is only valid for Postscript printer. `01 = The image type `02 = The printer type
006147	Invalid number of items in FOR-PRINTER list.	Too few or too many items in the FOR-PRINTER list. Correct the syntax.
006150	INTERNAL: Bad image index from stack (`01).	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Image name
006200	This report has already been defined.	Each report must be given a unique name.
006201	This layout has already been defined.	Each layout must be given a unique name.
006202	This printer has already been defined.	Each printer must be given a unique name.
006203	The values for '`01' must be > 0.	Correct the syntax. `01 = Qualifier name
006204	Qualifiers '`01' and '`02' are mutually exclusive.	Correct the syntax. `01 = Qualifier name `02 = Qualifier name
006205	Qualifier '`01' is not applicable with a 'default' printer.	Correct the syntax. `01 = Qualifier name
006206	The list must contain report names or ALL.	Correct the syntax.
006207	'ALL' must be specified by itself.	Correct the syntax.
006208	No report name was specified.	Correct the syntax.
006209	No layout name was specified.	Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
006210	No printer name was specified.	Correct the syntax.
006211	The name cannot be 'ALL'.	Correct the syntax.
006212	The name can only contain characters [0-9 A-Z _ -].	Correct the syntax.
006213	Report `01' is referenced by multiple `02' printers.	Correct the syntax. `01 = Report name `02 = Printer type
006214	Qualifier `01' is not allowed with a `02' printer.	Correct the syntax. `01 = Qualifier name `02 = Printer type
006215	The value for `01' must be `02 0.	Correct the syntax. `01 = Qualifier name `02 = Relation to zero (<,<=,=,>=,>)
006216	Report `01' does not exist.	Correct the syntax. `01 = Report name
006217	The report name can be a string literal, variable, or column.	Correct the syntax. `01 = Report name
006218	Referenced layouts not defined:	A list of undefined layouts follows this message.
006219	Referenced reports not defined:	A list of undefined reports follows this message.
006220	Referenced printers not defined:	A list of undefined printers follows this message.
006221	The following SQR commands (listed below) cannot be used when any of the following NEW SQR commands are also used in the same report:	Correct the syntax.
006224	No printer type was specified.	Correct the syntax.
006225	Incorrect value for printer type. Valid values are:	Correct the syntax. A list of valid printer types follows this message.
006226	Attempt to execute the `01 command while processing the `02 procedure.	SQR aborts the program run. `01 = SQR command `02 = Procedure name
006227	Incorrect value for 'paper-size'. Specify the actual dimensions or one of the following names:	Correct the syntax. A list of valid predefined paper-size names follows this message.
006228	Referenced TOC (Table Of Contents) not defined:	A list of undefined Table of Contents follows this message.
006229	This TOC (Table Of Contents) has already been defined.	Each Table of Contents must be given a unique name.
006230	The list must contain TOC (Table of Contents) names or ALL.	Correct the syntax.
006231	The TOC (Table Of Contents) entry cannot be positioned given the LEVEL (`01) and INDENTATION (`02) values.	The Table of Contents entry will not fit given the specified level and current indentation values. `01 = Specified LEVEL= value `02 = Current INDENTATION= value
006232	`01 command not allowed while generating the Table of Contents.	The specified command cannot be used while the Table of Contents is being generated. `01 = SQR command
006233	The TOC (Table of Contents) entry "A" cannot be processed because the existing entry "B" is positioned below it. A: Line = `01, Level = `02, Text = `03' B: Line = `04, Level = `05, Text = `06'	Correct the program logic to eliminate the conflict between the two TOC (Table of Contents) entries. `01 = A: Line number `02 = A: Level value `03 = A: Text value `04 = B: Line number `05 = B: Level value `06 = B: Text value

Error Number	Error Message	Suggestion/Interpretation
006303	Parameter (`01) is required, but has not been specified.	Correct the syntax. `01 = Parameter name
006304	Parameter (`01) already specified.	Correct the syntax. `01 = Parameter name
006308	Missing part of specification for parameter (`01).	Correct the syntax. `01 = Parameter name
006309	Parameter (`01) requires literal.	Correct the syntax. `01 = Parameter name
006352	INTERNAL: Unsupported option/request (`01) in (`02).	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Option/request code `02 = Function name
006400	Unsupported background color.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006401	Unsupported border color.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006402	Border width out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006403	X position out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006404	Y position out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006405	X size out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006406	Y size out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006407	Unsupported font.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006408	Unsupported font style.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006409	Unsupported font color.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006410	Unsupported horizontal text justification value.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006411	Unsupported vertical text justification value.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006412	Unsupported font path.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006413	Unsupported font rotation.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006414	Font size out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006415	Text line id# out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006416	Unsupported chart type.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006417	Unsupported chart sub-type.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006418	Unsupported chart orientation (not H or V).	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006419	Unsupported perspective (not 2D or 3D).	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006420	Unsupported axis (not X or Y).	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006421	Unsupported axis label data type.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006422	Dataset id# out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006423	Unsupported dataset type.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006424	Unsupported dataset color.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006425	Unsupported dataset line style.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006426	Unsupported dataset fill pattern.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006427	Unsupported dataset marker.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006428	Chart type does not support Y-axis datasets.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006429	Pie-chart segment id# is out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006430	Unsupported pie-segment color.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006431	Unsupported pie-segment border color.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006432	Unsupported pie-segment pattern.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006433	Unsupported pie-segment explode setting.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006434	Command only valid for charts of type 'pie'.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006435	Pie-chart radius out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006436	Pie-chart starting angle out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006437	Unsupported pie-chart fill direction. Must be clockwise or counter-clockwise.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006438	Unsupported pie-segment label position.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006439	Unsupported pie-segment quantity display position.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006440	Unsupported pie-segment per-cent display position.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006441	Unsupported legend style.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006442	Unsupported legend horizontal position.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006443	Unsupported legend vertical position.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006444	Text charts do not support legend.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006445	Number of datasets specified does not match data.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006446	Unsupported axis label position.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006447	Unsupported axis type (not LINEAR or LOG).	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006448	Pie and text charts do not support axis control.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006449	Unsupported axis min scaling.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006450	Unsupported axis max scaling.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006451	Unsupported axis max scaling.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006452	Beginning of tickmarks is after end.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006453	Unsupported tickmark type.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006454	Unsupported grid type.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006455	Unsupported grid color.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006456	Grid line width out of range.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006457	Unable to open grafcap file.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006458	Unsupported grafcap device.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006459	Error in grafcap entry specification.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006460	Unable to open chart output destination.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006461	Internal error during ggDraw.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006462	Improper parameters passed to gscale.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006463	The shared library specified in the grafcap file could not be found.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006464	A function called from the shared library specified in the grafcap file could not be found.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006500	The bar code could not be positioned on the page. Row: `01, Column: `02, Height: `03	Correct the source code. `01 = Row `02 = Column `03 = Height
006501	Unknown BCL error (`01) encountered.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = BCL error code
006502	Invalid bar code type (`01): Valid values are from 1 to 15.	Correct the source code. `01 = Bar code type.
006503	The length of the bar code text '01' must be between 1 and 30 characters.	Correct the source code. `01 = Bar code text
006504	The length of the caption text '01' must be between 1 and 30 characters.	Correct the source code. `01 = Caption text
006505	Invalid printer type (`01): Valid values are from 0 to 13.	Correct the source code. `01 = Printer type
006506	Invalid offset: Valid values are from 0 to 250.	Correct the source code.
006507	Invalid height (`01): Valid values are from 0.1 to 2.0 inches.	Correct the source code. `01 = Height

Error Number	Error Message	Suggestion/Interpretation
006508	Invalid checksum: Valid values are from 0 to 2.	Correct the source code.
006509	Invalid pass: Valid values are from 1 to 6.	Correct the source code.
006510	The bar code text '01' is not valid for the type of bar code (`02) selected.	Correct the source code. `01 = Bar code text `02 = Bar code type
006511	Internal error: Could not generate the bar code.	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006512	Internal error: Bar code buffer required too large (>32K).	Should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006601	Cannot allocate the device context for the default printer.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006602	Failed to start printing the document.	(Windows) Can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006603	New-page (start) failed on page `01.	(Windows) Can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator. `01 = Page number
006604	New-page (end) failed on page `01.	(Windows) Can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator. `01 = Page number
006605	End document failed.	(Windows) Can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006606	Error reading font information from the [Fonts] section in SQR.INI. Using the default font.	(Windows) Correct the [Fonts] section in SQR.INI.
006607	Failed to create a brush for shading.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006608	Failed to select font `01.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Font name
006609	Failed to modify font `01.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Font name
006610	Failed to create a pen that was required to draw a box.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006612	Failed to create a pen that was required to draw a vertical line.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.

Error Number	Error Message	Suggestion/Interpretation
006613	Failed to open the image bitmap file (`01). (`02): `03	(Windows) Can occur during normal operations due to the system environment (file locking, permissions). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message
006614	The file (`01) does not contain a valid bitmap.	(Windows) Specify a valid bitmap file. `01 = Name of the file
006615	Failed to create the palette for image (`01).	(Windows) Can occur due to lack of system resources or an invalid bitmap. Record the steps leading up to the error and contact your system administrator. `01 = Name of the file
006616	Failed to load RLE into memory for image (`01).	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Name of the file
006617	Failed to convert DIB to DDB for image (`01).	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Name of the file
006618	Failed to draw the bitmap image (`01).	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Name of the file
006619	Cannot access the default printer's driver.	(Windows) Can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006620	Cannot select the charting clip area onto the printers DC.	(Windows) Can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006621	Cannot select create a metafile required for business graphics.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006622	Cannot create a region required for business graphics.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006623	Cannot create a DC required for business graphics.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006624	Cannot create a bitmap required for business graphics.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006625	Business graphics failed while setting up the device (ggWinDevice).	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006626	Cannot draw business graphics.	(Windows) Can occur due to lack of system resources or it can be due to a damaged LIBSTI.INI file. The LIBSTI.INI file resides in the Windows main directory. Make sure that the GPATH= and IPT= entries point to a valid SQR bin directory.

Error Number	Error Message	Suggestion/Interpretation
		Record the steps leading up to the error and contact your system administrator.
006700	SQRDIR is not defined.	(Windows) The variable SQRDIR must be defined in SQR.INI.
006701	Could not allocate memory while attempting to register the .spf filename extension.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006704	Cannot open or read file (`01) (`02): `03	(Windows) Can occur during normal operations due to the system environment (e.g. file locking, permissions, etc.). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message
006705	File (`01) is not in SPF packet format.	(Windows) The file was not produced by SQR or it has been corrupted. `01 = Name of the file
006706	Failed to identify the start of the report (`01).	(Windows) The file was not produced by SQR or it has been corrupted. `01 = Name of the file
006707	An invalid seek was made for page `01.	(Windows) Internal error which should not occur under normal operations. Contact technical support. `01 = Page number
006708	Too many errors were encountered while processing the file. Processing has been stopped.	(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006709	Failed to open the image bitmap file (`01). (`02): `03 This message is displayed only once per SPF file.	(Windows) Can occur during normal operations due to the system environment (e.g. file locking, permissions, etc.). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message
006800	`01: Detected internal program error.	Internal error that should never occur during normal operation. Record the steps leading up to the error and contact technical support. `01 = Name of the routine
006801	`01: Null Operand Passed as input.	Internal error that should never occur during normal operation. Record the steps leading up to the error and contact technical support. `01 = Name of the routine
006802	`01: Decimal Exponent Under/Overflow.	Exponent Under/Overflow: Exponent of decimal number has exceeded the valid boundaries established for the decimal type. Check the documentation for the current upper and lower bounds of a decimal object. `01 = Name of the routine
006803	`01: Decimal to Integer Conversion Under/Overflow.	Integer Under/Overflow: Cannot convert input decimal object into a valid integer number. Decimal object exceeds the established integer boundaries for this machine architecture. Check the magnitude and sign of the decimal object to ensure that it falls within the upper and lower bounds of an integer number. `01 = Name of the routine
006805	`01: Decimal Precision Under/Overflow.	Decimal Precision Under/Overflow: Attempt made to initialize decimal object with an invalid precision. Check the

Error Number	Error Message	Suggestion/Interpretation
		input precision value against the documented upper and lower boundaries for a decimal object. `01 = Name of the routine
006806	`01: String to Decimal Object Conversion Error.	String To Decimal Conversion Error: Length of input string is greater than precision of underlying decimal object. Either increase the precision of the decimal object or reduce the size of the input mantissa to match the decimal object precision. `01 = Name of the routine
006807	`01: Truncation/Rounding Error - Outside Valid Range for Decimal Object.	Truncation/Rounding Error: Input truncation or round value is outside the valid range for this decimal object. Please ensure that the truncation/round value is greater than or equal to zero and less than the precision of the underlying decimal object. `01 = Name of the routine
006808	`01: Decimal Error: Cannot Divide by Zero.	Decimal Math Divide by Zero Error: Attempt made to divide a decimal object by zero. Please check divisor to ensure that it does not equal zero before attempting to divide. `01 = Name of the routine
006900	There is no default printer set up on your system. Use the Control Panel "Printers" applet to define it.	(Windows) SQR Print requires that a default printer be defined. Use the "Printers" applet in the Control Panel to define one.

Table 99 Numbered Messages 007000 to 007999

Error Number	Error Message	Suggestion/Interpretation
007000	The locale '`01' is not defined in SQR.INI.	Check for a misspelled locale name and/or the SQR.INI file. `01 = Locale name
007001	At least one qualifier must be specified.	Correct the source code.
007002	The value for '01' must be a list of 02 string literals, variables or columns.	Correct the source code. `01 = Qualifier `02 = Number of entities in list
007003	The values for '01' and '02' cannot be the same.	Correct the source code. `01 = Qualifier `02 = Qualifier
007004	The value for '01' (`02) must be a single character which is not in the list: "03".	Correct the source code. `01 = Qualifier `02 = Value `03 = List of invalid characters
007005	The value for '`01' (`02) is invalid. Valid values are:	Correct the source code. `01 = Qualifier `02 = Value
007006	The last character of the '`01' value (`02) cannot be a digit or the minus sign or the same as either of the separators.	Correct the source code. `01 = Qualifier `02 = Invalid character
007007	The first character of the '`01' value (`02) cannot be a digit or the minus sign or the same as either of the separators.	Correct the source code. `01 = Qualifier `02 = Invalid character
007008	The following errors occurred while processing the (`01) locale from SQR.INI.	This message precedes error messages encountered while processing the SQR.INI file. `01 = Locale name
007009	The value for '`01' cannot be 'DEFAULT' or 'SYSTEM'.	Correct the syntax. `01 = Qualifier

Error Number	Error Message	Suggestion/Interpretation
007010	The value for '`01' (`02) is not properly formatted: Did not find the '>' for the '<nnn>' construct.	Correct the syntax. `01 = Qualifier `02 = Value
007011	The value for '`01' (`02) is not properly formatted: The value of an '<nnn>' construct must be from 1 to 255.	Correct the syntax. `01 = Qualifier `02 = Value
007012	The default locale (`01) specified in the [`02] section of SQR.INI has not been defined.	Correct the syntax. `01 = Locale name `02 = Section name
007013	The value for '`01' (`02) must be a list of `03 quoted string literals.	Correct the syntax. `01 = Qualifier `02 = Value `03 = Number of entities in list
007014	The entry (`01 = `02) is not valid.	Correct the SQR.INI entry. `01 = Qualifier from the SQR.INI file `02 = Qualifier's value
007100	The use of an edit mask or the keywords NUMBER, MONEY, or DATE is not legal when storing numeric variables.	Correct the source code.
007101	The last keyword is not '`01'.	Correct the source code. `01 = Keyword
007102	Incompatible source and destination variable types.	Correct the source code.
007103	The keyword (`01) is not compatible with the variable (`02).	Correct the source code. `01 = Keyword `02 = Variable name
007104	The use of an edit mask or the keyword DATE is not legal if both variables are date variables.	Correct the source code.
007200	The specified precision (`01) is out of range (`02 - `03).	Correct the source code. `01 = Specified precision `02 = Minimum precision `03 = Maximum precision
007201	The precision is specified by a value from `01 to `02 surrounded by parentheses.	Correct the source code. `01 = Minimum precision `02 = Maximum precision
007202	Variable (`01) is not a decimal variable and cannot have a precision associated with it.	Correct the source code. `01 = Variable name
007203	A string variable name is required here.	Correct the source code.
007204	A numeric variable name is required here.	Correct the source code.
007205	The variable (`01) has already been defined as '`02' and may not be redefined.	Correct the source code. `01 = Variable name `02 = Variable type
007206	The variable type has not been specified.	Correct the source code.
007207	This command is only allowed within local procedures.	Correct the source code.
007208	This command must be before all other commands in the procedure.	Correct the source code.
007209	Only string (\$) and numeric (#) variables may be declared.	Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
007210	Invalid variable name specified.	Correct the source code.
007211	You cannot declare a global variable from within a procedure.	Correct the source code.
007400	The specified character is invalid in the current character set.	Correct the program logic.
007401	'`01' is not a valid value for the ENCODING environment variable.	The specified encoding scheme is not known by SQR. `01 = ENCODING environment variable setting.
007403	The Double-Byte SQR command '`01' is not supported in this version of SQR.	The SQT file contains a reference to an SQR command, which is not supported by this version of SQR. `01 = SQR command name
007405	The barcode text '`01' cannot contain double-byte characters.	Correct the source code. `01 = Bar code text
007501	Using `01 edit mask from (`02) against (`03)	A date edit mask element was detected which could cause date data to be incorrectly interpreted. This warning message can be turned off by setting the "OutputTwoDigitYearWarningMsg" entry to the [Default-Settings] section of SQR.INI to FALSE. `01 = Edit mask element `02 = Edit mask being used `03 = Value being applied to the edit mask
007601	Cannot access the Java file (`01) (`02): `03	SQR cannot access the required file. `01 = Name of the file `02 = System error code `03 = System error message
007602	-EH_Scale: value (`01) is out of range (`02 - `03).	Correct the command line. `01 = Specified scale `02 = Minimum allowed `03 = Maximum allowed
007603	-Printer:EH functionality is not available on this platform.	Enhanced HTML functionality is not available on this platform.
007604	-Printer:PD functionality is not available on this platform.	PDF functionality is not available on this platform.
007605	Cannot support Unicode internally. Please reset the UseUnicodeInternal setting in SQR.INI to FALSE.	Cannot support Unicode internally. Reset the UseUnicodeInternal setting in SQR.INI to FALSE.
007702	Invalid entry for keyword, '`01='`02'	Correct the source code.
007703	May only specify either PROCEDURE=, or COMMAND=, or GETDATA=, exclusive.	Correct the source code.
007704	Must specify a SCHEMA.	Correct the source code.
007705	Must specify either a PROCEDURE, COMMAND, or GETDATA.	Correct the source code.
007706	CONNECTION '`01' not found. No such connection.	Correct the source code.
007707	The returned set of Procedure parameters (INOUT and OUT) (length = `01 items) did not include one or more of the specified items.	Stored procedure error.

Error Number	Error Message	Suggestion/Interpretation
007708	Encountered a parameter of type '`01'. Valid types are either IN, OUT, or INOUT. If no type is entered, the type defaults to IN.	Stored procedure error.
007709	The datasource failed to provide the expected return status value. Verify the query metadata.	Datasource error.
007711	Failed to login to the requested datasource (Connection='`01', username='`02'). DETAILS: `03	Logon failed.
007712	The requested rowset (`01) was not available. Verify the query metadata.	Not enough rowsets.
007713	Missing or invalid Registry.properties file. Verify that the CLASSPATH includes SQRDIR, that SQRDIR contains the folder with the Registry.properties file, and that the Registry.properties file is valid.	Incorrect environment setup.
007714	The datasource (`01) does not support the requested capability (`02). Check the capabilities list for the datasource, located in the Properties folder.	Invalid query for datasource.
007715	Failed to start the Java Virtual Machine (JVM). Possible causes are: missing or invalid jdk files, incorrect CLASSPATH, or insufficient resources.	Incorrect environment setup.
007717	The query failed. DETAILS: `01	Query failed.
007718	Failure setting property '`01'. DETAILS: `02	Property-set failed.
007721	Parameter `01 (`02) was passed to the PROCEDURE as data type `03; expected (`04) type `05. Verify the query metadata.	A failure occurred during row fetch.
007722	Invalid query parameter: Reason: `01	Bad procedure parameter.
007723	Too many parameters (= `01) were supplied to the query. Verify the query metadata.	Bad procedure parameter.
007724	Parameter `01 (`02) was passed to the PROCEDURE as type `03; expected type `04. Verify the query metadata.	Bad procedure parameter.
007725	Parameter `01 (`02, JDO-type `03), specified 'NULL', is a required-parameter. Specify a value or variable name.	Bad procedure parameter.
007727	Unable to retrieve metadata for Procedure=`01, Schema=`02. DETAILS: `03	Metadata check failed.
007728	Parameter list type mismatch (# `01, SQR type = `02). The datasource expected a parameter of type `03. Verify the query metadata.	Parameter list mismatch.

Error Number	Error Message	Suggestion/Interpretation
007729	List size mismatch detected while fetching data of type ROW, `01 items, into SQR list-variable, `02 items. Fetching will proceed to the smaller size.	List size mismatch.
007730	Incorrect syntax for BEGIN-SELECT ... FROM. Options are: FROM ROWSETS=... FROM PARAMETER= \$strvar strlit	Bad begin-select syntax.
007732	Attempt to use a scalar SQR variable ('01') to reference a ROWSET procedure parameter ('02'). Use either the keyword 'NULL', or an SQR LIST variable (%var). Verify the query metadata.	Bad proc parameter.
007733	The list of keywords entered to the PARAMETERS keyword must be terminated with a semicolon.	Bad proc parameter. Correct the source code.
007734	Datasource '01' not found. The Connection being used by this query specifies a datasource which is not listed in the DDO Registry ('02'). DETAILS: `03.	Bad proc parameter. Correct the source code.
007735	Missing one or more DDO {fname} .jar files. Verify the location of the original-installation files, and that they are accessible. Error code: `01. Classpath: `02.	Bad environment.
007736	Unable to open Connection ('01') to datasource ('02'). Possible causes: (a) the Declare- or Alter-connection specification is invalid, or (b) the datasource is no longer available. DETAILS: `03.	Bad environment.
007738	At least one JNI method pointer was lost. This should never occur: record the steps leading up to this failure, and contact Technical Support. DETAILS: Schema=`01', Proc=`02'.	Bad environment.
007739	Unable to locate query object `01' in the specified schema (`02). DETAILS: `03.	Bad environment.
007740	Invalid &pseudonym or 'TYPE=' data-type specified for a begin-select column-variable. Valid types are: CHAR, TEXT, DATE, NUMBER, BINARY.	Correct the syntax.
007741	Illegal attempt to fetch a non-scalar field into a column variable. Correct the query.	Correct the syntax.
007742	The output parameter specified in 'Begin-Select ... From Parameter = `01' is not available. Available parameters: `02.	Bad command.
007743	The output parameter specified in 'Begin-Select ... From Parameter = `01' is not of type ROWSET. Verify the query metadata.	Bad command.
007744	Illegal attempt to assign an SQR variable ('01') of type '02' the value from a DDO object ('03') of type '04'. Verify the query metadata.	Bad var assignment.

Error Number	Error Message	Suggestion/Interpretation
007745	Illegal attempt to assign an SQR column variable ('01') of type '02' the value from a DDO object of type '03'. Verify the query metadata.	Bad var assignment.
007746	Failed to locate the requested Rowset (`01) while processing the query. The last available Rowset number is `02. Verify the query metadata.	Not enough RowSets.
007747	The query raised a DDO exception. DETAILS: `01.	Bad query.
007748	A BEGIN-SELECT paragraph was coded, but the query returned no Rows.	No data warning.
007749	Invalid syntax for PARAMETERS=(...) statement. Use: PARAMETERS=(%v \$v #v &v NULL SKIP numlit datelit textlit [IN INOUT], ...) All parameters must be specified. Optional parameters which are to be ignored may be specified by the keyword 'NULL' or 'SKIP'. Correct the syntax.	Incorrect syntax.
007750	FATAL: Failure creating Java object.	General failure.
007751	Attempt to create a List variable of size greater than the maximum size of `01 items.	General failure.
007753	Attempt to access List-row (`01) beyond the List size (`02 rows).	Bad list assignment/setup.
007754	Attempt to assign/modify a List row is not compatible with the List definition.	Bad list assignment/setup.
007755	Attempt to assign a row to a non-existent List variable. Define the List first, using the syntax: let %lname[size] = list(NUMBER DATE TEXT #var \$var [, ...])	Bad list assignment/setup.
007756	Incorrect syntax for List-variable reference. Use: let [\$ #]var = %listname[nlit #var].colname	Bad list assignment/setup.
007757	Alter-connection statement missing 'DSN=...'.	Improper alter-conn.
007758	List-definition size specifier must be literal.	Improper alter-conn.
007759	Attempt to access a non-existent List-column ('01').	No such list column name.
007760	Must specify one of the keywords, FROM-ROWSETS or FROM_PARAMETER.	Incorrect syntax for Load-lookup.
007761	Incorrect syntax to Load-lookup 'PARAMETERS=' keyword. Use: PARAMETERS=(slit nlit \$var #var %var &var, ...) No line wrapping is allowed for this usage.	Incorrect syntax for Load-lookup.
007762	Too many parameters (`02) entered to Load-Lookup command. Max parameters is `01.	Incorrect syntax for Load-lookup.

Error Number	Error Message	Suggestion/Interpretation
007763	Problem executing the cursor for LOAD-LOOKUP table '`01'. DETAILS: `02.	The database server returned an error while trying to execute the SQL statement needed to process the LOAD-LOOKUP command. `01 = Load lookup table name
007764	Bad return fetching row from database in LOAD-LOOKUP table '`01'. DETAILS: `02	The database server returned an error while fetching the data. `01 = Load lookup table name
007765	DC, DI sort options not supported with this SQR version. To sort, use SORT=SC or SORT=SI.	Database sort not supported for Load-Lookup with DDO.
007766	Must specify a query keyword; PROCEDURE=, COMMAND= or GETDATA=.	Incorrect syntax for Load-lookup. Specify a keyword representing the query.
007767	Unknown column variable type.	Unknown data type returned by the server.
007768	The property `01` was not found in the property sheet for the specified datasource (`02). Available property names are: `03. The datasource property sheet does not include the named property.	Verify the metadata and correct the syntax.
007771	Did not find value after ``01 ='	The code specified a Connection = keyword, but no matching literal. Correct the syntax.
007774	Invalid attempt to establish a second connection to datasource '`02', using Connection '`01'. The Connection ``01' is declared to allow only one active login (no-duplicate=TRUE).	Duplicate logins specified as not allowed. Correct the source code, declare a new Connection, or omit the use of no-duplicate in the subject Connection
007775	Bad value ('`01') for Alter-connection keyword (`02). Valid values are: `03.	Bad keyword value. Refer to the language reference and correct the syntax.
007778	Datasource login not available.	Connection non-existent. Correct the source. Possible causes: No BEGIN-EXECUTE statement.
007779	Unable to verify ResultSet column types due to use of variable. Variables are not allowed either for CONNECTION, SCHEMA, PROCEDURE, From-Parameter, or the first element of the From-Rowsets. Use literals or define column types using 'type=<datatype>'.	Can't verify colvar types with variable entry; use literals or define column types. Correct the source.
007780	Unable to verify ResultSet column types. Must specify column types using 'type=CHAR NUMBER DATE' construct when selecting from datasources which do not supply metadata, or when using the COMMAND= and GETDATA= keywords.	Can't verify colvar types with for COMMAND, GETDATA or from [TABLES]. Use type=<datatype> on select variables. Correct the source.
007781	Unable to log onto datasource to obtain query metadata. Specify the Connection for this query using a complete Declare-Connection statement, or specify type=<datatype> for each column variable in the Begin-Select.	Must declare a complete connection if use early binding of select column variables.
007783	Could not execute SQL. DETAILS: `01	An error occurred while trying to compile the SQL statement. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.

Error Number	Error Message	Suggestion/Interpretation
007784	Bad CONNECTION specification ('` 01'). Possible causes: Syntax error, Dimension name not found, Dimension attributes not found, Dimension name not found in Begin-Select list. DETAILS: ` 02	The OLAP-related members of the named CONNECTION could not be processed, either due to syntax or no such name.
007785	The column specified ('` 01') is ambiguous. It appears more than once in the data.	A column specified in the query appears multiple times in the data. Change the query to avoid that column, or rename the column in the data. ` 01 = The column name
007786	Column ('` 01') not found.	A column specified in the query was not found in the data. ` 01 = The column name
007787	Unsupported datatype ('` 02') found in column ('` 01'). Only Text, Numeric, and Date are currently supported.	A column specified in the query contained values which are not currently supported. Change the query to use a different column. ` 01 = The column name ` 02 = The unsupported datatype
007789	The query is improperly specified. The result would involve a Cartesian Join which is not currently supported. Rewrite the query to involve only columns in the same branch of the tree.	The columns specified in the query are found in different branches of a heirarchical tree in the data. Computing result rows would require a Cartesian Join, which is not currently supported. Rewrite the query to involve only columns on the same branch of the tree.
007790	Must specify PROCEDURE= to use PARAMETERS= in BEGIN-EXECUTE.	Parameters may only be specified for PROCEDURE queries, not for COMMAND or GETDATA queries. Remove the PARAMETERS= line, or specify PROCEDURE= instead.
007791	Unable to locate one or more JAVA classes in the DDO JAR file.	Verify that the original installation files are not corrupted.
007792	Unable to locate one or more JAVA methods in the DDO JAR file.	Verify that the original installation files are not corrupted.
007793	An incompatible version of the DDO JAR file was found.	Verify that the original installation files are not corrupted.

Table 100 Numbered Messages 008000 to 009999

Error Number	Error Message	Suggestion/Interpretation
008000	Delay not appropriate for database columns or literals.	The DELAY argument to the PRINT command can only be used with SQR #variables or \$variables.
008001	The width must also be specified when DELAY is used.	The DELAY argument to the PRINT command requires that the width argument be specified.
008003	The SET-PRINT-DELAY command cannot find a pending PRINT DELAYstatement.	An attempt was made to process an SET-DELAY-PRINT command against an SQR variable for which there was no pending PRINT DELAY statement.
008004	The PRINT DELAY statement did not have an SET-PRINT-DELAY command executed against it.	This PRINT DELAY statement did not have an SET-DELAY-PRINT command executed against it when SQR ended its run.

Error Number	Error Message	Suggestion/Interpretation
008005	The variable (`01`02) was referenced by a PRINT DELAY statement but the SQR program does not contain a matching SET-PRINT-DELAY command.	The referenced variable was used with a PRINT DELAY statement but the SQR program did not contain a SET-PRINT-DELAY command for that variable.
008006	The variable (`01`02) was referenced by a SET-PRINT-DELAY command but the SQR program does not contain a matching PRINT DELAY statement.	The referenced variable was used with a SET-PRINT-DELAY command but the SQR program did not contain a PRINT DELAY statement for that variable.
008101	The specified MODE value ``01' is not legal. Legal values are 'ON' or 'OFF'.	The MODE= qualifier values are ON or OFF. `01 = Invalid value
008102	At least one qualifier must be specified.	Correct the source line.
008200	The specified color (`01') is not defined.	The specified color is not defined in the color map. `01 = Undefined color
008201	Qualifier ``01' has a malformed color reference.	The specified color reference is not properly formed. It can reference a single string literal, column, or variable (i.e. (\$name)) or it can reference three numeric literals, columns, or variables (i.e. (10,20,30)) which represent the Red, Green, and Blue components of the color.
008202	Qualifier ``01' must reference SQR variables only.	The specified color reference is not properly formed. It can reference a single string variable (i.e. (\$name)) or it can reference three numeric variables (i.e. (#R,#G,#B)) which represent the Red, Green, and Blue components of the color.
008203	Invalid RGB value (`01,`02,`03)	The RGB values are out of range. Each value can be from 0 to 255. `01 = Red value `02 = Green value `03 = Blue value
008204	At least one qualifier must be specified.	Correct the source line.
008205	The Declare-Color-Map entry is not properly defined.	The Declare-Color-Map is not properly defined: 1) The color name can only contain characters [0-9 A-Z _ -]. 2) The color name cannot be 'none' 3) The RGB values are not valid (each can be 0 to 255)
008206	Duplicate palette name: `01	Change the name of the palette. `01 = Palette name in question
008207	The name can only contain characters [0-9 A-Z _ -].	Correct the syntax.
008208	The palette cannot have gaps. All colors up to the highest one defined (`01) must be specified.	An SQR palette cannot have gaps. Correct the source code. `01 = Highest color defined for this palette

Error Number	Error Message	Suggestion/Interpretation
008209	The specified palette (`01) does not exist.	Change the name of the palette. `01 = Palette name in question
008300	For font (`01) the specified typeface (`02) is not legal.	Correct the name of the CJK typeface `01 = Font id `02 = Typeface name
008301	For font (`01) the specified character map (`02) is not legal.	Correct the name of the CJK character map `01 = Font id `02 = Character map name
008302	For font (`01) both a typeface and character map must be specified.	If a CJK typeface is specified with a font then a character map must also be specified. `01 = Font id
008304	The current report encoding (`01) requires that a typeface and a character map be specified with each font.	PDF support requires that a proper typeface and character map be associated with a font in order to generate a PDF file with the following output encodings: <ul style="list-style-type: none"> ● Simplified Chinese: EUC-CN, GBK (CP936), UCS-2 ● Traditional Chinese: EUC-TW, BIG5, USC-2 ● Korean: EUC-KR, UHC (Johab), UCS-2 ● Japanese: EUC-JP, Shift-JIS, ISO-2022-JP, UCS-2
008305	For font (`01) the encoding (`02) is incompatible with the report output encoding (`03).	The encoding for the current font is incompatible with the encoding used for the PDF file. `01 = Font id `02 = Font encoding `03 = Report output encoding
008400	The table name (`01) can only contain the following characters [0-9 A-Z _ -].	The specified table name contains invalid characters. `01 = Table name
008401	The table name (`01) is already being used.	The specified table name is already being used. `01 = Table name
008402	The table definition (`01) is already being used.	The specified table definition is already being used. `01 = Table definition name
0008403	The table definition (`01) does not exist.	The specified table definition does not exist. `01 = Table definition name
008404	The table (`01) does not exist.	The specified table does not exist. `01 = Table name
008405	No table definition name was specified.	Correct the source line.
008406	When ACTION=ERASE no other parameters are allowed.	Correct the source line.

Error Number	Error Message	Suggestion/Interpretation
008407	When ACTION=INFO only the ROW and COUNT parameters are allowed. At least one must be specified.	Correct the source line.
008408	When ACTION=GET only the following combination of parameters are allowed: <ul style="list-style-type: none"> ● ARRAY with optional ROW and COUNT ● VALUES with optional ROW 	Correct the source line.
008409	When ACTION=REPLACE only the following combination of parameters are allowed: <ul style="list-style-type: none"> ● ARRAY with optional FIRST, ROW and COUNT ● BLANK with optional ROW and COUNT ● VALUES with optional ROW 	Correct the source line.
008410	When ACTION=INSERT only the following combination of parameters are allowed: <ul style="list-style-type: none"> ● ARRAY with optional MODE, FIRST, ROW and COUNT ● BLANK with optional MODE, ROW and COUNT ● VALUES with optional MODE and ROW 	Correct the source line.
008411	When ACTION=DELETE only the ROW and COUNT parameters are allowed.	Correct the source line.
008412	When ACTION=APPEND only the following combination of parameters are allowed: <ul style="list-style-type: none"> ● ARRAY with optional FIRST, and COUNT ● BLANK with optional COUNT VALUES 	Correct the source line.
008413	Qualifier '`01' requires a numeric variable.	Correct the source line.
008414	The value for '`01' (`02) must be `03 `04 and `05 `06.	Correct the source line. `01 = Qualifer name `02 = Value `03 = Minimum value relation `04 = Minimum value `05 = Maximum value relation `06 = Maximum value
008415	The first argument in '`01' must be the column number (>= 0 and <= `02).	Correct the source line. `01 = Qualifier name `02 = Maximum value
008416	The specified column number (`01) exceeds the number of columns for this table as specified by the COLUMN-COUNT qualifier.	Correct the source line. `01 = Column number
008417	The specified column number (`01) has already been defined by another `02 qualifier.	Correct the source line.

Error Number	Error Message	Suggestion/Interpretation
		`01 = Column number `02 = Qualifier name
008418	Incorrect value (`01) for qualifier ``02'. Valid values are:	Correct the source line. `01 = Value `02 = Qualifier name
008419	Unknown keyword (`01) for qualifier ``02'. Valid keywords are:	Correct the source line. `01 = Value `02 = Qualifier name
008420	Incorrect value (`01) for qualifier ``02'. It must be an integer value > 0.	Correct the source line. `01 = Value `02 = Qualifier name
008421	Incorrect value (`01) for qualifier ``02'. It must be a numeric value > 0.	Correct the source line. `01 = Value `02 = Qualifier name
008422	Incorrect value (`01) for qualifier ``02'. It must be a string literal.	Correct the source line. `01 = Value `02 = Qualifier name
008423	Incorrect value (`01) for qualifier ``02'. It must be either YES or NO.	Correct the source line. `01 = Value `02 = Qualifier name
008424	Incorrect value (`01) for qualifier ``02'. It must be either YES, NO, or the number of lines.	Correct the source line. `01 = Value `02 = Qualifier name
008425	Qualifier ``01' has already been defined.	Correct the source line. `01 = Qualifier name `02 = List Keyword
008426	There are TABLE-FORMAT entries for column numbers which exceed the number of columns as defined by the COLUMN-COUNT qualifier.	Correct the source line.
008427	The value for SIZE= (`01) must be > 0.	Correct the source line.
008428	The value for EXTENT= (`01) must be > 0.	Correct the source line.
008429	Encountered an invalid VALUES= argument. Legal arguments types are \$Variable and #Variable.	Correct the source line.

Error Number	Error Message	Suggestion/Interpretation
008430	Encountered an invalid (^ 01) VALUES= argument. Legal argument types are \$Variable, #Variable, &Variable, 'Literal', Numerics, and the word NULL.	Correct the source line. ^ 01 = List contents
008431	The type of VALUES= parameter (^ 01) is not compatible with the corresponding table column (^ 02) defined as '^ 03'.	The VALUES= parameter type is not compatible with the corresponding TABLE column type. ^ 01 = VALUES parameter ^ 02 = Table column number ^ 03 = Table column type
008432	The number of parameters in the VALUES= list exceeds the number of columns (^ 01) for table (^ 02).	The number of parameters in the VALUES= list exceeds the number of columns defined in the table. ^ 01 = Table column count ^ 02 = Table name
008433	The value for MODE= (^ 01) must be either BEFORE or AFTER.	Correct the source line.
008434	Cannot perform GET because the specified table (^ 01) is empty.	The ACTION=GET option cannot be used against an empty table. ^ 01 = Table name
008435	The number of columns (^ 01) defined for array (^ 02) does not match the number of columns (^ 03) defined for table (^ 04).	The number of columns defined for the specified array must be the same as the number of columns defined for the specified table. ^ 01 = Array column count ^ 02 = Array name ^ 03 = Table column count ^ 04 = Table name
008436	Column type mismatch in column (^ 01): Array (^ 02) is defined as (^ 03) and Table (^ 04) is defined as (^ 05)	The column types for the specified array must be compatible with the columns types for the specified table. ^ 01 = Column number ^ 02 = Array name ^ 03 = Array column type ^ 04 = Table name ^ 05 = Table column type
008437	Cannot perform GET from table (^ 01) for ^ 02 rows because the array (^ 03) can only support (^ 04) rows.	The ACTION=GET option cannot be used to increase the size of array. ^ 01 = Table name ^ 02 = Table rows ^ 03 = Array name ^ 04 = Array size
8600	Binary variables (^ 01) cannot be used with this command.	Correct the source syntax. ^ 01 = Variable name

Error Number	Error Message	Suggestion/Interpretation
8601	The use of an edit mask or the keywords NUMBER, MONEY or DATE is not legal when storing numeric or binary variables.	Correct the source code.
8602	Binary variables (`01) cannot be used in BEGIN-SQL or BEGIN-SELECT paragraphs.	Correct the source code. `01 = Variable name
8603	The use of an edit mask is not legal with binary variables.	Correct the source code.
8604	All variables must be binary if one variable is binary.	Correct the source code.
8605	COMPAR: Unknown relational (binary) operator.	Internal error that should never occur during normal operations. Record the steps leading up to the error and contact customer support.
8606	Incorrect relational operator for binary variables. Only = and != allowed.	Correct the source code.
8607	The ENCODING qualifier is not allowed when the record type is binary.	Correct the source code.
8608	The record type of the file is not binary.	Files must be opened for BINARY access when binary variables are used. Correct the program logic.
8609	The CODE-PRINTER qualifier is required when binary variables are used.	The CODE-PRINTER qualifier is required to PRINT binary variables. Correct the source code.
8610	Function or operator ``01' requires binary argument.	Correct the source line. `01 = Function or operator
8611	Function or operator ``01' does not support binary arguments.	Correct the source line. `01 = Operator
8612	Function or operator ``01' must be a binary or string argument.	Correct the source line. `01 = Function or operator
8613	Qualifier ``01' requires a binary literal.	Correct the source line. `01 = Qualifier name
8614	Qualifier ``01' requires a binary literal, variable, or column.	Correct the source line. `01 = Qualifier name
8615	Qualifier ``01' requires a binary or string literal, variable, or column.	Correct the source line. `01 = Qualifier name
8616	Error creating the image: ``01'. (`02): `03	SQR aborts the program run. `01 = Image file name `02 = System error code `03 = System error message
8617	Error closing the image: ``01'. (`02): `03	SQR aborts the program run. `01 = Image file name

Error Number	Error Message	Suggestion/Interpretation
		<p>`02 = System error code</p> <p>`03 = System error message</p>
8618	Error writing the image: ``01'. (`02): `03	<p>SQR aborts the program run.</p> <p>`01 = Image file name</p> <p>`02 = System error code</p> <p>`03 = System error message</p>
8619	Error opening the image: ``01'. (`02): `03	<p>SQR aborts the program run.</p> <p>`01 = Image file name</p> <p>`02 = System error code</p> <p>`03 = System error message</p>
8620	Error reading the image: ``01'. (`02): `03	<p>SQR aborts the program run.</p> <p>`01 = Image file name</p> <p>`02 = System error code</p> <p>`03 = System error message</p>
8621	The SQR compression logic failed, reason `01.	<p>The SQR compression logic failed while compressing an image.</p> <p>`01 = Reason code</p>
8622	The SQR decompression logic failed, reason `01.	<p>The SQR decompression logic failed while decompressing an image.</p> <p>`01 = Reason code</p>
8623	The embedded BMP image is not valid, reason `01.	<p>(Windows) The embedded bitmap image is not valid.</p> <p>`01 = Reason code</p>
009999	The printer (`01) specified with the -Printer:WP command line flag is invalid.	<p>(Windows) The specified printer is not valid.</p>



Production Reporting Language Quick Reference

Table 101 Production Reporting Commands

COMMAND	SYNTAX
ADD	<code>{src_num_lit _var _col} TO dst_num_var [ROUND=nn]</code>
ALTER-COLOR-MAP	<code>NAME={color_name_lit _var _col}</code> <code>VALUE=({color_name_lit _var _col} {rgb})</code>
ALTER-CONNECTION	<code>NAME=connection_name</code> <code>[DSN=or [USER=or [PASSWORD={ug_txt_lit _var}]</code> <code>[PARAMETERS=keyword_str=attr_str;</code> <code>[,keyword_str=attr_str;...]]</code> <code>[NO-DUPLICATE=TRUE FALSE]</code> <code>SET-GENERATIONS=({dimension1, hierarchy1}</code> <code>[,dimensioni, hierarchyi] ...)</code> <code>SET-LEVELS=({dimension1, level1} [,dimensioni, leveli] ...)</code> <code>SET-MEMBERS=({dimension1, level1} [,dimensioni, leveli] ...)</code>
ALTER-LOCALE	<code>[LOCALE=or [NUMBER-EDIT-MASK=or [MONEY-EDIT-MASK=</code> <code>or [DATE-EDIT-MASK=or [INPUT-DATE-EDIT-MASK=or [MONEY-SIGN=</code> <code>{txt_lit _var} DEFAULT SYSTEM]]</code> <code>[MONEY-SIGN-LOCATION={txt_var DEFAULT SYSTEM LEFT RIGHT}]</code> <code>[THOUSAND-SEPARATOR=or [DECIMAL-SEPARATOR= or</code> <code>[DATE-SEPARATOR=or [TIME-SEPARATOR=</code> <code>or [EDIT-OPTION-NA, AM, PM, BC, AD=</code> <code>{txt_lit _var} DEFAULT SYSTEM]]</code> <code>[DAY-OF-WEEK-CASE= or [MONTHS-CASE=</code> <code>{txt_var DEFAULT SYSTEM UPPER LOWER EDIT NO-CHANGE}]</code> <code>[DAY-OF-WEEK-FULL= or [DAY-OF-WEEK-SHORT=</code> <code>({txt_lit1 _var1}...{txt_lit7 _var7})]</code> <code>[MONTHS-FULL= or [MONTHS-SHORT=</code> <code>({txt_lit1 _var1}...{txt_lit12 _var12})]</code>
ALTER-PRINTER	<code>[POINT-SIZE={point_size_num_lit _var}]</code> <code>[FONT-TYPE={font_type txt_var}]</code> <code>[SYMBOL-SET={symbol_set_id txt_var}]</code> <code>[FONT={font_int_lit _var}]</code> <code>[PITCH={pitch_num_lit _var}]</code>
ALTER-REPORT	<code>[HEADING={heading_name_txt_lit _var _col}</code> <code>[HEADING-SIZE={heading_size_int_lit _var _col}</code> <code>[FOOTING={footing_name_txt_lit _var _col}</code> <code>[FOOTING-SIZE={footing_size_int_lit _var _col}]</code> <code>[PDF-APPEARANCE=(appearance_lit _var _col)]</code> <code>[PDF-INFORMATION=(information_lit _var _col, value_lit _var _col</code>

COMMAND	SYNTAX
	[, <i>information_lit</i> <i>_var</i> <i>_col</i> , <i>value_lit</i> <i>_var</i> <i>_col</i>]...] [PDF-OPEN-ACTION=(<i>openaction_lit</i> <i>_var</i> <i>_col</i> , [, <i>name_lit</i> <i>_var</i> <i>_col</i> , <i>value_lit</i> <i>_var</i> <i>_col</i>]...)] [PDF-PAGE-TRANSITION=(<i>transition_lit</i> <i>_var</i> <i>_col</i> , <i>duration_lit</i> <i>_var</i> <i>_col</i>)] [PDF-SECURITY=(<i>security_lit</i> <i>_var</i> <i>_col</i> , <i>value_lit</i> <i>_var</i> <i>_col</i> [, <i>security_lit</i> <i>_var</i> <i>_col</i> , <i>value_lit</i> <i>_var</i> <i>_col</i>]...)] [PDF-VIEWER-PREFERENCE=(<i>preference_lit</i> <i>_var</i> <i>_col</i> , <i>value_lit</i> <i>_var</i> <i>_col</i> [, <i>preference_lit</i> <i>_var</i> <i>_col</i> , <i>value_lit</i> <i>_var</i> <i>_col</i>]...)]
ALTER-TABLE	NAME= <i>table_name_var</i> <i>_lit</i> <i>_col</i> ACTION= <i>action_lit</i> [COUNT= <i>count_var</i> <i>_lit</i> <i>_col</i>] [ROW= <i>row_var</i> <i>_lit</i> <i>_col</i>] [ATTRIBUTES=(<i>{keyword1}</i>), <i>{value1}</i> , ..., <i>{keywordn}</i> , <i>{valuen}</i>)]
ARRAY-ADD	{ <i>src_num_lit</i> <i>_var</i> <i>_col</i> }...TO <i>dst_array_name</i> (<i>element_lit</i> <i>_var</i> <i>_col</i>) [<i>field</i> [(<i>occurs_lit</i> <i>_var</i> <i>_col</i>)]]...
ARRAY-DIVIDE	{ <i>src_num_lit</i> <i>_var</i> <i>_col</i> }...INTO <i>dst_array_name</i> (<i>element_int_lit</i> <i>_var</i> <i>_col</i>) [<i>field</i> [(<i>occurs_lit</i> <i>_var</i> <i>_col</i>)]]...
ARRAY-MULTIPLY	{ <i>src_num_lit</i> <i>_var</i> <i>_col</i> }...TIMES <i>dst_array_name</i> (<i>element_int_lit</i> <i>_var</i> <i>_col</i>) [<i>field</i> [(<i>occurs_lit</i> <i>_var</i> <i>_col</i>)]]...
ARRAY-SUBTRACT	{ <i>src_num_lit</i> <i>_var</i> <i>_col</i> }...FROM <i>dst_array_name</i> (<i>element_int_lit</i> <i>_var</i> <i>_col</i>) [<i>field</i> [(<i>occurs_lit</i> <i>_var</i> <i>_col</i>)]]...
ASK	<i>substitution_variable</i> [<i>prompt</i>]
BEGIN-DOCUMENT	{ <i>position</i> } END-DOCUMENT
BEGIN-EXECUTE	[CONNECTION= <i>uq_txt_lit</i>] [ON-ERROR= <i>sqr_procedure</i> [(<i>arg1</i> [, <i>argi</i>]...)]] [RSV= <i>num_var</i>] [STATUS= <i>list_var</i> <i>num_var</i> <i>txt_var</i>] [PROPERTIES=(<i>{key_txt_lit</i> <i>_var</i>)={ <i>{value_txt_lit</i> <i>_var</i> <i>_col</i> } <i>{num_lit</i> <i>_var</i> <i>_col</i> },...)] [SCHEMA= <i>{txt_lit</i> <i>_var</i>] [PROCEDURE= <i>{txt_lit</i> <i>_var</i>] [PARAMETERS=(<i>{arg1</i> [IN INOUT]} NULL) [[, <i>argi</i> [IN INOUT]] NULL] ...)] (or) COMMAND= <i>{txt_lit</i> <i>_var</i>] (or) GETDATA= <i>{txt_lit</i> <i>_var</i>] [BEGIN-SELECT [BEFORE= <i>sqr_procedure</i> [(<i>arg1</i> [, <i>argi</i>]...)]] [AFTER= <i>sqr_procedure</i> [(<i>arg1</i> [, <i>argi</i>]...)]] <i>col-name</i> TYPE=CHAR TEXT NUMBER DATE [<i>edit-mask</i>] [on-break]... [<i>{FROM ROWSETS=(<i>{m</i> <i>m-n</i> <i>m-</i> <i>-n</i>)</i> [, ...]} {ALL}})]

COMMAND	SYNTAX
	{FROM PARAMETER={txt_lit _var}} {FROM {table_name}}} END-SELECT END-EXECUTE
BEGIN-FOOTING	footing_lines_int_lit [FOR-REPORTS=(report_name1[, report_namei]...)] [FOR-TOCS=(toc_name1[, toc_namei]...)] [NAME={footing_name}] END-FOOTING
BEGIN-HEADING	heading_lines_int_lit [FOR-REPORTS=(report_name1[, report_namei]...)] [FOR-TOCS=(toc_name1[, toc_namei]...)] [NAME={heading_name}] END-HEADING
BEGIN-PROCEDURE	procedure_name [LOCAL (arg1 [, argi]...)] END-PROCEDURE
BEGIN-PROGRAM	END-PROGRAM
BEGIN-SELECT	[DISTINCT] [-Cnn] [-Bnn] [-XP] [-NR] [-SORTnn] [-LOCK{RR CS RO RL XX}] [-DBdatabase] [-DBconnectionstring] [LOOPS=nn] [ON-ERROR=procedure(arg1[, argi]...)] {column} [&synonym] {expression &synonym} {[\$columnname] &synonym=(char number date)} [sqr_commands] FROM {table,... [table:\$tablename]} [additional SQL] [\$variable] END-SELECT
BEGIN-SETUP	END-SETUP
BEGIN-SQL	[-Cnn] [-XP] [-NR] [-SORTnn] [-LOCK{RR CS RO RL XX}] [-DBdatabase] [-DBconnectionstring] [ON-ERROR=procedure(arg1[, argi]...)] (non-SETUP) [ON-ERROR={STOP WARN SKIP}] (inSETUP) END-SQL
BREAK	BREAK
CALL	subroutine USING {src_txt_lit _var _col} {src_num_lit _var _col} {dst_txt_var _num_var} [param]
CALL SYSTEM	Same as CALL
CLEAR-ARRAY	NAME=array_name
CLOSE	{filenum_lit _var _col}

COMMAND	SYNTAX
CLOSE-RS	NAME=row_set_name_var _lit _col
COLUMNS	{int_lit _var _col}[int_lit _var _col]...
COMMIT	COMMIT
CONCAT	{src_any_lit _var _col} WITH dst_txt_var[[:\$]edit_mask]
CONNECT	{txt_lit _var _col}[ON-ERROR=procedure[(arg1 [, argi]...)]]
CREATE-ARRAY	NAME=array_name SIZE=nn [EXTENT=nn] {FIELD=name:type[:occurs] [={init_value_txt_lit _num_lit _binary_lit}]}...
CREATE-COLOR-PALETTE	NAME={palette_name_txt_lit} COLOR_1={rgb} COLOR_2={rgb} [COLOR_n]={rgb}
CREATE-LIST	NAME=list_name_txt_lit _var _col LIST=(value_lit var _col (r,g,b)...))
CREATE-TABLE	NAME=table_name_var _lit _col USING=table_template_var _lit _col [COLUMN-COUNT=number_of_columns_var _lit _col] [COLUMN-ATTRIBUTES={({column-number},{keyword1},{value1}, ..., {keywordn},{valuen})}] [ROW-ATTRIBUTES={({keyword1},{value1}, ..., {keywordn},{valuen})}] [TABLE-ATTRIBUTES={({keyword1},{value1}, ..., {keywordn},{valuen})}]
#DEBUG	[x...] <i>sqr_command</i>
DECLARE-CHART	<i>chart_name</i> [Type=chart_type_lit] [CHART-SIZE=(chart_width_int_lit,chart_depth_int_lit)] [TITLE=title_txt_lit] [SUB-TITLE=subtitle_txt_lit] [FILL=fill_lit] [3D-EFFECTS=3d_effects_lit] [BORDER=border_lit] [COLOR-PALETTE=color_palette_lit] [POINT-MARKERS=point_markers_lit] [ATTRIBUTES={selector_lit LIST:{selector_list_name_lit (selector_lit,...)}, {decl_key_lit,{decl_value_lit LIST:{decl_val_list_name_lit (decl_val_lit,...)}} PALETTE:{color_palette_lit}}},...}}] [DATA-ARRAY=array_name] [DATA-ARRAY-ROW-COUNT=row_count_num_lit] [DATA-ARRAY-COLUMN-COUNT= column_count_num_lit] [DATA-ARRAY-COLUMN-LABELS={NONE array_name (txt_lit,...)}] [DATA-LABELS=data_labels_lit] [FOOTER-TEXT=NONE text_lit] [SUB-FOOTER-TEXT=NONE text_lit]

COMMAND	SYNTAX
	<pre> [ITEM-COLOR=(<i>chart_item_keyword_lit</i>, <i>color_value_lit</i> (r,g,b)] [ITEM-SIZE={<i>item_size_keyword_lit</i>, <i>item_size_num_lit</i>}] [LEGEND=<i>legend_lit</i>] [LEGEND-TITLE=<i>legend_title_txt_lit</i>] [LEGEND-PLACEMENT=<i>legend_placement_lit</i>] [LEGEND-PRESENTATION=<i>legend_presentation_lit</i>] [PIE-SEGMENT-QUANTITY-DISPLAY=<i>pie_segment_quantity_display_lit</i>] [PIE-SEGMENT-PERCENT-DISPLAY=<i>pie_segment_percent_display_lit</i>] [PIE-SEGMENT-EXPLODE=<i>pie_segment_explode_lit</i>] [X-AXIS-GRID=<i>x_axis_grid_lit</i>] [X-AXIS-LABEL=<i>x_axis_label_txt_lit</i>] [X-AXIS-MIN-VALUE={<i>x_axis_min_value_lit</i> <i>_num_lit</i>}] [X-AXIS-MAX-VALUE={<i>x_axis_max_value_lit</i> <i>_num_lit</i>}] [X-AXIS-MAJOR-INCREMENT={<i>x_axis_major_increment_lit</i> <i>_num_lit</i>}] [X-AXIS-MINOR-INCREMENT=<i>x_axis_minor_increment_num_lit</i>] [X-AXIS-MAJOR-TICK-MARKS=<i>x_axis_major_tick_marks_lit</i>] [X-AXIS-MINOR-TICK-MARKS=<i>x_axis_minor_tick_marks_lit</i>] [X-AXIS-TICK-MARK-PLACEMENT=<i>x_axis_tick_mark_placement_lit</i>] [X-AXIS-ROTATE=<i>x_rotate_num_lit</i>] [X-AXIS-SCALE=<i>x_axis_scale_lit</i>] [Y-AXIS-GRID=<i>y_axis_grid_lit</i>] [Y-AXIS-LABEL=<i>y_axis_label_lit</i>] [Y-AXIS-MASK=<i>mask_txt_lit</i>] [Y-AXIS-MIN-VALUE={<i>y_axis_min_value_lit</i> <i>_num_lit</i>}] [Y-AXIS-MAX-VALUE={<i>y_axis_max_value_lit</i> <i>_num_lit</i>}] [Y-AXIS-MAJOR-INCREMENT={<i>y_axis_major_increment_lit</i> <i>_num_lit</i>}] [Y-AXIS-MINOR-INCREMENT=<i>y_axis_minor_increment_num_lit</i>] [Y-AXIS-MAJOR-TICK-MARKS=<i>y_axis_major_tick_marks_lit</i>] [Y-AXIS-MINOR-TICK-MARKS=<i>y_axis_minor_tick_marks_lit</i>] [Y-AXIS-TICK-MARK-PLACEMENT=<i>y_axis_tick_mark_placement_lit</i>] [Y-AXIS-SCALE=<i>y_axis_scale_lit</i>] [Y2-AXIS-LABEL=<i>y2_axis_label_lit</i>] [Y2-AXIS-MASK=<i>mask_txt_lit</i>] [Y2-AXIS-MIN-VALUE={<i>y2_axis_min_value_lit</i> <i>_num_lit</i>}] [Y2-AXIS-MAX-VALUE={<i>y2_axis_max_value_lit</i> <i>_num_lit</i>}] [Y2-AXIS-MAJOR-INCREMENT={<i>y2_axis_major_increment_lit</i> <i>_num_lit</i>}] [Y2-AXIS-MINOR-INCREMENT=<i>y2_axis_minor_increment_num_lit</i>] [Y2-AXIS-MAJOR-TICK-MARKS=<i>y2_axis_major_tick_marks_lit</i>] [Y2-AXIS-MINOR-TICK-MARKS=<i>y2_axis_minor_tick_marks_lit</i>] [Y2-AXIS-SCALE=<i>y2_axis_scale_lit</i>] [Y2-AXIS-COLOR-PALETTE=<i>color_palette_lit</i>] [Y2-DATA-ARRAY=<i>array_name</i>] [Y2-DATA-ARRAY-ROW-COUNT=<i>row_count_num_lit</i>] [Y2-DATA-ARRAY-COLUMN-COUNT=<i>column_count_num_lit</i>] [Y2-DATA-ARRAY-COLUMN-LABELS={NONE <i>array_name</i> (<i>txt_lit</i>,...)}] [Y2-TYPE=<i>chart_type_lit</i>] END-DECLARE </pre>
DECLARE-COLOR-MAP	<pre> <i>color_name</i>=(<i>{rgb}</i>) <i>color_name</i>=(<i>{rgb}</i>) . . . END-DECLARE </pre>

COMMAND	SYNTAX
DECLARE-CONNECTION	<pre> <i>connection_name</i> DSN={<i>uq_txt_lit</i>} [USER= or [PASSWORD={<i>uq_txt_lit</i>}] [PARAMETERS=<i>keyword_str=attr_str</i>; [<i>keyword_str=attr_str</i>; ...]] [NO-DUPLICATE=TRUE FALSE] SET-GENERATIONS=({<i>dimension1, hierarchy1</i>}[,<i>dimensioni, hierarchyi</i>] ...) SET-LEVELS=({<i>dimension1, level1</i>}[,<i>dimensioni, leveli</i>] ...) SET-MEMBERS=({<i>dimension1, level1</i>}[,<i>dimensioni, leveli</i>] ...) END-DECLARE </pre>
DECLARE-IMAGE	<pre> <i>image_name</i> [TYPE=<i>image_type_lit</i>] [IMAGE-SIZE=(<i>width_num_lit,height_num_lit</i>)] [SOURCE=<i>file_name_lit</i>] [[FOR-PRINTER=({POSTSCRIPT HPLASERJET HTML PDF WINDOWS PS HP HT PD WP}, <i>image_type_lit,file_name_lit</i>) . . .] END-DECLARE </pre>
DECLARE-LAYOUT	<pre> DECLARE-LAYOUT <i>layout_name</i> [PAPER-SIZE=({<i>paper_width_num_lit[uom],paper_depth_num_lit[uom]</i>}[<i>paper_name</i>])] [FORMFEED=<i>form_feed_lit</i>] [ORIENTATION=<i>orientation_lit</i>] [LEFT-MARGIN=<i>left_margin_num_lit[uom]</i>] [TOP-MARGIN=<i>top_margin_num_lit[uom]</i>] [RIGHT-MARGIN=<i>right_margin_num_lit[uom]</i> LINE- WIDTH=<i>line_width_num_lit[uom]</i> MAX-COLUMNS=<i>columns_int_lit</i>] [BOTTOM-MARGIN= <i>bottom_margin_num_lit[uom]</i> PAGE- DEPTH=<i>page_depth_num_lit[uom]</i> MAX-LINES=<i>lines_int_lit</i>] [CHAR-WIDTH=<i>char_width_num_lit[uom]</i>] [LINE-HEIGHT=<i>line_height_num_lit[uom]</i>] END-DECLARE </pre>
DECLARE-PRINTER	<pre> <i>printer_name</i> [[TYPE=<i>printer_type_lit</i>]FOR-REPORTS=(<i>report_name1</i> [,<i>report_namei</i>]...)] [INIT-STRING=<i>initialization_string_txt_lit</i>] [RESET-STRING=<i>reset_string_txt_lit</i>] [COLOR=<i>color_lit</i>] [POINT-SIZE=<i>point_size_num_lit</i>] [FONT-TYPE=<i>font_type_int_lit</i>] [SYMBOL-SET=<i>symbol_set_id_lit</i>] [STARTUP-FILE=<i>file_name_txt_lit</i>] [PITCH=<i>pitch_num_lit</i>] [FONT=<i>font_int_lit</i>] [BEFORE-BOLD=<i>before_bold_string_txt_lit</i>] [AFTER-BOLD=<i>after_bold_string_txt_lit</i>] END-DECLARE </pre>
DECLARE-PROCEDURE	<pre> [FOR-REPORTS=(<i>report_name1</i>[,<i>report_namei</i>]...)] [BEFORE-REPORT=<i>procedure_name</i>[(<i>arg1</i>[,<i>argi</i>]...)]] </pre>

COMMAND	SYNTAX
	<pre>[AFTER-REPORT=<i>procedure_name</i>[(<i>arg1</i>[,<i>argi</i>]...)]] [BEFORE-PAGE=<i>procedure_name</i>[(<i>arg1</i>[,<i>argi</i>]...)]] [AFTER-PAGE=<i>procedure_name</i>[(<i>arg1</i>[,<i>argi</i>]...)]] END-DECLARE</pre>
DECLARE-REPORT	<pre><i>report_name</i> [TOC=<i>toc_name</i>] [LAYOUT=<i>layout_name</i>] [PRINTER-TYPE=<i>printer_type</i>] END-DECLARE</pre>
DECLARE-TABLE	<pre>NAME=<i>table_template_name</i> COLUMN-COUNT=<i>number_of_columns</i> [COLUMN-ATTRIBUTES=(<i>{column number}</i>,<i>{keyword1}</i>,<i>{value1}</i>, ..., <i>{keywordn}</i>,<i>{valuen}</i>)] [ROW-ATTRIBUTES=(<i>{keyword1}</i>,<i>{value1}</i>, ..., <i>{keywordn}</i>,<i>{valuen}</i>)] [TABLE-ATTRIBUTES=(<i>{keyword1}</i>,<i>{value1}</i>, ..., <i>{keywordn}</i>,<i>{valuen}</i>)]</pre>
DECLARE-TOC	<pre><i>toc_name</i> [FOR-REPORTS=(<i>report_name1</i>[,<i>report_namei</i>]...)] [DOT-LEADER=YES NO] [INDENTATION=<i>position_count_num_lit</i>] [BEFORE-TOC=<i>procedure_name</i>[(<i>arg1</i>[,<i>argi</i>]...)]] [AFTER-TOC=<i>procedure_name</i>[(<i>arg1</i>[,<i>argi</i>]...)]] [BEFORE-PAGE=<i>procedure_name</i>[(<i>arg1</i>[,<i>argi</i>]...)]] [AFTER-PAGE=<i>procedure_name</i>[(<i>arg1</i>[,<i>argi</i>]...)]] [ENTRY=<i>procedure-name</i> [(<i>argi</i> [,<i>argi</i>] ...)]] END-DECLARE</pre>
DECLARE-VARIABLE	<pre>[DEFAULT-NUMERIC= {DECIMAL[(<i>prec_lit</i>)] FLOAT INTEGER}] [DECIMAL[(<i>prec_lit</i>)]<i>num_var</i>[(<i>prec_lit</i>)] [<i>num_var</i>[(<i>prec_lit</i>)]]...] [FLOAT <i>num_var</i>[<i>num_var</i>]...] [DATE <i>date_var</i>[<i>date_var</i>]...] [INTEGER <i>num_var</i>[<i>num_var</i>]...] [TEXT <i>string_var</i>[<i>string_var</i>]...] [BINARY <i>binary_var</i>[<i>binary_var</i>]...] END-DECLARE</pre>
#DEFINE	<pre><i>substitution_variable value</i></pre>
DISPLAY	<pre>{<i>any_lit</i> <i>_var</i> <i>_col</i>} [[:\$]<i>edit_mask</i> NUMBER MONEY DATE] [NOLINE]</pre>
DIVIDE	<pre>{<i>src_num_lit</i> <i>_var</i> <i>_col</i>} INTO <i>dst_num_var</i> [ON-ERROR={HIGH ZERO}] [ROUND=<i>nn</i>]</pre>
DO	<pre><i>procedure_name</i>[(<i>arg1</i>[, <i>argi</i>]...)]</pre>
DRAW	<pre>DRAW {<i>position</i>} TYPE={<i>type_lit</i> <i>_var</i> <i>_col</i>} [HEIGHT={<i>height_lit</i> <i>_var</i> <i>_col</i>}] [WIDTH={<i>width_lit</i> <i>_var</i> <i>_col</i>}] [RULE={<i>rule_lit</i> <i>_var</i> <i>_col</i>}] [FILL COLOR=(<i>{color_name_lit</i> <i>_var</i> <i>_col</i>} <i>{rgb}</i>)] [LINE COLOR= <i>{color_name_lit</i> <i>_var</i> <i>_col</i>} <i>{rgb}</i>)]</pre>

COMMAND	SYNTAX
	[CAP={ <i>cap_lit</i> <i>_var</i> <i>_col</i> }] [LINE-STYLE={ <i>line_style_lit</i> <i>_var</i> <i>_col</i> }] [END-POINT=(<i>row_lit</i> <i>_var</i> <i>_col</i> , <i>column_lit</i> <i>_var</i> <i>_col</i>)]
DUMP-TABLE	NAME= <i>table_name_var</i> <i>_lit</i> <i>_col</i> [CONTINUATION= <i>continuation_var</i> <i>_lit</i> <i>_col</i>]
#ELSE	#ELSE
ELSE	ELSE
ENCODE	<i>src_code_string_lit</i> INTO <i>dst_txt_var</i>
END-DECLARE	END-DECLARE
END-DOCUMENT	END-DOCUMENT
END-EVALUATE	END-EVALUATE
END-EXECUTE	END-EXECUTE
END-FOOTING	END-FOOTING
END-HEADING	END-HEADING
#END-IF	#END-IF
#ENDIF	#ENDIF
END-IF	END-IF
END-PROCEDURE	END-PROCEDURE
END-PROGRAM	END-PROGRAM
END-SELECT	END-SELECT
END-SETUP	END-SETUP
END-SQL	END-SQL
END-WHILE	END-WHILE
EVALUATE	{ <i>any_lit</i> <i>_var</i> <i>_col</i> } WHEN <i>comparison_operator</i> { <i>any_lit</i> <i>_var</i> <i>_col</i> } <i>sqr_commands...</i> [BREAK] [WHEN-OTHER <i>sqr_commands...</i> [BREAK]] END-EVALUATE
EXECUTE	[-XC] [ON-ERROR= <i>procedure</i> [(<i>arg1</i> [, <i>argi</i>] ...)]] [DO= <i>procedure</i> [(<i>arg1</i> [, <i>argi</i>] ...)]] { [@# <i>status_var</i> =] <i>stored_procedure_name</i> } { [@\$ <i>return_var</i> =] <i>stored_procedure_name</i> }

COMMAND	SYNTAX
	[[@param=]{any_col _var _lit}[OUTPUT][, ...]] [INTO any_coldata_type[(length_int_lit)] [, ...]][WITH RECOMPILE]
EXIT-SELECT	EXIT-SELECT
EXTRACT	{dst_txt_var date_var} FROM {{src_txt_lit _var _col} {src_date_var _col}} {start_num_lit _var}{length_num_lit _var}
FILL-TABLE	NAME=table_name_var _lit _col VALUE=value_var _lit _col LOCATION=(row_var _lit _col, column_var _lit[,length_var _lit]) [ATTRIBUTES=({keyword1},{value1}, ..., {keywordn},{valuen})]
FIND	{{obj_txt_lit _var _col} {date_var _col}} IN {{src_txt_var _col} {date_var _col}} {start_int_lit _var} dst_location_int_var
GET	dst_any_var...FROM src_array_name(element)[field[(occurs)]]...
GET-COLOR	[PRINT-TEXT-FOREGROUND=({color_name_var})] [PRINT-TEXT-BACKGROUND=({color_name_var})] [PRINT-PAGE-BACKGROUND=({color_name_var})] [LINE-COLOR=({color_name_var})] [FILL-COLOR=({color_name_var})]
GOTO	label
#IF	{txt_lit num_lit}comparison_operator {txt_lit num_lit}
IF	logical_expression sqr_commands... [ELSE sqr_commands...] END-IF
#IFDEF #IFNDEF	substitution_variable
#INCLUDE	filename_lit
INPUT	input_var[MAXLEN=nn][prompt] [TYPE={CHAR TEXT NUMBER INTEGER DATE}] [STATUS=num_var][NOPROMPT][BATCH-MODE] [FORMAT={txt_lit _var _col}]
LAST-PAGE	position[pre_txt_lit[post_txt_lit]]
LET	dst_var=expression
LOAD-LOOKUP	In the SETUP section: NAME=lookup_table_name TABLE=database_table_name KEY=key_column_name RETURN_VALUE=return_column_name

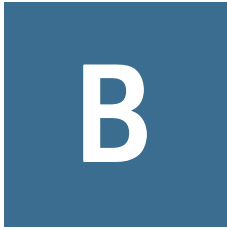
COMMAND	SYNTAX
	<pre> [ROWS=<i>initial_row_estimate_int_lit</i>] [EXTENT=<i>size_to_grow_by_int_lit</i>] [WHERE=<i>where_clause_txt_lit</i>] [<i>SORT=sort_mode</i>] [QUIET] [SCHEMA=<i>schema_txt_lit</i>] [PROCEDURE=<i>proc_txt_lit</i>] [PARAMETERS=(<i>{arg1 [IN INOUT]}</i> NULL) [<i>[,argi [IN INOUT]]</i>] NULL] ...)] (or) COMMAND=<i>command_txt_lit</i> (or) GETDATA=<i>getdata_txt_lit</i>] [<i>{FROM-ROWSETS=(<i>{m m-n m- -n} [,...]</i>)} {ALL}}</i>}] {FROM-PARAMETER=<i>parameter_txt_lit</i>}] In the body of the report: NAME=<i>lookup_table_name</i> TABLE=<i>database_table_name</i> KEY=<i>key_column_name</i> RETURN_VALUE=<i>return_column_name</i> [ROWS=<i>initial_row_estimate_lit _var _col</i>] [EXTENT=<i>size_to_grow_by_lit _var _col</i>] [WHERE=<i>where_clause_txt_lit _var _col</i>] [<i>SORT=sort_mode</i>] [QUIET] [SCHEMA=<i>{txt_lit _var}</i>] [PROCEDURE=<i>{txt_lit _var}</i>] [PARAMETERS=(<i>{arg1 [IN INOUT]}</i> NULL) [<i>[,argi [IN INOUT]]</i>] NULL] ...)] (or) COMMAND=<i>{txt_lit _var}</i> (or) GETDATA=<i>{txt_lit _var}</i>] [<i>{FROM-ROWSETS=(<i>{m m-n m- -n} [,...]</i>)} {ALL}}</i>}] {FROM-PARAMETER=<i>{txt_lit _var}</i>}] </pre>
LOOKUP	<i>lookup_table_name {key_any_lit _var _col} {ret_txt_var _date _var}</i>
LOWERCASE	<i>txt_var</i>
MBTOSBS	<i>{txt_var}</i>
MOVE	<i>{src_any_lit _var _col} TO dst_any_var</i> <i>[[:\$] format_mask NUMBER MONEY DATE]</i>
MULTIPLY	<i>{src_num_lit _var _col} TIMES dst_num_var</i> <i>[ROUND=nn]</i>
NEW-PAGE	<i>[erase_from_line_num_lit _var _col]</i>
NEW-REPORT	<i>{report_filename_txt_lit _var _col}</i>

COMMAND	SYNTAX
NEXT-COLUMN	[AT-END={NEWLINE NEWPAGE}] [GOTO-TOP={num_lit _var _col}] [ERASE-PAGE={num_lit _var _col}]
NEXT-LISTING	[NO-ADVANCE [SKIPLINES={num_lit _var _col}] [NEED={num_lit _var _col}]
OPEN	{filename_lit _var _col} AS {filenum_num_lit _var _col} {FOR-READING FOR-WRITING FOR-APPEND} {RECORD=length_num_lit[:FIXED :FIXED_NOLF :VARY :BINARY]}] [STATUS=num_var]] [ENCODING={_var _col ASCII ANSI SJIS JEUC EBCDIC EBCDIK290 EBCDIK1027 UCS-2 UTF-8 others... }]
OPEN-RS	OPEN-RS NAME=row_set_name_var _lit _col FILENAME=file_name_var _lit _col COLUMN=({name_var _lit _col},{type_var _lit _col})
PAGE-NUMBER	position [pre_txt_lit[post_txt_lit]]
POSITION	position [@document_marker[COLUMNS{num_lit _var _col} [num_lit _var _col]...]]
PRINT	{any_lit _var _col} position[format_command [format_cmd_params]...]....
PRINT-BAR-CODE	position {TYPE={bar_code_type_num_lit _var _col}} {HEIGHT={bar_code_height_num_lit _var _col}} {TEXT={bar_code_txt_lit _var _col}} [CAPTION={bar_code_caption_txt_lit _var _col}] [CHECKSUM={bar_code_checksum_txt_lit _var _col}]
PRINT-CHART	PRINT-CHART[chart_name]position [TYPE={chart_type_txt_lit _var _col}] [CHART-SIZE=(chart_width_num_lit _var _col, chart_depth_num_lit _var _col)] [TITLE={title_txt_lit _var _col}] [SUB-TITLE={subtitle_txt_lit _var _col}] [FILL={fill_txt_lit _var _col}] [3D-EFFECTS={3d_effects_txt_lit _var _col}] [BORDER={border_txt_lit _var _col}] [COLOR-PALETTE=color_palette_lit _var _col] [POINT-MARKERS={point_markers_txt_lit _var _col}] [ATTRIBUTES={selector_lit _var _col LIST:{selector_list_name_lit _var _col (selector_lit _var _col,...)},{decl_key_lit _var _col,{decl_value_lit _var _col LIST:{decl_val_list_name_lit _var _col (decl_val_lit _var

COMMAND	SYNTAX
	<pre> _col, ...)} PALETTE={color_palette_lit _var _col}}},...}}] [DATA-ARRAY=array_name] [DATA-ARRAY-ROW-COUNT={x_num_lit _var _col}] [DATA-ARRAY-COLUMN-COUNT={x_num_lit _var _col}] [DATA-ARRAY-COLUMN-LABELS={NONE array_name ({txt_lit var _col},...)}] [DATA-LABELS={data_labels_txt_lit _var _col}] [FOOTER-TEXT=NONE text_lit _var _lit] [SUB-FOOTER-TEXT=NONE text_lit _var _col] [ITEM-COLOR=(item_color_keyword _lit _var _col,{color_txt_lit_var _col} (r,g,b))] [ITEM-SIZE=(item_size_keyword_lit _var _col,item_size_num_lit _var _col)] [LEGEND={legend_txt_lit _var _col}] [LEGEND-TITLE={legend_title_txt_lit _var _col}] [LEGEND-PLACEMENT={legend_placement_txt_lit _var _col}] [LEGEND-PRESENTATION={legend_presentation_txt_lit _var _col}] [PIE-SEGMENT-QUANTITY-DISPLAY={pie_segment_quantity_display_txt_lit _var _col}] [PIE-SEGMENT-PERCENT-DISPLAY={pie_segment_percent_display_txt_lit _var _col}] [PIE-SEGMENT-EXPLODE={pie_segment_explode_txt_lit _var _col}] [X-AXIS-GRID={x_axis_grid_txt_lit _var _col}] [X-AXIS-LABEL={x_axis_label_txt_lit _var _col}] [X-AXIS-MIN-VALUE={x_axis_min_value_num_lit _var _col}] [X-AXIS-MAX-VALUE={x_axis_max_value_num_lit _var _col}] [X-AXIS-MAJOR-INCREMENT={x_axis_major_increment_num_lit _var _col}] [X-AXIS-MINOR-INCREMENT={x_axis_minor_increment_num_lit _var _col}] [X-AXIS-MAJOR-TICK-MARKS={x_axis_major_tick_marks_txt_lit _var _col}] [X-AXIS-MINOR-TICK-MARKS={x_axis_minor_tick_marks_txt_lit _var _col}] [X-AXIS-TICK-MARK-PLACEMENT={x_axis_tick_mark_placement_txt_lit _var _col}] [X-AXIS-ROTATE={x_num_lit _var _col}] [X-AXIS-SCALE={x_axis_scale_txt_lit _var _col}] [Y-AXIS-GRID={y_axis_grid_txt_lit _var _col}] [Y-AXIS-LABEL={y_axis_label_txt_lit _var _col}] [Y-AXIS-MASK={mask_txt_lit _var _col}] [Y-AXIS-MIN-VALUE={y_axis_min_value_num_lit _var _col}] [Y-AXIS-MAX-VALUE={y_axis_max_value_num_lit _var _col}] [Y-AXIS-MAJOR-INCREMENT={y_axis_major_increment_num_lit _var _col}] [Y-AXIS-MINOR-INCREMENT={y_axis_minor_increment_num_lit _var _col}] [Y-AXIS-MAJOR-TICK-MARKS={y_axis_major_tick_marks_txt_lit _var _col}] [Y-AXIS-MINOR-TICK-MARKS={y_axis_minor_tick_marks_txt_lit _var _col}] [Y-AXIS-TICK-MARK-PLACEMENT={y_axis_tick_mark_placement_txt_lit _var _col}] [Y-AXIS-SCALE={y_axis_scale_txt_lit _var _col}] [Y2-AXIS-LABEL={y2_axis_label_txt_lit _var _col}] [Y2-AXIS-MASK={mask_txt_lit _var _col}] [Y2-AXIS-MIN-VALUE={y2_axis_min_value_num_lit _var _col}] [Y2-AXIS-MAX-VALUE={y2_axis_max_value_num_lit _var _col}] </pre>

COMMAND	SYNTAX
	[Y2-AXIS-MAJOR-INCREMENT={y2_axis_major_increment_num_lit _var _col}] [Y2-AXIS-MINOR-INCREMENT={y2_axis_minor_increment_num_lit _var _col}] [Y2-AXIS-MAJOR-TICK-MARKS={y2_axis_major_tick_marks_txt_lit _var _col}] [Y2-AXIS-MINOR-TICK-MARKS={y2_axis_minor_tick_marks_txt_lit _var _col}] [Y2-AXIS-SCALE={y2_axis_scale_txt_lit _var _col}] [Y2-COLOR-PALETTE=color_palette_lit _var _col] [Y2-DATA-ARRAY=array_name] [Y2-DATA-ARRAY-ROW-COUNT={x_num_lit _var _col}] [Y2-DATA-ARRAY-COLUMN-COUNT={x_num_lit _var _col}] [Y2-DATA-ARRAY-COLUMN-LABELS={NONE array_name ({txt_lit _var _col},...)}] [Y2-TYPE={chart_type_txt_lit _var _col}]
PRINT-DIRECT	[NOLF] [PRINTER={LINEPRINTER POSTSCRIPT HPLASERJET HTML LP PS HP HT}] {txt_lit _var _col}...
PRINT-IMAGE	[image_name]position [TYPE={image_type_lit _var _col}] [IMAGE-SIZE= (width_num_lit _var _col,height_num_lit _var _col)] [SOURCE={file_name_lit _var _col}] [[FOR-PRINTER=({POSTSCRIPT HPLASERJET HTML PDF WINDOWS PS HP HT PD WP printer_type_lit _var _col}, {image_type_lit _var _col}, {file_name_lit _var _col})]...]
PRINT-TABLE	NAME=table_name_var _lit _col [CONTINUATION=continuation_var _lit _col]
PUT	{src_any_lit _var _col}... INTO dst_array_name(element) [field[(occurs)]]...
READ	{filenum_lit _var _col} INTO {any_var:length_int_lit}... [STATUS=status_num_var]
ROLLBACK	ROLLBACK
SBTOMBS	{txt_var}
SECURITY	[SET=(sid [,sid]...)] [APPEND=(sid [,sid]...)] [REMOVE=(sid [,sid]...)] [MODE=mode]
SET-COLOR	[PRINT-TEXT-FOREGROUND=({color_name_lit _var _col {rgb}})] [PRINT-TEXT-BACKGROUND=({color_name_lit _var _col {rgb}})] [PRINT-PAGE-BACKGROUND=({color_name_lit _var _col {rgb}})] [LINE-COLOR=({color_name_lit _var _col {rgb}})] [FILL-COLOR=({color_name_lit _var _col {rgb}})]
SET-DELAY-PRINT	delay_var WITH {src_lit _var _col}

COMMAND	SYNTAX
SHOW	[<i>cursor_position</i> [CLEAR-SCREEN CS CLEAR-LINE CL] [<i>any_lit</i> <i>_var</i> <i>_col</i>] [EDIT <i>edit_mask</i> NUMBER MONEY DATE] [BOLD] [BLINK] [UNDERLINE] [REVERSE] [NORMAL] [BEEP] [NOLINE]...
STOP	[QUIET]
STRING	{ <i>src_any_lit</i> <i>_var</i> <i>_col</i> }... BY { <i>delim_txt_lit</i> <i>_var</i> <i>_col</i> } INTO <i>dst_txt_var</i>
SUBTRACT	SUBTRACT { <i>src_num_lit</i> <i>_var</i> <i>_col</i> } FROM <i>dst_num_var</i> [ROUND= <i>nn</i>]
TOC-ENTRY	TEXT={ <i>src_txt_lit</i> <i>_var</i> <i>_col</i> } [LEVEL={ <i>level_num_lit</i> <i>_var</i> <i>_col</i> }]
UNSTRING	{{ <i>src_txt_lit</i> <i>_var</i> <i>_col</i> } { <i>src_date_var</i> <i>_col</i> }} BY { <i>delim_txt_lit</i> <i>_var</i> <i>_col</i> } INTO <i>dst_txt_var</i> ...
UPPERCASE	<i>txt_var</i>
USE	<i>database</i>
USE-COLUMN	{ <i>column_number_int_lit</i> <i>_var</i> <i>_col</i> }
USE-PRINTER-TYPE	<i>printer-type</i> See DECLARE-PRINTER for valid types.
USE-PROCEDURE	[FOR-REPORTS=(<i>report_name1</i> [, <i>report_namei</i>]...)] [BEFORE-REPORT= <i>procedure_name</i> [(<i>arg1</i> [, <i>argi</i>]...)]] [AFTER-REPORT= <i>procedure_name</i> [(<i>arg1</i> [, <i>argi</i>]...)]] [BEFORE-PAGE= <i>procedure_name</i> [(<i>arg1</i> [, <i>argi</i>]...)]] [AFTER-PAGE= <i>procedure_name</i> [(<i>arg1</i> [, <i>argi</i>]...)]]
USE-REPORT	{ <i>report_name_lit</i> <i>_var</i> <i>_col</i> }
WHILE	<i>logical_expression</i> <i>sqr_commands</i> ... [BREAK] [CONTINUE} <i>sqr_commands</i> ... END-WHILE
WRITE	{ <i>filenum_lit</i> <i>_var</i> <i>_col</i> } FROM {{ <i>txt_lit</i> <i>_var</i> <i>_col</i> } { <i>date_var</i> <i>_col</i> } <i>num_col</i> } [: <i>len_int_lit</i> <i>_var</i> <i>_col</i>]} { <i>num_lit</i> <i>_var</i> : <i>len_int_lit</i> <i>_var</i> <i>_col</i> }... [STATUS= <i>status_num_var</i>]
WRITE-RS	NAME= <i>row_set_name_var</i> <i>_lit</i> <i>_col</i> VALUE=(<i>{name_var</i> <i>_lit</i> <i>_col</i> },{ <i>data_var</i> <i>_lit</i> <i>_col</i> })



Deprecated Information

In This Appendix

Deprecated Production Reporting Command-line Flags.....	435
Deprecated SQR.INI Entries.....	436
Deprecated Transforms.....	438
Deprecated Production Reporting Commands	438
BEGIN-REPORT	439
DATE-TIME	440
DECLARE PRINTER	441
DECLARE PROCEDURE	445
DOLLAR-SYMBOL.....	446
GRAPHIC BOX	448
GRAPHIC FONT	449
GRAPHIC HORZ-LINE.....	450
GRAPHIC VERT-LINE	450
MONEY-SYMBOL	451
NO-FORMFEED	452
PAGE-SIZE	453
PRINT ...CODE	454
PRINTER-DEINIT	454
PRINTER-INIT	455

Deprecated Production Reporting Command-line Flags

Table 102 Deprecated Production Reporting Command-line Flags

Flag	Description
<code>-EH_FULLHTML:xx</code>	<p>Specifies the level of HTML that your browser supports so appropriate Enhanced HTML code is generated. Acceptable values for this flag are:</p> <ul style="list-style-type: none">● 40—Generates XHTML 1.1● 32—Generates HTML 3.2 <p>The following values are deprecated (no longer valid but still accepted for upward compatibility):</p> <ul style="list-style-type: none">● 30—Was used to specify HTML 3.0● TRUE—Was used to specify HTML 3.2

Flag	Description
	<ul style="list-style-type: none"> ● FALSE—Was used to specify HTML 3.0 <p>Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified. Only use this flag when needed. Reports that require this flag should be migrated to use the XHTML 1.1 generator as soon as possible.</p>
-Mfile	<p>Defines a startup file containing sizes to be assigned to various internal parameters—extremely small, large, or complex reports.</p> <p>-Mfiles are text files that have individual switches in the INI files unique to a specific run. (See “Deprecated SQR.INI Entries” on page 436 for more information.)</p>
-PRINTER:HT	Uses HTML 2.0 when creating output files.

Deprecated SQR.INI Entries

The following SQR.INI entries are deprecated:

- [Values for the FullHTML Keyword in the \[Enhanced-HTML\] Section](#)
- [\[Processing-Limits\] Section](#)
- [Values for PDFCompressionText and PDFCompressionGraphics in the \[Default-Settings\] Section](#)

Values for the FullHTML Keyword in the [Enhanced-HTML] Section

[Table 103](#) lists the deprecated values for the FullHTML keyword. (See [“Deprecated Production Reporting Command-line Flags” on page 435](#) on [page 435](#) for more information.)

Table 103 Deprecated Values for the FullHTML Keyword

Entry	Deprecated Values	Description
FullHTML	3.0—Was used to generate HTML 3.0. TRUE—Was used to generate HTML 3.2. FALSE—Was used to generate HTML 3.0.	<p>Specifies the level of HTML that the browser supports so appropriate Enhanced HTML code is generated.</p> <p>Note: See FullHTML under the “[Enhanced-HTML] Section” on page 343 for more information on current values.</p>

[Processing-Limits] Section

Production Reporting has built-in default values as to how much memory to allocate to certain Production Reporting internal structures. In versions of Production Reporting prior to 8.0, you were required to specify how much memory to allocate for some of these internal structures. Starting with version 8.0, Production Reporting automatically adjusts the internal structures until the architectural limit is reached.

The sizes and limitations of Production Reporting’s internal structures are defined in the [Processing-Limits] section in SQR.INI. Processing limits will still be supported in this release.

Unlike previous releases, however, you can only increase the default values (you cannot decrease them).

The following internal structures will now have their default sizes increased as indicated in [Table 104](#).

Table 104 Entries for [Processing-Limits] Section

Entry	Old Default	New Default	Maximum Value	Description
BREAKS	100	1024	65535	Number of BREAK arguments allowed per EVALUATE command.
DYNAMICARGS	70	4096	32767	Maximum number of dynamic SQL arguments.
EXPRESSIONSPACE	8192	65535	65535	Maximum length, in bytes, of temporary string storage used during LET operations.
FORWARDREFS	200	1024	32767	Maximum number of column forward references.
ONBREAKS	30	1024	65535	Maximum number of ON-BREAK LEVEL=values per SET.
POSITIONS	1800	32767	65535	Maximum number of placement parameters, "(10,5,30)".
PROGLINEPARS	18000	32767	65535	Maximum number of arguments for all program lines.
PROGLINES	5000	16384	32767	Maximum number of program lines (Production Reporting commands).
QUERIES	60	1024	32767	Maximum number of BEGIN-SQL and BEGIN-SELECT paragraphs.
QUERYARGS	240	4096	65535	Maximum number of arguments (bind variables) for all BEGIN-SQL or BEGIN-SELECT paragraphs.
SQLSIZE	4000	16384	65535	Maximum length of an SQL statement in characters.
STRINGSPACE	15000	32767	65535	Maximum size of string space for program line arguments, in bytes.
SUBVARS	100	4096	32767	Maximum number of substitution variables.
VARIABLES	1500	16384	32767	Maximum number of variables (string, float, integer, decimal, date), literal values, and database columns.
WHENS	70	1024	65535	Maximum number of WHEN arguments allowed per EVALUATE command.

Note:

The entries in the [Processing-Limits] section are the same as those specified with the *-Mfile* command line flag. If the *-Mfile* command line flag is used, then the [Processing-Limits] section in SQR.INI is not processed.

Values for PDFCompressionText and PDFCompressionGraphics in the [Default-Settings] Section

The PDFCompressionText and PDFCompressionGraphics settings in the [Default-Settings] section now appear in the [PDF Settings] section as CompressionText and CompressionGraphics. See “[PDF Settings] Section” on page 340 for more information.

Table 105 Deprecated Entries in the [Default-Settings] Section

Entry	Value	Description
PDFCompressionText PDFCompressionGraphics	0 - 9	Each of these entries specifies the amount of compression to apply. The values range from 0 (no compression) to 9 (maximum compression). The default value for both is 6, which is the best value for the compression verses speed.

Deprecated Transforms

Table 106 Deprecated Transforms

Transform	Description
ToCanonical	Transforms Unicode "compatibility characters" to their standard equivalents.
ToTraditionalChinese	Converts all Simplified Chinese characters to their Traditional Chinese equivalent.
ToSimplifiedChinese	Converts all Traditional Chinese characters to their Simplified Chinese equivalent.

Note:

A *transform* is a function of the LET command. See “Unicode Functions” on page 210 on page 211 for information on the available transforms.

Deprecated Production Reporting Commands

If you still have older Production Reporting commands in your program code, refer to [Table 107](#) to replace them with their updated alternatives. Even though the commands are technically supported in this release, they do not interact well with the current Production Reporting lexicon. Incorporating the deprecated commands into your Production Reporting code can cause unpredictable results.

Table 107 Deprecated Production Reporting Commands

Old Commands	Use Instead
BEGIN-REPORT (END-REPORT)	BEGIN-PROGRAM (END-PROGRAM)
DATE-TIME	datenow function

Old Commands	Use Instead
DECLARE PRINTER	DECLARE-PRINTER
DECLARE PROCEDURE	DECLARE-PROCEDURE
DOLLAR-SYMBOL	ALTER-LOCALE
GRAPHIC FONT	ALTER-PRINTER
GRAPHIC-BOX	DRAW
GRAPHIC HORZ-LINE	DRAW
GRAPHIC VERT-LINE	DRAW
MONEY-SYMBOL	ALTER-LOCALE
NO-FORMFEED	DECLARE-LAYOUT
PAGE-SIZE	DECLARE-LAYOUT
PRINTER-DEINIT	DECLARE-PRINTER
PRINTER-INIT	DECLARE-PRINTER
PRINT...CODE	PRINT...CODE-PRINTER

Note:

Two older commands, DECLARE PRINTER and DECLARE PROCEDURE, do not contain hyphens. The new commands, DECLARE-PRINTER and DECLARE-PROCEDURE, contain hyphens.

BEGIN-REPORT

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use BEGIN-PROGRAM.

Function

Begins a report.

Syntax

BEGIN-REPORT

Description

After processing the commands in the `SETUP` section, Production Reporting starts program execution at the `BEGIN-REPORT` section. The `PROGRAM` section typically contains a list of `DO` commands, though other commands can be used. This is the only required section in an Production Reporting program.

Examples

```
begin-report
  do startup
  do main
  do finish
end-report
```

DATE-TIME

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use the `datenow` function in the `LET` command.

Function

Retrieves the current date and/or time from the local machine (or from the database for Oracle and some DB2 platforms) and places it in the output file at the specified position or into a column variable.

Syntax

```
DATE-TIME position [date_format[col_var]]
```

Arguments

position

The position for printing the date.

date_format

A string literal containing the date format mask.

col_var

Places the retrieved date-time into a column variable rather than in the output file.

Description

If *col_var* is specified, a *date_format* must be supplied and the current date and time is retrieved each time this command is executed. Otherwise, the date is retrieved only at program start and the same date and/or time is printed each time.

If a *date_format* is not specified, then the date is returned in the default format for that database.

Table 108 Default Date-Time Formats

Database	Default Date-Time Format
DB2	YYYY-MM-DD-HH:MI
	YYYY-MM-DD-HH:MI:SS.NNNNNN
Informix	YYYY-MM-DD HH:MI
	YYYY-MM-DD HH:MI:SS.NNN
Oracle	DD-Mon-YYYY HH:MI PM
Sybase	DD-MON-YYYY HH:MI

For some databases, there are two default formats. The first format prints the date-time, as in the following example:

```
date-time (+1,1)
```

The second format retrieves the date-time into a column variable, as follows:

```
date-time () ' ' &date1
```

Obviously, for those databases with only one default format, that format is always used in either of these cases.

For information on the valid edit mask format codes, see [Table 52, “Miscellaneous Functions,” on page 212](#).

Examples

```
date-time (1,50) MM/DD/YY
date-time (1,1) 'Day Mon DD, YYYY'
date-time () HH:MI &time
date-time (+1,70) 'MON DD YYYY HH24:MI' &datetime
date-time (#i, #j) 'YYYY-MM-DD' &date1
```

DECLARE PRINTER

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use DECLARE-LAYOUT and DECLARE-PRINTER.

Function

Specifies the printer type and sets printer characteristics.

Syntax

```
DECLARE PRINTER
[TYPE=printer_type_lit]
[ORIENTATION=orientation_lit]
[LEFT-MARGIN=left_margin_num_lit]
[TOP-MARGIN=top_margin_num_lit]
[LINE-SIZE=line_size_num_lit]
[CHAR-SIZE=char_size_num_lit]
[LINES-INCH=lines_inch_int_lit]
[CHARS-INCH=chars_inch_num_lit]
[POINT-SIZE=point_size_num_lit]
[FONT-TYPE=font_type_txt_lit]
[SYMBOL-SET=symbol_set_id_lit]
[STARTUP-FILE=file_name_txt_lit]
[FONT=font_int_lit]
[BEFORE-BOLD=before_bold_string_txt_lit]
[AFTER-BOLD=after_bold_string_txt_lit]
```

Arguments

[Table 109](#) describes the arguments for the DECLARE PRINTER command.

Table 109 DECLARE PRINTER Command Arguments

Argument	Choice or Measure	Default Value	Description
TYPE	LINEPRINTER, POSTSCRIPT, HPLASERJET	LINEPRINTER	Production Reporting creates output specific to each printer. LINEPRINTER files generally consist of ASCII characters and can be viewed by a text editor. POSTSCRIPT files consist of ASCII characters, but you need to know PostScript to understand what will be shown on the printer. HP Laserjet files are binary files and cannot be edited or viewed.
ORIENTATION	PORTRAIT, LANDSCAPE	PORTRAIT	Portrait pages are printed vertically. Landscape pages are printed horizontally. Printing in landscape on HP Laserjet printers requires landscape fonts.
LEFT-MARGIN	inches	0.5	This argument does not apply to LINEPRINTER printers. This is the amount of blank space to leave at the left side of the page.
TOP-MARGIN	inches	0.5	This argument does not apply to LINEPRINTER printers. This is the amount of blank space to leave at the top of the page.
LINE-SIZE	points	12	This argument does not apply to LINEPRINTER printers.

Argument	Choice or Measure	Default Value	Description
			<p>This is the size of each Production Reporting line on the page. There are 72 points per inch.</p> <p>If LINE-SIZE is not specified, it follows the value for POINT-SIZE, if specified. The default value of 12 points yields 6 lines per inch.</p>
CHAR-SIZE	points	7.2	<p>This argument does not apply to LINEPRINTER printers.</p> <p>This is the size of each Production Reporting horizontal character column on the page (for example, the distance between the locations (1,12) and (1,13)). If CHAR-SIZE is not specified and the POINT-SIZE is less than 8.6, CHAR-SIZE is set to 4.32, which yields 16.6 characters per inch. The default value of 7.2 yields 10 characters per inch.</p>
LINES-INCH	lines	6	<p>This argument does not apply to Lineprinter printers.</p> <p>This is an alternate way of indicating the line size, in lines per inch, rather than in points for the LINE-SIZE.</p>
CHARS-INCH	characters	10	<p>This argument does not apply to LINEPRINTER printers.</p> <p>This is an alternate way of indicating the width of each Production Reporting character column, in characters per inch, rather than points for CHAR-SIZE.</p>
POINT-SIZE	points	12	<p>This argument does not apply to Lineprinter printers.</p> <p>This is the beginning size of the selected font.</p>
FONT-TYPE	PROPORTIONAL, FIXED	Depends on the font	<p>This argument applies only to HP Laserjet printers and needs to be specified only for font types not defined in Table 33, "Fonts Available for HP LaserJet Printers in Production Reporting," on page 140.</p>
SYMBOL-SET	HP defined sets	OU	<p>This argument applies only to HP Laserjet printers.</p> <p>The default value of "OU" is for the ASCII symbol set. For a complete list of the symbol sets, see the <i>HP Laserjet Technical Reference Manual</i>.</p>
STARTUP-FILE	filename	POSTSCRI.STR	<p>This argument applies only to PostScript printers.</p> <p>This is used to specify an alternate startup file. Unless otherwise specified, the default startup file is located in the directory specified by the environment variable SQRDIR.</p>
FONT	font_number	3	<p>This is the font number of the typeface to use. For HP Laserjet printers, this is the typeface value as defined by Hewlett-Packard. For a complete list of the typeface numbers, see the <i>HP Laserjet Technical Reference Manual</i>. For PostScript printers, Production Reporting supplies a list of fonts and arbitrary font number assignments in the file</p>

Argument	Choice or Measure	Default Value	Description
			<p>POSTSCRI.STR. The font numbers are the same as those for HP LaserJet printers, wherever possible, so that you can use the same font number for reports to be printed on both types of printers. You can modify the font list in POSTSCRI.STR to add or delete fonts. Read the POSTSCRI.STR file for instructions.</p> <p>Table 33, “Fonts Available for HP LaserJet Printers in Production Reporting,” on page 140 lists the fonts available in Production Reporting internally.</p> <p>Table 34, “Fonts Available for PostScript Printers,” on page 141 lists the fonts available in the Production Reporting POSTSCRI.STR file.</p>
BEFORE-BOLD	any string	(none)	<p>The BEFORE-BOLD and AFTER-BOLD arguments are for Lineprinter printers only. They specify the character string to turn bolding on and off. If the string contains blank characters, enclose it in single quote marks ('). To specify non-printable characters, such as ESC, enclose the decimal value inside angle brackets as follows:</p> <p>BEFORE-BOLD=<27>[r ! Turn on bold</p> <p>AFTER-BOLD=<27>[u ! Turn it off</p> <p>These arguments work in conjunction with the BOLD argument of the PRINT command.</p>
AFTER-BOLD	any string	(none)	See BEFORE-BOLD.

The font you choose—in orientation, typeface, and point size—must be an internal font, available in a font cartridge, or downloaded to the printer.

For fonts not listed in [Table 33, “Fonts Available for HP LaserJet Printers in Production Reporting,” on page 140](#), you must indicate the font style using the FONT-TYPE argument, or the correct typeface cannot be selected by the printer.

Description

DECLARE PRINTER can be used in either the SETUP section or in the body of the report. Generally, you should use it in the SETUP section. However, if you do not know what type of printer you will be using until the report is run, or if you need to change some of the arguments depending on user selection, you could put several DECLARE PRINTER commands in the body of the report and execute the one you need.

The following arguments take effect only once, upon execution of the first PRINT command, and thereafter have no effect even if changed:

- LINE-SIZE
- CHAR-SIZE
- LINES-INCH
- CHARS-INCH
- ORIENTATION

Production Reporting maps its line and column positions on the page by using a grid determined by the `LINE-SIZE` and `CHAR-SIZE` (or `LINES-INCH` and `CHARS-INCH`) arguments. Each printed piece of text is placed on the page using this grid. Because the characters in proportional fonts vary in width, it is possible that a word or string is wider than the horizontal space you have allotted, especially in words containing uppercase letters. To account for this behavior, you can either move the column position in the `PRINT` statement or indicate a larger `CHAR-SIZE` in the `DECLARE PRINTER` command.

DECLARE PROCEDURE

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use `DECLARE-PROCEDURE`.

Function

Defines specific event procedures.

Syntax

```
DECLARE PROCEDURE  
[BEFORE-REPORT=procedure_name]  
[AFTER-REPORT=procedure_name]  
[BEFORE-PAGE=procedure_name]  
[AFTER-PAGE=procedure_name]
```

Arguments

`BEFORE-REPORT`

A procedure to execute at the time of the first `PRINT` command. It may be used, for example, to create a report heading.

`AFTER-REPORT`

A procedure to execute just before the report file is closed at the end of the report. It can be used to print totals or other closing summary information. If no report was generated, the procedure does not execute.

`BEFORE-PAGE`

A procedure to execute at the beginning of every page, just before the first `PRINT` command for the page. It can be used, for example, to set up page totals.

`AFTER-PAGE`

A procedure to execute just before each page is written to the file. It can be used, for example, to display page totals.

Description

DECLARE PROCEDURE can be issued either in the SETUP section or in the body of the report. You can use the command as often as you like.

If you issue multiple DECLARE PROCEDURE commands, the last one takes precedence. In this way, you can turn procedures on and off while the report is executing. The referenced procedures do not take any arguments; however, the variables can be local by using the LOCAL argument. In addition, they can only PRINT into the body of the report, that is, they cannot PRINT into the header and/or footer areas.

Examples

```
declare procedure
  before-page=page_setup
  after-page=page_totals
```

DOLLAR-SYMBOL

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use ALTER-LOCALE.

Function

Redefines the currency symbol within numeric edit masks.

Syntax

```
DOLLAR-SYMBOL new_symbol
```

Arguments

new_symbol

A new, single character to be used in edit masks instead of the dollar sign (\$).

Description

The dollar sign (\$) is the default currency symbol for coding edit masks in the program that prints on report listings. The DOLLAR-SYMBOL provides a way to change that symbol for both the edit mask and for printing.

If you wish to change the symbol that prints on the report, use MONEY-SYMBOL in the PROCEDURE section. DOLLAR-SYMBOL and MONEY-SYMBOL can be used together to customize your Production Reporting programs and the reports they produce.

This command is used only in the SETUP section.

Note:

MONEY-SYMBOL has the same effect as these options of the ALTER-LOCALE command: MONEY-SIGN and MONEY-SIGN-LOCATION=LEFT.

Table 110 lists the characters that DOLLAR-SYMBOL cannot take.

Table 110 Characters Disallowed in the DOLLAR-SYMBOL Command

Type	Characters	
Numbers	0, 8, 9	
Alphabetical	b	B
	e	E
	n	N
	r	R
	v	V
Symbols	.	,
	-	+
	!	*
	—	`
	<	>
	()

Examples

The following example shows how to use the DOLLAR-SYMBOL command:

```
begin-setup
  dollar-symbol £      ! Define £ as the currency symbol
end-setup
begin-procedure
...
print #amount () edit £££,999.99
...
end-procedure
```

In the previous example, if you used the dollar sign in the edit mask after defining the dollar symbol as £, the following error message appears:

```
Bad numeric 'edit' format: $$$,999.99
```

GRAPHIC BOX

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use the DRAW command.

Function

Draws a box.

Syntax

```
GRAPHIC ({line_int_lit|_var},{column_int_lit|_var},  
{width_int_lit|_var}) BOX {depth_int_lit|_var}  
[rule_width_int_lit|_var [shading_int_lit|_var]]
```

Arguments

width and *depth*

The *width* is the horizontal size in character columns; *depth* is the vertical size in lines. The top left corner of the box is drawn at the line and column specified. The bottom right corner is calculated using the *width* and *depth*. You can specify relative placement with (+), (-), or numeric variables, as with regular print positions.

rule_width

The default rule width is 2 decipoints (there are 720 decipoints per inch). The top horizontal line is drawn just below the base of the line above the starting point. The bottom horizontal line is drawn just below the base of the ending line. Therefore, a one-line deep box surrounds a single line.

shading

A number between 1 and 100, specifying the percentage of shading to apply. 1 is very light, and 100 is black. If no shading is specified, the box is blank. Specify a *rule-width* of zero, if a border is not desired.

Description

Draws a box of any size at any location on the page. Boxes can be drawn with any size rule and can be shaded or left empty. After GRAPHIC commands execute, Production Reporting changes the current print location to the starting location of the graphic. (This is different than the way the PRINT command works.)

Examples

```
graphic (1,1,66) box 58 20! Draw box around page  
graphic (30,25,10) box 10! Draw a 10-characters-wide-by-10- characters-long  
box  
graphic (1,1,66) box 5 0 8! Draw 5 line shaded box (without ! border)
```



```
graphic (50,8,30) box 1! Draw box around 1 line
```

GRAPHIC FONT

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use ALTER-PRINTER and DECLARE-PRINTER to set the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

Function

Changes a font.

Syntax

```
GRAPHIC ()
```

```
FONT {font_number_int_lit|_var} [point_size_int_lit|_var[1|0]]  
[pitch_int_lit|_var]]
```

Arguments

font_number

For HP LaserJet printers, the specified font must be installed in the printer. For PostScript printers, the font must be defined in the POSTSCRI.STR file.

point_size

If the *point_size* is omitted, the size from the most recent DECLARE-PRINTER or GRAPHIC FONT command is used.

[1|0]

This argument is for HP LaserJet printers only. It is needed only if you are using a font that Production Reporting does not know about. (See [Table 34, “Fonts Available for PostScript Printers,” on page 141](#) under the DECLARE-PRINTER command.) 1 indicates a proportional font, and 0 indicates a fixed pitch font. The default is proportional.

pitch

If the specified font is fixed pitch, you should also indicate the pitch in characters per inch.

Examples

The following example shows the GRAPHIC FONT command:

```
graphic () font 23 8.5  
graphic () font 6 12 0 10  
graphic () font :#font_number :#point_size
```

GRAPHIC HORZ-LINE

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use the DRAW command.

Function

Draws a horizontal line.

Syntax

```
GRAPHIC ({line_int_lit|_var},{column_int_lit|_var},  
{length_int_lit|_var}) HORZ-LINE [rule_width_int_lit |_var]
```

Arguments

rule_width

The default rule width is 2 decipoints.

Description

Draws a horizontal line from the location specified, for the length specified. Horizontal lines are drawn just below the base. After GRAPHIC commands execute, Production Reporting changes the current print location to the starting location of the graphic. (This is different than the way the PRINT command works.)

Examples

```
graphic (4,1,66) horz-line 10! Put line under page heading  
graphic (+1,62,12) horz-line! Put line under final total
```

GRAPHIC VERT-LINE

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use the DRAW command.

Function

Draws a vertical line.

Syntax

```
GRAPHIC ({line_int_lit|_var},{column_int_lit|_var},  
{length_int_lit|_var}) VERT-LINE [rule_width_int_lit|_var]
```

Arguments

rule_width

The default rule width is 2 decipoints.

Description

Draws a vertical line from the location specified for the length (in lines) specified. Vertical lines are drawn just below the base line of the line position specified to just below the base line of the line reached by the length specified. To draw a vertical line next to a word printed on line 27, position the vertical line to begin on line 26, for a length of 1 line.

After GRAPHIC commands execute, Production Reporting changes the current print location to the starting location of the graphic. (This is different than the way the PRINT command works.)

Examples

```
graphic (1,27,54) vert-line! Draw lines between columns  
graphic (1,52,54) vert-line  
graphic (3,+2,4) vert-line 6! Red line the paragraph
```

MONEY-SYMBOL

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use the ALTER-LOCALE command.

Function

Redefines the currency symbol to be printed.

Syntax

```
MONEY-SYMBOL new_symbol
```

Arguments

new_symbol

A new, single character to replace the dollar sign (\$) or DOLLAR-SYMBOL character on the printed report.

Description

To change the symbol that prints on the report, use the `MONEY-SYMBOL` in the programs PROCEDURE sections. When the `MONEY-SYMBOL` is set, that value is used until the next `MONEY-SYMBOL` command executes.

The `DOLLAR-SYMBOL` and `MONEY-SYMBOL` can be used together to customize your Production Reporting application programs and the reports they produce.

To indicate a non-edit character, surround its decimal value with angle brackets (<>). Refer to [Table 110](#) under [Table 110](#) for characters that cannot be used with `MONEY-SYMBOL`.

Note:

`MONEY-SYMBOL` has the same effect as these options of the `ALTER-LOCALE` command: `MONEY-SIGN` and `MONEY-SIGN-LOCATION=LEFT`.

Examples

The following example shows how to use the `DOLLAR-SYMBOL` and `MONEY-SYMBOL` commands:

```
begin-setup
  dollar-symbol £      ! Define £ as the
    ! currency symbol
end-setup
begin-procedure      ! If #Amount=1234.56
...
money-symbol £
print #Amount () Edit £££,999.99      ! Prints as: £1,234.56
...
money-symbol $
print #Amount () Edit £££,999.99      ! Prints as: $1,234.56
...
money-symbol
print #Amount () Edit £££,999.99      ! Prints as: 1,234.56
...
end-procedure
```

NO-FORMFEED

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use the `FORMFEED` parameter of the `DECLARE-LAYOUT` command.

Function

Prevents form feed characters from being written to the output file.

Syntax

NO-FORMFEED

Description

NO-FORMFEED is useful for certain types of reports; for example, flat file output. It is used only in the SETUP section.

Do not write form feed control characters directly into the output file between pages.

Examples

```
begin-setup
  no-formfeed
end-setup
```

PAGE-SIZE

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use the MAX-LINES and MAX-COLUMNS parameters of the DECLARE-LAYOUT command.

Function

Sets the page size.

Syntax

PAGE-SIZE *page_depth_num_lit* *page_width_num_lit*

Description

If you are printing multiple reports, you must use the PAPER-SIZE parameter of the DECLARE-LAYOUT command.

This command is used in the SETUP section only.

Specify the *page_depth* in lines and the *page_width* in columns. An average report printed on 8 1/2 by 11 inch paper might have a page size of 60 lines by 80 columns. A 3 inch by 5 inch sales lead card might have a size of 18 by 50.

If the page size is not specified, the default of 62 lines by 132 columns is used.

For line printers, Production Reporting stores one complete page in a buffer before writing the page to the output file when you issue a NEW-PAGE command or when a page overflow occurs.

You could define a page to be 1 line deep and 4,000 characters wide. This could be used for writing large flat files, perhaps for copying to magnetic tape. Each time a NEW-PAGE occurs, one record would be written. The NO-FORMFEED command in the SETUP section can be used to suppress form feed characters between pages.

Use a page width at least one character larger than the right-most position that will be written. This prevents unwanted wrapping when printing. When the last column position on a line is printed, the current position becomes the first position of the next line. This can cause confusion when using relative line positioning with the `NEXT-LISTING` command. Having a wider page than necessary does not waste any file space since Production Reporting trims trailing blanks on each line before writing the report file.

The size of the internal page buffer used to store a complete page in memory can be determined by multiplying the page depth by the width in the `PAGE-SIZE` command. For PCs, the page buffer is limited to 64K bytes. On other computers, the page buffer is limited only by the amount of memory available.

Examples

```
begin-setup
  page-size 57 132      ! 57 lines long by 132 columns wide
end-setup
```

PRINT ...CODE

The `PRINT` command has the following format option :

`CODE`

`CODE` is a qualifier that may be discontinued in a future release. Use `CODE-PRINTER` instead.

If you use `CODE`, the sequence is assumed to be for the printer type specified in the `DECLARE-REPORT` or default printer, if none is specified.

PRINTER-DEINIT

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Production Reporting functionality, use the `RESET-STRING` parameter of the `DECLARE-PRINTER` command.

Function

Sends control or other characters to the printer at the end of a report.

Syntax

```
PRINTER-DEINIT initialization_string
```

Description

Specify nondisplay characters by placing their decimal values inside angled brackets. For example, `<27>` is the ESC or escape character.

PRINTER-DEINIT is used only in the SETUP section and is designed for use with Line-Printer style output. It has limited functionality with HP LaserJet and PostScript printers.

Examples

```
begin-setup
  printer-deinit<27>[7J    ! Reset the printer
end-setup
```

PRINTER-INIT

Note:

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer Oracle's Hyperion® SQR® Production Reporting functionality, use the INIT-STRING parameter of the DECLARE-PRINTER command.

Function

Sends control or other characters to the printer at the beginning of a report.

Syntax

```
PRINTER-INIT initialization_string
```

Description

Specify non-display characters by placing their decimal values inside angled brackets. For example, <27> is the ESC or escape character.

PRINTER-INIT is used only in the SETUP section and is designed for use with Line-Printer output. It has limited functionality with HP LaserJet and PostScript printers.

Examples

```
begin-setup
  printer-init<27>[7J    ! Set the printer
end-setup
```


Index

Symbols

- #character, defined, 14
- #current-column, 16
- #current-line, 16
- #DEFINE, 153, 187, 188
- #ELSE, 164, 185
- #end-file, 17, 276
- #END-IF, 166
- #ENDIF, 166, 185
- #IF, 184, 185
- #IFDEF, 187
- #IFDEF, 185, 188
- #INCLUDE, 25, 188
- #page-count, 17
- #return-status, 17
- #sql-count, 17
- #sql-status, 17
- #sqr-max-columns, 19
- #sqr-max-lines, 19
- #sqr-pid, 19
- \$character, defined, 14
- \$current-date, 16
- \$sql-error, 17
- \$sql-test, 17
- \$sqr-connected-db, 17
- \$sqr-connected-db-name, 18
- \$sqr-database, 18
- \$sqr-dbcs, 18
- \$sqr-encoding, 18
- \$sqr-encoding-console, 18
- \$sqr-encoding-database, 18
- \$sqr-encoding-file-input, 18
- \$sqr-encoding-file-output, 18
- \$sqr-encoding-hostname, 18
- \$sqr-encoding-report-output, 18
- \$sqr-encoding-source, 18
- \$sqr-locale, 18
- \$sqr-platform, 19
- \$sqr-program, 19
- \$sqr-report, 19
- \$sqr-ver, 19
- \$username, 19
- &character, defined, 14
- A, 21
- Bnn
 - BEGIN-SELECT command, 73
 - definition, 21
- BURST,{xx}, 21
- C, 22
- CB, 22
- Cnn argument
 - BEGIN-SELECT command, 73
 - BEGIN-SQL command, 77
- DBconnectionstring argument, BEGIN-SELECT, 74, 78
- DBdatabase, 22
- DEBUG, 22, 185, 187, 188
- Dnn, 22
- DNT, 22
- E, 22
- EH_APPLETS, dir, 22
- EH_BQD, 23
- EH_BQD, file, 23
- EH_BROWSER, xx, 23
- EH_CSV, 23
- EH_CSV, file, 23
- EH_CSVONLY, 23
- EH_DEBUG, 24
- EH_FULLHTML, xx, 435
- EH_ICONS, dir, 24
- EH_IMAGES, dir, 24
- EH_KEEP, 24
- EH_LANGUAGE, xx, 24
- EH_PDF, 24
- EH_SCALE, nn, 24
- EH_XIMG, 25

- EH_XML, file, 25
- EH_ZIP, file, 25
- F, 25
- ID, 25
- KEEP, 26, 145
- ldir_list, 25
- LL, 26
- Mfile, 436
- NOLIS, 26, 145
- O, 26
- PB, 26
- PRINTER
 - EH, 26
 - EP, 26
 - GD, 26
 - HP, 27
 - LP, 27
 - PD, 27
 - PS, 27
 - WP, 27, 141
 - xx, 26, 436
- RS, 28
- RT, 28
- S, 28
- Tnn, 28
- T{B}, 28
- T{Z}, 28
- U, 28
- Vserver, 29
- XB, 29
- XC
 - definition of, 29
 - in the EXECUTE command, 170
- XCB, 29
- XFRM, 29
- XI, 29
- XL, 29
- XLFF, 29
- XMB, 29
- XNAV, 29
- XP
 - BEGIN-SELECT command, 73
 - BEGIN-SQL command, 77
 - definition of, 30
- XTB, 30
- XTOC, 30
- ZEN{name}, 30

- ZIF{file}, 30
- ZIV, 30
- ZMF{file}, 30, 31
- ZRF{file}, 31
- @character, defined, 14

A

- abs, 197, 207
- acos, 198
- ADD, 36
- AFTER, 62, 63
- AFTER qualifier, ON-BREAK argument, 255
- AFTER-BOLD argument
 - DECLARE-PRINTER command, 138, 444
 - PRINT command, 241
- AFTER-PAGE argument
 - DECLARE PROCEDURE command, 445
 - DECLARE-PROCEDURE command, 143
 - DECLARE-TOC command, 149
 - USE-PROCEDURE command, 295
- AFTER-REPORT argument
 - DECLARE PROCEDURE command, 445
 - DECLARE-PROCEDURE command, 143
 - USE-PROCEDURE command, 295
- AFTER-TOC argument
 - DECLARE-TOC command, 149
- ALTER-COLOR-MAP, 37
- ALTER-CONNECTION, 38
- ALTER-LOCALE, 41
- ALTER-PRINTER, 48, 138, 142
- ALTER-REPORT, 50
- ALTER-TABLE, 55
- arccosine, 198
- arcsine, 198
- arctangent, 198
- array
 - defined, 212
 - elements, 94
- arrays
 - ARRAY-ADD command, 58
 - ARRAY-DIVIDE, 58
 - ARRAY-MULTIPLY, 58
 - ARRAY-SUBTRACT command, 58
 - maximums, 93
- ascii, 203
- asciic, 203
- asin, 198

ASK, [60](#), [187](#), [188](#)
 AT-END argument, NEXT-COLUMN command, [231](#)
 atan, [198](#)
 ATTRIBUTES argument
 declaration keywords, [115](#)
 definition of, [112](#)
 selector/sub-selector keywords, [113](#)

B

bar codes, [260](#)
 BATCH-MODE, INPUT command, [190](#)
 BEEP argument, SHOW command, [284](#)
 BEFORE, [62](#), [63](#)
 BEFORE qualifier, ON-BREAK argument, [255](#)
 BEFORE-BOLD argument
 DECLARE-PRINTER command, [138](#), [444](#)
 PRINT command, [241](#)
 BEFORE-PAGE argument
 DECLARE PROCEDURE command, [445](#)
 DECLARE-PROCEDURE command, [143](#)
 DECLARE-TOC command, [149](#)
 USE-PROCEDURE command, [295](#)
 BEFORE-REPORT argument
 DECLARE PROCEDURE command, [445](#)
 DECLARE-PROCEDURE command, [143](#)
 USE-PROCEDURE command, [295](#)
 BEFORE-TOC argument, DECLARE-TOC command, [149](#)
 BEGIN-DOCUMENT, [61](#)
 BEGIN-EXECUTE, [62](#)
 BEGIN-FOOTING, [66](#)
 BEGIN-HEADING, [68](#). *See also* Page Header
 BEGIN-PROCEDURE, [69](#)
 BEGIN-PROGRAM, [72](#)
 BEGIN-REPORT, [439](#)
 BEGIN-SELECT, [62](#), [72](#)
 BEGIN-SETUP, [76](#)
 BEGIN-SQL, [77](#)
 bind variables, [73](#)
 BLINK argument, SHOW command, [284](#)
 BOLD argument
 PRINT command, [241](#)
 SHOW command, [284](#)
 BOTTOM-MARGIN argument
 DECLARE-LAYOUT command, [134](#)
 BOX and SHADE, [242](#)

BOX argument
 GRAPHIC command, [448](#), [450](#)
 PRINT command, [241](#)
 BREAK, [80](#)
 BREAK argument, EVALUATE command, [169](#)
 break processing, field changes, [254](#)

C

CALL, [80](#)
 CALL-SYSTEM, [80](#)
 callable Production Reporting, [22](#), [29](#)
 cancel dialog box, [22](#)
 CAPTION argument, PRINT-BAR-CODE, [261](#)
 ceil, [198](#)
 CHAR-SIZE argument, DECLARE-PRINTER command, [443](#)
 CHAR-WIDTH argument, DECLARE-LAYOUT command, [135](#)
 CHARS-INCH argument, DECLARE-PRINTER command, [443](#)
 charts, [263](#)
 CHECKSUM argument, PRINT-BAR-CODE, [261](#)
 chr, [204](#)
 CLEAR-ARRAY, [85](#), [92](#)
 CLEAR-LINE argument, SHOW command, [283](#)
 CLEAR-SCREEN argument, SHOW command, [283](#)
 CLOSE, [86](#)
 CLOSE-RS, [86](#)
 COLOR argument
 ALTER-COLOR-MAP command, [37](#)
 DECLARE-COLOR-MAP command, [126](#)
 DECLARE-PRINTER command, [138](#)
 GET-COLOR command, [181](#)
 SET-COLOR command, [280](#)
 COLUMNS, [14](#), [88](#), [293](#)
 COLUMNS argument, POSITION command, [238](#)
 COMMAND, [62](#), [63](#), [218](#), [220](#)
 command line
 flags, [21](#)
 command_line, [212](#)
 COMMIT, [88](#)
 compiler directives, [164](#), [166](#), [185](#), [187](#), [188](#)
 CONCAT, [89](#)
 concatenation, [288](#)
 cond, [212](#)
 conditional processing, [168](#), [186](#), [297](#)
 CONNECT, [91](#)

- CONNECTION, [62](#), [63](#)
 - connection_name, [37](#), [39](#), [128](#), [182](#), [280](#)
 - copyright banner, [25](#)
 - cosh, [198](#)
 - cosine, [198](#)
 - cosValue type, [198](#)
 - CREATE-ARRAY, [92](#), [274](#)
 - CREATE-COLOR-PALETTE, [94](#)
 - CREATE-LIST, [95](#)
 - CREATE-TABLE, [96](#)
 - csv file, [23](#)
 - csv icon, [23](#)
 - currency symbol, [446](#), [451](#)
- D**
- data input, [189](#)
 - database cursors, [91](#)
 - database type, [18](#)
 - DATE argument
 - DECLARE-VARIABLE command, [151](#)
 - DISPLAY command, [155](#)
 - MOVE command, [225](#)
 - SHOW command, [284](#)
 - date edit format characters, [247](#)
 - date functions
 - dateadd, [208](#)
 - datediff, [209](#)
 - datenow, [209](#)
 - datetostr, [209](#)
 - strtodate, [209](#)
 - DATE-EDIT-MASK argument, ALTER-LOCALE command, [42](#)
 - DATE-SEPARATOR argument, ALTER-LOCALE command, [43](#)
 - DATE-TIME, [440](#)
 - dateadd, [208](#)
 - datediff, [209](#)
 - datenow, [209](#)
 - datetostr, [209](#)
 - DAY-OF-WEEK-CASE argument, ALTER-LOCALE command, [43](#)
 - DAY-OF-WEEK-FULL argument, ALTER-LOCALE command, [43](#), [336](#)
 - DAY-OF-WEEK-SHORT argument, ALTER-LOCALE command, [43](#), [336](#)
 - DB2, [441](#)
 - DDO, [219](#), [239](#)
 - ALTER-CONNECTION command, [39](#)
 - BEGIN-EXECUTE command, [62](#)
 - BEGIN-SELECT command, [74](#)
 - BEGIN-SQL command, [78](#)
 - COMMIT command, [89](#)
 - CONNECT command, [91](#)
 - DECLARE-CONNECTION command, [128](#)
 - DECLARE-VARIABLE command, [152](#)
 - LET command, [192](#)
 - LOAD-LOOKUP command, [218](#)
 - PRINT command, [239](#)
 - ROLLBACK command, [278](#)
 - variables in the Environment section, [334](#)
 - DECIMAL argument, DECLARE-VARIABLE command, [151](#)
 - DECIMAL-SEPARATOR argument, ALTER-LOCALE command, [43](#)
 - DECLARE PRINTER, [441](#)
 - DECLARE PROCEDURE, [445](#)
 - DECLARE-CHART, [100](#), [265](#)
 - ATTRIBUTES argument, [112](#)
 - DECLARE-COLOR-MAP, [126](#)
 - DECLARE-CONNECTION, [63](#), [128](#)
 - DECLARE-IMAGE, [130](#)
 - DECLARE-LAYOUT, [132](#)
 - PAPER-SIZE parameter, [453](#)
 - DECLARE-PRINTER, [111](#), [137](#)
 - DECLARE-PROCEDURE, [143](#)
 - DECLARE-REPORT, [144](#)
 - DECLARE-TABLE, [146](#)
 - DECLARE-TOC, [148](#)
 - DECLARE-VARIABLE, [150](#)
 - DEFAULT-NUMERIC argument, [151](#)
 - Default-Settings section, in SQR.INI, [328](#)
 - deg, [199](#)
 - delete, [203](#)
 - destination field, [225](#)
 - DISPLAY, [155](#)
 - DISTINCT argument, BEGIN-SELECT command, [73](#)
 - DIVIDE, [158](#)
 - DO, [72](#), [159](#), [440](#)
 - DO argument, EXECUTE command, [171](#)
 - document marker, [61](#), [238](#)
 - DOCUMENT paragraphs, [348](#)
 - DOLLAR-SYMBOL, [446](#), [452](#)

DOT-LEADER argument, DECLARE-TOC
 command, 149
 DSN, 37, 39, 126, 128
 DSQUERY, 333
 DUMP-TABLE, 163
 dynamic query variables, 73

E

edit, 204
 EDIT argument
 SHOW command, 284
 edit masks
 case sensitivity, 247
 default formats, 251
 description of, 247
 samples, 249
 uses, 250
 with specified width values, 250
 edit types, 244
 EDIT-OPTION-AD argument, ALTER-LOCALE
 command, 43, 336
 EDIT-OPTION-AM argument, ALTER-LOCALE
 command, 42, 43
 EDIT-OPTION-BC argument, ALTER-LOCALE
 command, 43, 336
 EDIT-OPTION-NA argument, ALTER-LOCALE
 command, 43
 EDIT-OPTION-PM argument, ALTER-LOCALE
 command, 43
 ELSE, 164
 ENCODE, 164
 Encoding, 210, 211, 212, 234
 END-DECLARE, 126, 165
 END-DOCUMENT, 165
 END-EVALUATE, 165
 END-EXECUTE, 62
 END-FOOTING, 165
 END-HEADING, 165
 END-IF, 167
 END-PROCEDURE, 167
 END-PROGRAM, 167
 END-SELECT, 62, 167
 END-SETUP, 167
 END-SQL, 167
 END-WHILE, 167
 ending a query, 175
 Enhanced-HTML, 343, 345

environment variables, 333
 Environment, Common, 333
 ERASE-PAGE argument, NEXT-COLUMN
 command, 232
 error message file, 30
 EVALUATE, 80, 168
 Examples, 38
 EXECUTE, 170
 exists, 203
 EXIT-SELECT, 175
 exp, 199
 exponents, 199
 expressions, 192, 193
 EXTENT argument, LOAD-LOOKUP command,
 219
 external source files, 188
 EXTRACT, 176

F

FIELD argument, CREATE-ARRAY command, 92
 fields See columns, 92
 file number, 234
 file-related functions
 delete, 203
 exists, 202, 203
 rename, 203
 files, reading, 275
 FILL-TABLE, 177
 FIND, 179
 FIXED argument, OPEN command, 234
 FIXED_NOLF argument, OPEN command, 234
 flags, in the Production Reporting command line, 21
 FLOAT argument, DECLARE-VARIABLE command,
 151
 floor, 199
 FONT argument, DECLARE-PRINTER command,
 139, 444
 FONT-TYPE argument, DECLARE-PRINTER
 command, 139, 443
 fonts
 HP Laser Jet, 140
 PostScript, 141
 Windows printers, 141
 Fonts section, in SQR.INI file, 337
 FOOTING, 66
 footings, 66
 FOOTING section, 12

FOR-APPEND argument, OPEN command, [234](#)
 FOR-READING argument, OPEN command, [234](#)
 FOR-REPORTS argument
 DECLARE-PRINTER command, [139](#)
 DECLARE-PROCEDURE command, [143](#)
 DECLARE-TOC command, [149](#)
 USE-PROCEDURE command, [295](#)

FOR-REPORTS, BEGIN-HEADING command, [68](#)

FOR-WRITING argument, OPEN command, [234](#)

FORMFEED argument, DECLARE-LAYOUT
 command, [134](#)

FROM PARAMETER, [64](#), [220](#)

FROM ROWSET, [63](#), [220](#)

functions

 date, [208](#)

 file-related, [203](#)

 miscellaneous, [212](#)

 numeric, [197](#)

 string, [203](#)

 transcendental, [202](#)

 unicode, [210](#)

 writing custom functions, [215](#)

G

general purpose procedures, in HTML, [303](#)

GET, [180](#)

GET-COLOR, [181](#)

GETDATA, [62](#), [63](#), [218](#), [220](#)

getenv, [213](#)

getfilemapname, [213](#)

global variables, [15](#)

GOTO, [183](#)

GOTO-TOP argument, NEXT-COLUMN command,
[231](#)

GRAPHIC BOX, [448](#)

GRAPHIC FONT, [449](#)

GRAPHIC HORZ-LINE, [450](#)

GRAPHIC VERT-LINE, [450](#)

H

halting Production Reporting, [287](#)

hardware/operating system, [19](#)

HEADING, [68](#)

heading procedures, in HTML, [305](#)

Hebrew language support, [259](#)

HEIGHT argument, PRINT-BAR CODE command,
[260](#)

hex, [199](#)

highlighting procedures, in HTML, [307](#)

HOLE-VALUE, [118](#)

HORZ-LINE argument, GRAPHIC command, [450](#)

HPLASERJET, [139](#)

HTML

 general purpose procedures, [303](#)

 heading procedures, [305](#)

 highlighting procedures, [307](#)

 hypertext link procedures, [309](#)

 list procedures, [309](#)

 table procedures, [312](#)

HTML-Images, [342](#)

hyperbolic cosine, [199](#)

hyperbolic sine, [201](#)

hyperbolic tangent, [202](#)

hypertext link procedures, in HTML, [309](#)

I

IF, [186](#)

IF nested within a WHILE loop, [298](#)

IFDEF, [185](#)

IMAGE-SIZE argument

 DECLARE-IMAGE command, [131](#)

 PRINT-IMAGE command, [271](#)

images

 DECLARE-IMAGE command, [130](#)

 printing, [271](#)

INDENTATION argument, DECLARE-TOC
 command, [149](#)

Informix, [31](#), [288](#), [441](#)

INIT-STRING argument, DECLARE-PRINTER
 command, [139](#)

INPUT, [189](#)

instr, [204](#)

instrb, [204](#)

INTEGER argument, DECLARE-VARIABLE
 command, [151](#)

INTO argument, EXECUTE command, [171](#)

isblank, [205](#)

isnull, [213](#)

K

KEY argument, LOAD-LOOKUP command, [219](#)

L

labels, 183
 LAST-PAGE, 192
 LAYOUT argument, DECLARE-REPORT command, 145
 LEFT-MARGIN argument
 DECLARE-LAYOUT command, 134
 DECLARE-PRINTER command, 442
 length, 205
 lengthb, 205
 lengthh, 210
 lengthp, 210
 LET, 192
 LEVEL argument, TOC-ENTRY command, 290
 LEVEL qualifier, ON-BREAK argument, 255
 LINE-HEIGHT argument, DECLARE-LAYOUT command, 134
 LINE-INCH argument, DECLARE-PRINTER command, 443
 LINE-SIZE argument, DECLARE-PRINTER command, 443
 LINE-WIDTH argument, DECLARE-LAYOUT command, 134
 LINEPRINTER, 139
 list procedures, in HTML, 309
 literals, 14
 LOAD-LOOKUP, 26, 217, 223
 loading an internal table, 217
 local procedures, 70
 local variables, 15
 LOCALE, 335
 LOCALE argument, ALTER-LOCALE command, 42
 log, 200
 log10, 200
 logical expressions, 186
 LOOKUP, 221, 222
 LOOPS argument, BEGIN-SELECT command, 74
 lower, 205
 LOWERCASE, 223
 lpad, 205
 ltrim, 206

M

MAX-COLUMNS argument, DECLARE-LAYOUT command, 134
 MAX-LINES argument, DECLARE-LAYOUT command, 134

MAXLEN arguments, INPUT command, 189
 MBTOSBS, 224
 Microsoft SQL Server, 32
 miscellaneous functions
 array, 212
 command_line, 212
 cond, 212
 getenv, 213
 getfilemapname, 213
 isnull, 213
 nvl, 213
 range, 214
 roman, 214
 wrapdepth, 215
 mod, 200
 MONEY argument
 DISPLAY command, 155
 MOVE command, 225
 PRINT command, 254
 SHOW command, 284
 MONEY-EDIT-MASK argument, ALTER-LOCALE command, 42
 MONEY-SIGN argument, ALTER-LOCALE command, 42
 MONEY-SIGN-LOCATION argument, ALTER-LOCALE command, 42
 MONEY-SYMBOL, 446, 451
 MONTHS-CASE argument, ALTER-LOCALE command, 44
 MONTHS-FULL argument, ALTER-LOCALE command, 44, 336
 MONTHS-SHORT argument, ALTER-LOCALE command, 44
 MOVE, 224
 multiple reports, 49
 MULTIPLY, 228

N

NAME, 39
 NAME argument, 85
 CREATE-ARRAY command, 92
 LOAD-LOOKUP command, 219
 natural log base e raised to x power, 199
 NEED argument, NEXT-LISTING command, 233
 nesting
 arguments, 197
 IF command, 186

nesting levels, #INCLUDE command, 188
 NEW-PAGE, 229, 453
 NEW-REPORT, 230
 NewGraphics, 102, 330
 NEXT-COLUMN, 88, 231
 NEXT-LISTING, 232, 454
 no logon, 29
 NO-ADVANCE argument, NEXT-LISTING
 command, 232
 NO-DUPLICATE, 39, 128
 NO-FORMFEED, 452, 453
 NOLINE argument
 DISPLAY command, 155
 SHOW command, 284
 non-Windows, 22, 29
 NOPROMPT arguments, INPUT command, 189
 NORMAL argument, SHOW command, 284
 NOWAIT, CALL command, 81
 NUMBER argument
 DISPLAY command, 155
 MOVE command, 225
 PRINT command, 254
 SHOW command, 284
 NUMBER-EDIT-MASK argument, ALTER-LOCALE
 command, 42
 numeric functions
 10 raised to x power, 199
 absolute value, 197, 207
 arccosine, 198
 arcsine, 198
 arctangent, 198
 cosine, 198
 degrees, 199
 hyperbolic cosine, 199
 hyperbolic sine, 201
 hyperbolic tangent, 202
 largest integer, 199
 log base 10, 200
 log base e, 200
 natural log base e raised to x power, 199
 power, 200
 radians, 201
 round, 201
 sign, 201
 sine, 201
 square root, 201
 tangent, 202

nvI, 213

O

ODBC, 31
 ON-BREAK
 AFTER qualifier, 255
 BEFORE qualifier, 255
 in PRINT command, 254
 SET qualifier, 255
 ON-ERROR argument, 348
 BEGIN-EXECUTE command, 62
 BEGIN-SELECT command, 74
 BEGIN-SQL command, 78
 CONNECT command, 91
 DIVIDE command, 158
 EXECUTE command, 170
 OPEN, 233
 open a new report, 230
 OPEN-RS, 235
 operands, 193
 operators, 194
 Oracle, 32, 73, 441
 ORIENTATION argument
 DECLARE-LAYOUT command, 134
 DECLARE-PRINTER command, 442
 OUTPUT argument, EXECUTE command, 171
 output file, 25

P

page numbering, 192
 page overflow, 230
 PAGE-DEPTH argument, DECLARE-LAYOUT
 command, 134
 PAGE-NUMBER, 237
 PAGE-SIZE, 453
 PAPER-SIZE argument, DECLARE-LAYOUT
 command, 134, 453
 PARAMETER_LIST, 63, 220
 PARAMETERS, 39, 128
 PASSWORD, 39, 128
 PC, 81
 PDF, 340
 PITCH argument, DECLARE-PRINTER command,
 139
 PL/SQL, 79
 point labels, 94

POINT-SIZE argument, DECLARE-PRINTER
 command, [139](#), [443](#)
 POSITION, [238](#)
 position, PRINT-IMAGE command, [271](#)
 POSTSCRIPT, [139](#)
 power, [200](#)
 precision, [194](#)
 PRINT
 BOLD format command, [241](#)
 BOX format command, [241](#)
 CENTER format command, [242](#)
 CODE-PRINTER format command, [242](#)
 DATE format command, [243](#)
 definition, [239](#)
 DELAY format command, [243](#)
 EDIT format command, [244](#)
 FILL format command, [252](#)
 FONT format command, [252](#)
 FOREGROUND/BACKGROUND format
 command, [253](#)
 format commands, [239](#), [454](#)
 MATCH format command, [253](#)
 MONEY format command, [254](#)
 NOP format command, [254](#)
 NUMBER format command, [254](#)
 ON-BREAK format command, [254](#)
 POINT-SIZE format command, [257](#)
 SHADE format command, [257](#)
 UNDERLINE format command, [257](#)
 URL format command, [257](#)
 URL-TARGET format command, [258](#)
 WRAP format command, [258](#)
 PRINT qualifier, ON-BREAK argument, [255](#)
 PRINT-BAR-CODE, [260](#)
 PRINT-CHART, [110](#), [263](#)
 PRINT-DIRECT, [270](#)
 PRINT-IMAGE
 definition, [271](#)
 position, [271](#)
 PRINT-TABLE, [273](#)
 PRINTER qualifier, [270](#)
 printer type, [27](#), [294](#)
 PRINTER-DEINIT, [454](#)
 PRINTER-INIT, [455](#)
 PRINTER-TYPE argument, DECLARE-REPORT
 command, [145](#)
 printing

 Hebrew, [244](#)
 on Windows, [26](#), [142](#), [436](#)
 PRINT...;CODE, [454](#)
 PROCEDURE, [62](#), [63](#), [218](#), [220](#)
 PROCEDURE section, [12](#)
 procedures
 running, [159](#)
 USE-PROCEDURE command, [295](#)
 process ID, [19](#)
 Production Reporting
 banner suppression, [29](#)
 page buffer, [270](#)
 program structure, [11](#)
 version, [19](#)
 PROGRAM section, [12](#)
 prompt, [190](#)
 PUT, [274](#)

Q

QUIET argument
 LOAD-LOOKUP command, [220](#)
 STOP command, [279](#), [288](#)

R

rad, [200](#)
 range, [214](#)
 READ, [275](#)
 RECORD argument, OPEN command, [234](#)
 rename, [203](#)
 replace, [206](#)
 report arguments file on command line, [31](#)
 report layout, [132](#)
 report output file, [19](#)
 reserved variables
 #current-column, [16](#)
 #current-line, [16](#)
 #end-file, [17](#)
 #INCLUDE, [25](#)
 #max-columns, [19](#)
 #page-count, [17](#)
 #return-status, [17](#)
 #sql-count, [17](#)
 #sql-status, [17](#), [71](#)
 #sqr-max-lines, [19](#)
 #sqr-pid, [19](#)
 \$current-date, [16](#)

- \$sql-error, [17, 71](#)
- \$sql-text, [17](#)
- \$sqr-connected-db, [17](#)
- \$sqr-connected-db-name, [18](#)
- \$sqr-database, [18](#)
- \$sqr-dbcs, [18](#)
- \$sqr-encoding, [18](#)
- \$sqr-encoding-console, [18](#)
- \$sqr-encoding-database, [18](#)
- \$sqr-encoding-file-input, [18](#)
- \$sqr-encoding-file-output, [18](#)
- \$sqr-encoding-hostname, [18](#)
- \$sqr-encoding-report-output, [18](#)
- \$sqr-encoding-source, [18](#)
- \$sqr-local, [18](#)
- \$sqr-platform, [19](#)
- \$sqr-program, [19](#)
- \$sqr-report, [19](#)
- \$sqr-ver, [19](#)
- \$username, [19, 91](#)
- RESET-STRING argument, DECLARE-PRINTER command, [139](#)
- RETURN_VALUE argument, LOAD-LOOKUP command, [219](#)
- REVERSE argument, SHOW command, [284](#)
- reversed characters, [259](#)
- RIGHT-MARGIN argument, DECLARE-LAYOUT command, [134](#)
- ROLLBACK, [277, 288](#)
- roman, [214](#)
- round, [201](#)
- ROUND argument
 - ADD command, [158, 229](#)
 - SUBTRACT command, [290](#)
- row set
 - close, [86](#)
 - open, [235](#)
 - SQR.INI setting, [330](#)
 - write, [300](#)
- ROWS argument, LOAD-LOOKUP command, [219](#)
- ROWSET, [63, 220](#)
- rpad, [206](#)
- RSV, [62, 63](#)
- rtrim, [206](#)
- run-time report files, [28](#)
- S**
 - SAVE qualifier, ON-BREAK argument, [255](#)
 - SBTOMBS, [278](#)
 - SCHEMA, [63, 220](#)
 - screen I/O, [283](#)
 - SECURITY, [278](#)
 - security issues, username/password, [92](#)
 - sequential processing, FOR-READING, [234](#)
 - SET qualifier, ON-BREAK argument, [255](#)
 - SET-COLOR, [280](#)
 - SET-DELAY-PRINT, [282](#)
 - SETUP, [126, 128](#)
 - SETUP section, [11, 76](#)
 - SHADE argument, PRINT command, [257](#)
 - SHOW, [283](#)
 - sign, [201](#)
 - sin, [201](#)
 - sine, [201](#)
 - sinh, [201](#)
 - SIZE argument, CREATE-ARRAY command, [92](#)
 - SKIPLINES argument, NEXT-LISTING command, [233](#)
 - SKIPLINES qualifier, ON-BREAK argument, [255](#)
 - SORT argument, LOAD-LOOKUP command, [219](#)
 - SOURCE argument
 - DECLARE-IMAGE command, [131](#)
 - PRINT-IMAGE command, [272](#)
 - source field, [225](#)
 - SPF files, [26](#)
 - SQR Extension, [335](#)
 - SQR Remote section, in SQR.INI, [345](#)
 - SQRDIR, [333](#)
 - SQRFLAGS, [333](#)
 - sqrt, [201](#)
 - square root, [201](#)
 - starting, [439](#)
 - startup file, [436](#)
 - STARTUP-FILE argument, DECLARE-PRINTER command, [139, 443](#)
 - STATUS, [62, 63](#)
 - STATUS argument
 - INPUT command, [189](#)
 - OPEN command, [235](#)
 - STOP, [287](#)
 - stored procedures
 - XP, [30](#)
 - Oracle, [79](#)

- STRING, 288
 - string functions
 - ascii, 203
 - asciic, 203
 - chr, 204
 - edit, 204
 - instr, 204
 - instrb, 204
 - isblank, 205
 - length, 205
 - lengthb, 205
 - lower, 205
 - lpad, 205
 - ltrim, 206
 - replace, 206
 - rpad, 206
 - rtrim, 206
 - substr, 207
 - substrb, 207
 - to_char, 207
 - to_multi_byte, 207
 - to_number, 207
 - to_single_byte, 208
 - translate, 208
 - upper, 208
 - string values, 212
 - strtodate, 209
 - structure, 11
 - STRUCTURE, See List Variable, 15
 - subroutines
 - calling, 81
 - writing, 82
 - substitution variables
 - #DEFINE command, 153
 - #IF command, 184
 - #IFDEF command, 187
 - #IFNDEF command, 188
 - ASK command, 60
 - substring, 176, 291
 - substringb, 207
 - substrp, 210
 - substrt, 210
 - SUBTRACT, 289
 - SYBASE, 73
 - Sybase
 - Default Date-Time Format, 441
 - Description, 32
 - server, 29
 - STOP command, 288
 - USE command, 293
 - Sybase DB-Lib, 73, 77
 - SYMBOL-SET argument, DECLARE-PRINTER
 - command, 139, 443
 - syntax conventions, 12
 - SYSTEM argument, CALL command, 83
- T**
- TABLE argument, LOAD-LOOKUP command, 219
 - Table of Contents file, 21
 - table procedures, in HTML, 312
 - tabular reports
 - NEXT-LISTING command, 232
 - ON-BREAK argument, 254
 - tan, 202
 - tangent, 202
 - tanh, 202
 - testing, 28
 - TEXT argument
 - DECLARE-VARIABLE command, 151
 - PRINT-BAR-CODE, 260
 - TOC-ENTRY command, 290
 - THOUSAND-SEPARATOR argument, ALTER-LOCALE
 - command, 43
 - time, 440
 - TIME-SEPARATOR argument, ALTER-LOCALE
 - command, 43
 - to_char, 207
 - to_multi_byte, 207
 - to_number, 207
 - to_single_byte, 208
 - TOC argument, DECLARE-REPORT command, 145
 - TOC-ENTRY, 290
 - TOP-MARGIN argument
 - DECLARE-LAYOUT command, 134
 - DECLARE-PRINTER command, 442
 - trailing blanks, 30
 - transcendental functions, 202
 - transform, 211
 - translate, 208
 - trunc, 202
 - truncate, 202
 - TYPE argument
 - DECLARE-IMAGE command, 131, 271
 - DECLARE-PRINTER command, 140, 442

INPUT command, 189
 PRINT-BAR-CODE, 260

U

ucall, 335
 UCALL.C, 82, 83
 Ufunc, 335
 UNDERLINE argument
 SHOW command, 284
 unicode, 212
 unicode functions
 lengthp, 210
 lengtht, 210
 substrp, 210
 substrt, 210
 transform, 211
 unicode, 212
 Unix, 25, 81
 UNSTRING, 291
 upper, 208
 UPPERCASE, 292
 USE, 293
 USE-COLUMN, 293
 USE-PRINTER-TYPE, 294
 USE-PROCEDURE, 295
 USE-REPORT, 296
 USER, 39, 128

V

valid uom suffixes, 133
 value determination, 168
 variables, 14
 global, 15
 local, 15
 rules, 15
 VARY argument, OPEN command, 234
 VERT-LINE argument, GRAPHIC command, 451

W

WAIT, CALL command, 81
 WHEN argument, EVALUATE command, 168
 WHEN-OTHER argument, EVALUATE command,
 169
 WHERE argument, LOAD-LOOKUP command, 219
 WHILE, 80, 297
 Windows

-C, 22
 -XCB, 29
 CALL command, 81
 log messages, 26
 WITH RECOMPILE argument, EXECUTE
 command, 174
 WRAP argument
 ON, 259
 wrapdepth, 215
 WRITE, 299
 WRITE-RS, 300
 writing to a page, 239

Z

ZIP+4 Postnet, 262