# Contents

## Chapter 1. Introduction
- 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
- Production Reporting Command-line Flags ........................................... 21
- Production Reporting Command-line Arguments .................................. 31

## Chapter 3. Production Reporting Command Reference
- 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
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN-SELECT</td>
<td>72</td>
</tr>
<tr>
<td>BEGIN-SETUP</td>
<td>76</td>
</tr>
<tr>
<td>BEGIN-SQL</td>
<td>77</td>
</tr>
<tr>
<td>BREAK</td>
<td>80</td>
</tr>
<tr>
<td>CALL, CALL SYSTEM</td>
<td>80</td>
</tr>
<tr>
<td>CLEAR-ARRAY</td>
<td>85</td>
</tr>
<tr>
<td>CLOSE</td>
<td>86</td>
</tr>
<tr>
<td>CLOSE-RS</td>
<td>86</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>88</td>
</tr>
<tr>
<td>COMMIT</td>
<td>88</td>
</tr>
<tr>
<td>CONCAT</td>
<td>89</td>
</tr>
<tr>
<td>CONNECT</td>
<td>91</td>
</tr>
<tr>
<td>CREATE-ARRAY</td>
<td>92</td>
</tr>
<tr>
<td>CREATE-COLOR-PALETTE</td>
<td>94</td>
</tr>
<tr>
<td>CREATE-LIST</td>
<td>95</td>
</tr>
<tr>
<td>CREATE-TABLE</td>
<td>96</td>
</tr>
<tr>
<td>#DEBUG</td>
<td>99</td>
</tr>
<tr>
<td>DECLARE-CHART</td>
<td>100</td>
</tr>
<tr>
<td>Attributes Argument</td>
<td>112</td>
</tr>
<tr>
<td>DECLARE-COLOR-MAP</td>
<td>126</td>
</tr>
<tr>
<td>DECLARE-CONNECTION</td>
<td>128</td>
</tr>
<tr>
<td>DECLARE-IMAGE</td>
<td>130</td>
</tr>
<tr>
<td>DECLARE-LAYOUT</td>
<td>132</td>
</tr>
<tr>
<td>DECLARE-PRINTER</td>
<td>137</td>
</tr>
<tr>
<td>DECLARE-PROCEDURE</td>
<td>143</td>
</tr>
<tr>
<td>DECLARE-REPORT</td>
<td>144</td>
</tr>
<tr>
<td>DECLARE-TABLE</td>
<td>146</td>
</tr>
<tr>
<td>DECLARE-TOC</td>
<td>148</td>
</tr>
<tr>
<td>DECLARE-VARIABLE</td>
<td>150</td>
</tr>
<tr>
<td>#DEFINE</td>
<td>153</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>155</td>
</tr>
<tr>
<td>DIVIDE</td>
<td>158</td>
</tr>
<tr>
<td>DO</td>
<td>159</td>
</tr>
<tr>
<td>DRAW</td>
<td>160</td>
</tr>
<tr>
<td>DUMP-TABLE</td>
<td>163</td>
</tr>
<tr>
<td>#ELSE</td>
<td>164</td>
</tr>
<tr>
<td>ELSE</td>
<td>164</td>
</tr>
<tr>
<td>ENCODE</td>
<td>164</td>
</tr>
</tbody>
</table>
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
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
Contents

Index ................................................................................................................. 457
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 \texttt{BEGIN-section} and \texttt{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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ }</td>
<td>Braces enclose required items.</td>
</tr>
<tr>
<td>[]</td>
<td>Square brackets enclose optional items.</td>
</tr>
<tr>
<td>...</td>
<td>An ellipsis shows that the preceding parameter can be repeated.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>'</td>
<td>A single quote starts and ends a literal text constant or any argument with more than one word. <strong>Caution:</strong> 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.</td>
</tr>
<tr>
<td>!</td>
<td>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 &quot;!=&quot; denotes a relational operator, and Production Reporting could confuse it with a comment where a relational argument could occur.</td>
</tr>
<tr>
<td>,</td>
<td>A comma separates multiple arguments.</td>
</tr>
<tr>
<td>()</td>
<td>Parentheses must enclose an argument or element.</td>
</tr>
</tbody>
</table>
Production Reporting Language Syntax Abbreviation Conventions

Table 2  Syntax Abbreviation Conventions

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>any_col</td>
<td>A column of any type.</td>
<td>&amp;string &amp;number &amp;date</td>
</tr>
<tr>
<td>date_col</td>
<td>Date or datetime column retrievable from a database.</td>
<td>&amp;date1</td>
</tr>
<tr>
<td>num_col</td>
<td>Numeric column retrievable from a database.</td>
<td>&amp;price</td>
</tr>
<tr>
<td>txt_col</td>
<td>Text column retrievable from a database.</td>
<td>&amp;address</td>
</tr>
<tr>
<td>any_lit</td>
<td>A literal of any type.</td>
<td>'abc' 12</td>
</tr>
<tr>
<td>int_lit</td>
<td>Integer literal defined in a program.</td>
<td>12345</td>
</tr>
<tr>
<td>num_lit</td>
<td>Numeric literal defined in a program.</td>
<td>12345.67</td>
</tr>
<tr>
<td>txt_lit</td>
<td>Text literal defined in a program.</td>
<td>'Company Confidential'</td>
</tr>
<tr>
<td>any_var</td>
<td>A variable of any type.</td>
<td>$string #number $date</td>
</tr>
<tr>
<td>date_var</td>
<td>A variable explicitly defined as a date variable.</td>
<td>$date1</td>
</tr>
<tr>
<td>num_var</td>
<td>Numeric variable defined in a program.</td>
<td>#total_cost</td>
</tr>
<tr>
<td>txt_var</td>
<td>String variable defined in a program.</td>
<td>$your_name</td>
</tr>
<tr>
<td>nn</td>
<td>Integer literal used as an argument to a command.</td>
<td>123</td>
</tr>
<tr>
<td>position</td>
<td>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.</td>
<td>(5, 10, 30)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#current-column</td>
<td>Current column on the page.</td>
</tr>
<tr>
<td>$current-date</td>
<td>Current date-time on the local machine when Production Reporting starts running the program.</td>
</tr>
<tr>
<td>#current-line</td>
<td>Current line on the page. This value is the physical line on the page, not the line in the report body. Line numbers are referenced in PRINT and other Production Reporting commands used for positioning data on the page. Optional page headers and footers, defined BEGIN-HEADING and BEGIN-FOOTING, have their own line sequences. Line 2 of the heading is different from line 2 of the report body or footing.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>#end-file</td>
<td>Set to one (1) if end of file occurs when reading a flat file; otherwise, set to zero (0). Your program should check this variable after each READ command. (See “READ” on page 275 for more information.)</td>
</tr>
<tr>
<td>#page-count</td>
<td>Current page number.</td>
</tr>
<tr>
<td>#return-status</td>
<td>Value returned to the operating system when Production Reporting exits. Can be set in your report. #return-status is initialized to the “success” return value for the operating system.</td>
</tr>
<tr>
<td>#sql-count</td>
<td>Count of rows affected by a DML statement (INSERT, UPDATE, or DELETE). This is equivalent to ROWCOUNT in Oracle and Sybase.</td>
</tr>
<tr>
<td>$sql-error</td>
<td>Text message from the database explaining an error. This variable is rewritten when a new error is encountered.</td>
</tr>
<tr>
<td>#sql-status</td>
<td>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 $SQL-ERROR variable contains the error message). The actual meaning of #SQL-STATUS is database dependent. Consult the proper database manual to fully interpret its meaning.</td>
</tr>
<tr>
<td>$sql-text</td>
<td>Last SQL statement sent to the database by Production Reporting. 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. The variable is populated for ODBC, Sybase, Oracle, Informix, Teradata, DB2, and DDO.</td>
</tr>
<tr>
<td>$sqr-connected-db</td>
<td>Class of the backend database from the initial connection or from CONNECT. For ODBC, the class is: ODBC, Sybase, Oracle, Informix, Redbrick, Teradata, or DB2. For DDO, the class is: XML, CSV, SAP, MSOLAP, Essbase, SQLServer, Sybase, Oracle, Informix, or DB2. The information is derived from $sqr-connected-db-name / {sqr-connected-db-name}.</td>
</tr>
<tr>
<td>$sqr-connected-db-name</td>
<td>Name of the database driver from the initial connection or from CONNECT.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>$sqr$-database</td>
<td>Database type for which Production Reporting was compiled. Valid values are: DB2, ODBC, Sybase, Informix, and Oracle.</td>
</tr>
<tr>
<td>$sqr$dbcs</td>
<td>Defines whether Production Reporting recognizes double-byte character strings. The value can be either YES or NO.</td>
</tr>
<tr>
<td>$sqr$-encoding</td>
<td>Name of the default encoding as defined by the ENCODING environment variable when Production Reporting is invoked.</td>
</tr>
<tr>
<td>$sqr$-encoding-console</td>
<td>Name of encoding for character data written to the log file or console.</td>
</tr>
<tr>
<td>$sqr$-encoding-database</td>
<td>Character data retrieved from and inserted into the database.</td>
</tr>
<tr>
<td>$sqr$-encoding-file-input</td>
<td>Name of encoding for character data read from files used with OPEN.</td>
</tr>
<tr>
<td>$sqr$-encoding-file-output</td>
<td>Name of encoding for character data written to files used with OPEN.</td>
</tr>
<tr>
<td>$sqr$-encoding-report-output</td>
<td>Report generated by Production Reporting (for example, an LIS file or a PostScript file).</td>
</tr>
<tr>
<td>$sqr$-encoding-source</td>
<td>Name of encoding for Production Reporting source files and include files.</td>
</tr>
<tr>
<td>$sqr$-hostname</td>
<td>Name of the computer on which Production Reporting is currently executing.</td>
</tr>
<tr>
<td>$sqr$-locale</td>
<td>Name of the current locale. A + at the end of the name indicates an argument used in the locale has changed.</td>
</tr>
<tr>
<td>#sqr-max-columns</td>
<td>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.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>#sqr-max-lines</td>
<td>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.</td>
</tr>
<tr>
<td>#sqr-pid</td>
<td>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.</td>
</tr>
<tr>
<td>$sqr-platform {sqr-platform}</td>
<td>Hardware/operating system type for which Production Reporting was compiled. Valid values are WINDOWS and UNIX.</td>
</tr>
<tr>
<td>$sqr-program</td>
<td>Name of the Production Reporting program file.</td>
</tr>
<tr>
<td>$sqr-report</td>
<td>Name of the report output file. $sqr-report reflects the actual name of the file to be used (as specified by the -F flag or the NEW- REPORT command).</td>
</tr>
<tr>
<td>$sqr-ver</td>
<td>Text string shown with the -ID flag. Production Reporting version.</td>
</tr>
<tr>
<td>$username</td>
<td>Database user name specified on the command line.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
<th>Program</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>-A</td>
<td>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. Restrictions: 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.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>-Bnn</td>
<td>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.</td>
<td>Production Reporting Server</td>
<td>ODBC Oracle Sybase</td>
</tr>
</tbody>
</table>
| -BURST:xx | Specifies the bursting type.  
- -BURST:T generates the Table of Contents.  
- -BURST:S generates report output according to the symbolic Table of Contents entries defined with the level 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 | All |
<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
<th>Program</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>-BURST:P0,1</td>
<td>-BURST:P0,1 when using -PRINTER:HT or -BURST:P1 when using -PRINTER:EH.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-C</td>
<td>(Windows) Displays the Cancel dialog box while the program runs to terminate program execution.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td>-CB</td>
<td>(Windows, Callable Production Reporting) Forces the communication box to be used.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td>-Dnn</td>
<td>(non-Windows) Displays the report output at the same time it is written to the output file. nn is the maximum number of lines to display before pausing. If no number is entered after -D, the display scrolls continuously. Note: The printer type must be LP or the display is ignored. If the program produces multiple reports, the display is for the first report only.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-DBdatabase</td>
<td>Uses the specified database, which overrides any USE command in the Production Reporting program.</td>
<td>Production Reporting Server</td>
<td>Sybase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td>-DEBUG[xxx]</td>
<td>Compiles lines preceded by #DEBUG. Without this flag, Production Reporting ignores these lines. (See “#DEBUG” on page 99.)</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td>-DNT:{xx}</td>
<td>Defines the default behavior for numeric variables. The value for xx 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.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>-E[file]</td>
<td>Directs error messages to the named file, or to the default file program.err. If no errors occur, no file is created.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-EH_APPLETS:dir</td>
<td>Defines the directory for Enhanced HTML applets. (The default directory for applets is IMAGES.) Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>Flag</td>
<td>Description</td>
<td>Program</td>
<td>Database</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>-EH_BQD</td>
<td>Generates a <code>{report}.bqd</code> file from the report data and associates a BQD (Brio Query Format File) icon with <code>{report}.bqd</code> in the navigation bar. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Print</td>
<td>All</td>
</tr>
<tr>
<td>-EH_BQD:file</td>
<td>Associates BQD icons with the specified file. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
</tbody>
</table>
| -EH_BROWSER:xx | Defines the browser and generates the HTML.  
  - 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_frm.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                 | All      |
<p>| -EH_CSV      | Generates a <code>{report}.csv</code> file from report data. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.                                                                                                           | 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 Execute                 | 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 Execute                 | All      |</p>
<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
<th>Program</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>-EH_DEBUG[:opts]</td>
<td>Produces a DBG output file containing compiler and internal error messages. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Print</td>
<td>All</td>
</tr>
<tr>
<td>-EH_ICONS:dir</td>
<td>Defines the directory where HTML should look for the referenced icons. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>-EH_IMAGES:dir</td>
<td>Defines the directory path for the GIF files used by the Navigation Bar. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>-EH_KEEP</td>
<td>Copies (does not move) the files when used in conjunction with -EH_ZIP. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>-EH_LANGUAGE:xx</td>
<td>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.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>-EH_PDF</td>
<td>Associates a PDF icon with {report}.pdf in the navigation bar. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>-EH_SCALE:{nn}</td>
<td>Sets the scaling factor from 50 to 200. Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>Flag</td>
<td>Description</td>
<td>Program</td>
<td>Database</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>-EH_XIMG</td>
<td>Specifies to not remove the directory path from the IMAGE reference.</td>
<td>Production Reporting Print</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Server</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-EH_XML:file</td>
<td>Associates an XML icon with a file.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-EH_ZIP[:file]</td>
<td>Moves generated files to the specified file or {report}.zip if {file} is</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>not specified.</td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: Only applicable with -PRINTER:EH or -PRINTER:EP.</td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-F[ file</td>
<td>directory]</td>
<td>Overrides the default output file name, program.lis.</td>
<td>Production Reporting Server</td>
</tr>
<tr>
<td></td>
<td>The default action places the program.lis file in the same directory as the</td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>program.sqr file. To use the current directory, specify -F without an</td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td></td>
<td>argument. To change the name of the output file, specify -F with the new</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>name. If the new name does not specify a directory, the file is created in</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>the current directory. The output file is not created until data actually</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>prints on the page. If no data prints, no output file is created.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The following shows how to specify file names and directories for different</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>operating systems.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNIX</td>
<td>/</td>
<td>-F$HOME/reports/</td>
<td>Production Reporting Server</td>
</tr>
<tr>
<td>Windows</td>
<td>\</td>
<td>-F:\Oracle\Files\</td>
<td>Production Reporting Server</td>
</tr>
<tr>
<td>-ID</td>
<td>(non-Windows) Displays the copyright banner.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-Idir_list</td>
<td>Specifies the directories that Production Reporting searches when processing</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>the #INCLUDE directive if the include file does not exist in the current</td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>directory and no path was specified for the file.</td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flag</td>
<td>Description</td>
<td>Program</td>
<td>Database</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>directory names must be separated by either commas (,) or semicolons (;).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|            | 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. For example, under UNIX: sqr myreport sammy/baker -I/home/sqr/inc/,
|            | usr/sqr/incl/                                                                                                                                                                                              |                                               |          |
| -KEEP      | Creates SPF files in addition to LIS files for each report generated. (See Chapter 28, “Printing Issues,” in Volume 1 of the Hyperion SQR Production Reporting Developer’s Guide for more information on LIS and SPF files.) | Production Reporting Server | 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                  | All      |
| -O[file]   | Directs log messages to the specified file or to program.log if no file is specified. By default, the file sqr.log is used in the current working directory.                                                                 | Production Reporting Server                  | All      |
| -PB        | Retains trailing blanks in column data.                                                                                                                                                                       | Production Reporting Server                  | Informix |
| -PRINTER:xx| Uses printer type xx when creating output files.                                                                                                                                                              | Production Reporting Server                  | All      |

<table>
<thead>
<tr>
<th>xx</th>
<th>Printer Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>EH</td>
<td>Enhanced HTML</td>
<td>-PRINTER:EH</td>
</tr>
<tr>
<td>GD</td>
<td>Generic Output Driver</td>
<td>-PRINTER:GD</td>
</tr>
<tr>
<td>-GD_DRIVER:name defines the name</td>
<td>To generate Excel documents, enter: -PRINTER:GD -GD_DRIVER:EXCEL</td>
<td></td>
</tr>
</tbody>
</table>
of the Generic Output Driver.

- 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.)

To generate Word documents, enter:

-PRINTER:GD -GD_DRIVER:WORD

To generate Power Point documents, enter:

-PRINTER:GD -GD_DRIVER:PP

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.)

Note: You can use the -f flag to change the output file name to report.doc, report.ppt, or report.xls.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
<th>Program</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP</td>
<td>HP LaserJet</td>
<td>-PRINTER:HP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Production Reporting does not support color for -PRINTER:HP (HP PCL) output generation.</td>
<td></td>
</tr>
<tr>
<td>LP</td>
<td>Line Printer</td>
<td>-PRINTER:LP</td>
<td></td>
</tr>
<tr>
<td>PD</td>
<td>PDF</td>
<td>-PRINTER:PD</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>PostScript</td>
<td>-PRINTER:PS</td>
<td></td>
</tr>
<tr>
<td>WP</td>
<td>Windows</td>
<td>-PRINTER:WP</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**

- LP, HP, and PS produce files with the .lis extension.
- EH and HT produce .htm file output.
  - HT is controlled by the PrinterHT setting in the [Default-Settings] section of SQR.INI. If PrinterHT is set to standard, HT produces version 2.0 HTML files with the report content inside <PRE></PRE> tags. If PrinterHT is set to enhanced, HT is mapped to EH.
    (See “[Default-Settings] Section” on page 328 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...
<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
<th>Program</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-PSnn</td>
<td>Sets the TDS (Tabular Data Stream) packet size to the specified value.</td>
<td>Production Reporting Server Production</td>
<td>Sybase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reporting Execute</td>
<td></td>
</tr>
<tr>
<td>-RS</td>
<td>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 program.sqt. Note that Production Reporting does not prompt for ASK variables after compilation.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>-RT</td>
<td>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.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>-S</td>
<td>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.</td>
<td>Production Reporting Server Production</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reporting Execute</td>
<td></td>
</tr>
<tr>
<td>-Tnn</td>
<td>Specifies that you want to test your report for nn 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.</td>
<td>Production Reporting Server Production</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reporting Execute</td>
<td></td>
</tr>
<tr>
<td>-T[B]</td>
<td>Trims trailing blanks from database character columns.</td>
<td>Production Reporting Server</td>
<td>DB2 Sybase ODBC Teradata</td>
</tr>
<tr>
<td>-T[Z]</td>
<td>Trims trailing zeros from the decimal portion of numeric columns.</td>
<td>Production Reporting Server</td>
<td>DB2 Teradata</td>
</tr>
<tr>
<td>-U{priv_connectivity}</td>
<td>Directs SQR to connect to a privileged user and then proxy as the &quot;user&quot; from the normal connectivity piece. sqr (program) (connectivity) -Upriv_connectivity [flags] [args] For example: sqr scott {prg} -Upriv/priv@instance Logs in in as &quot;priv/priv@instance&quot; and then proxies to user &quot;scott&quot;. All access is then be based &quot;scott&quot; and not &quot;priv&quot;.</td>
<td>Production Reporting Server Production</td>
<td></td>
</tr>
<tr>
<td>Flag</td>
<td>Description</td>
<td>Program</td>
<td>Database</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>-Vserver</td>
<td>Uses the named server.</td>
<td>Production Reporting Server</td>
<td>Sybase</td>
</tr>
<tr>
<td>XB</td>
<td>(non-Windows) Suppresses the Production Reporting banner and the “Production Reporting... End of Run” message.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td>XC</td>
<td>(Callable Production Reporting) Suppresses the database commit when the report finishes running.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>XCB</td>
<td>(Windows) Defines to not use the communication box. Requests for input are made in Windows dialog boxes.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td>XFRM</td>
<td>Prevents Production Reporting from creating a frame in HTML files generated with -PRINTER:EH.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td>XI</td>
<td>Prevents user interaction during a program run. If ASK or INPUT requires user input, an error displays and the program ends.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>XL</td>
<td>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: sqr myprog / -xl</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td>XLFF</td>
<td>Prevents trailing form feed.</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>XMB</td>
<td>(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})</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>XNAV</td>
<td>Prevents Production Reporting from creating the Navigation Bar in HTML files generated with -PRINTER:HT and -PRINTER:EH. This</td>
<td>Production Reporting Server</td>
<td>All</td>
</tr>
<tr>
<td>Flag</td>
<td>Description</td>
<td>Program</td>
<td>Database</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>occurs when only a single HTML file is produced. Multiple HTML files generated from a single report always contain the Navigation Bar.</td>
<td>Production Reporting Execute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-XP</td>
<td>Prevents Production Reporting from creating temporary stored procedures. See “BEGIN-SELECT” on page 72 for more information.</td>
<td>Production Reporting Execute</td>
<td>Sybase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-XTB</td>
<td>Preserves trailing blanks in LIS files at the end of a line. Adamant.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-XTOC</td>
<td>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.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-ZEN(name)</td>
<td>Sets the default encoding name.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-ZIF(file)</td>
<td>Sets the full path and name of the SQR initialization file, SQR.INI.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-ZIV</td>
<td>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.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
<tr>
<td>-ZMF(file)</td>
<td>Defines the full path and name of the Production Reporting error message file, sqrerr.dat.</td>
<td>Production Reporting Execute</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production Reporting Print</td>
<td></td>
</tr>
</tbody>
</table>
Flag | Description | Program | Database
---|---|---|---
-ZRF{file} | (DDO) Sets the full path and name of an alternate registry.properties file. Following is a common default path to the registry.properties file on a Windows system: c:\program files\hyperion\properties\registry.properties The registry.properties file lists datasources that Production Reporting can access. The information in the registry.properties file makes it possible for Production Reporting to access datasources for which DDO drivers have been loaded and configured. | Production Reporting Server  
Production Reporting Execute | All

Production Reporting Command-line Arguments

Table 5  Production Reporting Command-line Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
<th>Program</th>
</tr>
</thead>
</table>
| program | 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 (\). | Production Reporting Server  
Production Reporting Execute |
| -flags | Any flag in Production Reporting Command-line Flags. | Production Reporting Server  
Production Reporting Execute  
Production Reporting Print |
| pars... | 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. | Production Reporting Server  
Production Reporting Execute |
| @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. | Production Reporting Server  
Production Reporting Execute |
| connectivity | Information needed to connect to the database. If entered as “?” or omitted, Production Reporting prompts for the information. | Production Reporting Server  
Production Reporting Execute |

DB2 | [Database]/[Username]/[Password]  
● Database—Database name.  
● Username—User name for the database.  
● Password—Password for the database. |

Informix | Database[/username/password]  
● Database—Database name.  
● Username—User name for the database.  
● Password—Password for the database. |

ODBC | Data_Source_Name[/Username][/Password] |
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
|          | ● *Data_Source_Name*—ODBC driver name.  
          | ● *Username*—User name for the database.  
          | ● *Password*—Password for the database.  
          | Note: This port has been certified against Microsoft SQL Server. |
| Oracle   | [Username]/[Password[@Database]]  
          | ● *Database*—(Optional) Database connection string (for example, @sales.2cme.com).  
          | ● *Username*—Username for the database.  
          | ● *Password*—Password for the database. |
| Sybase   | Username/[Password]  
          | ● *Username*—User name for the database.  
          | ● *Password*—Password for the database. |
| Teradata | [tdpid/]username[,password]  
          | ● *tdpid*—Teradata tdp name.  
          | ● *username*—User name for the database.  
          | ● *password*—Password for the database. |
In This Chapter

About Production Reporting Commands ................................................................. 36
ADD .......................................................................................................................... 36
ALTER-COLOR-MAP ............................................................................................... 37
ALTER-CONNECTION ............................................................................................. 39
ALTER-LOCAL .......................................................................................................... 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-PALLETTE ................................................................................. 94
CREATE-LIST ......................................................................................................... 95
CREATE-TABLE .................................................................................................... 96
#DEBUG ............................................................................................................... 99
<table>
<thead>
<tr>
<th>Statement</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOWERCASE</td>
<td>223</td>
</tr>
<tr>
<td>MBTOSBS</td>
<td>224</td>
</tr>
<tr>
<td>MOVE</td>
<td>224</td>
</tr>
<tr>
<td>MULTIPLY</td>
<td>228</td>
</tr>
<tr>
<td>NEW-PAGE</td>
<td>229</td>
</tr>
<tr>
<td>NEW-REPORT</td>
<td>230</td>
</tr>
<tr>
<td>NEXT-COLUMN</td>
<td>231</td>
</tr>
<tr>
<td>NEXT-LISTING</td>
<td>232</td>
</tr>
<tr>
<td>OPEN</td>
<td>233</td>
</tr>
<tr>
<td>OPEN-RS</td>
<td>235</td>
</tr>
<tr>
<td>PAGE-NUMBER</td>
<td>237</td>
</tr>
<tr>
<td>POSITION</td>
<td>238</td>
</tr>
<tr>
<td>PRINT</td>
<td>239</td>
</tr>
<tr>
<td>PRINT-BAR-CODE</td>
<td>260</td>
</tr>
<tr>
<td>PRINT-CHART</td>
<td>263</td>
</tr>
<tr>
<td>PRINT-DIRECT</td>
<td>270</td>
</tr>
<tr>
<td>PRINT-IMAGE</td>
<td>271</td>
</tr>
<tr>
<td>PRINT-TABLE</td>
<td>273</td>
</tr>
<tr>
<td>PUT</td>
<td>274</td>
</tr>
<tr>
<td>READ</td>
<td>275</td>
</tr>
<tr>
<td>ROLLBACK</td>
<td>277</td>
</tr>
<tr>
<td>SBTOMBS</td>
<td>278</td>
</tr>
<tr>
<td>SECURITY</td>
<td>278</td>
</tr>
<tr>
<td>SET-COLOR</td>
<td>280</td>
</tr>
<tr>
<td>SET-DELAY-PRINT</td>
<td>282</td>
</tr>
<tr>
<td>SHOW</td>
<td>283</td>
</tr>
<tr>
<td>STOP</td>
<td>287</td>
</tr>
<tr>
<td>STRING</td>
<td>288</td>
</tr>
<tr>
<td>SUBTRACT</td>
<td>289</td>
</tr>
<tr>
<td>TOC-ENTRY</td>
<td>290</td>
</tr>
<tr>
<td>UNSTRING</td>
<td>291</td>
</tr>
<tr>
<td>UPPERCASE</td>
<td>292</td>
</tr>
<tr>
<td>USE</td>
<td>293</td>
</tr>
<tr>
<td>USE-COLUMN</td>
<td>293</td>
</tr>
<tr>
<td>USE-PRINTER-TYPE</td>
<td>294</td>
</tr>
<tr>
<td>USE-PROCEDURE</td>
<td>295</td>
</tr>
<tr>
<td>USE-REPORT</td>
<td>296</td>
</tr>
<tr>
<td>WHILE</td>
<td>297</td>
</tr>
<tr>
<td>WRITE</td>
<td>299</td>
</tr>
<tr>
<td>WRITE-RS</td>
<td>300</td>
</tr>
</tbody>
</table>
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)
grays=(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, leveli1] [,dimensioni, leveli] ...])
SET-MEMBERS=([[dimension1, leveli1] [,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=pswd
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}]
```
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:

- \textbf{UPPER, LOWER}—Forces output to all uppercase or lowercase, ignoring the case of the format code in the edit mask.
- \textbf{EDIT}—Uses the case specified with the format code in the edit mask.
- \textbf{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  \textit{SYSTEM} Locale Settings

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

Examples

The following code:

```plaintext
! The following program segments will illustrate the various ALTER-LOCALE features.
!
beginsetup
declare-variable
date $date $date1 $date2 $date3
dend-declare
dsetup
!
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:
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 = strtotime('Jan 01 1995', 'Mon DD YYYY')
let $date2 = strtotime('Feb 28 1995', 'Mon DD YYYY')
let $date3 = strtotime('Mar 15 1995', 'Mon DD YYYY')
show 'System:'
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 $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 $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.
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

<table>
<thead>
<tr>
<th>Appearance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Neither bookmarks nor thumbnails are visible. (Default value when the document does not contain bookmarks.)</td>
</tr>
<tr>
<td>Bookmarks</td>
<td>Opens documents with bookmarks visible. (Default value when the document contains bookmarks.)</td>
</tr>
<tr>
<td>Thumbnails</td>
<td>Opens documents with thumbnails visible.</td>
</tr>
<tr>
<td>Fullscreen</td>
<td>Opens in full-screen mode. (This value does not work in the browser.)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>Document subject.</td>
</tr>
<tr>
<td>Title</td>
<td>Document title. (The default is the product name and version string.)</td>
</tr>
<tr>
<td>Creator</td>
<td>Software used to create the document.</td>
</tr>
<tr>
<td>Author</td>
<td>Document author.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Keywords describing the document contents.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Open Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed</td>
<td>Use a fixed destination view defined with Zoom, Left, and Top.</td>
</tr>
</tbody>
</table>
### Table 10  Additional Values

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

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

<table>
<thead>
<tr>
<th>Transition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split</td>
<td>Two lines sweeping across the screen reveal the page.</td>
</tr>
<tr>
<td>Blinds</td>
<td>Multiple lines sweeping across the screen reveal the page.</td>
</tr>
<tr>
<td>Box</td>
<td>A box reveals the page.</td>
</tr>
<tr>
<td>Wipe</td>
<td>A single line sweeping across the screen reveals the page.</td>
</tr>
<tr>
<td>Dissolve</td>
<td>The old page dissolves to reveal the new page.</td>
</tr>
<tr>
<td>Glitter</td>
<td>The dissolve effect moves from one screen edge to another.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Security</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-Password</td>
<td>User level password applied to the generated PDF. Up to 32 characters. Needed to open the document.</td>
</tr>
<tr>
<td>Master-Password</td>
<td>Master level password applied to the generated PDF. Up to 32 characters. Used to open documents or override the permissions.</td>
</tr>
<tr>
<td>Permissions</td>
<td>Permissions applied to the generated PDF.</td>
</tr>
<tr>
<td></td>
<td>● NoPrint—Prevents printing the file.</td>
</tr>
<tr>
<td></td>
<td>● NoModify—Prevents users from adding form fields or making any other changes.</td>
</tr>
<tr>
<td></td>
<td>● NoCopy—Prevents copying and extracting text and graphics and disables the accessibility interface.</td>
</tr>
<tr>
<td></td>
<td>● NoAnnots—Prevents adding or changing comments or form fields.</td>
</tr>
<tr>
<td></td>
<td>● NoForm—Prevents form field filling, even if NoAnnots is not specified. (Requires Acrobat 5 or higher)</td>
</tr>
<tr>
<td></td>
<td>● NoAccessible—Prevents extracting text or graphics for accessibility purposes. For example, a screen reader program.(Requires Acrobat 5 or higher)</td>
</tr>
<tr>
<td></td>
<td>● NoAssemble—Prevents inserting, deleting, or rotating pages and creating bookmarks and thumbnails, even if NoModify is not specified. (Requires Acrobat 5 or higher)</td>
</tr>
<tr>
<td></td>
<td>● NoHiResPrint—Prevents high-resolution printing. If NoPrint is not specified, printing this setting is restricted to the Print As Image feature, which prints a low-resolution rendition of the page. (Requires Acrobat 5 or higher)</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Preference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toolbar</td>
<td>True hides Acrobat’s tool bar. Default = False</td>
</tr>
<tr>
<td>MenuBar</td>
<td>True hides Acrobat’s menu bar. Default = False</td>
</tr>
<tr>
<td>WindowUI</td>
<td>True hides Acrobat’s windows controls. Default = False</td>
</tr>
<tr>
<td>FitWindow</td>
<td>True resizes the document’s window to the size of the first page. Default = False</td>
</tr>
<tr>
<td>CenterWindow</td>
<td>True positions the document’s windows in the center of the screen. Default = False</td>
</tr>
<tr>
<td>DisplayDocTitle</td>
<td>True displays the document information field in Acrobat’s title bar; False displays the file name. Default = False</td>
</tr>
<tr>
<td>NonFullscreen-PageMode</td>
<td>How to display the document on exiting full-screen mode:</td>
</tr>
<tr>
<td></td>
<td>● UseOutlines—Displays page and document outlines.</td>
</tr>
<tr>
<td></td>
<td>● UseThumbs—Displays page and thumbnails.</td>
</tr>
<tr>
<td></td>
<td>● UseNone—Displays neither document outlines nor thumbnails.</td>
</tr>
<tr>
<td></td>
<td>Default = UseNone</td>
</tr>
<tr>
<td>Direction</td>
<td>Reading order of the document. (Affects scroll ordering in double-page view.)</td>
</tr>
<tr>
<td></td>
<td>● L2R—Left to right</td>
</tr>
<tr>
<td></td>
<td>● R2L—Right to left (including vertical writing systems).</td>
</tr>
<tr>
<td></td>
<td>Default = L2R</td>
</tr>
</tbody>
</table>

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
```
See Also

BEGIN-FOOTING and BEGIN-HEADING

ALTER-TABLE

Function

Manipulates table attributes.

Syntax

\[
\text{ALTER-TABLE} \\
\text{NAME=} \text{table\_name\_var|_lit|_col} \\
\text{ACTION=} \text{action\_lit} \\
[\text{COUNT=} \text{count\_var|_lit|_col}] \\
[\text{ROW=} \text{row\_var|_lit|_col}] \\
[\text{ATTRIBUTES=} (\text{keyword1}, \text{value1}, \ldots, \text{keywordn}, \text{valuen})]
\]

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>
<thead>
<tr>
<th>Table 14</th>
<th>ALTER-TABLE Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>Causes all attributes to be set to their default values as defined by DECLARE-TABLE and CREATE-TABLE before applying any other attributes.</td>
</tr>
<tr>
<td>BACKGROUND</td>
<td>Background color name or RGB triplet</td>
</tr>
<tr>
<td>BOLD</td>
<td>YES</td>
</tr>
<tr>
<td>CENTER</td>
<td>YES</td>
</tr>
<tr>
<td>COLUMN-LEADING</td>
<td>Expressed in decipoints</td>
</tr>
<tr>
<td>COLUMN-LINE COLOR</td>
<td>Color name or RGB triplet</td>
</tr>
<tr>
<td>COLUMN-LINE-THICKNESS</td>
<td>Expressed in decipoints</td>
</tr>
<tr>
<td>COLUMN-LINE-STYLE</td>
<td>SOLID</td>
</tr>
<tr>
<td>HEADER</td>
<td>YES</td>
</tr>
<tr>
<td>FILL-COLOR</td>
<td>Fill color name or RGB triplet. Default=NONE</td>
</tr>
<tr>
<td>FONT</td>
<td>Font number</td>
</tr>
<tr>
<td>FOOTER</td>
<td>YES</td>
</tr>
<tr>
<td>FOREGROUND</td>
<td>Foreground color name or RGB triplet</td>
</tr>
<tr>
<td>GROUP-HEADER</td>
<td>YES</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>GROUP-FOOTER</td>
<td>YES</td>
</tr>
<tr>
<td>ITALIC</td>
<td>YES</td>
</tr>
<tr>
<td>POINT-SIZE</td>
<td>Point size of the font</td>
</tr>
<tr>
<td>ROW-BORDER-COLOR</td>
<td>Color name or RGB triplet</td>
</tr>
<tr>
<td>ROW-BORDER-THICKNESS</td>
<td>Expressed in decipoints</td>
</tr>
<tr>
<td>ROW-LINE-COLOR</td>
<td>Color name or RGB triplet</td>
</tr>
<tr>
<td>ROW-LINE-STYLE</td>
<td>SOLID</td>
</tr>
<tr>
<td>ROW-LINE-THICKNESS</td>
<td>Expressed in decipoints</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>YES</td>
</tr>
<tr>
<td>WRAP</td>
<td>YES</td>
</tr>
<tr>
<td>WRAP-HEIGHT</td>
<td>Number of lines between each wrapped line</td>
</tr>
<tr>
<td>WRAP-ON</td>
<td>Characters on which to force a WRAP</td>
</tr>
<tr>
<td>WRAP-STRIP</td>
<td>Characters to change to a space before the WRAP is done</td>
</tr>
</tbody>
</table>

**Description**

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

**Example**

```sql
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' (+1, 25) bold underline

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

Function
Retrieves values for compile-time substitution variables. Retreival 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
Function
Begin 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)
Dear $firstname

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]...)]]
  [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:

```plaintext
XML_DATA.desc=Sample XML files
XML_DATA.class=com.sqribe.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:

```xml
<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>Everretsville</CUSTOMERS.CITY>
        <CUSTOMERS.STATE>OH</CUSTOMERS.STATE>
        <CUSTOMERS.ZIP>40231000</CUSTOMERS.ZIP>
        <CUSTOMERSPHONE>2165531109</CUSTOMERSPHONE>
        <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
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 SYYYYMMDD [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


\{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:

'\(DSN=ora8;UID=scott;PWD=tiger\)'

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.

\(LOOPS\)

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
   [{$col_var_char}] loops=100  
   &coll1=char
end-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.
(ODBC) The ODBC connection string for this SQL paragraph only. A connection string has the following syntax:

```markdown
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>
<thead>
<tr>
<th>System</th>
<th>Value Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX</td>
<td>Zero (0) indicates success. Any other value is the system error code.</td>
</tr>
<tr>
<td>PC/Windows</td>
<td>A value less than 32 indicates an error.</td>
</tr>
</tbody>
</table>

**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:

```plaintext
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>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
<th>How Passed</th>
</tr>
</thead>
<tbody>
<tr>
<td>callname</td>
<td>Subroutine name.</td>
<td>By reference with a maximum of 31 characters, null terminated.</td>
</tr>
<tr>
<td>strsrc</td>
<td>Source string.</td>
<td>By reference with a maximum of 511 characters, null terminated.</td>
</tr>
<tr>
<td>strdes</td>
<td>Destination string.</td>
<td>By reference with a maximum of 511 characters, null terminated.</td>
</tr>
<tr>
<td>dblsrc</td>
<td>Source double floating point.</td>
<td>By reference.</td>
</tr>
<tr>
<td>dbldes</td>
<td>Destination double floating point.</td>
<td>By reference.</td>
</tr>
<tr>
<td>param</td>
<td>Subroutine parameter string. It must be a literal.</td>
<td>By reference with a maximum of 2047 characters, null terminated.</td>
</tr>
</tbody>
</table>

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`.  

Table 16 UCALL Subroutine Arguments
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:

```plaintext
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.

```plaintext
! 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:

```c
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.

```c
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.

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
```
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**

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

For Sybase:

```sql
...  ! 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

\texttt{CONCAT \{} src\_any\_lit|\_var|\_col \texttt{WITH} dst\_txt\_var\{[:$]edit\_mask\}\texttt{\}}

Arguments

\texttt{src\_any\_lit|\_var|\_col}
Source field to concatenate with \texttt{dst\_txt\_var}.

\texttt{dst\_txt\_var}
Result after execution.

\texttt{edit\_mask}
Optional edit mask.

Description

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

\texttt{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 \texttt{DATETIME} columns and Production Reporting \texttt{DATE} variables, Production Reporting uses the format specified by \texttt{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 \texttt{DATE} columns, Production Reporting uses the format specified by \texttt{SQR\_DB\_DATE\_ONLY\_FORMAT}. If not set, Production Reporting uses the format in Table 62, “DATE Column Formats,” on page 252.

- For \texttt{TIME} columns, Production Reporting uses the format specified by \texttt{SQR\_DB\_TIME\_ONLY\_FORMAT}. If not set, Production Reporting uses the format in Table 63, “TIME Column Formats,” on page 252.

Examples

\texttt{concat \&zip\_plus\_4 with \$zip \'-xxxx' ! Edit zip plus 4.}
\texttt{concat \&descr with \$rec :\$desc\_edit ! Edit mask in variable.}
\texttt{concat \$date1 with \$string ! Concatenate a date.}

See Also

- \texttt{PRINT} or information on edit masks
- \texttt{LET} for string functions.
- \texttt{STRING} and \texttt{UNSTRING}
### CONNECT

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

**Syntax**

```sql
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**

```sql
connect $new-user on-error=bad-logon($new_user)
```
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[NNNNNN]]]]'. 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**

![Sample Array with Three Fields](image)

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

```
create-array name=emps size=10
  field=name:char='Unknown'
```
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 \(c\) 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/Instruction}),{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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKGROUND</td>
<td>Background color name or RGB triplet. Default=NONE.</td>
</tr>
<tr>
<td>BOLD</td>
<td>YES</td>
</tr>
<tr>
<td>CENTER</td>
<td>YES</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>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.</td>
</tr>
<tr>
<td>FILL-COLOR</td>
<td>Fill color name or RGB triplet. Default=NONE</td>
</tr>
<tr>
<td>FONT</td>
<td>Font number. (Must be defined.)</td>
</tr>
<tr>
<td>FOREGROUND</td>
<td>Foreground color name or RGB triplet. Default=BLACK.</td>
</tr>
<tr>
<td>ITALIC</td>
<td>YES</td>
</tr>
<tr>
<td>LEADING</td>
<td>Expressed in decipoints</td>
</tr>
<tr>
<td>LINE-COLOR</td>
<td>Color name or RGB triplet of column line (line after column). Default=NONE (no line).</td>
</tr>
<tr>
<td>LINE-STYLE</td>
<td>Column line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT,LONG-DASH-DOT-DOT). Default=SOLID.</td>
</tr>
<tr>
<td>LINE-THICKNESS</td>
<td>Column line thickness expressed in decipoints. Default=two decipoints.</td>
</tr>
<tr>
<td>POINT-SIZE</td>
<td>Point size of the font. (Must be specified.)</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>YES</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Width expressed in coordinate units. (Must be specified.)</td>
</tr>
<tr>
<td>WRAP</td>
<td>YES</td>
</tr>
<tr>
<td>WRAP-HEIGHT</td>
<td>Number of lines between each wrapped line. Default=one line.</td>
</tr>
<tr>
<td>WRAP-ON</td>
<td>Characters on which to force a WRAP. The default is not to force a WRAP.</td>
</tr>
<tr>
<td>WRAP-STRIP</td>
<td>Characters to change to a space before the WRAP is done. The default is not to strip any characters.</td>
</tr>
</tbody>
</table>
ROW–ATTRIBUTES

Attributes to apply to rows.

Table 18  Row Attributes

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

TABLE–ATTRIBUTES

Attributes for the appearance of the table.

Table 19  Table Attributes

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

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**

```plaintext
#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 "#debuge", 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.'
#debuge 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-PALLETTE=color_palette_lit]
    [POINT-MARKERS=point_markers_lit]
    [ATTRIBUTES={selector_lit|
        LIST:{selector_list_name_lit|(selector_lit,...)},
    ]
```
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>
<thead>
<tr>
<th>Argument</th>
<th>Choices</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3D-EFFECTS</strong></td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Argument</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ATTRIBUTES</strong></td>
<td>See &quot;Attributes Argument&quot; on page 112 for information on the valid choices.</td>
<td>Defines the appearance of a chart. 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. 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. <strong>Note</strong>: 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.</td>
</tr>
<tr>
<td><strong>BORDER</strong></td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>chart_name</strong></td>
<td>User defined chart name.</td>
<td>Name for referencing a chart.</td>
</tr>
<tr>
<td><strong>CHART-SIZE</strong></td>
<td>User defined chart size.</td>
<td>Size of the chart frame in standard Production Reporting coordinate units.</td>
</tr>
<tr>
<td><strong>COLOR-PALETTE</strong></td>
<td>palette_name</td>
<td>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. <strong>Note</strong>: The defined color palette is only valid when NewGraphics=TRUE. <strong>Note</strong>: 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.</td>
</tr>
<tr>
<td>Argument</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DATA-ARRAY Y2-DATA-ARRAY</td>
<td>array_name</td>
<td>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.</td>
</tr>
<tr>
<td>DATA-ARRAY-ROW-COUNT</td>
<td>row_count</td>
<td>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.</td>
</tr>
<tr>
<td>Y2-DATA-ARRAY-ROW-COUNT</td>
<td>row_count</td>
<td>(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.</td>
</tr>
<tr>
<td>DATA-ARRAY-COLUMN-COUNT</td>
<td>column_count</td>
<td>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.</td>
</tr>
<tr>
<td>Y2-DATA-ARRAY-COLUMN-COUNT</td>
<td>column_count</td>
<td>(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.</td>
</tr>
<tr>
<td>DATA-ARRAY-COLUMN-LABELS</td>
<td>NONE</td>
<td>array_name</td>
</tr>
<tr>
<td>Y2-DATA-ARRAY-COLUMN-LABELS</td>
<td>NONE</td>
<td>array_name</td>
</tr>
<tr>
<td>DATA-LABELS</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>FILL</td>
<td>GRAYSCALE</td>
<td>COLOR</td>
</tr>
<tr>
<td>Argument</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Argument</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>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. <strong>Note:</strong> 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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOOTER-TEXT</td>
<td>NONE</td>
<td>text</td>
</tr>
<tr>
<td>ITEM-COLOR</td>
<td>ChartBackgroundColor</td>
<td>Background color of entire chart area.</td>
</tr>
<tr>
<td>ChartForegroundColor</td>
<td>Text and Line color of chart area.</td>
<td></td>
</tr>
<tr>
<td>HeaderBackgroundColor</td>
<td>Area within the text box specified for the title and sub-title.</td>
<td></td>
</tr>
<tr>
<td>HeaderForeground</td>
<td>Text color of the Title and sub-title.</td>
<td></td>
</tr>
<tr>
<td>LegendBackgroundColor</td>
<td>Area within the box defining the legend.</td>
<td></td>
</tr>
<tr>
<td>LegendForeground</td>
<td>Text and Outline color of the legend.</td>
<td></td>
</tr>
<tr>
<td>ChartAreaBackgroundColor</td>
<td>Area that includes the body of the chart.</td>
<td></td>
</tr>
<tr>
<td>ChartAreaForeground</td>
<td>Text and Line colors of the chart area.</td>
<td></td>
</tr>
<tr>
<td>PlotAreaBackgroundColor</td>
<td>Area within the X and Y Axis of a chart.</td>
<td></td>
</tr>
<tr>
<td>ITEM-SIZE</td>
<td>Title</td>
<td>SubTitle</td>
</tr>
<tr>
<td>(NewGraphics) Size of the following chart objects. The value is based on HTML text sizes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● Title—Chart title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITEM-SIZE=&quot;('Title',value)&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● SubTitle—Chart subtitle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITEM-SIZE=&quot;('SubTitle',value)&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Argument</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>XAxisLabel</td>
<td></td>
<td>● XAxisLabel—Label below the X Axis of the chart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITEM-SIZE= ('XAxisLabel',value)</td>
</tr>
<tr>
<td>XAxisMarkers</td>
<td></td>
<td>● XAxisMarkers—Point labels on the X Axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITEM-SIZE= ('XAxisMarkers',value)</td>
</tr>
<tr>
<td>YAxisLabel</td>
<td></td>
<td>● YAxisLabel—Rotated label to the left of the chart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITEM-SIZE= ('YAxisLabel',value)</td>
</tr>
<tr>
<td>YAxisMarkers</td>
<td></td>
<td>● YAxisMarkers—Point labels on the Y Axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITEM-SIZE= ('YAxisMarkers',value)</td>
</tr>
<tr>
<td>Y2AxisLabel</td>
<td></td>
<td>● Y2AxisLabel—Rotated Label to the right of the chart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITEM-SIZE= ('Y2AxisLabel',value)</td>
</tr>
<tr>
<td>Y2AxisMarkers</td>
<td></td>
<td>● Y2AxisMarkers—Point labels on the Y Axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITEM-SIZE= ('Y2AxisMarkers',value)</td>
</tr>
<tr>
<td>LegendText</td>
<td></td>
<td>● LegendText—Legend text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITEM-SIZE= ('LegendText',value)</td>
</tr>
<tr>
<td>LegendTitle</td>
<td></td>
<td>● LegendTitle—Legend title</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITEM-SIZE= ('LegendTitle',value)</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>If you do not define an ITEM-SIZE value, Production Reporting uses the HTML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>text value of 3 (normal size).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The POINT-SIZE declaration keyword in the ATTRIBUTES argument replaces the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>functionality provided by ITEM-SIZE. As a result, values set with POINT-SIZE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>override values set with ITEM-SIZE.</td>
</tr>
<tr>
<td>LEGEND</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>LEGEND-PLACEMENT</td>
<td>CENTER-RIGHT</td>
<td>CENTER-LEFT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: The LOCATION declaration keyword in the ATTRIBUTES argument replaces the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>functionality provided by LEGEND-PLACEMENT. As a result, values set with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOCATION override values set with LEGEND-PLACEMENT.</td>
</tr>
<tr>
<td>LEGEND-PRESENTATION</td>
<td>INSIDE</td>
<td>OUTSIDE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>axes. If OUTSIDE, then the legend is presented within the chart border, but</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outside of the region represented by the two axes.</td>
</tr>
<tr>
<td>Argument</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LEGEND-TITLE</td>
<td>NONE</td>
<td>text</td>
</tr>
<tr>
<td>PIE-SEGMENT-EXPLODE</td>
<td>NONE</td>
<td>MAX</td>
</tr>
<tr>
<td>PIE-SEGMENT-PERCENT-DISPLAY</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>PIE-SEGMENT-QUANTITY-DISPLAY</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>POINT-MARKERS</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>SUB-FOOTER-TEXT</td>
<td>NONE</td>
<td>text</td>
</tr>
<tr>
<td>SUB-TITLE</td>
<td>NONE</td>
<td>text</td>
</tr>
<tr>
<td>TITLE</td>
<td>NONE</td>
<td>text</td>
</tr>
<tr>
<td>TYPE Y2-TYPE (no pie charts)</td>
<td>PIE</td>
<td>BAR</td>
</tr>
<tr>
<td>Y2-AXIS-COLOR-PALETTE</td>
<td>palette_name</td>
<td>(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.</td>
</tr>
<tr>
<td>X-AXIS-G2 Y-AXIS-G2 Y2-AXIS-G2</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Argument</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>X-AXIS-LABEL  Y-AXIS-LABEL  Y2-AXIS-LABEL</td>
<td>NONE</td>
<td>text</td>
</tr>
<tr>
<td>X-AXIS-MAX-VALUE</td>
<td>AUTOSCALE</td>
<td>number</td>
</tr>
<tr>
<td>Y-AXIS-MAX-VALUE</td>
<td>AUTOSCALE</td>
<td>number</td>
</tr>
<tr>
<td>Y2-AXIS-MAX-VALUE</td>
<td>AUTOSCALE</td>
<td>number</td>
</tr>
<tr>
<td>X-AXIS-MIN-VALUE</td>
<td>AUTOSCALE</td>
<td>number</td>
</tr>
<tr>
<td>Y-AXIS-MIN-VALUE</td>
<td>AUTOSCALE</td>
<td>number</td>
</tr>
<tr>
<td>Y2-AXIS-MIN-VALUE</td>
<td>AUTOSCALE</td>
<td>number</td>
</tr>
<tr>
<td>X-AXIS-MAJOR-TICK-MARKS</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Y-AXIS-MAJOR-TICK-MARKS</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Y2-AXIS-MAJOR-TICK-MARKS</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Y-AXIS-MASK</td>
<td>‘$999,999.99’</td>
<td></td>
</tr>
<tr>
<td>Argument</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Y2-AXIS-MASK</td>
<td>‘099999’</td>
<td>(NewGraphics) Numeric mask used to format the Y2 Axis. Follows the edit mask rules defined in Table 56.</td>
</tr>
<tr>
<td>X-AXIS-MINOR-TICK-MARKS</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Y-AXIS-MINOR-TICK-MARKS</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Y2-AXIS-MINOR-TICK-MARKS</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>X-AXIS-MAJOR-INCREMENT Y-AXIS-MAJOR-INCREMENT Y2-AXIS-MAJOR-INCREMENT</td>
<td>AUTOSCALE</td>
<td>number</td>
</tr>
<tr>
<td>X-AXIS-MINOR-INCREMENT Y-AXIS-MINOR-INCREMENT Y2-AXIS-MINOR-INCREMENT</td>
<td>number</td>
<td>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)</td>
</tr>
<tr>
<td>X-AXIS-ROTATE</td>
<td>X-Axis-Rotate = 0</td>
<td>No Rotation</td>
</tr>
<tr>
<td>X-AXIS-TICK-MARK-PLACEMENT Y-AXIS-TICK-MARK-PLACEMENT</td>
<td>INSIDE</td>
<td>OUTSIDE</td>
</tr>
<tr>
<td>X-AXIS-SCALE Y-AXIS-SCALE Y2-AXIS-SCALE</td>
<td>LOG</td>
<td>LINEAR</td>
</tr>
</tbody>
</table>
**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.

```plaintext
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)
```

110  Production Reporting Command Reference
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**

<table>
<thead>
<tr>
<th>CHART.FILL</th>
<th>PRINTER.COLOR=Y</th>
<th>PRINTER.COLOR=N</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAYSCALE</td>
<td>GRAYSCALE</td>
<td>GRAYSCALE</td>
</tr>
<tr>
<td>COLOR</td>
<td>COLOR</td>
<td>GRAYSCALE</td>
</tr>
<tr>
<td>CROSSHATCH</td>
<td>COLOR-CROSSHATCH</td>
<td>CROSSHATCH</td>
</tr>
<tr>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
</tbody>
</table>

**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.

```plaintext
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'
```

```plaintext
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-title = '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 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.
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.)


**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

<table>
<thead>
<tr>
<th>Sub_SELECTOR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FILL</strong></td>
<td>Color of the fill area on a chart.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>ATTRIBUTES=('fill','style','verticalstripe')</code></td>
</tr>
<tr>
<td></td>
<td>Creates a fill style of VerticalStripe for all charts.</td>
</tr>
<tr>
<td><strong>GRADIENT</strong></td>
<td>Color of the gradient fill area on a chart.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>ATTRIBUTES=('chart1.gradient','color','none')</code></td>
</tr>
<tr>
<td></td>
<td>Declares that fill areas are not gradient.</td>
</tr>
<tr>
<td></td>
<td><code>ATTRIBUTES=('chart2.gradient','color','black')</code></td>
</tr>
<tr>
<td></td>
<td>Declares that fill areas are gradient and that black will be used to create the fill area.</td>
</tr>
<tr>
<td><strong>GRID</strong></td>
<td>Grid attributes (color, style, and line thickness) on a chart.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>ATTRIBUTES=('x-axis.grid','style','solid')</code></td>
</tr>
<tr>
<td></td>
<td>Sets the style of the x-axis grid to a solid line.</td>
</tr>
<tr>
<td>Sub_SELECTOR</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HILO-LINE</td>
<td>For HiLo and Candle charts, defines color, style, and line thickness of the Hi-Low line.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>ATTRIBUTES=('chart1.hilo-line', 'color', ('blue'))</td>
</tr>
<tr>
<td></td>
<td>Sets the HILO-LINE (line) color to blue.</td>
</tr>
<tr>
<td>LABEL</td>
<td>Label attributes.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>ATTRIBUTES=('x-axis.label', 'foreground', ('yellow'))</td>
</tr>
<tr>
<td></td>
<td>Sets the X-Axis label to the colors specified for all charts.</td>
</tr>
<tr>
<td>LINE</td>
<td>For Line charts, defines color, style, and thickness of data lines.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>ATTRIBUTES=('chart1.line', 'color', list:(('red'), ('blue'), ('green')))</td>
</tr>
<tr>
<td></td>
<td>Sets the first, second, and third data lines to red, blue, and green,  respectively.</td>
</tr>
<tr>
<td>MARKER</td>
<td>Marker label attributes.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>attributes=('X-axis.marker', 'point-size', 0.1)</td>
</tr>
<tr>
<td></td>
<td>Sets the point size for X-axis markers to ten.</td>
</tr>
<tr>
<td>OTHER</td>
<td>For Pie charts, defines the “Other” pie chart slice.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>attributes=('other', 'color', ('red'))</td>
</tr>
<tr>
<td></td>
<td>Sets the “Other” pie slice to red.</td>
</tr>
<tr>
<td>OUTLINE</td>
<td>Attributes (color, style, border thickness) for outlines (bars, pie slices, legend, and legend symbols).</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>ATTRIBUTES=('chart1.outline', 'style', 'shortdash')</td>
</tr>
<tr>
<td></td>
<td>Sets the outline of chart areas to be a short dashed line.</td>
</tr>
<tr>
<td>RISING-FILL/FALLING-FILL</td>
<td>For Candle charts, attributes (fill pattern, color, width of candle) for the fill area in the rising and falling candle.</td>
</tr>
<tr>
<td>Sub_SELECTOR</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 23  Valid Selector/Sub-selector Combinations

<table>
<thead>
<tr>
<th>MARKER</th>
<th>LABEL</th>
<th>LINE</th>
<th>GRID</th>
<th>SYMBOL</th>
<th>OTHER</th>
<th>OUTLINE</th>
<th>GRADIENT</th>
<th>FILL</th>
<th>HILO-LINE</th>
<th>RISING-FILL</th>
<th>FALLING-FILL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>CHART1</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>CHART2</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>ALL-AXIS</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X-AXIS</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y-AXIS</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y2-AXIS</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEGEND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>LEGEND-SYMBOL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Declaration Keyword</th>
<th>Choices</th>
<th>Description</th>
</tr>
</thead>
</table>
| 3D-DEPTH            | 0 – 100 | Depth of the three-dimensional display in percent.  
|                     | Default = 10 |  
|                     | Note: You must set the 3d-EFFECTS argument to YES for this keyword to work.  
|                     | Example:     | ATTRIBUTES=('All', '3D-DEPTH', 15)  
<p>|                     | Displays all three-dimensional charts with a depth of 15 percent. |</p>
<table>
<thead>
<tr>
<th>Declaration Keyword</th>
<th>Choices</th>
<th>Description</th>
</tr>
</thead>
</table>
| **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:**  
ATTRIBUTES=('All','3D-ELEVATION',45)  
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:**  
ATTRIBUTES=('All','3D-ROTATION',50)  
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:**  
ATTRIBUTES=('All','BackGround',('White'))  
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:**  
ATTRIBUTES=('All','CLUSTER-OVERLAP',45)  
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:**  
ATTRIBUTES=('All','CLUSTER-WIDTH',60)  
Displays all bar charts with a width of 60 percent between each bar cluster. |
<table>
<thead>
<tr>
<th>Declaration Keyword</th>
<th>Choices</th>
<th>Description</th>
</tr>
</thead>
</table>
| COLOR               | List generated from CREATE-LIST, or an in-line list of values. 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. | Defines a single color or a group of colors for an individual chart element. 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. An error occurs if the contents of the list do not match the data type expected. COLOR can reference a color palette or a single color. **Example:**

ATTRIBUTES= ('CHART2.LINE', 'COLOR', LIST: (('red'), ('yellow'), ('maroon')))

Creates a group of line colors of 'red', 'yellow' and 'maroon' for chart2.

ATTRIBUTES= ('OTHER', 'COLOR', ('red'))

Sets the color for the 'Other' pie chart slice to 'red' for all pie charts. **Note:** COLOR replaces the functionality provided by ITEM-COLOR. |
| FONT                | Values specified under [Fonts] in SQR.INI. Each entry consists of a font number assigned to a named font. For example, 3 may represent Courier. Default = Times New Roman | Font for all text and/or for specific text areas in a chart image. **Example:**

ATTRIBUTES= ('ALL', 'Font', 3, 'Point-Size', 12)

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. |
| FONT-STYLE          | PLAIN | BOLD | ITALIC | UNDERLINE |
|                     | Default = PLAIN | Style of the font for all text and/or for specific text areas in a chart image. Separate multiple values with a space. **Example:**

ATTRIBUTES= ('Title', 'Font-Style', 'Bold Underline')

Sets the font style to 'BOLD' and 'Underline' for all charts. |
| FOREGROUND          | Named color or values in the range of RGB. Default = Black | Text, outline, and line color for a chart area. **Example:**
<table>
<thead>
<tr>
<th>Declaration Keyword</th>
<th>Choices</th>
<th>Description</th>
</tr>
</thead>
</table>
| **DECLARE-COLOR-MAP** | See "DECLARE-COLOR-MAP" on page 126 in the for an explanation of RGB values. | Attributes=('X-Axis.Label', 'ForeGround', ('Yellow'))
Sets the X-Axis label for all charts to yellow. **Note**: Along with BACKGROUND, FOREGROUND replaces the functionality provided by COLOR-PALETTE. |
<p>| <strong>HALF-RANGE</strong> | YES | NO Default = NO Determines how the x-axis is displayed in Polar charts. NO—X-axis is displayed as one full range from 0 to 360 degrees. YES—X-axis is displayed as two half-ranges from −180 to 180 degrees. <strong>Example</strong>: Attributes=('Chart1', 'HALF-RANGE', 'Yes') Sets the axis label range from -180 to 180 degrees. |
| <strong>HOLE-VALUE</strong> | A value between the following minimum and maximum values: Min Value: -1.7976931348623157E+308 Max Value: 1.7976931348623157E+308 Default = None Value in the data to ignore. You can only have one HOLE-VALUE per chart. <strong>Example</strong>: Attributes=('All', 'HOLE-VALUE', -1) 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. |
| <strong>LABEL-LOCATION</strong> | INNER | OUTER | AUTO Location of labels in pie charts. INNER—Labels are placed within the chart area. OUTER—Labels are placed outside the chart area. AUTO—The label location is controlled by the charting application. <strong>Example</strong>: Attributes=('All', 'LABEL-LOCATION', 'OUTER') Displays the pie chart labels outside of the chart area. |
| <strong>LEGEND-COLUMNS</strong> | Any numeric value. Default = 0 A value of 0 means the charting application will determine the proper value. Number of columns to use when generating the chart legend. You can only have one LEGEND-COLUMNS value per chart. <strong>Example</strong>: |</p>
<table>
<thead>
<tr>
<th>Declaration Keyword</th>
<th>Choices</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEGEND-ROWS</td>
<td>Any numeric value. &lt;br&gt;Default = 0 &lt;br&gt;A value of 0 means the charting application will determine the proper value.</td>
<td>Number of rows to use when generating the chart legend. You can only have one LEGEND-ROWS value per chart. &lt;br&gt;&lt;b&gt;Example:&lt;/b&gt; &lt;br&gt;ATTRIBUTES=({'All', 'LEGEND-ROWS', 2}) &lt;br&gt;Displays the chart legend with two rows.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>LOWER-RIGHT</td>
<td>CENTER-RIGHT</td>
</tr>
<tr>
<td>ORIGIN-BASE-ANGLE</td>
<td>0 – 360 &lt;br&gt;Default = 0</td>
<td>Position of the x-axis for Polar, Radar, and Area Radar charts in degrees. &lt;br&gt;&lt;b&gt;Example:&lt;/b&gt; &lt;br&gt;ATTRIBUTES=({'CHART1', 'ORIGIN-BASE-ANGLE', '90'}) &lt;br&gt;Sets the ORIGIN-BASE-ANGLE to 90 degrees.</td>
</tr>
<tr>
<td>OTHER-LABEL</td>
<td>Any value. &lt;br&gt;Default = Other</td>
<td>Name of the label used for the “Other” pie chart segment. &lt;br&gt;&lt;b&gt;Example:&lt;/b&gt; &lt;br&gt;ATTRIBUTES=({'ALL', 'OTHER-LABEL', 'OTHER-PRODUCTS'}) &lt;br&gt;Declares the label used for the “Other” pie chart segment is “Other-Products”.</td>
</tr>
<tr>
<td>PATTERN</td>
<td>CIRCULAR</td>
<td>WEBBED</td>
</tr>
<tr>
<td>Declaration Keyword</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **PERCENTAGE-PRECISION** | 0 – 15  | Number of digits to the right of the decimal point in a pie chart.  
Example:  
ATTRIBUTES= ('All', 'PERCENTAGE-PRECISION', 5)  
Displays the pie chart percentage with five digits to the right of the decimal point. |
| **POINT-SIZE** | Any point size.  
Default = 12 | 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.  
Example:  
ATTRIBUTES= ('Title', 'Style', 'Bold Underline', 'Point-Size', 16)  
Sets the font style to ‘Bold’ and ‘Underline’ and font size to 16 point for the ‘Title’ text item for all charts.  
Note: POINT-SIZE replaces the functionality provided by ITEM-SIZE. |
| **SIZE** | Depends on the SUB-SELECTOR:  
- LINE – 1 to 10 pixels  
- SYMBOL – 1 to 100 pixels  
- GRID – 1 to 10 pixels | Size of the lines, symbols, and grids for a chart element.  
The SIZE keyword overrides the default values established for the size property of a chart.  
The default line and grid width is 1 pixel. The default symbol width is a bounding box of 6x6 pixels.  
Example:  
ATTRIBUTES= ('CHART1.LINE', 'SIZE', 2)  
Sets the line size for all lines to 2 pixels for chart1. |
| **SORT-ORDER** | LARGEST | Defines whether to display pie-chart slices largest-to-smallest, smallest-to-largest, or the order they appear in the data.  
- LARGEST—Largest-to-smallest  
- SMALLEST—Smallest-to-largest  
- DATAORDER—Order they appear in the data. |
<table>
<thead>
<tr>
<th>Declaration Keyword</th>
<th>Choices</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATTRIBUTES</strong></td>
<td>('All', 'SORT-ORDER', 'Largest')</td>
<td>Sets the display order of the pie slices from largest to smallest for all charts.</td>
</tr>
<tr>
<td><strong>START-ANGLE</strong></td>
<td>0 - 359&lt;br&gt;Default = 0</td>
<td><strong>For Pie Charts:</strong>&lt;br&gt;Position in the pie chart where the first pie slice is drawn.&lt;br&gt;A value of zero degrees represents a horizontal line from the center of the pie to the right-hand side of the pie chart.&lt;br&gt;A value of 90 degrees represents a vertical line from the center to the top of the pie.&lt;br&gt;Slices are drawn clockwise from the specified angle.&lt;br&gt;The default position for the first pie segment is 90 degrees.&lt;br&gt;<strong>Example:</strong>&lt;br&gt;ATTRIBUTES=('All','START-ANGLE',45)&lt;br&gt;Sets the starting location of the first pie slice to a line 45 degrees to the right and down of a horizontal line.&lt;br&gt;<strong>For Polar, Radar, and Area Radar Charts:</strong>&lt;br&gt;Angle that the y-axis makes with the ORIGIN-BASE-ANGLE.&lt;br&gt;<strong>Example:</strong>&lt;br&gt;ATTRIBUTES=('Chart1','START-ANGLE',135)&lt;br&gt;Sets the START-ANGLE to 135 degrees.</td>
</tr>
<tr>
<td><strong>STYLE</strong></td>
<td>Depends on the SUB-SELECTOR:&lt;br&gt;- LINE – Solid</td>
<td>LongDash</td>
</tr>
<tr>
<td>Declaration Keyword</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>Declaration Keyword</td>
<td>Choices</td>
<td>Description</td>
</tr>
<tr>
<td>THRESHOLD-METHOD</td>
<td>VALUE</td>
<td>Grouping method to use for the Other slice in a pie chart.</td>
</tr>
<tr>
<td></td>
<td>PERCENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default = VALUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Grouping method to use for the Other slice in a pie chart.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATTRIBUTES=('All', 'THRESHOLD-METHOD', 'Value')</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets the THRESHOLD-METHOD to indicate that the THRESHOLD-VALUE is a data value for all charts.</td>
</tr>
<tr>
<td>THRESHOLD-VALUE</td>
<td>Depends on the THRESHOLD-METHOD:</td>
<td>Data value to group into the Other slice in a pie chart.</td>
</tr>
<tr>
<td></td>
<td>● When THRESHOLD-METHOD is set to Value, valid values are numbers greater than or equal to 0.</td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>● When THRESHOLD-METHOD is set to Percent, valid values are from 0 to 100.</td>
<td>ATTRIBUTES=('All', 'THRESHOLD-VALUE', 10)</td>
</tr>
<tr>
<td></td>
<td>Default = 0 (No Other Slice)</td>
<td>Together with THRESHOLD-METHOD establishes that values less than 10 will be grouped into the 'Other' pie slice for all charts.</td>
</tr>
<tr>
<td>UNITS</td>
<td>DEGREES</td>
<td>Angular units used for Polar, Radar, and Area Radar charts. Affects the ORIGIN-BASE-ANGLE and START-ANGLE.</td>
</tr>
<tr>
<td></td>
<td>RADIANS</td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>GRADS</td>
<td>ATTRIBUTES=('Chart1', 'UNITS', 'Radians')</td>
</tr>
<tr>
<td></td>
<td>Default = DEGREES</td>
<td>Displays the ORIGIN-BASE-ANGLE and START-ANGLE in radians.</td>
</tr>
</tbody>
</table>

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:

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

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

```plaintext
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:

```plaintext
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>
<thead>
<tr>
<th>Selectors</th>
<th>Color</th>
<th>Style</th>
<th>Size</th>
<th>Pattern</th>
<th>Font</th>
<th>Font Style</th>
<th>Point Size</th>
<th>Sort Order</th>
<th>Background Colors</th>
<th>ForeGround Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHART1</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHART2</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEADER</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBTITLE</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOOTER</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOOTER-TEXT</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB-FOOTER-TEXT</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHART AREA</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLOT AREA</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEGEND</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Color</td>
<td>Style</td>
<td>Size</td>
<td>Pattern</td>
<td>Font Style</td>
<td>Font Point Size</td>
<td>Sort Order</td>
<td>Background Colors</td>
<td>ForeGround Colors</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>---------</td>
<td>------------</td>
<td>-----------------</td>
<td>------------</td>
<td>-------------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>LEGEND-TITLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>LEGEND-SYMBOL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>ALL-AXIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>X-AXIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Y-AXIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Y2-AXIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Sub-selectors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>MARKER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>LABEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>LINE</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>GRID</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>SYMBOL</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>FILL</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>OTHER</td>
<td>•</td>
<td></td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Start Angle</th>
<th>3d-Depth</th>
<th>3d-Elevation</th>
<th>3d-Rotation</th>
<th>Cluster-Width</th>
<th>Cluster-Overlap</th>
<th>Threshold Value</th>
<th>Location</th>
<th>Hole Value</th>
<th>Legend Columns</th>
<th>Legend Rows</th>
<th>Percentage</th>
<th>Precision</th>
<th>Label Location</th>
<th>Other Label</th>
<th>Other Units</th>
<th>Other Base-Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>CHART1</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>CHART2</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>HEADER</td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>FOOTER</td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>LEGEND</td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td>•</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Sub-selector</th>
<th>Declaration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE</td>
<td>STYLE</td>
<td>Solid, LongDash, ShortDash, LongShort, Dot, DashDot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value = Solid</td>
</tr>
<tr>
<td>LINE</td>
<td>SIZE</td>
<td>1 – 10 pixels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value = 1</td>
</tr>
<tr>
<td>LINE</td>
<td>COLOR</td>
<td>Named colors or values in the range of RGB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See “DECLARE-COLOR-MAP” on page 88 in the for an explanation of RGB values.</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>STYLE</td>
<td>None, Dot, Box, Triangle, Diamond, Star, VerticalLine, HorizontalLine, Cross, Circle, Square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value = Dot</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>SIZE</td>
<td>1 – 100 pixels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value = bounding box 6x6</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>COLOR</td>
<td>Named color or Values in the range of RGB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See “DECLARE-COLOR-MAP” on page 88 in the for an explanation of RGB values.</td>
</tr>
<tr>
<td>FILL</td>
<td>STYLE</td>
<td>None, Solid, 25Per, 50Per, 75Per, HorizontalStripes, VerticalStripe, 45Stripe, 135Stripe, DiagonalHatch, CrossHatch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value = Solid</td>
</tr>
<tr>
<td>FILL</td>
<td>COLOR</td>
<td>Named color or values in the range of RGB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See “DECLARE-COLOR-MAP” on page 88 in the for an explanation of RGB values.</td>
</tr>
<tr>
<td>GRID</td>
<td>SIZE</td>
<td>1 – 10 pixels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value = 1</td>
</tr>
<tr>
<td>GRID</td>
<td>COLOR</td>
<td>Named color or values in the range of RGB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See “DECLARE-COLOR-MAP” on page 88 in the for an explanation of RGB values.</td>
</tr>
<tr>
<td>GRID</td>
<td>STYLE</td>
<td>None, Solid, LongDash, ShortDash, LongShort, DashDot</td>
</tr>
<tr>
<td>Sub-selector</td>
<td>Declaration</td>
<td>Value</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>OUTLINE</td>
<td>STYLE</td>
<td>None, Solid, LongDash, ShortDash, LongShort, DashDot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value = Solid</td>
</tr>
<tr>
<td>OUTLINE</td>
<td>SIZE</td>
<td>1 – 10 pixels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value = 1</td>
</tr>
<tr>
<td>OUTLINE</td>
<td>COLOR</td>
<td>Named color or values in the range of RGB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See “DECLARE-COLOR-MAP” on page 88 in the for an explanation of RGB values.</td>
</tr>
<tr>
<td>OTHER</td>
<td>COLOR</td>
<td>Named color or values in the range of RGB. See SQR Language Reference/Declare-Color-Map for explanation of RGB values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value = Purple - RGB(170,00,255)</td>
</tr>
</tbody>
</table>

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.

\[\text{rgb}\]

\(\text{red}_\text{lit}_\text{var}_\text{col}, \text{green}_\text{lit}_\text{var}_\text{col}, \text{blue}_\text{lit}_\text{var}_\text{col}\) 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 a 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:

```sql
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’.


**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:

```sql
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:

```sql
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>
<thead>
<tr>
<th>Table 28</th>
<th>Valid Images Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BMP</td>
</tr>
<tr>
<td>HPLaserJet</td>
<td>X</td>
</tr>
<tr>
<td>HTML</td>
<td>X</td>
</tr>
<tr>
<td>PDF</td>
<td>X</td>
</tr>
<tr>
<td>Postscript</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td></td>
</tr>
</tbody>
</table>
Examples

declare-image officer-signature
   type= eps-file
   source='off_sherman.eaps'
   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.hpgl')
   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.hpgl’ (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

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]]
[LIGHT-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]]
[LIGHT-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

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Meaning</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dp</td>
<td>decipoint</td>
<td>0.001388 inch</td>
</tr>
<tr>
<td>pt</td>
<td>point</td>
<td>0.01388 inch</td>
</tr>
<tr>
<td>mm</td>
<td>millimeter</td>
<td>0.03937 inch</td>
</tr>
<tr>
<td>cm</td>
<td>centimeter</td>
<td>0.3937 inch</td>
</tr>
<tr>
<td>in</td>
<td>inch</td>
<td>1.0000 inch</td>
</tr>
</tbody>
</table>

paper_name

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

Table 30 Valid Paper Names

<table>
<thead>
<tr>
<th>Name</th>
<th>Width</th>
<th>Depth</th>
<th>Orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letter</td>
<td>8.5 in</td>
<td>11 in</td>
<td>Portrait</td>
</tr>
<tr>
<td>Legal</td>
<td>8.5 in</td>
<td>14 in</td>
<td>Portrait</td>
</tr>
<tr>
<td>A4</td>
<td>8.27 in</td>
<td>11.69 in</td>
<td>Portrait</td>
</tr>
<tr>
<td>Name</td>
<td>Width</td>
<td>Depth</td>
<td>Orientation</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>A3</td>
<td>11.69 in</td>
<td>16.54 in</td>
<td>Portrait</td>
</tr>
<tr>
<td>Executive</td>
<td>7.25 in</td>
<td>10.5 in</td>
<td>Portrait</td>
</tr>
<tr>
<td>B5</td>
<td>7.17 in</td>
<td>10.12 in</td>
<td>Portrait</td>
</tr>
<tr>
<td>Com-10</td>
<td>4.125 in</td>
<td>9.5 in</td>
<td>Landscape</td>
</tr>
<tr>
<td>Monarch</td>
<td>3.875 in</td>
<td>7.5 in</td>
<td>Landscape</td>
</tr>
<tr>
<td>DL</td>
<td>4.33 in</td>
<td>8.66 in</td>
<td>Landscape</td>
</tr>
<tr>
<td>C5</td>
<td>6.378 in</td>
<td>9.016 in</td>
<td>Landscape</td>
</tr>
</tbody>
</table>

Table 31  DECLARE-LAYOUT Command Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Choice or Default uom</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPER-SIZE</td>
<td>inches</td>
<td>8.5 in, 11 in</td>
<td>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.</td>
</tr>
<tr>
<td>FORMFEED</td>
<td>YES, NO</td>
<td>YES</td>
<td>Whether to write form feeds at the end of each page.</td>
</tr>
<tr>
<td>ORIENTATION</td>
<td>PORTRAIT, LANDSCAPE</td>
<td>PORTRAIT</td>
<td>Portrait = vertical. Landscape = horizontal. Printing in landscape for the printer type HPLASERJET requires landscape fonts.</td>
</tr>
<tr>
<td>LEFT-MARGIN</td>
<td>inches</td>
<td>0.5 in</td>
<td>Amount of blank space to leave at the left side of the page.</td>
</tr>
<tr>
<td>TOP-MARGIN</td>
<td>inches</td>
<td>0.5 in</td>
<td>Amount of blank space to leave at the top of the page.</td>
</tr>
<tr>
<td>RIGHT-MARGIN</td>
<td>inches</td>
<td>0.5 in</td>
<td>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.</td>
</tr>
<tr>
<td>LINE-WIDTH</td>
<td>inches</td>
<td>7.5 in</td>
<td>Length of the line. If you specify RIGHT-MARGIN or MAX-COLUMNS, you cannot use this parameter.</td>
</tr>
<tr>
<td>MAX-COLUMNS</td>
<td></td>
<td>75</td>
<td>Maximum number of columns in a line. If you specify RIGHT-MARGIN or LINE-WIDTH, you cannot use this parameter.</td>
</tr>
<tr>
<td>BOTTOM-MARGIN</td>
<td>inches</td>
<td>0.5 in</td>
<td>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.</td>
</tr>
<tr>
<td>PAGE-DEPTH</td>
<td>inches</td>
<td>10 in</td>
<td>Depth of the page. If you specify BOTTOM-MARGIN or MAX-LINES, you cannot use this parameter.</td>
</tr>
<tr>
<td>MAX-LINES</td>
<td></td>
<td>60</td>
<td>Maximum number of lines printed on the page. If you specify PAGE-DEPTH or BOTTOM-MARGIN, you cannot use this parameter.</td>
</tr>
<tr>
<td>LINE-HEIGHT</td>
<td>points</td>
<td>12 pt</td>
<td>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).</td>
</tr>
</tbody>
</table>
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)
 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
```
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>
<thead>
<tr>
<th>Argument</th>
<th>Choice or Measure</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFTER-BOLD</td>
<td>any string</td>
<td>(none)</td>
<td>See BEFORE-BOLD.</td>
</tr>
<tr>
<td>BEFORE-BOLD</td>
<td>any string</td>
<td>(none)</td>
<td>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 (‘...’). To specify non-printable characters, such as ESC, enclose the decimal value inside angle brackets as follows: BEFORE–BOLD=&lt;27&gt;[r ! Turn on bold AFTER–BOLD=&lt;27&gt;[u ! Turn it off These arguments work with the BOLD argument of PRINT.</td>
</tr>
<tr>
<td>COLOR</td>
<td>Yes, No</td>
<td>No</td>
<td>Defines whether the printer can print in color.</td>
</tr>
<tr>
<td>FONT</td>
<td>font_number</td>
<td>3</td>
<td>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 Technical Reference Manual. For POSTSCRIPT printers, Production Reporting supplies a list of fonts and arbitrary font number assignments in the file POSTSCRI.STR. 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</td>
</tr>
<tr>
<td>Argument</td>
<td>Choice or Measure</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FONT-TYPE</td>
<td>PROPORTIONAL, FIXED</td>
<td>Depends on the font</td>
<td>Applies only to HP LASERJET printers and needs to be specified only for font types not defined in Table 33.</td>
</tr>
<tr>
<td>FOR-REPORTS</td>
<td></td>
<td>ALL</td>
<td>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.</td>
</tr>
<tr>
<td>INIT-STRING</td>
<td>(none)</td>
<td></td>
<td>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, &lt;27&gt; is the ESC or escape character.</td>
</tr>
<tr>
<td>PITCH</td>
<td>characters/inch</td>
<td>10</td>
<td>Required for HPLASERJET printers and SPF Viewer. Fixed—pitched fonts should indicate the pitch.</td>
</tr>
<tr>
<td>POINT-SIZE</td>
<td>points</td>
<td>12</td>
<td>Beginning size of the font. Does not apply to line printers</td>
</tr>
<tr>
<td>RESET-STRING</td>
<td>(none)</td>
<td></td>
<td>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, &lt;27&gt; is the ESC or escape character.</td>
</tr>
<tr>
<td>STARTUP-FILE</td>
<td>filename</td>
<td>POSTSCRI.STR</td>
<td>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 SQRDIR.</td>
</tr>
<tr>
<td>SYMBOL-SET</td>
<td>HP defined sets</td>
<td>0U</td>
<td>HP LASERJET printers only. The default value of &quot;0U&quot; is for the ASCII symbol set. For a complete list of the symbol sets, see the HP LaserJet Technical Reference Manual.</td>
</tr>
</tbody>
</table>
| TYPE              | LINEPRINTER, POSTSCRIPT, HPLASERJET, HTML, LP, PS, HP, HT | LP          | Production Reporting creates output specific to each printer.  
  ● 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 |
PostScript to understand what is shown on the printer.

- 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 HPLaserJet printer types.

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

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

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

Other type faces can be added to POSTSCRL.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>
<thead>
<tr>
<th>Value</th>
<th>Windows Font/Name</th>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Courier New</td>
<td>Fixed</td>
</tr>
<tr>
<td>300</td>
<td>Courier New</td>
<td>Bold</td>
</tr>
<tr>
<td>4</td>
<td>Arial</td>
<td>Proportional</td>
</tr>
<tr>
<td>400</td>
<td>Arial</td>
<td>Bold</td>
</tr>
<tr>
<td>5</td>
<td>Times New Roman</td>
<td>Proportional</td>
</tr>
<tr>
<td>500</td>
<td>Times New Roman</td>
<td>Bold</td>
</tr>
<tr>
<td>6</td>
<td>AvantGarde</td>
<td>Proportional</td>
</tr>
<tr>
<td>8</td>
<td>Palatino</td>
<td>Proportional</td>
</tr>
<tr>
<td>800</td>
<td>Palatino</td>
<td>Bold</td>
</tr>
<tr>
<td>11</td>
<td>Symbol</td>
<td>Proportional</td>
</tr>
</tbody>
</table>

**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**

```plaintext
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                          ! Times-Roman
font=5                           !
```
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-report=(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**

```plaintext
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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKGROUND</td>
<td>Background color name or RGB triplet. Default=NONE</td>
</tr>
<tr>
<td>BOLD</td>
<td>YES</td>
</tr>
<tr>
<td>CENTER</td>
<td>YES</td>
</tr>
<tr>
<td>FILL-COLOR</td>
<td>Fill color name or RGB triplet. Default=NONE</td>
</tr>
</tbody>
</table>
### Attribute Description

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

### ROW-ATTRIBUTES

Attributes to apply to rows.

#### Table 37 Row Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BORDER-COLOR</td>
<td>Color name or RGB triplet. Default=NONE (no border)</td>
</tr>
<tr>
<td>BORDER-LINE-STYLE</td>
<td>The border line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT,LONG-DASH-DOT-DOT). Default=SOLID</td>
</tr>
<tr>
<td>BORDER-THICKNESS</td>
<td>Border thickness expressed in decipoints. Default=two decipoints</td>
</tr>
<tr>
<td>FILL-COLOR</td>
<td>Fill color name or RGB triplet. Default=NONE</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Number of lines between each printed row. Default=one line</td>
</tr>
<tr>
<td>LINE-COLOR</td>
<td>Color name or RGB triplet. Default=NONE (no line)</td>
</tr>
<tr>
<td>LINE-STYLE</td>
<td>Row line style (SOLID, SQUARE-DOT, DASH, DASH-DOT, LONG-DASH, LONG-DASH-DOT,LONG-DASH-DOT-DOT). Default=SOLID</td>
</tr>
<tr>
<td>LINE-THICKNESS</td>
<td>Line thickness expressed in decipoints. Default=two decipoints</td>
</tr>
</tbody>
</table>
Attributes for the appearance of the table.

Table 38  Table Attributes

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

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]...)]]
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**

```plaintext
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**

```plaintext
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
```
Arguments

DEFAULT-NUMERIC

Default type for numeric variables. Unless explicitly declared otherwise, a numeric variable assumes the variable type. This qualifier overrides any setting from the command-line flag -DNT or the DEFAULT-NUMERIC entry in the [Default-Settings] section of SQR.INI. If -DNT was not specified on the command line and SQR.INI has no DEFAULT-NUMERIC entry, then the default numeric type is FLOAT.

DECIMAL

Defines that the numeric variables that follow are decimal variables with a precision specified with prec_lit. The precision can be assigned to the group of variables or to each individual variable. The precision is the total number of digits used to represent the number. This precision can range from 1 to 38. The default value is 16. The range of decimal numbers is from -9.9999999999999999999999999999999999999E±4096 to +9.9999999999999999999999999999999999999E±4096

FLOAT

Defines that the numeric variables that follow are used as double precision floating point. The range and precision of these numbers are machine-dependent.

DATE

Defines that the date variables that follow can contain a date in the range of January 1, 4713 BC to December 31, 9999 AD.

INTEGER

Defines that the numeric variables that follow are used as integers with a range of -2147483648 to +2147483647.

TEXT

Defines that the string variables that follow are text (character) variables.

BINARY

Defines that the variables that follow support BINARY data.

Description

You can set the default numeric type externally, using the -DNT command-line flag or the DEFAULT-NUMERIC setting in the [Default-Settings] section of SQR.INI. However, the setting in DECLARE-VARIABLE takes precedence over all other settings. If the command has not been used, then the -DNT command-line flag takes precedence over the setting in SQR.INI.

Besides FLOAT, INTEGER, or DECIMAL, the DEFAULT-NUMERIC setting in SQR.INI and -DNT command-line flag can be set to V30. With V30, the program acts in the same manner as in pre-version 4.0 releases; that is, all variables are FLOAT. Incidentally, V30 is not a valid setting for the DEFAULT-NUMERIC setting in DECLARE-VARIABLE.
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 \texttt{ASK} is used to obtain the value of a substitution variable that has already been defined, \texttt{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, \texttt{ASK} requests values for only those variables that have not had a value assigned.

You can use \texttt{#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 \texttt{value} in the \texttt{#DEFINE} command can have embedded spaces, and needs no enclosing quotes. The entire string is used as is.

The \texttt{#DEFINE} command cannot be broken across program lines.

\textbf{Examples}

The following code defines several constants:

\begin{verbatim}
#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
\end{verbatim}

The following excerpt from a report uses the preceding definitions:

\begin{verbatim}
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
\end{verbatim}
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 SYYYYMMDD[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:

```plaintext
! Display a string using an edit mask
! display '123456789' xxx-xx-xxxx
```

Produces the following output:

```
123-45-6789
```

The following code:

```plaintext
! 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:

```plaintext
! Display a number using the default edit mask (specified in SQR.INI)
! display 123.78
```

Produces the following output:
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:

```
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 'Hello' noline
display ' World'
```

Produces the following output:

Hello World

The following code:

```
! Display two values on the same line
! display 'Hello' noline
display ' World'
```
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-LOCAL 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

\[
\text{divide 37.5 into } \#\text{price} \quad \#\text{price} / 37.5 \\
\text{divide } \&\text{rate into } \#\text{tot} \quad \text{on-error=high} \\
\text{divide } \#j \text{ into } \#\text{subtot} \quad \text{on-error=zero}
\]

Note:

\text{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

\[\text{DO procedure\_name[(arg1[, argi]...)]}\]

Arguments

\text{procedure\_name}

Name of the procedure to execute.

\text{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 \text{DO}. You can use arguments to send values to or receive values from a procedure.

Arguments passed by \text{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}
```
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-SYTLE 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

<table>
<thead>
<tr>
<th>Line Style</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLID</td>
<td></td>
</tr>
<tr>
<td>SQUARE-DOT</td>
<td>•••••••</td>
</tr>
<tr>
<td>DASH</td>
<td>• • • •</td>
</tr>
<tr>
<td>DASH-DOT</td>
<td>• • • •</td>
</tr>
<tr>
<td>LONG-DASH</td>
<td>••••••</td>
</tr>
<tr>
<td>LONG-DASH-DOT</td>
<td>•• • •</td>
</tr>
<tr>
<td>LONG-DASH-DOT-DOT</td>
<td>• • • •</td>
</tr>
</tbody>
</table>

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)

\[ \text{DRAW} \ (1,1) \ \text{TYPE}='\text{OVAL}' \ \text{HEIGHT}=10 \ \text{WIDTH}=10 \ \text{RULE}=20 \ \text{FILL COLOR}=('\text{BLUE}') \ \text{LINE COLOR}=('\text{RED}') \]

Draw a box using a dashed line

\[ \text{DRAW} \ (1,1) \ \text{TYPE}='\text{BOX}' \ \text{WIDTH}=78 \ \text{HEIGHT}=66 \ \text{RULE}=20 \ \text{LINE STYLE}='\text{DASH}' \]

See Also

\[ \text{GET-COLOR}, \ \text{PRINT}, \ \text{and SET-COLOR} \]

**DUMP-TABLE**

**Function**

Dumps table data into an SPF file.

**Syntax**

\[ \text{DUMP-TABLE} \ \\
\text{Name=table_name_var|_lit|_col} \ \\
[\text{CONTINUATION=continuation_var|_lit|_col}] \]

**Arguments**

\[ \text{NAME} \]

Name of the table created by \text{CREATE-TABLE}.

\[ \text{CONTINUATION} \]

Defines whether the table data is a continuation of a previous \text{DUMP-TABLE} command. Valid values are \text{YES} and \text{NO}. The default is \text{NO}.

**Description**

Use \text{DUMP-TABLE} in any section except \text{BEGIN-SETUP}, \text{BEGIN-SQL}, and \text{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**

\[ \text{dump-table} \ \\
\text{name='customers'} \ \\
\text{continuation='yes'} \]

See Also

\[ \text{ALTER-TABLE}, \ \text{CREATE-TABLE}, \ \text{DECLARE-TABLE}, \ \text{FILL-TABLE}, \ \text{PRINT-TABLE} \]
#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.
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>
<thead>
<tr>
<th>Database</th>
<th>Valid Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oracle</strong></td>
<td>CHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>DATE</td>
</tr>
<tr>
<td></td>
<td>DECIMAL[(p,[s])]</td>
</tr>
<tr>
<td></td>
<td>FLOAT[(b)]</td>
</tr>
<tr>
<td></td>
<td>INTEGER</td>
</tr>
<tr>
<td></td>
<td>LONG</td>
</tr>
<tr>
<td></td>
<td>NCHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>NVARCHAR2[(n)]</td>
</tr>
<tr>
<td></td>
<td>NUMBER[(p,[s])]</td>
</tr>
<tr>
<td></td>
<td>NUMERIC[(p,[s])]</td>
</tr>
<tr>
<td></td>
<td>REAL</td>
</tr>
<tr>
<td></td>
<td>ROWID</td>
</tr>
<tr>
<td></td>
<td>SMALLINT</td>
</tr>
<tr>
<td></td>
<td>VARCHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>VARCHAR2[(n)]</td>
</tr>
<tr>
<td><strong>ODBC</strong></td>
<td>BIT</td>
</tr>
<tr>
<td></td>
<td>TINYINT</td>
</tr>
<tr>
<td></td>
<td>SMALLINT</td>
</tr>
<tr>
<td></td>
<td>INT</td>
</tr>
<tr>
<td></td>
<td>CHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>NCHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>VARCHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>NVARCHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>NTEXT</td>
</tr>
<tr>
<td></td>
<td>TEXT</td>
</tr>
<tr>
<td></td>
<td>REAL</td>
</tr>
<tr>
<td></td>
<td>FLOAT[(b)]</td>
</tr>
<tr>
<td></td>
<td>IMAGE</td>
</tr>
<tr>
<td></td>
<td>SMALLMONEY</td>
</tr>
<tr>
<td></td>
<td>MONEY</td>
</tr>
<tr>
<td></td>
<td>DECIMAL[(p,[s])]</td>
</tr>
<tr>
<td></td>
<td>NUMERIC [(p,[s])]</td>
</tr>
<tr>
<td></td>
<td>SYSNAME</td>
</tr>
<tr>
<td></td>
<td>SMALLDATETIME</td>
</tr>
<tr>
<td></td>
<td>DATETIME</td>
</tr>
<tr>
<td></td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>Database</td>
<td>Valid Data Types</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>BINARY</td>
</tr>
<tr>
<td></td>
<td>VARBINARY</td>
</tr>
<tr>
<td>Sybase</td>
<td>BIT</td>
</tr>
<tr>
<td></td>
<td>TINYINT</td>
</tr>
<tr>
<td></td>
<td>SMALLINT</td>
</tr>
<tr>
<td></td>
<td>INT</td>
</tr>
<tr>
<td></td>
<td>CHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>NCHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>VARCHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>NVARCHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>TEXT</td>
</tr>
<tr>
<td></td>
<td>REAL</td>
</tr>
<tr>
<td></td>
<td>FLOAT[(b)]</td>
</tr>
<tr>
<td></td>
<td>IMAGE</td>
</tr>
<tr>
<td></td>
<td>SMALLMONEY</td>
</tr>
<tr>
<td></td>
<td>MONEY</td>
</tr>
<tr>
<td></td>
<td>DECIMAL[(p[,s])]</td>
</tr>
<tr>
<td></td>
<td>NUMERIC [(p[,s])]</td>
</tr>
<tr>
<td></td>
<td>SYSNAME</td>
</tr>
<tr>
<td></td>
<td>SMALLDATETIME</td>
</tr>
<tr>
<td></td>
<td>DATETIME</td>
</tr>
<tr>
<td></td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td></td>
<td>BINARY</td>
</tr>
<tr>
<td></td>
<td>VARBINARY</td>
</tr>
<tr>
<td></td>
<td>UNICHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>UNIVARCHAR[(n)]</td>
</tr>
<tr>
<td>DB2</td>
<td>CHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>VARCHAR[(n)]</td>
</tr>
<tr>
<td></td>
<td>DATE</td>
</tr>
<tr>
<td></td>
<td>TIME</td>
</tr>
<tr>
<td></td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td></td>
<td>FLOAT</td>
</tr>
<tr>
<td></td>
<td>DOUBLE</td>
</tr>
<tr>
<td></td>
<td>NUMERIC</td>
</tr>
<tr>
<td></td>
<td>DECIMAL[(p[,s])]</td>
</tr>
<tr>
<td></td>
<td>INTEGER</td>
</tr>
<tr>
<td></td>
<td>GRAPHIC[(n)]</td>
</tr>
<tr>
<td>Database</td>
<td>Valid Data Types</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>VARGRAPHIC[(n)]</td>
</tr>
</tbody>
</table>

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`. 
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`.

```sql
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
```

**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**

```sql
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
```
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>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANNOTATION</td>
<td>Arbitrary text string associated with the specified location. SQR passes the data to the backend printer driver—it does not validate the data.</td>
</tr>
<tr>
<td>BACKGROUND</td>
<td>Background color name or RGB triplet</td>
</tr>
<tr>
<td>BOLD</td>
<td>YES</td>
</tr>
<tr>
<td>CENTER</td>
<td>YES</td>
</tr>
<tr>
<td>COLUMN-LEADING</td>
<td>Expressed in decipoints</td>
</tr>
<tr>
<td>COLUMN-LINE-COLOR</td>
<td>Color name or RGB triplet</td>
</tr>
<tr>
<td>COLUMN-LINE-STYLE</td>
<td>SOLID</td>
</tr>
<tr>
<td>COLUMN-LINE-THICKNESS</td>
<td>Expressed in decipoints</td>
</tr>
<tr>
<td>EDIT-MASK</td>
<td>Edit mask or keyword to use</td>
</tr>
<tr>
<td>FILL-COLOR</td>
<td>Fill color name or RGB triplet. Default=NONE</td>
</tr>
<tr>
<td>FONT</td>
<td>Font number</td>
</tr>
<tr>
<td>FOREGROUND</td>
<td>Foreground color name or RGB triplet</td>
</tr>
<tr>
<td>FORMULAE</td>
<td>Arbitrary formula associated with the specified location. SQR passes the data to the backend printer driver—it does not validate the data.</td>
</tr>
<tr>
<td>ITALIC</td>
<td>YES</td>
</tr>
<tr>
<td>POINT-SIZE</td>
<td>Point size of the font</td>
</tr>
<tr>
<td>ROW-LINE-COLOR</td>
<td>Color name or RGB triplet</td>
</tr>
<tr>
<td>ROW-LINE-STYLE</td>
<td>SOLID</td>
</tr>
<tr>
<td>ROW-LINE-THICKNESS</td>
<td>Expressed in decipoints</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>YES</td>
</tr>
<tr>
<td>URL</td>
<td>Specifies the hypertext link for the specified location. SQR does not validate the address.</td>
</tr>
<tr>
<td>URL-TARGET</td>
<td>Specifies the target within the URL. SQR does not validate the target.</td>
</tr>
<tr>
<td>WRAP</td>
<td>YES</td>
</tr>
<tr>
<td>WRAP-HEIGHT</td>
<td>Expressed as the number of lines between each wrapped line.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>WRAP-ON</td>
<td>Characters on which to force a WRAP</td>
</tr>
<tr>
<td>WRAP-STRIP</td>
<td>Characters to change to a space before the WRAP is done</td>
</tr>
</tbody>
</table>

**Description**

Use **FILL-TABLE** in any section except **BEGIN-SETUP**, **BEGIN-SQL**, and **BEGIN-DOCUMENT** to manipulate table attributes.

**Example**

```plaintext
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**

```plaintext
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

```plaintext
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

```plaintext
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[NNNNNN]]]]'.

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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal</td>
</tr>
<tr>
<td>!=</td>
<td>Not Equal</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not Equal</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Directive</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#IF</td>
<td>#IF {option}='A'</td>
<td>Compiles the commands following the #IF directive if the substitution variable option is equal to 'A'. The test is case-insensitive. Only one simple expression is allowed per #IF command.</td>
</tr>
<tr>
<td>#ELSE</td>
<td>#ELSE</td>
<td>Compiles the commands following the #ELSE directive when the #IF expression is FALSE.</td>
</tr>
<tr>
<td>#ENDIF</td>
<td>#ENDIF</td>
<td>Ends the #IF directive. #ENDIF can also be typed #END-IF (with a hyphen).</td>
</tr>
<tr>
<td>#IFDEF</td>
<td>#IFDEF option</td>
<td>Compiles the commands following the #IFDEF directive if the substitution variable option is defined.</td>
</tr>
<tr>
<td>#IFNDEF</td>
<td>#IFNDEF option</td>
<td>Compiles the command following the #IFNDEF directive if the substitution variable option is not defined.</td>
</tr>
</tbody>
</table>

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][NNN]]`.

- 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**

```plaintext
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
#IFNDEF

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. (#IFNDEF is a compiler directive.)

Syntax
#IFNDEF 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

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR, TEXT</td>
<td>Any character. This is the default datatype.</td>
</tr>
<tr>
<td>NUMBER</td>
<td>A floating point number in the format [+</td>
</tr>
<tr>
<td>INTEGER</td>
<td>An integer in the format [+</td>
</tr>
<tr>
<td>DATE</td>
<td>A date in one of the following formats:</td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY [BC</td>
</tr>
<tr>
<td></td>
<td>MM-DD-YYYY [BC</td>
</tr>
<tr>
<td></td>
<td>MM.DD.YYYY [BC</td>
</tr>
<tr>
<td></td>
<td>SYYYYMMDD[HH24][MI][SS][NNNNN]]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Status Value</th>
<th>Indicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful.</td>
</tr>
<tr>
<td>Status Value</td>
<td>Indicates</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Bad type (did not match the datatype of TYPE).</td>
</tr>
<tr>
<td>2</td>
<td>Too long (longer than MAXLEN or the input for an INTEGER variable is &lt; -2147483648 or</td>
</tr>
<tr>
<td></td>
<td>&gt; +2147483647).</td>
</tr>
<tr>
<td>3</td>
<td>No arguments remain on the command line. The command was ignored.</td>
</tr>
</tbody>
</table>

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-LOCAL**
- 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

\[
\text{LAST-PAGE position [pre_txt_lit][post_txt_lit]}
\]

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 \( \text{pre_txt_lit} \) and \( \text{post_txt_lit} \) are printed immediately before and after the number.

Using \text{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

\[
\begin{align*}
\text{begin-footing 1} \\
\quad \text{page-number(1,37) \ 'Page ' Will appear as} \\
\quad \text{last-page () \ ' of ' \ '! ' ! \ ''Page 12 of 25.'} \\
\text{end-footing}
\end{align*}
\]

See Also

\text{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

\[
\text{LET } \text{dst_var}=\text{expression}
\]
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:

```plaintext
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:

```plaintext
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:

```plaintext
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>
<thead>
<tr>
<th>Operator</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>`</td>
<td></td>
</tr>
</tbody>
</table>
### Operator Explanation

<table>
<thead>
<tr>
<th>Operator</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>+, -</td>
<td>Sign prefix (positive or negative)</td>
</tr>
<tr>
<td>^</td>
<td>Exponent</td>
</tr>
<tr>
<td>*, /, %</td>
<td>Multiply, divide, remainder: a % b = mod(a, b) for integers</td>
</tr>
<tr>
<td>+, -</td>
<td>Plus, minus</td>
</tr>
<tr>
<td>&gt;, &lt;, &gt;=, &lt;=, !=, =</td>
<td>Comparison operators: greater than, less than, greater or equal to, less than or equal to, not equal (!= or &lt;&gt;), equal</td>
</tr>
<tr>
<td>not</td>
<td>Logical NOT</td>
</tr>
<tr>
<td>and</td>
<td>Logical AND</td>
</tr>
<tr>
<td>or, xor</td>
<td>Logical OR, XOR (exclusive OR)</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BITAND</td>
<td>Acts just like Logical AND except returns the actual result instead of TRUE or FALSE</td>
</tr>
<tr>
<td>BitOR</td>
<td>Acts just like Logical OR except returns the actual result instead of TRUE or FALSE</td>
</tr>
<tr>
<td>BitXOR</td>
<td>Acts just like Logical XOR except returns the actual result instead of TRUE or FALSE</td>
</tr>
</tbody>
</table>

**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
   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 'SYYYYMMDD [HH24 [MI [SS [NNNNNN] ]]]'

## Numeric Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>Returns the absolute value of num_value. Value type: Same as num_value Syntax: dst_var = abs(num_value)</td>
</tr>
<tr>
<td></td>
<td>● num_value = decimal, float, or integer literal, column, variable, or expression.</td>
</tr>
<tr>
<td></td>
<td>● dst_var = decimal, float, or integer variable. Example: let#dabsvar = abs(#dvar)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| acos | Returns the arccosine of `num_value` in the range of 0 to π radians. The value of `num_value` must be between -1 and 1.  
  **Value type**: float  
  **Syntax**: `dst_var = acos(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
  - `dst_var` = decimal, float, or integer variable.  
  **Example**: `let #facosvar = acos(#fvar)` |
| asin | Returns the arcsine of `num_value` in the range of -π/2 to π/2 radians. The value of `num_value` must be between -1 and 1.  
  **Value type**: float  
  **Syntax**: `dst_var = asin(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
  - `dst_var` = decimal, float, or integer variable.  
  **Example**: `let #fasinvar = asin(#fvar)` |
| atan | Returns the arctangent of `num_value` in the range of -π/2 to π/2 radians. The value of `num_value` must be between -1 and 1.  
  **Value type**: float  
  **Syntax**: `dst_var = atan(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
  - `dst_var` = decimal, float, or integer variable.  
  **Example**: `let #fatanvar = atan(#fvar)` |
| ceil | Returns a value representing the smallest integer that is greater than or equal to `num_value`.  
  **Value type**: Same as `num_value`.  
  **Syntax**: `dst_var = ceil(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression.  
  - `dst_var` = decimal, float, or integer variable.  
  **Example**: `let #fceilvar = ceil(#fvar)` |
| cos | Returns the cosine of `num_value`.  
  **Value type**: float  
  **Syntax**: `dst_var = cos(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
  - `dst_var` = decimal, float, or integer variable.  
  **Example**: `let #fcosvar = cos(#fvar)` |
<p>| cosh | Returns the hyperbolic cosine of <code>num_value</code>. This function returns a float value. |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| `cosh` | Syntax: `dst_var = cosh(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
  - `dst_var` = decimal, float, or integer variable.  
  Example: let #fcoshvar = cosh(#fvar) |
| `deg` | Returns a value expressed in degrees of `num_value` which is expressed in radians.  
  Value type: float  
  Syntax: `dst_var = deg(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
  - `dst_var` = decimal, float, or integer variable.  
  Example: let #degvar = deg(#fvar) |
| `e10` | Returns the value of 10 raised to `num_value`.  
  Value type: float  
  Syntax: `dst_var = e10(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
  - `dst_var` = decimal, float, or integer variable.  
  Example: let #fe10var = e10(#fvar) |
| `exp` | Returns the value of e raised to `num_value`.  
  Value type: float  
  Syntax: `dst_var = exp(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
  - `dst_var` = decimal, float, or integer variable.  
  Example: let #fexpvar = exp(#fvar) |
| `floor` | Returns a value representing the largest integer that is less than or equal to `num_value`.  
  Value type: Same as `num_value`  
  Syntax: `dst_var = floor(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression.  
  - `dst_var` = decimal, float, or integer variable.  
  Example: let #ffloornvar = floor(#fvar) |
| `hex` | Returns a string, in the form of 0x?, which is the hexadecimal representation of the argument  
  Syntax: `dst_var = hex(num_value)`  
  - `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to an integer.  
  - `dst_var` = string variable.  
  Example: let $hexvar = hex(#fvar) |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| **log** | Returns the natural logarithm of num_value.  
*Value type:* float  
*Syntax:* dst_var = log(num_value)  
- num_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
- dst_var = decimal, float, or integer variable.  
*Example:* let #flogvar = log(#fvar) |
| **log10** | Returns the base-10 logarithm of num_value.  
*Value type:* float  
*Syntax:* dst_var = log10(num_value)  
- num_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
- dst_var = decimal, float, or integer variable.  
*Example:* let #flog10var = log10(#fvar) |
| **mod** | Returns the fractional remainder, f, of x_value/y_value such that x_value = i * y_value + f, where i is an integer, f has the same sign as x_value, and the absolute value of f is less than the absolute value of y_value. The arguments are promoted to the type of the greatest precision and the function returns a value of that type.  
*Syntax:* dst_var = mod(x_value, y_value)  
- x_value = decimal, float, or integer literal, column, variable, or expression.  
- y_value = decimal, float, or integer literal, column, variable, or expression.  
- dst_var = decimal, float, or integer variable.  
*Example:* let #fmodvar = mod(#fxvar, #fyvar) |
| **power** | Returns the value of x_value raised to the power of y_value.  
*Value type:* float  
*Syntax:* dst_var = power(x_value, y_value)  
- x_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
- y_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
- dst_var = decimal, float, or integer variable.  
*Example:* let #fpowervar = power(#fxvar, #fyvar) |
| **rad** | Returns a value expressed in radians of num_value which is expressed in degrees.  
*Value type:* float  
*Syntax:* dst_var = rad(num_value)  
- num_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
- place_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| `rad`    | `dst_var` = decimal, float, or integer variable.  
Example: let #fradvar = rad(#fvar) |
| `round`  | Returns a value that is `num_value` rounded to `place_value` digits after the decimal separator.  
**Value type**: Same as `num_value`  
**Syntax**: `dst_var = round(num_value, place_value)`  
- `num_value` = decimal, float, or integer literal, column, variable, or expression.  
- `place_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
- `dst_var` = decimal, float, or integer variable.  
Example: let #frndvar = round(#fvar, #fplace) |
| `sign`   | Returns a -1, 0, or +1 depending on the sign of `num_value`.  
**Value type**: float  
**Syntax**: `dst_var = sign(num_value)`  
- `num_value` = decimal, float, or integer literal, column, variable, or expression.  
- `dst_var` = decimal, float, or integer variable.  
Example: let #fsignvar = sign(#fvar) |
| `sin`    | Returns the sine of `num_value`.  
**Value type**: float  
**Syntax**: `dst_var = sin(num_value)`  
- `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
- `dst_var` = decimal, float, or integer variable.  
Example: let #fsinvar = sin(#fvar) |
| `sinh`   | Returns the hyperbolic sine of `num_value`.  
**Value type**: float  
**Syntax**: `dst_var = sinh(num_value)`  
- `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
- `dst_var` = decimal, float, or integer variable.  
Example: let #fsinhvar = sinh(#fvar) |
| `sqrt`   | Returns the square root of `num_value`.  
**Value type**: float  
**Syntax**: `dst_var = sqrt(num_value)`  
- `num_value` = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
- `dst_var` = decimal, float, or integer variable.  
Example: let #fsqrtvar = sqrt(#fvar) |
Function | Description
--- | ---
**tan** | Returns the tangent of num_value.

Value type: float

Syntax: dst_var = tan(num_value)

- num_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.
- dst_var = decimal, float, or integer variable.

Example: let #ftanvar = tan(#fvar)

**tanh** | Returns the hyperbolic tangent of num_value.

Value type: float

Syntax: dst_var = tanh(num_value)

- num_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.
- dst_var = decimal, float, or integer variable.

Example: let #ftanhvar = tanh(#fvar)

**trunc** | Returns a value that is num_value truncated to place_value digits after the decimal separator.

Value type: Same as num_value

Syntax: dst_var = trunc(num_value, place_value)

- num_value = decimal, float, or integer literal, column, variable, or expression.
- place_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.
- dst_var = decimal, float, or integer variable.

Example: let #ftruncvar = trunc(#fvar, #fplace)

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| delete   | Deletes filename.  
**Syntax:** stat_var = delete(filename)  
- filename = text literal, column, variable, or expression.  
- stat_var = decimal, float, or integer variable.  
**Example:** let #fstatus = delete($filename) |
| exists   | Determines if filename exists.  
**Syntax:** stat_var = exists(filename)  
- filename = text literal, column, variable, or expression.  
- stat_var = decimal, float, or integer variable.  
**Example:** let #fstatus = exists($filename) |
| filesize | 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.  
**Syntax:** dst_var = filesize(source_value)  
- source_value = text literal, column, variable, or expression  
- dst_var = decimal, float, or integer variable  
**Example:** let #size = filesize($file) |
| rename   | Renames old_filename to new_filename.  
**Syntax:** stat_var = rename(old_filename, new_filename)  
- old_filename = text literal, column, variable, or expression  
- new_filename = text literal, column, variable, or expression  
- stat_var = decimal, float, or integer variable  
**Example:** let #fstatus = rename($old_filename, $new_filename) |

String Functions

Table 49 String Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| ascii    | Returns the ASCII value for the first character in str_value.  
**Value type:** float  
**Syntax:** ascii_var = ascii(str_value)  
- str_value = date or text literal, column, variable, or expression  
- ascii_var = decimal, float, or integer variable  
**Example:** let #fascii = ascii($filename) |
| asciic   | Returns the numeric value for the first character (rather than byte) of the specified string.  
**Syntax:** ascii_var = asciic(str_value) |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>chr</strong></td>
<td>Returns a string composed of a character with the ASCII value of <code>num_value</code>.  &lt;br&gt;&lt;br&gt; <strong>Syntax:</strong> <code>dst_var = chr(num_value)</code>  &lt;br&gt; ● <code>num_value</code> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  &lt;br&gt; ● <code>dst_var</code> = text variable  &lt;br&gt; <strong>Example:</strong> let $svar = chr(#num)`</td>
</tr>
<tr>
<td><strong>edit</strong></td>
<td>Formats <code>source_value</code> according to <code>edit_mask</code> and returns a string containing the result.  &lt;br&gt;&lt;br&gt; <strong>Syntax:</strong> <code>dst_var = edit(source_value, edit_mask)</code>  &lt;br&gt; ● <code>source_value</code> = Any literal, column, variable, or expression  &lt;br&gt; ● <code>edit_mask</code> = text literal, column, variable, or expression  &lt;br&gt; ● <code>dst_var</code> = text variable  &lt;br&gt; <strong>Example:</strong> let $phone = edit(&amp;phone, '(xxx) xxx-xxxxx') let $price = edit(#price, '999.99') let $today = edit($date, 'DD/MM/YYYY')</td>
</tr>
<tr>
<td><strong>fromhex</strong></td>
<td>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.  &lt;br&gt;&lt;br&gt; <strong>Syntax:</strong> <code>dst_var = fromhex(source_value)</code>  &lt;br&gt; ● <code>source_value</code> = text literal, column, variable, or expression  &lt;br&gt; ● <code>dst_var</code> = binary variable  &lt;br&gt; <strong>Example:</strong> let $image = fromhex($hexchars)</td>
</tr>
<tr>
<td><strong>instr</strong></td>
<td>Returns the numeric position of <code>sub_value</code> in <code>source_value</code> or zero (0) if not found. The search begins at offset <code>offset_value</code>.  &lt;br&gt;&lt;br&gt; <strong>Value type:</strong> float  &lt;br&gt;&lt;br&gt; <strong>Syntax:</strong> <code>dst_var = instr(source_value, sub_value, offset_value)</code>  &lt;br&gt; ● <code>source_value</code> = date or text literal, column, variable, or expression  &lt;br&gt; ● <code>sub_value</code> = text literal, column, variable, or expression  &lt;br&gt; ● <code>offset_value</code> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.  &lt;br&gt; ● <code>dst_var</code> = decimal, float, or integer variable  &lt;br&gt; <strong>Example:</strong> let #offset = instr(&amp;description, 'auto', 10)</td>
</tr>
<tr>
<td><strong>instrb</strong></td>
<td>Performs the same functionality as the instr function except that the starting point and returned value are expressed in bytes rather than characters.  &lt;br&gt;&lt;br&gt; <strong>Syntax:</strong> <code>dst_var = instrb(source_value, sub_value, offset_value)</code>  &lt;br&gt; ● <code>source_value</code> = date or text literal, column, variable, or expression  &lt;br&gt; ● <code>sub_value</code> = text literal, column, variable, or expression  &lt;br&gt; ● <code>offset_value</code> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| dst_var = decimal, float, or integer variable  
**Example:** let #offset = instrb(&description, 'auto', 10)  
**Note:** instrb does not allow you to specify the target encoding. If you are using Unicode internally, specify the target encoding with lengthp, lengtht, substrp, substrt, or transform. |
| **isblank** | Returns a value of one (1) if source_val is an empty string, null string, or composed entirely of whitespace characters; otherwise, returns a value of zero (0).  
**Syntax:** dst_var = isblank(source_value)  
- source_value = date or text literal, column, variable, or expression (character data type columns only, no numeric data type columns)  
- dst_var = decimal, float, or integer variable  
**Example:** let #blank = isblank(&description)  
**Note:** isblank can only be used for character data type columns. |
| **length** | Returns the number of characters in source_value.  
**Syntax:** dst_var = length(source_value)  
- source_value = date or text literal, column, variable, or expression  
- dst_var = decimal, float, or integer variable  
**Example:** let #length = length(&description) |
| **lengthb** | Same functionality as length except that the return value is expressed in bytes, rather than characters.  
**Syntax:** dst_var = lengthb(source_value)  
- source_value = date or text literal, column, variable, or expression  
- dst_var = decimal, float, or integer variable  
**Example:** let #length = lengthb(&description)  
**Note:** lengthb does not allow you to specify the target encoding. If you are using Unicode internally, specify the target encoding with lengthp, lengtht, substrp, substrt, or transform. |
| **lower** | Converts the contents of source_value to lowercase and returns the result.  
**Syntax:** dst_var = lower(source_value)  
- source_value = date or text literal, column, variable, or expression  
- dst_var = text variable  
**Example:** let $lower = lower(&description) |
| **lpad** | Pads the source_value on the left to a length of length_value using pad_value and returns the result.  
**Syntax:** dst_var = lpad(source_value, length_value, pad_value)  
- source_value = date or text literal, column, variable, or expression  
- length_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer  
- pad_value = text literal, column, variable, or expression  
- dst_var = text variable |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: let $lpad = lpad($notice, 25, '.')</td>
<td></td>
</tr>
<tr>
<td><strong>ltrim</strong></td>
<td>Trims characters in source_value from the left until a character is not in set_value and returns the result.</td>
</tr>
<tr>
<td>Syntax: $dst_var = ltrim(source_value, set_value)</td>
<td></td>
</tr>
<tr>
<td>● source_value = date or text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● set_value = text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● dst_var = text variable</td>
<td></td>
</tr>
<tr>
<td>Example: let $ltrim = ltrim(&amp;description, '.')</td>
<td></td>
</tr>
<tr>
<td><strong>replace</strong></td>
<td>Inspects the contents of source_value and replaces all occurrences of from_string with to_string and returns the modified string.</td>
</tr>
<tr>
<td>Syntax: $dst_var = replace(source_value, from_string, to_string)</td>
<td></td>
</tr>
<tr>
<td>● source_value = date or text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● from_string = text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● to_string = text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● dst_var = text variable</td>
<td></td>
</tr>
<tr>
<td>Example: let $replaced = replace($paragraph, 'good', 'excellent')</td>
<td></td>
</tr>
<tr>
<td><strong>rpad</strong></td>
<td>Pads the source_value on the right to a length of length_value using pad_value and returns the result.</td>
</tr>
<tr>
<td>Syntax: $dst_var = rpad(source_value, length_value, pad_value)</td>
<td></td>
</tr>
<tr>
<td>● source_value = date or text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● length_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</td>
<td></td>
</tr>
<tr>
<td>● pad_value = text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● dst_var = text variable</td>
<td></td>
</tr>
<tr>
<td>Example: let $rpad = rpad($notice, 25, '.')</td>
<td></td>
</tr>
<tr>
<td><strong>rtrim</strong></td>
<td>Trims characters in source_value from the right until a character is not in set_value and returns the result.</td>
</tr>
<tr>
<td>Syntax: $dst_var = rtrim(source_value, set_value)</td>
<td></td>
</tr>
<tr>
<td>● source_value = date, or text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● set_value = text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● dst_var = text variable</td>
<td></td>
</tr>
<tr>
<td>Example: let $rtrim = rtrim(&amp;description, '.')</td>
<td></td>
</tr>
<tr>
<td><strong>substr</strong></td>
<td>Extracts the specified portion source_value. The extraction begins at offset_value (origin is 1) for a length of length_value characters.</td>
</tr>
<tr>
<td>Syntax: $dst_var = substr(source_value, offset_value, length_value)</td>
<td></td>
</tr>
<tr>
<td>● source_value = date or text literal, column, variable, or expression</td>
<td></td>
</tr>
<tr>
<td>● offset_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **length_value** = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.  
**dst_var** = text variable  
*Example*: let $piece = substr(&record, 10, #len) |
| **substrb** | Has the same functionality as **substr** except that the starting point and length are expressed in bytes, rather than in characters.  
**Syntax**:  
```
 dst_var = substrb(source_value, offset_value, length_value)
```
  
  - **source_value** = date or text literal, column, variable, or expression  
  - **offset_value** = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.  
  - **length_value** = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.  
  - **dst_var** = text variable  
  
  *Example*: let $piece = substrb(&record, 10, #len)  
  
  *Note*: **substrb** does not allow you to specify the target encoding. If you are using Unicode internally, specify the target encoding with **lengthp**, **lengtht**, **substrp**, **substrt**, or **transform**. |
| **to_char** | Converts **source_value** to a string, using maximum precision.  
**Syntax**:  
```
 dst_var = to_char(source_value)
```
  
  - **source_value** = decimal, float, or integer literal, column, variable, or expression  
  - **dst_var** = text variable  
  
  *Example*: let $string = to_char(#number) |
| **tohex** | 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.  
**Syntax**:  
```
 dst_var = tohex(source_value)
```
  
  - **source_value** = binary literal, column, variable, or expression  
  - **dst_var** = text variable  
  
  *Example*: let $hexchars = tohex($vargraphic) |
| **to_multi_byte** | 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.  
**Syntax**:  
```
 dst_var = to_multi_byte (source_value)
```
  
  - **source_value** = date or text literal, column, variable, or expression  
  - **dst_var** = text variable  
  
  *Example*: let $multi = to_multi_byte (&text) |
| **to_number** | Converts **source_value** to a number.  
**Value type**: float  
**Syntax**:  
```
 dst_var = to_number(source_value)
```
  
  - **source_value** = decimal, float, or integer literal, column, variable, or expression  
  - **dst_var** = decimal, float, or integer variable |
## Function

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| to_single_byte    | 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. 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. 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. Syntax: dst_var = to_single_byte(source_value)  
  - source_value = date or text literal, column, variable, or expression  
  - dst_var = text variable  
  Example: let $single = to_single_byte(&text) |
| translate         | Inspects the contents of source_value and converts characters that match those in from_set to the corresponding character in to_set and returns the translated string. If to_set does not contain a matching translation character in the corresponding from_set, then the original is left unchanged with regard to that character. If the translation string in to_set is empty, then all characters specified in the from_set string are removed. Syntax: dst_var = translate(source_value, from_set, to_set)  
  - source_value = date or text literal, column, variable, or expression  
  - from_set = text literal, column, variable, or expression  
  - to_set = text literal, column, variable, or expression  
  - dst_var = text variable  
  Example: let $translated = translate(edit(&price, '999,999.99'), ',', '.') |
| upper             | Converts the contents of source_value to uppercase and returns the result. Syntax: dst_var = upper(source_value)  
  - source_value = date or text literal, column, variable, or expression  
  - dst_var = text variable  
  Example: let $upper = upper(&description) |

### Date Functions

#### Table 50  Date Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| dateadd  | Returns a date after adding (or subtracting) the specified units to the date_value. Syntax: date_var = dateadd(date_value, units_value, quantity_value)  
  - date_value = date variable or expression  
  - units_value = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second' |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| ● quantity_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.  
● date_var = date variable  
Example: let $date = dateadd($startdate, 'day', 7.5) | datediff Returns the difference between the specified dates expressed in units_value. The function returns a float value. The result can be negative if the first date is earlier than the second date.  
Syntax: dst_var = datediff(date1_value, date2_value, units_value)  
● date1_value = date variable or expression  
● date2_value = date variable or expression  
● units_value = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second'  
● dst_var = decimal, float, or integer variable  
Example: let #diff = datediff($date1, $date2, 'hour') |
| datenow Returns the current local date and time from the client machine.  
Syntax: dst_var = datenow()  
● dst_var = date variable  
Example: let $date = datenow() | datetostr Converts the date date_value to a string in the format format_mask.  
Syntax: dst_var = datetostr(date_value, [format_mask])  
● date_value = date variable or expression  
● format_mask = 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.  
● dst_var = text variable  
Example: let $formdate = datetostr($date, 'Day Mon DD, YYYY') let $localedate = datetostr($date, DATE) |
| strtodate Converts the string source_value in the format format_mask to a date type.  
Syntax: dst_var = strtodate(source_value [, format_mask])  
● source_value = text literal, column, variable, or expression  
● format_mask = text literal, column, variable, or expression that describes the exact format of the source_value. DATE can specify the DATE-EDIT-MASK setting from the current locale. If not specified, then source_value 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][NNNNNN]'. 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.  
● dst_var = date variable  
Example: let $date = strtodate($str_date, 'Mon DD, YYYY') let $date = strtodate($str_date, DATE) |
Unicode Functions

Note:
Unicode functions are only allowed when converting to Unicode internally.

Table 51  Unicode Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| lengthp | Returns the string length in print position. Half-width characters take one print position, full-width characters take two, and combining characters take zero.  
**Syntax:** dst_var = lengthp(source_value)  
- source_value = date or text literal, column, variable, or expression  
- dst_var = decimal, float, or integer variable  
**Example:** let #printLen = lengthp($string)) |
| lengtht | Returns the string length in bytes when converted (transformed) to a specified encoding. Encoding names are the same as those allowed in OPEN or in SQR.INI. String and column variables can be used in place of the literal encoding name.  
**Syntax:** dst_var = lengtht(source_value, encoding_value)  
- source_value = date or text literal, column, variable, or expression  
- encoding_value = text literal, column, variable, or expression  
- dst_var = decimal, float, or integer variable  
**Example:** let #sjisLen = lengtht($string, 'shift-jis') |
| substrp | 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.  
**Syntax:** dst_var = substrb(source_value, offset_value, length_value)  
- source_value = date or text literal, column, variable, or expression.  
- offset_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.  
- length_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.  
- dst_var = text variable  
**Example:** let $sub = substrp(&string, #printPos, #printlen) |
| substrt | 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.  
**Syntax:** dst_var = substrb(source_value, offset_value, length_value, encoding_value))  
- source_value = date or text literal, column, variable, or expression  
- offset_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| ● **length_value** = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.  
● encoding_value = text literal, column, variable, or expression  
● dst_var = text variable  
**Example:** let $sjisPrep = SUBSTR ($string, 1, 10, 'Shift-JIS') | **transform** Returns a Unicode string which is specified transform of a given string.  
**Syntax:** dst_var = transform (source_value, transform_value)  
● source_value = date or text literal, column, variable or expression  
● transform_value = text literal, column, variable, or expression  
● dst_var - text variable  
**Example:** let $hiragana = transform (&string, 'ToHiragana') |

Production Reporting supports the following transforms:  

(**Source: Rosette API Reference)**  
● **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.  
Characters with full-width representations are: Roman alphabet characters (A-z), digits (0-9), Japanese **katakana** characters, and the most commonly used punctuation characters (including Space).  
● **ToHalfwidth**—Transforms all full-width characters that also have a half-width representation to their half-width form.  
Characters with half-width representations are: Roman alphabet characters (A-z), digits (0-9), Japanese **katakana** characters, and the most commonly used punctuation characters (including Space).  
● **ToHiragana**—Transforms all full-width **katakana** characters to **hiragana**.  
To convert half-width **katakana** characters to **hiragana**, you must first convert the characters to full-width using the FullWidth transform.  
● **ToParagraphSeparator**—Standardizes the line/paragraph separators in the text according to the following standards:  
Standard Code Point Line/Paragraph Separator  
Windows 0x0D0A 0x0D0A  
Macintosh 0x0D ToCR  
UNIX 0x0A ToLF  
Unicode U+2028 ToLineSeparator  
Unicode U+2029 ToParagraphSeparator  
EBCDIC 0x15 ToEBCDICNewLine  
● **HankakuKatakanaToZenkaku**—Converts half-width (**hankaku**) Japanese katakana characters to the full-width (**zenkaku**) form.
Function | Description
--- | ---
 | This conversion is almost identical to ToFullwidth, except that it automatically composes and combines katakana "accent" marks (dakuten and handakuten) appropriately, whereas ToFullwidth does not provide any special treatment for these marks.
 | ZenkakuKatakanaToHankaku—Converts full-width (zenkaku) Japanese katakana characters to the half-width (hankaku) form.
 | This conversion is almost identical to ToHalfwidth, except that it automatically decomposes and separates katakana "accent" marks (dakuten and handakuten) appropriately, whereas ToHalfwidth does not provide any special treatment for these marks.
 | unicode | Returns a Unicode string from the string of hexadecimal values provided. The syntax of the literal for UNICODE is `
 | {'whitespace | U+ | \uXXXX...'
 | where X is a valid hexadecimal digit: 0-9, a-f, or A-F. The hexadecimal value will always be in big-endian form.
 | Syntax: dst_var = unicode(source_value)
 | source_value = text literal, column, variable or expression
 | dst_var = text variable
 | Example: let $uniStr = unicode ('U+5E73 U+2294')

## Miscellaneous Functions

Note:

Miscellaneous functions return a string value unless otherwise indicated.

<table>
<thead>
<tr>
<th>Function</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| array | 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 printarray in UFUNC.C.
 | Syntax: array_var = array(array_name, field_name)
 | array_name = text literal, column, variable, or expression
 | field_name = text literal, column, variable, or expression
 | array_var = text variable
 | Example: let #fstatus = printarray(array('products', 'name'), 10, 2, 'c')
| command_line | Returns command line arguments passed to SQR (or SQRT).
 | Syntax: dst_var = command_line( )
 | dst_var = text variable
 | Example: let $cmdline = command_line( )
| cond | Returns y_value if x_value is nonzero (0); otherwise, returns z_value. If y_value is numeric, then z_value must also be numeric; otherwise, date and textual arguments are compatible. If either

---

212  Production Reporting Command Reference
<table>
<thead>
<tr>
<th>Function</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>y_value</strong> or <strong>z_value</strong> is a date variable, column, or expression, a date is returned. The return value of the function depends on which value is returned.</td>
</tr>
<tr>
<td></td>
<td>Syntax: ( \text{dst}_\text{var} = \text{cond} (x_value, y_value, z_value) )</td>
</tr>
<tr>
<td></td>
<td>* ( x_value ) = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</td>
</tr>
<tr>
<td></td>
<td>* ( y_value ) = Any literal, column, variable, or expression</td>
</tr>
<tr>
<td></td>
<td>* ( z_value ) = Any literal, column, variable, or expression</td>
</tr>
<tr>
<td></td>
<td>* ( \text{dst}_\text{var} ) = Any variable</td>
</tr>
<tr>
<td></td>
<td>Example: let #avg = #total / \text{cond}(&amp;rate != 0, &amp;rate, 1)</td>
</tr>
<tr>
<td></td>
<td><strong>getenv</strong></td>
</tr>
<tr>
<td></td>
<td>Returns the value of the environment variable. If the environment variable does not exist, an empty string is returned.</td>
</tr>
<tr>
<td></td>
<td>Syntax: ( \text{dst}_\text{var} = \text{getenv} (\text{env}_value) )</td>
</tr>
<tr>
<td></td>
<td>* ( \text{env}_value ) = text literal, column, variable, or expression</td>
</tr>
<tr>
<td></td>
<td>* ( \text{dst}_\text{var} ) = text variable</td>
</tr>
<tr>
<td></td>
<td>Example: let $myuser = getenv('USER')</td>
</tr>
<tr>
<td></td>
<td><strong>getfilemapname</strong></td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Syntax: ( \text{dst}_\text{var} = \text{getfilemapname} (\text{source}_value) )</td>
</tr>
<tr>
<td></td>
<td>* ( \text{source}_value ) = text literal, column, variable, or expression</td>
</tr>
<tr>
<td></td>
<td>* ( \text{dst}_\text{var} ) = text variable</td>
</tr>
<tr>
<td></td>
<td>Example: let $realfile = getfilemapname('data.fil')! get real filename is run under EPM Workspace let #Status = System('cp '</td>
</tr>
<tr>
<td></td>
<td><strong>isnull</strong></td>
</tr>
<tr>
<td></td>
<td>Returns one (1) if ( \text{source}_value ) is null; otherwise, returns zero (0).</td>
</tr>
<tr>
<td></td>
<td>Syntax: ( \text{dst}_\text{var} = \text{isnull} (\text{source}_value) )</td>
</tr>
<tr>
<td></td>
<td>* ( \text{source}_value ) = date or text literal, column, variable, or expression</td>
</tr>
<tr>
<td></td>
<td>* ( \text{dst}_\text{var} ) = decimal, float, or integer variable</td>
</tr>
<tr>
<td></td>
<td>Example: let #null = isnull($date)</td>
</tr>
<tr>
<td></td>
<td><strong>isnumber</strong></td>
</tr>
<tr>
<td></td>
<td>Returns one (1) if ( \text{source}_value ) 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.</td>
</tr>
<tr>
<td></td>
<td>Syntax: ( \text{dst}_\text{var} = \text{isnumber} (\text{source}_value) )</td>
</tr>
<tr>
<td></td>
<td>* ( \text{source}_value ) = text literal, column, variable, or expression</td>
</tr>
<tr>
<td></td>
<td>* ( \text{dst}_\text{var} ) = decimal, float, or integer variable</td>
</tr>
<tr>
<td></td>
<td>Example: let #isnumber = isnumber($string)</td>
</tr>
<tr>
<td></td>
<td><strong>nvl</strong></td>
</tr>
<tr>
<td></td>
<td>Returns ( y_value ) if the ( x_value ) is null; otherwise, returns ( x_value ). If ( x_value ) is numeric, ( y_value ) must also be numeric; otherwise, date and textual arguments are compatible. In any case, the ( x_value ) determines the type of expression returned. The return value of the function depends on which value is returned.</td>
</tr>
<tr>
<td></td>
<td>Syntax: ( \text{dst}_\text{var} = \text{nvl} (x_value, y_value) )</td>
</tr>
<tr>
<td></td>
<td>* ( x_value ) = Any literal, column, variable, or expression</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Explanation</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| ● $y\_value$ = Any literal, column, variable, or expression  
● $dst\_var$ = Any variable  
**Example:** let $\text{city} = \text{nvl}(&\text{city}, '--\ not\ city--$')  
If $x\_value$ is a date and $y\_value$ is textual, then $y\_value$ is validated according to the following rules:  
● 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][NNNNNNNNNN]').  
● 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 | Returns one (1) if $x\_value$ is between $y\_value$ and $z\_value$; 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 $x\_value$ is a date and $y\_value$ and/or $z\_value$ are text arguments. In a comparison of this sort, $y\_value$ must represent a date that is earlier than that of $z\_value$.  
**Syntax:** $dst\_var$ = range($x\_value$, $y\_value$, $z\_value$)  
● $x\_value$ = Any literal, column, variable, or expression  
● $y\_value$ = Any literal, column, variable, or expression  
● $z\_value$ = Any literal, column, variable, or expression  
● $dst\_var$ = decimal, float, or integer variable  
**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)  
If $x\_value$ is a date and $y\_value$ and/or $z\_value$ is textual, then $y\_value$ and/or $z\_value$ is validated according to the following rules:  
● 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][NNNNNNNNNN]').  
● 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 | Returns a string that is the character representation of $source\_value$ expressed in lower case roman numerals.  
**Syntax:** $dst\_var$ = roman($source\_value$)  
● $source\_value$ = text literal, column, variable, or expression.  
● $dst\_var$ = text variable |
### Function

<table>
<thead>
<tr>
<th>Function</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> let $\text{roman} = \text{roman (#page-count)}$</td>
<td></td>
</tr>
<tr>
<td><strong>wrapdepth</strong></td>
<td>Returns the number of print lines required by $\text{source_value}$. See the WRAP argument in PRINT for detailed descriptions of the parameters to this function. This function returns a float value.</td>
</tr>
</tbody>
</table>

**Syntax:**
```
dst\_var = \text{wrapdepth}(source\_value, wrap\_width, line\_height, on, strip)
```

- $\text{source\_value} = \text{text literal, column, variable, or expression}$
- $\text{wrap\_width} = \text{decimal, float, or integer literal, column, variable, or expression}$
- $\text{line\_height} = \text{decimal, float, or integer literal, column, variable, or expression}$
- $\text{on} = \text{text literal, column, variable, or expression}$
- $\text{strip} = \text{text literal, column, variable, or expression}$
- $\text{dst\_var} = \text{decimal, float, or integer variable}$

**Example:** let $\#depth = \text{wrapdepth(\&description,40,1,'<13>',"\")}$

### 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.

   ```c
   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.

3. At the end of the UFUNC.C module add the `sqr_ufunc_rand` routine.

   ```c
   /*
   * 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_ARGL(int, argc)       /* Number of actual arguments */
   CC_ARGL(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]
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} [,...] | [null] ... )]
(or)
COMMAND={command_txt_lit}
(or)
GETDATA={getdata_txt_lit}
]
[(FROM-ROWSETS=({m|m-n|m-|-n} [,...])|{ALL})]
{FROM-PARAMETER=parameter_txt_lit}]

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
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_abbr $state_name

This example searches the states lookup table for a matching &state_abbr 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`

/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:

```plaintext
! 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:

```plaintext
! 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:

```plaintext
! 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:

```plaintext
! 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-LOCAL** 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
COLUMN 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**

```plaintext
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**

```plaintext
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
ame
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:

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKGROUND</td>
<td>EDIT</td>
<td>POINT-SIZE</td>
</tr>
<tr>
<td>BOLD</td>
<td>FILL</td>
<td>SHADE</td>
</tr>
<tr>
<td>BOX</td>
<td>FONT</td>
<td>UNDERLINE</td>
</tr>
<tr>
<td>BOX-FILL-COLOR</td>
<td>FOREGROUND</td>
<td>URL</td>
</tr>
<tr>
<td>BOX-LINE-COLOR</td>
<td>ITALIC</td>
<td>URL-TARGET</td>
</tr>
<tr>
<td>CENTER</td>
<td>MATCH</td>
<td>WRAP</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Command Combination</th>
<th>BOLD</th>
<th>BOX</th>
<th>BOX-FILL-COLOR</th>
<th>BOX-LINE-COLOR</th>
<th>CENTER</th>
<th>CENTER-WITHIN</th>
<th>CODE-PRINTER</th>
<th>DELAY</th>
<th>NUMBER</th>
<th>MONEY</th>
<th>DATE</th>
<th>FILL</th>
<th>FONT</th>
<th>F/B</th>
<th>ITALIC</th>
<th>MATCH</th>
<th>NOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOLD</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>BOX</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>BOX-FILL-COLOR</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>BOX-LINE-COLOR</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>CENTER</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>CENTER-WITHIN</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>CODE-PRINTER</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>DELAY</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>EDIT NUMBER MONEY DATE</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>FILL</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>FONT</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>F/B</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>ITALIC</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>MATCH</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>NOP</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>BOLD</td>
<td>BOX</td>
<td>BOX-FILL-COLOR</td>
<td>BOX-LINE-COLOR</td>
<td>CENTER</td>
<td>CENTER-WITHIN</td>
<td>CODE-PRINTER</td>
<td>DELAY</td>
<td>EDIT</td>
<td>EDIT</td>
<td>NUMBER</td>
<td>M N E Y</td>
<td>F I L L</td>
<td>F/B</td>
<td>I T /A L I C</td>
<td>MATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----</td>
<td>----------------</td>
<td>----------------</td>
<td>--------</td>
<td>---------------</td>
<td>--------------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>----</td>
<td>--------------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ON-BREAK</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POINT-SIZE</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHADE</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL-TARGET</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRAP</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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:

```plaintext
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

<table>
<thead>
<tr>
<th>Column Type</th>
<th>Default Mask</th>
<th>If not set</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATETIME</td>
<td>SQR_DB_DATE_FORMAT</td>
<td>See Table 61, “Default Formats by Database,” on page 251 for the format.</td>
</tr>
<tr>
<td>DATE</td>
<td>SQR_DB_DATE_ONLY_FORMAT</td>
<td>See Table 61, “Default Formats by Database,” on page 251 for the format.</td>
</tr>
<tr>
<td>TIME</td>
<td>SQR_DB_TIME_ONLY_FORMAT</td>
<td>See Table 63, “TIME Column Formats,” on page 252 for the format.</td>
</tr>
</tbody>
</table>
```

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
```
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>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Use character in field.</td>
</tr>
<tr>
<td>B</td>
<td>Insert blank.</td>
</tr>
<tr>
<td>~ (tilde)</td>
<td>Skip character in field.</td>
</tr>
<tr>
<td>R[n]</td>
<td>Reverse sequence of string, for languages such as Hebrew. The optional number indicates right justification within length indicated.</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Digit, zero fill to the right of the decimal point, trim leading blanks (left justify the number).</td>
</tr>
<tr>
<td>9</td>
<td>Digit, zero fill to the right of the decimal point, space fill to the left.</td>
</tr>
<tr>
<td>0</td>
<td>Digit, zero fill to the left.</td>
</tr>
<tr>
<td>$</td>
<td>Dollar sign, optionally floats to the right.</td>
</tr>
<tr>
<td>Character</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>B</td>
<td>Treated as a “9”, but if a value is zero, the field is converted to blanks.</td>
</tr>
<tr>
<td>C</td>
<td>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).</td>
</tr>
<tr>
<td>E</td>
<td>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.</td>
</tr>
<tr>
<td>V</td>
<td>Implied decimal point.</td>
</tr>
<tr>
<td>MI</td>
<td>Entered at the end of the mask, causes a minus to display at the right of the number.</td>
</tr>
<tr>
<td>PR</td>
<td>Entered at the end of the mask, causes angle brackets (&lt; &gt;) to display around the number if the number is negative.</td>
</tr>
<tr>
<td>PS</td>
<td>Entered at the end of the mask, causes parentheses to display around the number if the number is negative.</td>
</tr>
<tr>
<td>PF</td>
<td>Entered at the end of the mask, causes floating parentheses to display around the number if the number is negative.</td>
</tr>
<tr>
<td>NA</td>
<td>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.</td>
</tr>
<tr>
<td>NU</td>
<td>Entered at the end of the mask, causes blanks to display if the numeric column variable is null.</td>
</tr>
<tr>
<td>.</td>
<td>Decimal point.</td>
</tr>
<tr>
<td>,</td>
<td>Comma.</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYY YY Y</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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 “-”.
<p>| BC AD     | BC/AD indicator. |</p>
<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>Quarter of year (1,2,3,4; JAN-MAR=1).</td>
</tr>
<tr>
<td>RM</td>
<td>Roman numeral month (I-XII; JAN=I).</td>
</tr>
<tr>
<td>WW</td>
<td>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.</td>
</tr>
<tr>
<td>W</td>
<td>Week of the month (1-5) where week 1 starts on the first day of the month and ends on the seventh.</td>
</tr>
<tr>
<td>DDD</td>
<td>Day of year (1-366).</td>
</tr>
<tr>
<td>DD</td>
<td>Day of month (1 - 31).</td>
</tr>
<tr>
<td>D</td>
<td>Day of week (1-7). Sunday is first day of week.</td>
</tr>
<tr>
<td>DAY</td>
<td>Name of day.</td>
</tr>
<tr>
<td>DY</td>
<td>Abbreviated name of day.</td>
</tr>
<tr>
<td>ER</td>
<td>Japanese Imperial Era. Returns the name of the of the Japanese Imperial Era in the appropriate kanji (‘Heisei’ is the current era).</td>
</tr>
</tbody>
</table>
| 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** | 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

<table>
<thead>
<tr>
<th>Last 2 digits of current year</th>
<th>2-digit year is 00 - 49</th>
<th>2-digit year is 50 - 99</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 - 49</td>
<td>The return date is in the current century.</td>
<td>The return date is in the century before the current one.</td>
</tr>
<tr>
<td>50 - 99</td>
<td>The return date is in the century after the current one.</td>
<td>The return date is in the current century.</td>
</tr>
</tbody>
</table>

### 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 `strtotime` 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 current month 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`, and `AD` 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

<table>
<thead>
<tr>
<th>Edit Mask</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>dd/mm/yy</td>
<td>14/03/96</td>
</tr>
<tr>
<td>DD-MON-YYYY</td>
<td>14-MAR-1996</td>
</tr>
<tr>
<td>'Month dd, YYYY'</td>
<td>March 14, 1996</td>
</tr>
<tr>
<td>MONTH-YYYY</td>
<td>MARCH-1996</td>
</tr>
</tbody>
</table>
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:

```plaintext
let $date1 = strtoRdate('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:

```plaintext
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>
<thead>
<tr>
<th>Mask</th>
<th>Value</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>999.99</td>
<td>34.568</td>
<td>34.57</td>
</tr>
<tr>
<td>9,999,9999</td>
<td>123,456.7890</td>
<td>123,4567890</td>
</tr>
<tr>
<td>8,888,888,888</td>
<td>123,456.789</td>
<td>123,456.789</td>
</tr>
<tr>
<td>9,999</td>
<td>1234</td>
<td>1,234</td>
</tr>
<tr>
<td>9,999</td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>09999</td>
<td>1234</td>
<td>01234</td>
</tr>
<tr>
<td>9999</td>
<td>-123</td>
<td>-123</td>
</tr>
<tr>
<td>9999</td>
<td>-1234</td>
<td>****</td>
</tr>
<tr>
<td>9999</td>
<td>12345</td>
<td>****</td>
</tr>
<tr>
<td>9999mi</td>
<td>-123</td>
<td>123-</td>
</tr>
<tr>
<td>9999pr</td>
<td>-123</td>
<td>&lt; 123&gt;</td>
</tr>
<tr>
<td>99999ps</td>
<td>-123</td>
<td>( 123)</td>
</tr>
<tr>
<td>99999pf</td>
<td>-123</td>
<td>(123)</td>
</tr>
<tr>
<td>9999na</td>
<td>(null)</td>
<td>n/a</td>
</tr>
<tr>
<td>9999nu</td>
<td>(null)</td>
<td>(blank)</td>
</tr>
<tr>
<td>$$9,999.99c</td>
<td>1234.56</td>
<td>$1,234.56</td>
</tr>
<tr>
<td>$$9,999.99</td>
<td>1234.56</td>
<td>$1,234.56</td>
</tr>
<tr>
<td>$$9,999.99</td>
<td>12.34</td>
<td>$12.34</td>
</tr>
<tr>
<td>$$9,999.99</td>
<td>12.34</td>
<td>$12.34</td>
</tr>
<tr>
<td>9.999e</td>
<td>123456</td>
<td>1.235e+05</td>
</tr>
<tr>
<td>B9,999</td>
<td>0</td>
<td>(blank)</td>
</tr>
<tr>
<td>B9,999</td>
<td>12345</td>
<td>12,345</td>
</tr>
<tr>
<td>(xxx)bxxx-xxxx</td>
<td>2169910551</td>
<td>(216) 991-0551</td>
</tr>
<tr>
<td>xxx-xx-xxxx</td>
<td>123456789</td>
<td>123-45-6789</td>
</tr>
<tr>
<td><del>xx</del></td>
<td>ABCDEFGHU</td>
<td>CDFG</td>
</tr>
<tr>
<td>r10</td>
<td>ABCDEFG</td>
<td>GFEDCBA</td>
</tr>
</tbody>
</table>
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 999,999.99! < 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 £££9,999.99! 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.

<table>
<thead>
<tr>
<th>Print Statement</th>
<th>Display</th>
<th>Current Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print 'ABCDEFGHIJ' (1,1,5)</td>
<td>ABCDE</td>
<td>(1,1,6)</td>
</tr>
<tr>
<td>Edit xxxxxxxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print 'ABCDEFGHIJ' (1,1,5)</td>
<td>ABCDE</td>
<td>(1,1,6)</td>
</tr>
<tr>
<td>Edit xxxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print 'ABCDE' (1,1,10)</td>
<td>ABCDE</td>
<td>(1,1,11)</td>
</tr>
<tr>
<td>Edit xxxxxxxxxxxxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print 'ABCDE' (1,1,10)</td>
<td>ABCDE</td>
<td>(1,1,11)</td>
</tr>
<tr>
<td>Edit xxxxxx</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Date Edit Masks** – Print width sets the length allocated.

<table>
<thead>
<tr>
<th>Print Statement</th>
<th>Display</th>
<th>Current Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print $current-date (1,1,5)</td>
<td>16/05</td>
<td>(1,1,6)</td>
</tr>
<tr>
<td>Edit DD/MM/YYYY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print $current-date (1,1,5)</td>
<td>16/05</td>
<td>(1,1,6)</td>
</tr>
<tr>
<td>Edit DD/MM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print $current-date (1,1,10)</td>
<td>16/05/2003</td>
<td>(1,1,11)</td>
</tr>
<tr>
<td>Edit DD/MM/YYYY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print $current-date (1,1,10)</td>
<td>16/05</td>
<td>(1,1,11)</td>
</tr>
<tr>
<td>Edit DD/MM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Numeric Scientific Edit Masks** – Print width sets the length allocated.
Print Statement | Display | Current Position
--- | --- | ---
Print 1234567890 (1,1,5) Edit 9e99999999 | 1.23 | (1,1,6)

Print 1234567890 (1,1,10) Edit 9e99999999 | 1.234568e | (1,1,11)

Print 1234567890 (1,1,20) Edit 9e99999999 | 1.234568e+009 | (1,1,21)

- All other Numeric Edit Masks – Edit mask sets the length allocated.

<table>
<thead>
<tr>
<th>Print Statement</th>
<th>Display</th>
<th>Current Position</th>
</tr>
</thead>
</table>
| Print 1234567890 (1,1,5) Edit 99999999999 | 12345 | (1,1,11)

Print 1234567890 (1,1,5) Edit 9999999999 | 12345 | (1,1,11)

Print 1234567890 (1,1,20) Edit 8888888888 | 12345 | (1,1,21)

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**

<table>
<thead>
<tr>
<th>Database</th>
<th>Default Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>YYY-MM-DD-HH:MI:SS.NNNNNN YYYY-MM-DD</td>
</tr>
<tr>
<td>Database</td>
<td>Default Formats</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
</tr>
<tr>
<td>Informix</td>
<td>YYYY-MM-DD HH:MI:SS.NNN MM/DD/YYYY MM-DD-YYYY MM.DD.YYYY</td>
</tr>
<tr>
<td>ODBC</td>
<td>'MON DD YYYY HH:MIPM'</td>
</tr>
<tr>
<td>Oracle</td>
<td>DD-MON-YY</td>
</tr>
</tbody>
</table>

Table 62  DATE Column Formats

<table>
<thead>
<tr>
<th>Database</th>
<th>DATE Column Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>YYYY-MM-DD</td>
</tr>
<tr>
<td>Informix</td>
<td>MM/DD/YYYY</td>
</tr>
<tr>
<td>ODBC</td>
<td>DD-MON-YY</td>
</tr>
</tbody>
</table>

Table 63  TIME Column Formats

<table>
<thead>
<tr>
<th>Database</th>
<th>TIME Column Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>HH24.MI:SS</td>
</tr>
<tr>
<td>ODBC</td>
<td>HH24:MI:SS</td>
</tr>
</tbody>
</table>

**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:

```plaintext
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:

```plaintext
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
To use relative line and fixed column positioning, you could enter:

```plaintext
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:

```plaintext
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

```plaintext
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:
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:

```plaintext
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**

```plaintext
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:

```plaintext
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:

```plaintext
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:

```plaintext
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:

```plaintext
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):

```plaintext
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.

```plaintext
! 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Text Length</th>
<th>Text Type</th>
<th>CHECKSUM RECOGNIZED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UPC-A</td>
<td>11, 13, or 16</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>UPC-E</td>
<td>11, 13, or 16</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>EAN/JAN-13</td>
<td>12, 14, or 17</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>EAN/JAN-8</td>
<td>7, 9, or 12</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3 of 9 (Code 39)</td>
<td>1 to 100</td>
<td>9, X, p</td>
<td>y</td>
</tr>
<tr>
<td>6</td>
<td>Extended 3 of 9</td>
<td>1 to 100</td>
<td>9, X, x, p, c</td>
<td>y</td>
</tr>
<tr>
<td>7</td>
<td>Interleaved 2 of 5</td>
<td>2 to 100</td>
<td>9</td>
<td>y</td>
</tr>
<tr>
<td>8</td>
<td>Code 128</td>
<td>1 to 100</td>
<td>9, X, x, p, c</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Codabar</td>
<td>1 to 100</td>
<td>9, p</td>
<td>y</td>
</tr>
<tr>
<td>10</td>
<td>Zip+4 Postnet</td>
<td>5, 9, or 11</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>MSI Plessey</td>
<td>1 to 100</td>
<td>9</td>
<td>y</td>
</tr>
<tr>
<td>12</td>
<td>Code 93</td>
<td>1 to 100</td>
<td>9, X, p</td>
<td>y</td>
</tr>
<tr>
<td>13</td>
<td>Extended 93</td>
<td>1 to 100</td>
<td>9, X, x, p</td>
<td>y</td>
</tr>
<tr>
<td>14</td>
<td>UCC-128</td>
<td>19</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>HIBC</td>
<td>1 to 100</td>
<td>9</td>
<td>y</td>
</tr>
</tbody>
</table>

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

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
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}]}
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='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>
<thead>
<tr>
<th>Chart Type</th>
<th>Field 0</th>
<th>Field 1</th>
<th>Field 2</th>
<th>Field 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUBBLE</td>
<td>Type=num</td>
<td>Type=num</td>
<td>Type=num</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X-Axis values</td>
<td>Y-Axis values</td>
<td>radius of bubble at (x,y)</td>
<td></td>
</tr>
<tr>
<td>PIE</td>
<td>Type=char</td>
<td>Type=num</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pie segment labels, the names associated with each segment</td>
<td>The value associated with each pie segment</td>
<td>(Optional)</td>
<td></td>
</tr>
<tr>
<td>LINE BAR</td>
<td>Type=char</td>
<td>Type=num</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Optional)</td>
<td>(Optional)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chart Type</td>
<td>Field 0</td>
<td>Field 1</td>
<td>Field 2</td>
<td>Field 3</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>STACKED-BAR</td>
<td>X-Axis values</td>
<td>Series 1</td>
<td>Type=num Series 2</td>
<td>Type=num</td>
</tr>
<tr>
<td>100%-BAR</td>
<td></td>
<td>Y-Axis values</td>
<td></td>
<td>Series 3...</td>
</tr>
<tr>
<td>OVERLAPPED-BAR</td>
<td></td>
<td></td>
<td></td>
<td>Y-Axis values</td>
</tr>
<tr>
<td>HISTOGRAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AREA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STACKED-AREA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%-AREA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XY-SCATTER-POINT</td>
<td>Type=num</td>
<td>Type=num</td>
<td>(Optional)</td>
<td>(Optional)</td>
</tr>
<tr>
<td></td>
<td>Series 1</td>
<td>Series 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X-Axis values</td>
<td>Y-Axis values</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOATING-BAR</td>
<td>Type=char</td>
<td>Type=num</td>
<td>Type=num</td>
<td>(Optional)</td>
</tr>
<tr>
<td></td>
<td>X-Axis values</td>
<td>Series 1</td>
<td>Series 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y-Axis offset</td>
<td>Y-Axis duration</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 66**  Chart Array Field Types for HIGH-LOW-CLOSE

<table>
<thead>
<tr>
<th>Chart Type</th>
<th>Field 0</th>
<th>Field 1</th>
<th>Field 2</th>
<th>Field 3</th>
<th>Field 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH-LOW-CLOSE</td>
<td>Type=char X-Axis values</td>
<td>Type=num High value</td>
<td>Type=num Low value</td>
<td>Type=num Closing value</td>
<td>(Optional) Type=num Opening value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Examples**

![Chart Image]

**Employee Sales Overlapped-Bar Chart**

**Sales in Percentages**

<table>
<thead>
<tr>
<th>Field 0</th>
<th>Field 1</th>
<th>Field 2</th>
<th>Field 3</th>
<th>Field 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales</td>
<td>Field 1</td>
<td>Field 2</td>
<td>Field 3</td>
<td>Field 4</td>
</tr>
<tr>
<td>Field 1</td>
<td>Field 2</td>
<td>Field 3</td>
<td>Field 4</td>
<td>Field 5</td>
</tr>
</tbody>
</table>

**Print-Chart 267**
Note:

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:

```plaintext
print-image office-signature (50, 20)
print-image (50, 20)
  type = eps-file
  source = 'sherman.eps'
  image-size = (10, 3)
```

For Windows:

```plaintext
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

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 `"YYYYMMDD[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 (#).
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

```sql
if #error-status = 1
  rollback
  stop
end-if
```

See Also

COMMIT

**SBTOMBS**

**Function**

Converts a single-byte character into a multi-byte equivalent.

**Syntax**

```sql
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 \texttt{SHOW} without any cursor position when you want to scroll. You cannot mix \texttt{SHOW} and \texttt{DISPLAY} while referencing relative cursor positions.

\texttt{SHOW} does not advance to the next line if a cursor location (...) or \texttt{BEEP} is used. (\texttt{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 \texttt{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 \texttt{DATETIME} columns and Production Reporting \texttt{DATE} variables, Production Reporting uses the format specified by \texttt{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 \texttt{DATE} columns, Production Reporting uses the format specified by \texttt{SQR_DB_DATE_ONLY_FORMAT}. If not set, Production Reporting uses the format in Table 62, “DATE Column Formats,” on page 252.

- For \texttt{TIME} columns, Production Reporting uses the format specified by \texttt{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 \texttt{EDIT} or \texttt{DATE}, the string must be in the format specified by \texttt{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]]]].

\textbf{Examples}

The following code:

\begin{verbatim}
! Show a string using an edit mask
! let $ssn = '123456789'
show $ssn edit xxx-xx-xxxx
\end{verbatim}

Produces the following output:

123-45-6789

The following code:

\begin{verbatim}
! Show a number using an edit mask
! show 1234567.89 edit 999,999,999.99
\end{verbatim}

Produces the following output:

1,234,567.89
The following code:

```plaintext
!  
! Show a number using the default edit mask
! show 123.78
```

Produces the following output:

123.780000

The following code:

```plaintext
!  
! 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:

```plaintext
!  
! 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:

```plaintext
!  
! 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:

```plaintext
!  
! Show two values on the same line
! show 'Hello' ' World'
```

Produces the following output:
Hello World

The following code:

```plaintext
!  ! 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.

```plaintext
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**

**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 no 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 \textit{dst\textunderscore num\textunderscore var}.

\textit{dst\textunderscore num\textunderscore 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.

\textbf{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.

\textbf{Examples}

\begin{verbatim}
subtract 1 from #total      ! #total - 1
subtract &discount from #price  #price - &discount
\end{verbatim}

\textbf{See Also}

\begin{itemize}
  \item \textbf{ADD}
  \item \textbf{LET} for information on complex arithmetic expressions
\end{itemize}
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'
upercase $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**

```plaintext
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**

```plaintext
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:

```plaintext
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.

```plaintext
move 1 to #have_data
...
while #have_data
    ...processing...
end-while
```

The following example sets up an infinite loop:

```plaintext
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:

```plaintext
while #count < 100 and (not #end-file or isnull(&state))
    ...
end-while
```

The following example shows the use of CONTINUE in a WHILE loop:

```plaintext
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
{{(text_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.

{{(text_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**

```plaintext
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**

```plaintext
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**

```sql
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
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>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>html_br</strong></td>
<td>Line break in a paragraph.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_br(number count, string attributes)</td>
</tr>
<tr>
<td></td>
<td>● count = Number of &lt;BR&gt; tags</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;BR&gt;</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>print 'Here is some text' ()</td>
</tr>
<tr>
<td></td>
<td>do html_br(3,'')</td>
</tr>
<tr>
<td></td>
<td>print 'Here is some three lines down' ()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>html_center</strong></th>
<th>Start of centered text. (You can also use PRINT and specify CENTER in the code.)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>html_center(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;CENTER&gt;</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>print 'Here is some centered text' ()</td>
</tr>
<tr>
<td>Procedure</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| do html_center('')
print 'Here is some text' ()
do html_center_end | End of centered text. Syntax: html_center_end |
| do html_hr('')
print 'And some more text' () | Horizontal divider between sections of text. Syntax: html_hr(string attributes)
- attributes = HTML attributes inside <HR>
Example:
print 'Here is some text' ()
do html_hr('')
print 'And some more text' () |
| do html_img('src="/images/stop.gif"') | Image. (You can also use PRINT-IMAGE; however, html_img allows you to define the full set of available HTML attributes.) Syntax: html_img(string attributes)
- attributes = HTML attributes inside <IMG>
Common attributes: src=URL of image (Example: src="/images/abc.gif")
height=Image height in pixels (Example: height=200)
width=Image width in pixels (Example: width=400)
Example:
do html_img('src="/images/stop.gif"') |
| do html_nobr('')
print 'Here's long text that should not wrap' ()
do html_nobr_end | Start of text that cannot be wrapped. Syntax: html_nobr
Example:
do html_nobr('')
print 'Here's long text that should not wrap' ()
do html_nobr_end |
| do html_on | Turns on HTML procedures. Called at the start of Production Reporting programs. Syntax: html_on
Example
do html_on |
| do html_p('') | Start of a paragraph. Syntax: html_p(string attributes)
- attributes = HTML attributes inside <P>
Some common attributes:
align = left | right | center - Alignment of the paragraph. |
### Procedure Description

**Example:**
```
  do html_p('ALIGN=RIGHT')
  print 'Right aligned text' (1,1)
  do html_p_end
  print 'Normal text' (+1,1)
```

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>html_p_end</strong></td>
<td>End of a paragraph. Typically implied; however, can be defined for completeness.</td>
</tr>
<tr>
<td><strong>html_set_body_attributes</strong></td>
<td>Attributes inside <code>&lt;BODY&gt;</code>. Called at the start of Production Reporting programs.</td>
</tr>
<tr>
<td><strong>html_set_head_tags</strong></td>
<td>Tags between <code>&lt;HEAD&gt;</code> and <code>&lt;/HEAD&gt;</code>. (Empty by default.) Called at the start of Production Reporting programs.</td>
</tr>
</tbody>
</table>

### HTML Heading Procedures

**Table 68  HTML Heading Procedures**

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>html_h1</strong></td>
<td>Start of heading level one text.</td>
</tr>
<tr>
<td><strong>html_h1_end</strong></td>
<td>End of heading level one text.</td>
</tr>
</tbody>
</table>

**Example:**
```
do html_h1('')
print 'This is a heading' ()
do html_h1_end
```
<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html_h2</td>
<td>Start of heading level two text.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_h2(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;H2&gt;</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>do html_h2('')</td>
</tr>
<tr>
<td></td>
<td>print 'This is a heading' ()</td>
</tr>
<tr>
<td></td>
<td>do html_h2_end</td>
</tr>
<tr>
<td>html_h2_end</td>
<td>End of heading level two text.</td>
</tr>
<tr>
<td>html_h3</td>
<td>Start of heading level three text. (This heading is the default.)</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_h3(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;H3&gt;</td>
</tr>
<tr>
<td>html_h3_end</td>
<td>End of heading level three text.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_h3_end</td>
</tr>
<tr>
<td>html_h4</td>
<td>Start of heading level four text.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_h4(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;H4&gt;</td>
</tr>
<tr>
<td>html_h4_end</td>
<td>End of heading level four text.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_h4_end</td>
</tr>
<tr>
<td>html_h5</td>
<td>Start of heading level five text.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_h5(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;H5&gt;</td>
</tr>
<tr>
<td>html_h5_end</td>
<td>End of heading level five text.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_h5_end</td>
</tr>
<tr>
<td>html_h6</td>
<td>Start of heading level six text.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_h6(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;H6&gt;</td>
</tr>
<tr>
<td>html_h6_end</td>
<td>End of heading level six text.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_h6_end</td>
</tr>
</tbody>
</table>
## HTML Highlighting Procedures

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
</table>
| **html_blink** | Start of blinking text.  
**Syntax:** html_blink(string attributes)  
- attributes = HTML attributes inside <BLINK>  
**Example:**  
do html_blink("")  
print 'This is blinking' ()  
do html_blink_end |
| **html_blink_end** | End of blinking text.  
**Syntax:** html_blink_end |
| **html_cite** | Start of citation text.  
**Syntax:** html_cite(string attributes)  
- attributes = HTML attributes inside <CITE>  
**Example:**  
do html_cite("")  
print 'This is a citation' ()  
do html_cite_end |
| **html_cite_end** | End of citation text.  
**Syntax:** html_cite_end |
| **html_code**  | Start of code text.  
**Syntax:** html_code(string attributes)  
- attributes = HTML attributes inside <CODE>  
**Example:**  
do html_code("")  
print 'Here is the code' ()  
do html_code_end |
| **html_code_end** | End of code text.  
**Syntax:** html_code_end |
| **html_kbd**  | Start of keyboard input text.  
**Syntax:** html_kbd(string attributes)  
- attributes = HTML attributes inside <KBD>  
**Example:**  
do html_kbd("")  
print 'Here is keyboard' ()  
do html_kbd_end |
<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html_kbd_end</td>
<td>End of keyboard input text.</td>
</tr>
<tr>
<td></td>
<td>Syntax: html_kbd_end</td>
</tr>
<tr>
<td>html_samp</td>
<td>Start of sample text.</td>
</tr>
<tr>
<td></td>
<td>Syntax: html_samp(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;SAMP&gt;</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>do html_samp('')</td>
</tr>
<tr>
<td></td>
<td>print 'Here is sample' ()</td>
</tr>
<tr>
<td></td>
<td>do html_samp_end</td>
</tr>
<tr>
<td>html_samp_end</td>
<td>End of sample text.</td>
</tr>
<tr>
<td></td>
<td>Syntax: html_samp_end</td>
</tr>
<tr>
<td>html_strike</td>
<td>Start of strike-through text.</td>
</tr>
<tr>
<td></td>
<td>Syntax: html_strike(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;STRIKE&gt;</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>do html_strike('')</td>
</tr>
<tr>
<td></td>
<td>print 'Here is strike-through' ()</td>
</tr>
<tr>
<td></td>
<td>do html_strike_end</td>
</tr>
<tr>
<td>html_strike_end</td>
<td>End of strike-through text.</td>
</tr>
<tr>
<td></td>
<td>Syntax: html_strike_end</td>
</tr>
<tr>
<td>html_sub</td>
<td>Start of subscript text.</td>
</tr>
<tr>
<td></td>
<td>Syntax: html_sub(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;SUB&gt;</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>print 'Here is' ()</td>
</tr>
<tr>
<td></td>
<td>do html_sub('')</td>
</tr>
<tr>
<td></td>
<td>print 'subscript text' ()</td>
</tr>
<tr>
<td></td>
<td>do html_sub_end</td>
</tr>
<tr>
<td>html_sub_end</td>
<td>End of subscript text.</td>
</tr>
<tr>
<td></td>
<td>Syntax: html_sub_end</td>
</tr>
<tr>
<td>html_sup</td>
<td>Start of superscript text.</td>
</tr>
<tr>
<td></td>
<td>Syntax: html_sup(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside &lt;SUP&gt;</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>print 'Here is' ()</td>
</tr>
<tr>
<td></td>
<td>do html_sup('')</td>
</tr>
<tr>
<td></td>
<td>print 'superscript text' ()</td>
</tr>
<tr>
<td></td>
<td>do html_sup_end</td>
</tr>
</tbody>
</table>
### HTML Hypertext Link Procedures

**Table 70** HTML Hypertext Link Procedures

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html_a</td>
<td>Start of a hypertext link.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_a(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside <code>&lt;A&gt;</code>. At a minimum, define HREF, which specifies the URL of HTML documents.</td>
</tr>
<tr>
<td></td>
<td>Common attributes:</td>
</tr>
<tr>
<td></td>
<td>• href–Where the hypertext link points. (Example: href=home.html)</td>
</tr>
<tr>
<td></td>
<td>• name–Anchor to which a hypertext link can point. (Example: name=marker1)</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> Create an anchor with two hypertext links. Position the anchor at the top of the document. Point the first hypertext link to otherdoc.html. Point the second hypertext link to the anchor named TOP.</td>
</tr>
<tr>
<td></td>
<td>do html_a('NAME=TOP')</td>
</tr>
<tr>
<td></td>
<td>do html_a_end</td>
</tr>
<tr>
<td></td>
<td>print 'At the top of document' ()</td>
</tr>
<tr>
<td></td>
<td>do html_br(20, '')</td>
</tr>
<tr>
<td></td>
<td>do html_a('HREF=otherdoc.html')</td>
</tr>
<tr>
<td></td>
<td>print 'Goto other document' ()</td>
</tr>
<tr>
<td></td>
<td>do html_a_end</td>
</tr>
<tr>
<td></td>
<td>do html_p('')</td>
</tr>
<tr>
<td></td>
<td>do html_a('HREF=#TOP')</td>
</tr>
<tr>
<td></td>
<td>print 'Goto top of document' ()</td>
</tr>
<tr>
<td></td>
<td>do html_a_end</td>
</tr>
</tbody>
</table>

| html_a_end  | End of a hypertext link.                                                   |
|            | **Syntax:** html_a_end                                                     |

### HTML List Procedures

**Table 71** HTML List Procedures

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html_dd</td>
<td>Start of a definition in a definition list.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong> html_dd(string attributes)</td>
</tr>
<tr>
<td></td>
<td>● attributes = HTML attributes inside <code>&lt;DD&gt;</code></td>
</tr>
<tr>
<td>Procedure</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **html_dd_end** | End of a definition in a definition list. Typically implied; however, can be defined for completeness.  
*Syntax:* `html_dd_end`  |
| **html_dir** | Start of a directory list.  
*Syntax:* `html_dir(string attributes)`  
● `attributes` = HTML attributes inside `<DIR>`  
*Example:*  
```  
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  
```  |
| **html_dir_end** | End of a directory list.  
*Syntax:* `html_dir_end`  |
| **html_dl** | Start of a definition list. Terms display above and to the left of definitions. `html_dt` displays terms. `html_dd` displays definitions.  
*Syntax:* `html_dl(string attributes)`  
● `attributes` = HTML attributes inside `<DL>`  
*Example:* Definition list with two terms and definitions:  
```  
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  
```  |
| **html_dl_end** | End of a definition list.  
*Syntax:* `html_dl_end`  |
| **html_dt** | Start of a term in a definition list.  
*Syntax:* `html_dt(string attributes)`  
● `attributes` = HTML attributes inside `<DT>`  |
| **html_dt_end** | End of a term in a definition list.  
*Syntax:* `html_dt_end`  |
| **html_li** | Start of a list item.  
*Syntax:* `html_li(string attributes)`  |
<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html_li_end</td>
<td>End of a list item. Typically implied; however, can be defined for completeness.</td>
</tr>
<tr>
<td>html_menu</td>
<td>Start of a menu. html_li identifies menu items.</td>
</tr>
<tr>
<td>html_ol</td>
<td>Start of an ordered list. List items typically display indented to the right with a number to the left. html_li identifies list items.</td>
</tr>
<tr>
<td>html_ul</td>
<td>Start of an unordered list. List items typically display indented to the right with a bullet to the left. html_li identifies list items.</td>
</tr>
</tbody>
</table>

**Syntax:**

- `html_menu(string attributes)
- `html_ol(string attributes)
- `html_ul(string attributes)"
### HTML Table Procedures

**Table 72  HTML Table Procedures**

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
</table>
| `html_caption` | Start of a table caption.  
**Syntax:** `html_caption(string attributes)`  
- `attributes` = 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(string attributes)`  
- `attributes` = 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.  
!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 |
<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>do html_td_end</td>
<td>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</td>
</tr>
<tr>
<td>html_table_end</td>
<td>End of a table.</td>
</tr>
<tr>
<td>Syntax:</td>
<td>html_table_end</td>
</tr>
<tr>
<td>html_td</td>
<td>Start of a new column in a table row. The text that follows displays within the column.</td>
</tr>
<tr>
<td>Syntax:</td>
<td>html_td(string attributes) * attributes = HTML attributes inside &lt;TD&gt;</td>
</tr>
<tr>
<td>html_td_end</td>
<td>End of a column in a table. Typically implied; however, can be defined for completeness.</td>
</tr>
<tr>
<td>Syntax:</td>
<td>html_td_end</td>
</tr>
<tr>
<td>html_th</td>
<td>Start of a new column header in a table row. The text that follows displays as the column header.</td>
</tr>
<tr>
<td>Syntax:</td>
<td>html_th(string attributes) * attributes = HTML attributes inside &lt;TH&gt;</td>
</tr>
<tr>
<td>html_th_end</td>
<td>End of a column header in a table. Typically implied; however, can be defined for completeness.</td>
</tr>
<tr>
<td>Syntax:</td>
<td>html_th_end</td>
</tr>
<tr>
<td>html_tr</td>
<td>Start of a new table row.</td>
</tr>
<tr>
<td>Syntax:</td>
<td>html_tr(string attributes) * attributes = HTML attributes inside &lt;TR&gt;</td>
</tr>
<tr>
<td>html_tr_end</td>
<td>End of a table row. Typically implied; however, can be defined for completeness.</td>
</tr>
<tr>
<td>Syntax:</td>
<td>html_tr_end</td>
</tr>
</tbody>
</table>
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 SQR.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>
<thead>
<tr>
<th>Encoding Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseUnicodeInternal=TRUE</td>
<td>TRUE uses Unicode internally.</td>
</tr>
<tr>
<td>AutoDetectUnicodeFiles=TRUE</td>
<td>TRUE automatically detects UCS-2 encoded files.</td>
</tr>
</tbody>
</table>
**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>
<thead>
<tr>
<th>Encoding Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>Default encoding.</td>
</tr>
<tr>
<td>Encoding-Console</td>
<td>Encoding used for console input and output.</td>
</tr>
<tr>
<td>Encoding-Database</td>
<td>Encoding used for interfacing with the database.</td>
</tr>
<tr>
<td>Encoding-File-Input</td>
<td>Default encoding used for “OPEN” for-reading files and argument files.</td>
</tr>
<tr>
<td>Encoding-File-Output</td>
<td>Default encoding used for “OPEN” for-writing files.</td>
</tr>
<tr>
<td>Encoding-Report-Output</td>
<td>Default encoding used for report output, such as, SPF, LIS, HTM, etc.</td>
</tr>
<tr>
<td>Encoding-SQR-Source</td>
<td>Encoding used for Production Reporting source and include files.</td>
</tr>
</tbody>
</table>

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.
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 a 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

<table>
<thead>
<tr>
<th>Encoding-File-Input</th>
<th>Encoding-Database</th>
<th>Encoding-File-Output</th>
<th>Encoding-Report-Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>ASCII</td>
<td>ASCII</td>
<td>ASCII</td>
</tr>
</tbody>
</table>

318  Encoding in Production Reporting
<table>
<thead>
<tr>
<th>Encoding-File-Input</th>
<th>Encoding-Database</th>
<th>Encoding-File-Output</th>
<th>Encoding-Report-Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>ASCII</td>
<td>ASCII</td>
<td>Shift-JIS</td>
</tr>
<tr>
<td>ASCII</td>
<td>ASCII</td>
<td>ASCII</td>
<td>JEUC</td>
</tr>
<tr>
<td>ASCII</td>
<td>ASCII</td>
<td>ASCII</td>
<td>UTF-8/UCS-2</td>
</tr>
<tr>
<td>ASCII</td>
<td>Shift-JIS</td>
<td>ASCII</td>
<td>ASCII</td>
</tr>
<tr>
<td>ASCII</td>
<td>Shift-JIS</td>
<td>ASCII</td>
<td>Shift-JIS</td>
</tr>
<tr>
<td>ASCII</td>
<td>JEUC</td>
<td>ASCII</td>
<td>ASCII</td>
</tr>
<tr>
<td>ASCII</td>
<td>JEUC</td>
<td>ASCII</td>
<td>JEUC</td>
</tr>
<tr>
<td>ASCII</td>
<td>UTF-8/UCS-2</td>
<td>ASCII</td>
<td>ASCII</td>
</tr>
<tr>
<td>ASCII</td>
<td>UTF-8/UCS-2</td>
<td>ASCII</td>
<td>UTF-8/UCS-2</td>
</tr>
<tr>
<td>ASCII</td>
<td>Shift-JIS</td>
<td>Shift-JIS</td>
<td>Shift-JIS</td>
</tr>
<tr>
<td>ASCII</td>
<td>JEUC</td>
<td>JEUC</td>
<td>JEUC</td>
</tr>
<tr>
<td>ASCII</td>
<td>UTF-8/UCS-2</td>
<td>UTF-8/UCS-2</td>
<td>UTF-8/UCS-2</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>EBCDIC</td>
<td>EBCDIC</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>EBCDIC</td>
<td>EBCDIC</td>
<td>EBCDIC290 or 1027</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>EBCDIK290 or 1027</td>
<td>EBCDIC</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>EBCDIK290 or 1027</td>
<td>EBCDIC</td>
<td>EBCDIK290 or 1027</td>
</tr>
<tr>
<td>EBCDIK290 or 1027</td>
<td>EBCDIK290 or 1027</td>
<td>EBCDIK290 or 1027</td>
<td>EBCDIK290 or 1027</td>
</tr>
<tr>
<td>EBCDIK290 or 1027</td>
<td>EBCDIK290 or 1027</td>
<td>EBCDIK290 or 1027</td>
<td>EBCDIK290 or 1027</td>
</tr>
<tr>
<td>EBCDIK290 or 1027</td>
<td>Shift-JIS</td>
<td>ASCII</td>
<td>Shift-JIS</td>
</tr>
<tr>
<td>EBCDIK290 or 1027</td>
<td>Shift-JIS</td>
<td>Shift-JIS</td>
<td>Shift-JIS</td>
</tr>
<tr>
<td>EBCDIK290 or 1027</td>
<td>JEUC</td>
<td>ASCII</td>
<td>JEUC</td>
</tr>
<tr>
<td>EBCDIK290 or 1027</td>
<td>JEUC</td>
<td>JEUC</td>
<td>JEUC</td>
</tr>
<tr>
<td>UTF-8/UCS-2</td>
<td>UTF-8/UCS-2</td>
<td>ASCII</td>
<td>UTF-8/UCS-2</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Encoding</th>
<th>Character Set</th>
<th>Also Known As</th>
<th>Vendor / Standard Body</th>
<th>Acceptable Production Reporting Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP10004</td>
<td>Arabic</td>
<td>Macintosh Arabic</td>
<td>Microsoft &amp; IBM</td>
<td>CP10004</td>
</tr>
<tr>
<td>CP1256</td>
<td>Arabic</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP1256</td>
</tr>
<tr>
<td>CP20420</td>
<td>Arabic</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20420</td>
</tr>
<tr>
<td>CP28596</td>
<td>Arabic</td>
<td>Arabic Alphabet (ISO)</td>
<td>Microsoft &amp; IBM</td>
<td>CP28596</td>
</tr>
<tr>
<td>CP720</td>
<td>Arabic</td>
<td>Transparent ASMO</td>
<td>Microsoft &amp; IBM</td>
<td>CP720</td>
</tr>
<tr>
<td>CP864</td>
<td>Arabic</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP864</td>
</tr>
<tr>
<td>ISO 8859-6</td>
<td>Arabic</td>
<td>ISO-Latin Arabic</td>
<td>International or National Standard</td>
<td>Arabic</td>
</tr>
<tr>
<td>CP708</td>
<td>Arabic</td>
<td>ASMO708</td>
<td>Microsoft &amp; IBM</td>
<td>CP708</td>
</tr>
<tr>
<td>CP1257</td>
<td>Baltic</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP1257</td>
</tr>
<tr>
<td>CP28594</td>
<td>Baltic</td>
<td>Baltic Alphabet (ISO)</td>
<td>Microsoft &amp; IBM</td>
<td>CP28594</td>
</tr>
<tr>
<td>CP775</td>
<td>Baltic</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP775</td>
</tr>
<tr>
<td>ISO 8859-4</td>
<td>Baltic</td>
<td>Latin4</td>
<td>International or National Standard</td>
<td>Latin4, ISO-8859-4</td>
</tr>
<tr>
<td>ISO 8859-13</td>
<td>Baltic</td>
<td>Latin7</td>
<td>International or National Standard</td>
<td>Latin7, ISO-8859-13</td>
</tr>
<tr>
<td>ISO 8859-14</td>
<td>Celtic</td>
<td>Latin8</td>
<td>International or National Standard</td>
<td>ISO-8859-14</td>
</tr>
<tr>
<td>GB18030</td>
<td>Chinese</td>
<td></td>
<td>International or National Standard</td>
<td>GB18030</td>
</tr>
<tr>
<td>HKSCS</td>
<td>Chinese</td>
<td>Big5-HKSCS</td>
<td>International or National Standard</td>
<td>Big5-HKSCS</td>
</tr>
<tr>
<td>CP936</td>
<td>Chinese, Simplified</td>
<td>GBK</td>
<td>Microsoft &amp; IBM</td>
<td>CP936</td>
</tr>
<tr>
<td>GB2312</td>
<td>Chinese, Simplified</td>
<td>EUC-CN, EUC-SC</td>
<td>International or National Standard</td>
<td>GB2312, EUC-CN</td>
</tr>
<tr>
<td>Encoding</td>
<td>Character Set</td>
<td>Also Known As</td>
<td>Vendor / Standard Body</td>
<td>Acceptable Production Reporting Names</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>HZ-GB-2312</td>
<td>Chinese, Simplified</td>
<td>HZ-GB-2312</td>
<td>International or National Standard</td>
<td>HZ</td>
</tr>
<tr>
<td>Big5</td>
<td>Chinese, Traditional</td>
<td></td>
<td>International or National Standard</td>
<td>Big5</td>
</tr>
<tr>
<td>BIG5+</td>
<td>Chinese, Traditional</td>
<td></td>
<td>International or National Standard</td>
<td>BIG5+</td>
</tr>
<tr>
<td>CNS-11643-1986</td>
<td>Chinese, Traditional</td>
<td>EUC-TW</td>
<td>International or National Standard</td>
<td>CNS-11643-1986</td>
</tr>
<tr>
<td>GB12345</td>
<td>Chinese, Traditional</td>
<td></td>
<td>International or National Standard</td>
<td>GB12345</td>
</tr>
<tr>
<td>CP10002</td>
<td>Chinese, Traditional</td>
<td>Macintosh Traditional Chinese</td>
<td>Microsoft &amp; IBM</td>
<td>CP10002</td>
</tr>
<tr>
<td>CP950</td>
<td>Chinese, Traditional</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP950</td>
</tr>
<tr>
<td>CP10007</td>
<td>Cyrillic</td>
<td>Macintosh Cyrillic</td>
<td>Microsoft &amp; IBM</td>
<td>CP10007</td>
</tr>
<tr>
<td>CP1251</td>
<td>Cyrillic</td>
<td>MS Windows Cyrillic (Slavic)</td>
<td>Microsoft &amp; IBM</td>
<td>CP1251</td>
</tr>
<tr>
<td>CP20866</td>
<td>Cyrillic</td>
<td>Cyrillic Alphabet, KOI8-R</td>
<td>Microsoft &amp; IBM</td>
<td>CP20866</td>
</tr>
<tr>
<td>CP20880</td>
<td>Cyrillic</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20880</td>
</tr>
<tr>
<td>CP21025</td>
<td>Cyrillic</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP21025</td>
</tr>
<tr>
<td>CP21866</td>
<td>Cyrillic</td>
<td>Ukrainian KOI8-RU</td>
<td>Microsoft &amp; IBM</td>
<td>CP21866</td>
</tr>
<tr>
<td>CP28595</td>
<td>Cyrillic</td>
<td>Cyrillic Alphabet (ISO)</td>
<td>Microsoft &amp; IBM</td>
<td>CP28595</td>
</tr>
<tr>
<td>CP855</td>
<td>Cyrillic</td>
<td>IBM Cyrillic</td>
<td>Microsoft &amp; IBM</td>
<td>CP855</td>
</tr>
<tr>
<td>CP866</td>
<td>Cyrillic</td>
<td>MS DOS Russian</td>
<td>Microsoft &amp; IBM</td>
<td>CP866</td>
</tr>
<tr>
<td>ISO 8859-5</td>
<td>Cyrillic</td>
<td>ISOLatinCyrillic</td>
<td>International or National Standard</td>
<td>ISOLatinCyrillic</td>
</tr>
<tr>
<td>CP10006</td>
<td>Greek</td>
<td>Macintosh Greek 1</td>
<td>Microsoft &amp; IBM</td>
<td>CP10006</td>
</tr>
<tr>
<td>CP1253</td>
<td>Greek</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP1253</td>
</tr>
<tr>
<td>CP20423</td>
<td>Greek</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20423</td>
</tr>
<tr>
<td>Encoding</td>
<td>Character Set</td>
<td>Also Known As</td>
<td>Vendor / Standard Body</td>
<td>Acceptable Production Reporting Names</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>---------------------------------------------------</td>
<td>------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>CP28597</td>
<td>Greek</td>
<td>Greek Alphabet (ISO)</td>
<td>Microsoft &amp; IBM</td>
<td>CP28597</td>
</tr>
<tr>
<td>CP737</td>
<td>Greek</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP737</td>
</tr>
<tr>
<td>CP869</td>
<td>Greek</td>
<td>IBM Modern Greek</td>
<td>Microsoft &amp; IBM</td>
<td>CP869</td>
</tr>
<tr>
<td>ISO 8859-7</td>
<td>Greek</td>
<td>ISOLatinGreek</td>
<td>International or National Standard</td>
<td>Greek</td>
</tr>
<tr>
<td>CP10010</td>
<td>Gurmukhi</td>
<td>Macintosh Gurmukhi</td>
<td>Microsoft &amp; IBM</td>
<td>CP10010</td>
</tr>
<tr>
<td>CP10005</td>
<td>Hebrew</td>
<td>Macintosh Hebrew</td>
<td>Microsoft &amp; IBM</td>
<td>CP10005</td>
</tr>
<tr>
<td>CP1255</td>
<td>Hebrew</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP1255</td>
</tr>
<tr>
<td>CP28598</td>
<td>Hebrew</td>
<td>Hebrew Alphabet (ISO)</td>
<td>Microsoft &amp; IBM</td>
<td>CP28598</td>
</tr>
<tr>
<td>CP38598</td>
<td>Hebrew</td>
<td>ASCII + Hebrew and private use characters</td>
<td>Microsoft &amp; IBM</td>
<td>CP38598</td>
</tr>
<tr>
<td>CP862</td>
<td>Hebrew</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP862</td>
</tr>
<tr>
<td>ISO 8859-8</td>
<td>Hebrew</td>
<td>ISOLatinHebrew</td>
<td>International or National Standard</td>
<td>Hebrew</td>
</tr>
<tr>
<td>CP10079</td>
<td>Icelandic</td>
<td>Macintosh Icelandic</td>
<td>Microsoft &amp; IBM</td>
<td>CP10079</td>
</tr>
<tr>
<td>CP861</td>
<td>Icelandic</td>
<td>MS DOS Icelandic</td>
<td>Microsoft &amp; IBM</td>
<td>CP861</td>
</tr>
<tr>
<td>CCSID 1027</td>
<td>Japanese</td>
<td>EBCDIK</td>
<td>Microsoft &amp; IBM</td>
<td>CCSID-1027, EBCDIK1027</td>
</tr>
<tr>
<td>CCSID 290</td>
<td>Japanese</td>
<td>EBCDIK</td>
<td>Microsoft &amp; IBM</td>
<td>CCSID-290, EBCDIK290</td>
</tr>
<tr>
<td>CCSID 942</td>
<td>Japanese</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CCSID-942</td>
</tr>
<tr>
<td>CP10001</td>
<td>Japanese</td>
<td>Macintosh Japanese</td>
<td>Microsoft &amp; IBM</td>
<td>CP10001</td>
</tr>
<tr>
<td>CP20290</td>
<td>Japanese</td>
<td>(full/half width Latin &amp; halfwidth katakana)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20290</td>
</tr>
<tr>
<td>CP21027</td>
<td>Japanese</td>
<td>(halfwidth Latin, halfwidth katakana&amp;private use)</td>
<td>Microsoft &amp; IBM</td>
<td>CP21027</td>
</tr>
<tr>
<td>CP932</td>
<td>Japanese</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP932</td>
</tr>
<tr>
<td>EUC-JP</td>
<td>Japanese</td>
<td></td>
<td>UNIX</td>
<td>EUC-J, JEUC</td>
</tr>
<tr>
<td>JIS_X_0201</td>
<td>Japanese</td>
<td>HalfWidthKatakana</td>
<td>International or National Standard</td>
<td>JIS_X_0201, IBM897</td>
</tr>
<tr>
<td>Encoding</td>
<td>Character Set</td>
<td>Also Known As</td>
<td>Vendor / Standard Body</td>
<td>Acceptable Production Reporting Names</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>---------------</td>
<td>--------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>JIS_X_0208</td>
<td>Japanese</td>
<td></td>
<td>International or National Standard</td>
<td>JIS_X_0208</td>
</tr>
<tr>
<td>Shift-JIS</td>
<td>Japanese</td>
<td>MS_Kanji</td>
<td>Microsoft &amp; IBM</td>
<td>Shift-JIS, SJIS</td>
</tr>
<tr>
<td>CP10003</td>
<td>Korean</td>
<td>Macintosh Korean</td>
<td>Microsoft &amp; IBM</td>
<td>CP10003</td>
</tr>
<tr>
<td>CP1361</td>
<td>Korean</td>
<td>Korean Johab (based on KSC 5861-1992)</td>
<td>Microsoft &amp; IBM</td>
<td>CP1361</td>
</tr>
<tr>
<td>CP949</td>
<td>Korean</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP949</td>
</tr>
<tr>
<td>Johab</td>
<td>Korean</td>
<td></td>
<td>International or National Standard</td>
<td>Johab</td>
</tr>
<tr>
<td>CP10000</td>
<td>Latin</td>
<td>Macintosh Roman</td>
<td>Microsoft &amp; IBM</td>
<td>CP10000</td>
</tr>
<tr>
<td>CP10029</td>
<td>Latin</td>
<td>Macintosh Latin2</td>
<td>Microsoft &amp; IBM</td>
<td>CP10029</td>
</tr>
<tr>
<td>CP10082</td>
<td>Latin</td>
<td>(with mathematical symbols)</td>
<td>Microsoft &amp; IBM</td>
<td>CP10082</td>
</tr>
<tr>
<td>CCSID1047</td>
<td>Latin</td>
<td>EBCDIC (for IBM Open Systems platform)</td>
<td>Microsoft &amp; IBM</td>
<td>CCSID1047</td>
</tr>
<tr>
<td>CP20261</td>
<td>Latin</td>
<td>(with private use characters)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20261</td>
</tr>
<tr>
<td>CP20269</td>
<td>Latin</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP20269</td>
</tr>
<tr>
<td>CP20273</td>
<td>Latin</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20273</td>
</tr>
<tr>
<td>CP20277</td>
<td>Latin</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20277</td>
</tr>
<tr>
<td>CP20278</td>
<td>Latin</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20278</td>
</tr>
<tr>
<td>CP20280</td>
<td>Latin</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20280</td>
</tr>
<tr>
<td>CP20284</td>
<td>Latin</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20284</td>
</tr>
<tr>
<td>CP20285</td>
<td>Latin</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20285</td>
</tr>
<tr>
<td>CP20297</td>
<td>Latin</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20297</td>
</tr>
<tr>
<td>Encoding</td>
<td>Character Set</td>
<td>Also Known As</td>
<td>Vendor / Standard Body</td>
<td>Acceptable Production Reporting Names</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------</td>
<td>---------------------------------------------------</td>
<td>------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>CP20833</td>
<td>Latin</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20833</td>
</tr>
<tr>
<td>CP20871</td>
<td>Latin</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20871</td>
</tr>
<tr>
<td>CP28591</td>
<td>Latin</td>
<td>ASCII + Latin accented vowels</td>
<td>Microsoft &amp; IBM</td>
<td>CP28591</td>
</tr>
<tr>
<td>CP28593</td>
<td>Latin</td>
<td>Latin 3 Alphabet (ISO)</td>
<td>Microsoft &amp; IBM</td>
<td>CP28593</td>
</tr>
<tr>
<td>CP850</td>
<td>Latin</td>
<td>MS DOS Multilingual, MS-DOS Latin1</td>
<td>Microsoft &amp; IBM</td>
<td>CP850</td>
</tr>
<tr>
<td>CP870</td>
<td>Latin</td>
<td>(with fullwidth punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP870</td>
</tr>
<tr>
<td>HP-ROMAN8</td>
<td>Latin</td>
<td>csHPRoman8, r8, roman8</td>
<td>HP</td>
<td>HP-ROMAN8</td>
</tr>
<tr>
<td>ISO 8859-1</td>
<td>Latin</td>
<td>Latin1</td>
<td>International or National Standard</td>
<td>Latin1, ISO-8859-1</td>
</tr>
<tr>
<td>ISO 8859-15</td>
<td>Latin</td>
<td>Latin1 + Euro symbol &amp; accented characters</td>
<td>International or National Standard</td>
<td>ISO-8859-15</td>
</tr>
<tr>
<td>ISO 8859-2</td>
<td>Latin</td>
<td>Latin2</td>
<td>International or National Standard</td>
<td>Latin2, ISO-8859-2</td>
</tr>
<tr>
<td>UTF8-EBCDIC</td>
<td>Latin</td>
<td></td>
<td>Unicode</td>
<td>UTF8-EBCDIC</td>
</tr>
<tr>
<td>CP863</td>
<td>Latin, Canadian French</td>
<td>MS DOS Canadian French</td>
<td>Microsoft &amp; IBM</td>
<td>CP863</td>
</tr>
<tr>
<td>CP28592</td>
<td>Latin, Central European</td>
<td>Central European Alphabet (ISO)</td>
<td>Microsoft &amp; IBM</td>
<td>CP28592</td>
</tr>
<tr>
<td>CP1250</td>
<td>Latin, Eastern Europe</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP1250</td>
</tr>
<tr>
<td>CP20905</td>
<td>Latin, Esperanto</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20905</td>
</tr>
<tr>
<td>CP860</td>
<td>Latin, Portuguese</td>
<td>MS DOS Portuguese</td>
<td>Microsoft &amp; IBM</td>
<td>CP860</td>
</tr>
<tr>
<td>ISO 8859-3</td>
<td>Latin, Southeast European</td>
<td>Latin3</td>
<td>International or National Standard</td>
<td>Latin3, ISO-8859-3</td>
</tr>
<tr>
<td>ASCII</td>
<td>Latin, US English</td>
<td></td>
<td>International or National Standard</td>
<td>ASCII, ANSI</td>
</tr>
<tr>
<td>CP037</td>
<td>Latin, US English</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP037</td>
</tr>
<tr>
<td>CP1026</td>
<td>Latin, US English</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP1026</td>
</tr>
<tr>
<td>CP1252</td>
<td>Latin, US English</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP1252</td>
</tr>
<tr>
<td>CP20105</td>
<td>Latin, US English</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP20105</td>
</tr>
<tr>
<td>Encoding</td>
<td>Character Set</td>
<td>Also Known As</td>
<td>Vendor / Standard Body</td>
<td>Acceptable Production Reporting Names</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>----------------------------</td>
<td>------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>CP437</td>
<td>Latin, US English</td>
<td>MS-DOS Latin US</td>
<td>Microsoft &amp; IBM</td>
<td>CP437</td>
</tr>
<tr>
<td>CP500</td>
<td>Latin, US English</td>
<td>EBCDIC</td>
<td>Microsoft &amp; IBM</td>
<td>CP500</td>
</tr>
<tr>
<td>CP875</td>
<td>Latin, US English</td>
<td>EBCDIC</td>
<td>Microsoft &amp; IBM</td>
<td>CP875</td>
</tr>
<tr>
<td>CP10017</td>
<td>Malayan</td>
<td>Macintosh Malayan</td>
<td>Microsoft &amp; IBM</td>
<td>CP10017</td>
</tr>
<tr>
<td>CP865</td>
<td>Nordic</td>
<td>MS DOS Nordic</td>
<td>Microsoft &amp; IBM</td>
<td>CP865</td>
</tr>
<tr>
<td>ISO 8859-10</td>
<td>Nordic</td>
<td>Latin6</td>
<td>International or National Standard</td>
<td>Latin6, ISO-8859-10</td>
</tr>
<tr>
<td>CP852</td>
<td>Slavic</td>
<td>MS DOS Slavic</td>
<td>Microsoft &amp; IBM</td>
<td>CP852</td>
</tr>
<tr>
<td>Adobe-Symbol-Encoding</td>
<td>Symbol</td>
<td>(used in PS printers)</td>
<td>Adobe</td>
<td>Adobe-Symbol-Encoding</td>
</tr>
<tr>
<td>CP10008</td>
<td>Symbol</td>
<td>Macintosh RSymbol (Right-left symbol)</td>
<td>Microsoft &amp; IBM</td>
<td>CP10008</td>
</tr>
<tr>
<td>CP20838</td>
<td>Thai</td>
<td>(with fullwidth Latin &amp; punctuation)</td>
<td>Microsoft &amp; IBM</td>
<td>CP20838</td>
</tr>
<tr>
<td>CP874</td>
<td>Thai</td>
<td>IBMThai</td>
<td>Microsoft &amp; IBM</td>
<td>CP874</td>
</tr>
<tr>
<td>ISO 8859-11 (draft)</td>
<td>Thai</td>
<td>ISOLatinThai</td>
<td>International or National Standard</td>
<td>Thai</td>
</tr>
<tr>
<td>CP10081</td>
<td>Turkish</td>
<td>Macintosh Turkish</td>
<td>Microsoft &amp; IBM</td>
<td>CP10081</td>
</tr>
<tr>
<td>CP1254</td>
<td>Turkish</td>
<td>Macintosh Turkish</td>
<td>Microsoft &amp; IBM</td>
<td>CP1254</td>
</tr>
<tr>
<td>CP28599</td>
<td>Turkish</td>
<td>Turkish (ISO)</td>
<td>Microsoft &amp; IBM</td>
<td>CP28599</td>
</tr>
<tr>
<td>CP857</td>
<td>Turkish</td>
<td>IBM Turkish</td>
<td>Microsoft &amp; IBM</td>
<td>CP857</td>
</tr>
<tr>
<td>ISO 8859-9</td>
<td>Turkish</td>
<td>Latin5</td>
<td>International or National Standard</td>
<td>Latin5, ISO-8859-9</td>
</tr>
<tr>
<td>BMP</td>
<td>Unicode</td>
<td></td>
<td>Unicode</td>
<td>BMP</td>
</tr>
<tr>
<td>Java</td>
<td>Unicode</td>
<td></td>
<td>Sun</td>
<td>Java</td>
</tr>
<tr>
<td>UCS2</td>
<td>Unicode</td>
<td>ISO-10646-UCS2, UTF16</td>
<td>Unicode</td>
<td>UCS2</td>
</tr>
<tr>
<td>Unicode Big-endian</td>
<td>Unicode</td>
<td></td>
<td>Unicode</td>
<td>big-endian</td>
</tr>
<tr>
<td>Unicode Little-endian</td>
<td>Unicode</td>
<td></td>
<td>Unicode</td>
<td>little-endian</td>
</tr>
<tr>
<td>UTF7</td>
<td>Unicode</td>
<td></td>
<td>Unicode</td>
<td>UTF7</td>
</tr>
<tr>
<td>Encoding</td>
<td>Character Set</td>
<td>Also Known As</td>
<td>Vendor / Standard Body</td>
<td>Acceptable Production Reporting Names</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>---------------</td>
<td>------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>UTF8</td>
<td>Unicode</td>
<td></td>
<td>Unicode</td>
<td>UTF8</td>
</tr>
<tr>
<td>UTF8-EBCDIC</td>
<td>Unicode</td>
<td></td>
<td>Unicode</td>
<td>UTF8-EBCDIC</td>
</tr>
<tr>
<td>CP1258</td>
<td>Vietnamese</td>
<td></td>
<td>Microsoft &amp; IBM</td>
<td>CP1258</td>
</tr>
</tbody>
</table>
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 \textit{SQRDIR} is defined in the initialization file, Production Reporting produces an error message if it cannot find the file.

\section*{For All Other Platforms}

Production Reporting looks for the initialization file in the following order:

1. The file name specified by \texttt{-ZIF(file)}.
2. The current working directory.
3. The directory specified with \texttt{SQRDIR}.

Since the required environment variable \texttt{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:

\begin{verbatim}
; 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
.
.
\end{verbatim}

\section*{[Default-Settings] Section}

[Default-Settings] defines Production Reporting default actions.

\begin{table}[h!]
\centering
\caption{Entries in [Default-Settings]}
\begin{tabular}{|l|l|l|}
\hline
Entry & Value & Description \\
\hline
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, \\
\hline
\end{tabular}
\end{table}
Production Reporting allows characters to equal either character or date columns. When a date column is “type cast” as a character, Production Reporting creates the string according to the following rules:

- 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.

In the following example, AllowDateAsChar=True. This allows $Col1 to be either date or text.

```sql
Begin-Select
  [$Col1] &col1=Text
  [$Col2] &col2=Date
  [$Col3] &col3=Number
from MyTable
End-Select
```

<table>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoDetectUnicodeFiles</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>CSVSeparator</td>
<td>Comma</td>
<td>Semicolon</td>
</tr>
<tr>
<td>DEFAULT-NUMERIC</td>
<td>INTEGER</td>
<td>FLOAT</td>
</tr>
<tr>
<td>ExpirationWarningMessage</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>Entry</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FormatforRowsetXML</td>
<td>BIP</td>
<td>SQR</td>
</tr>
<tr>
<td>ImageCompression</td>
<td>0 - 9</td>
<td>Defines the compression level when PRINT-IMAGE references a BINARY variable.</td>
</tr>
<tr>
<td>LOCALE</td>
<td>Name of a locale defined in SQR.INI or the name SYSTEM.</td>
<td>Defines the initial locale that Production Reporting loads when the program starts to execute. The value of SYSTEM is used to reference the default locale. (See “ALTER-LOCALE” on page 41.)</td>
</tr>
<tr>
<td>NewGraphics</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td></td>
<td>Default = FALSE</td>
<td>Set NewGraphics=TRUE to use the following features:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● COLOR-PALETTE (See “Specifying Chart Data Series Colors” in Volume 1 of the Hyperion SQR Production Reporting Developer’s Guide.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● ITEM-COLOR (See “Specifying Chart Item Colors” in Volume 1 of the Hyperion SQR Production Reporting Developer’s Guide.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● ITEM-SIZE (See “DECLARE-CHART” on page 100 and “PRINT-CHART” on page 263.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Y-AXIS-MASK and Y2-AXIS-MASK (See “DECLARE-CHART” on page 100 and “PRINT-CHART” on page 263.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Y2 Syntax (See “DECLARE-CHART” on page 100 and “PRINT-CHART” on page 263.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Combination Charts (See “Creating Combination Charts” in Volume 1 of the Hyperion SQR Production Reporting Developer’s Guide.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Bubble Charts (See “Creating Bubble Charts” in Volume 1 of the Hyperion SQR Production Reporting Developer’s Guide.)</td>
</tr>
<tr>
<td>ODBCExecuteRetry</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td></td>
<td>Default = FALSE</td>
<td>FALSE retries EXECUTE and returns an error from the database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE does not attempt to retry.</td>
</tr>
<tr>
<td>Entry</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **OracleWeakCursor** | STOP | WARN | SKIP | Default = WARN | **STOP** displays an error or warning message and terminates the Production Reporting procedure.  
**WARN** displays an error or warning message, and the Production Reporting procedure continues.  
**SKIP** does *not* display an error or warning message, and the Production Reporting procedure continues. |
| **OUTPUT-FILE-MODE** | LONG | SHORT | Default = LONG | 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) **DECLARE-TOC** and **-Burst** force **Output-File-Mode = LONG**.  
The following represent the file formats for UNIX, DOS, and Windows.  
**SQR and SQRT:** [Program] is the name of the SQR/SQT file without the extension  
For **Output-File-Mode = SHORT**, SQR-generated filenames are limited to a DOS 8.3 format  
- 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  
  - bb ranges from 00 to 99 and represents the report number.  
  - nn ranges from 01 to 99 and represents the report number.  
  - xx ranges from 00 to ZZ and represents the graphic number.  
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.  
- Output file = [Output].LIS  
- SPF file = [Output].SPF  
- PDF file = [Output].PDF |
<table>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIF file</td>
<td>{Output}.zz.SPF</td>
<td>● GIF file = {Output}.zz.SPF</td>
</tr>
<tr>
<td>HTM files</td>
<td>(Output).HTM, (Output)_bb.HTM, (Output)_frm.HTM, (Output)_toc.HTM, (Output)_nav.htm</td>
<td>bb ranges from 01 to ZHJOZI and represents the bursted page group number in radix 36.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nn ranges from 01 to 99 and represents the report number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>zz ranges from 01 to ZHJOZI and represents the graphic number in radix 36.</td>
</tr>
<tr>
<td>SQRP: (Filename)</td>
<td>is the name of the SPF file without the extension</td>
<td></td>
</tr>
<tr>
<td>For Output-File-Mode = SHORT, SQR-</td>
<td>generated filenames are limited to a DOS 8.3 format.</td>
<td></td>
</tr>
<tr>
<td>generated filenames</td>
<td></td>
<td>● Output file = (Filename).LIS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● GIF file = (Filename).Gxx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● PDF file = (Filename).PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● HTM file = .HTM and (Filename).H00</td>
</tr>
<tr>
<td>xx ranges from 00 to ZZ and represents</td>
<td></td>
<td>For Output-File-Mode = LONG, SQR-generated filenames are not limited to a DOS 8.3 format.</td>
</tr>
<tr>
<td>the graphic number</td>
<td></td>
<td>● Output file = (Filename).LIS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● PDF file = (Filename).PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● GIF file = (Filename).zz.SPF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● HTM files = (Filename).HTM, (Filename)_bb.HTM, (Filename)_frm.HTM, (Filename)_toc.HTM, (Filename)_nav.htm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bb ranges from 01 to ZHJOZI and represents the bursted page group number in radix 36.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>zz ranges from 01 to ZHJOZI and represents the graphic number in radix 36.</td>
</tr>
<tr>
<td>OutputFormFeedWithDashD</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>Default</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>OutputTwoDigitYearWarningMsg</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>Default</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>Entry</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>SUBSTITUTION-CHARACTER=XX EncodingName1 [,xx EncodingName2...]</td>
<td>XX is the hexadecimal representation of the substitution variable</td>
<td>Allows substitution characters to be defined on a character set by character set basis. See “Substitution-Character Key” on page 316 for more information.</td>
</tr>
<tr>
<td>TreatBinaryColumnAsText</td>
<td>TRUE</td>
<td>Defines whether to treat BINARY columns as TEXT columns.</td>
</tr>
<tr>
<td>UseUnicodeInternal</td>
<td>TRUE</td>
<td>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.</td>
</tr>
<tr>
<td>UseY2kCenturyAlgorithm</td>
<td>TRUE</td>
<td>When set to TRUE, Production Reporting treats the YY date edit mask as though it is an RR edit mask.</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| SQR_USEJVM       | TRUE | FALSE | 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

<table>
<thead>
<tr>
<th>DDO Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQR_DDO_JRE_CLASS</td>
<td>Classpath for DDO drivers and support files.</td>
</tr>
<tr>
<td>SQR_DDO_JRE_CLASSn</td>
<td>(Optional) Additional entries to the classpath.</td>
</tr>
<tr>
<td>SQR_DDO_JRE_PATH</td>
<td>Classpath information for the local JRE.</td>
</tr>
<tr>
<td>SQR_DDO_JRE_INIT_HEAP</td>
<td>Initial Java heap size.</td>
</tr>
<tr>
<td>SQR_DDO_JRE_MAX_HEAP</td>
<td>Maximum Java heap size.</td>
</tr>
<tr>
<td>SQR_DDO_JRE_NOCLASSGC</td>
<td>Disables class garbage collection.</td>
</tr>
<tr>
<td>SQR_DDO_JRE_NOJIT</td>
<td>Disables the JIT compiler (same as Java.compiler=NONE)</td>
</tr>
<tr>
<td>SQR_DDO_JRE_VERBOSE</td>
<td>Logs Java class loading and garbage collection events into the <code>vm.out</code> output file.</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER-EDIT-MASK</td>
<td>Default numeric edit mask format when the keyword NUMBER accompanies DISPLAY, MOVE, PRINT, or SHOW.</td>
</tr>
<tr>
<td>MONEY-EDIT-MASK</td>
<td>Default numeric edit mask format when the keyword MONEY accompanies DISPLAY, MOVE, PRINT, or SHOW.</td>
</tr>
<tr>
<td>DATE-EDIT-MASK</td>
<td>Default date edit mask format when the keyword DATE accompanies DISPLAY, MOVE, PRINT, or SHOW, or the LET datetostr() or strtodate() functions.</td>
</tr>
<tr>
<td>INPUT-DATE-EDIT-MASK</td>
<td>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.</td>
</tr>
<tr>
<td>MONEY-SIGN</td>
<td>Character(s) to replace the '$' edit character.</td>
</tr>
<tr>
<td>MONEY-SIGN-LOCATION</td>
<td>MONEY-SIGN character(s) location. Valid values are LEFT and RIGHT.</td>
</tr>
<tr>
<td>THOUSAND-SEPARATOR</td>
<td>Character to replace the ',' edit character.</td>
</tr>
<tr>
<td>DECIMAL-SEPARATOR</td>
<td>Character to replace the '.' edit character.</td>
</tr>
<tr>
<td>DATE-SEPARATOR</td>
<td>Character to replace the '/' character.</td>
</tr>
<tr>
<td>TIME-SEPARATOR</td>
<td>Character to replace the ':' character.</td>
</tr>
<tr>
<td>EDIT-OPTION-NA</td>
<td>Character(s) to replace the 'na' option.</td>
</tr>
<tr>
<td>EDIT-OPTION-AM</td>
<td>Character(s) to replace 'AM'.</td>
</tr>
<tr>
<td>EDIT-OPTION-PM</td>
<td>Character(s) to replace 'PM'.</td>
</tr>
<tr>
<td>EDIT-OPTION-AD</td>
<td>Character(s) to replace 'AD'.</td>
</tr>
<tr>
<td>EDIT-OPTION-BC</td>
<td>Character(s) to replace 'BC'.</td>
</tr>
<tr>
<td>DAY-OF-WEEK-CASE</td>
<td>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.</td>
</tr>
<tr>
<td>DAY-OF-WEEK-FULL</td>
<td>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.</td>
</tr>
<tr>
<td>DAY-OF-WEEK-SHORT</td>
<td>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.</td>
</tr>
<tr>
<td>MONTHS-CASE</td>
<td>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.</td>
</tr>
<tr>
<td>MONTHS-FULL</td>
<td>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.</td>
</tr>
</tbody>
</table>
### Table 81  Date Column Formats

<table>
<thead>
<tr>
<th>Database</th>
<th>DATE Column Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>YYYY-MM-DD</td>
</tr>
<tr>
<td>INFORMIX</td>
<td>MM/DD/YYYY</td>
</tr>
<tr>
<td>ODBC</td>
<td>DD-MON-YYYY</td>
</tr>
</tbody>
</table>

### Table 82  Time Column Formats

<table>
<thead>
<tr>
<th>Database</th>
<th>TIME Column Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>HH24.MI.SS</td>
</tr>
<tr>
<td>ODBC</td>
<td>HH24:MI:SS</td>
</tr>
</tbody>
</table>

### [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`
- `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 
Character Set
 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

<table>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| Embed | All | Whether to embed fonts; and if so, what fonts to embed. Enter one of the following values:
| | [font numbers] | • 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:

\[ \text{font_number=} \text{Roman\_Typeface[,] CJK\_Typeface, Character\_Map} \]

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

<table>
<thead>
<tr>
<th>Local</th>
<th>CJK Typeface</th>
<th>Character Map</th>
<th>Encoding Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified Chinese</td>
<td>STSong-Light</td>
<td>GB-EUC-H</td>
<td>GB2312</td>
</tr>
<tr>
<td></td>
<td>STSongStd-Light-Acro</td>
<td>GBpc-EUC-H</td>
<td>GB2312</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GBK-EUC-H</td>
<td>CP936</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GBKp-EUC-H</td>
<td>CP936</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GBK2K-EUC-H</td>
<td>GB18030</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UniGB-UCS2-H</td>
<td>UCS-2</td>
</tr>
<tr>
<td>Traditional Chinese</td>
<td>MHei-Medium</td>
<td>B5pc-H</td>
<td>Big5</td>
</tr>
<tr>
<td></td>
<td>MSung-Light</td>
<td>HKscs-B5-H</td>
<td>Big5-HKCS</td>
</tr>
<tr>
<td></td>
<td>MSung-Std-Light-Acro</td>
<td>Big5</td>
<td>Big5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETen-B5-H</td>
<td>EUC-TW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETenms-B5-H</td>
<td>UCS-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CNS-EUC-H</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UniCNS-UCS2-H</td>
<td></td>
</tr>
<tr>
<td>Korean</td>
<td>HYGoThic-Medium</td>
<td>KSC-EUC-H</td>
<td>EUC-KR</td>
</tr>
<tr>
<td></td>
<td>HYSMyeongJo-Medium</td>
<td>KSCms-UHC-H</td>
<td>JOHAB</td>
</tr>
<tr>
<td></td>
<td>HYSMyeongJoStd-Medium-Acro</td>
<td>KSCms-UHC-HW-H</td>
<td>JOHAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KSCpc-EUC-H</td>
<td>EUC-KR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UniKS-UCS2-H</td>
<td>UCS-2</td>
</tr>
<tr>
<td>Japanese</td>
<td>HeiseiKakuGo-W5</td>
<td>83pv-RKSJ-H</td>
<td>Shift-JIS</td>
</tr>
<tr>
<td></td>
<td>HeiseiMin-W3</td>
<td>90ms-RKSJ-H</td>
<td>Shift-JIS</td>
</tr>
</tbody>
</table>
**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>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bookmarks</strong></td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td><strong>Compatibility</strong></td>
<td>4 - 6</td>
<td>Default = 5</td>
</tr>
<tr>
<td><strong>CompressionText</strong></td>
<td>0 - 9</td>
<td>Default = 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - no compression, 9 = maximum compression</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default value or 6 is the best value for compression versus speed.</td>
</tr>
<tr>
<td><strong>CompressionGraphics</strong></td>
<td>0 - 9</td>
<td>Default = 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - no compression, 9 = maximum compression</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default value of 6 is the best value for compression versus speed.</td>
</tr>
<tr>
<td><strong>EmbedAction</strong></td>
<td>Stop</td>
<td>Warn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EncodingNotInFontAction</strong></td>
<td>Stop</td>
<td>Warn</td>
</tr>
<tr>
<td>Entry</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Default = Warn</td>
<td>● Stop—An error message is issued and program execution stops.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Warn—A warning message is issued and CP1252 is used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Skip—No message is issued and CP1252 is used.</td>
</tr>
</tbody>
</table>

**Subsetting**

<table>
<thead>
<tr>
<th>Subsetting</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>Description</td>
<td>True enables subsetting for all fonts where subsetting is possible. False disables subsetting.</td>
<td></td>
</tr>
</tbody>
</table>

**SubsetLimit**

<table>
<thead>
<tr>
<th>SubsetLimit</th>
<th>1-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>100</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

**SubsetMinSize**

<table>
<thead>
<tr>
<th>SubsetMinSize</th>
<th>Numbers &gt; zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>100</td>
</tr>
<tr>
<td>Description</td>
<td>If the original font file is smaller than the specified size (in KB), then font subsetting is disabled for that particular font.</td>
</tr>
</tbody>
</table>

**UnsupportedEncodingAction**

<table>
<thead>
<tr>
<th>UnsupportedEncodingAction</th>
<th>Stop</th>
<th>Warn</th>
<th>Skip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Skip</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Stop—An error message is issued and program execution stops.</td>
<td>Warn—A warning message is issued and CP1252 is used.</td>
<td>Skip—No message is issued and CP1252 is used.</td>
</tr>
</tbody>
</table>

**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 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>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST-PAGE</td>
<td>HEIGHT, WIDTH, NAME</td>
<td>60,60,firstpg.gif</td>
<td>NAME of the graphic image file that accesses the first page of the report. Specify HEIGHT and WIDTH values in pixels.</td>
</tr>
<tr>
<td>PREV-PAGE</td>
<td>HEIGHT, WIDTH, NAME</td>
<td>60,60,prevpg.gif</td>
<td>NAME of the graphic image file that accesses the previous page of the report. Specify HEIGHT and WIDTH values in pixels.</td>
</tr>
<tr>
<td>NEXT-PAGE</td>
<td>HEIGHT, WIDTH, NAME</td>
<td>60,60,nextpg.gif</td>
<td>NAME of the graphic image file that accesses the next page of the report. Specify HEIGHT and WIDTH values in pixels.</td>
</tr>
<tr>
<td>LAST-PAGE</td>
<td>HEIGHT, WIDTH, NAME</td>
<td>60,60,lastpg.gif</td>
<td>NAME of the graphic image file that accesses the last page of the report. Specify HEIGHT and WIDTH values in pixels.</td>
</tr>
<tr>
<td>WALLPAPER</td>
<td>NAME</td>
<td></td>
<td>NAME of the graphic image file used as the report's background image.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Table 87 Entries in [Enhanced-HTML]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entry</strong></td>
</tr>
<tr>
<td>Browser</td>
</tr>
<tr>
<td>DefaultTemplate</td>
</tr>
<tr>
<td>IncludePDFinZip</td>
</tr>
<tr>
<td>Language</td>
</tr>
<tr>
<td>FullHTML</td>
</tr>
<tr>
<td>Entry</td>
</tr>
<tr>
<td>-------</td>
</tr>
</tbody>
</table>

*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)
```

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]

<table>
<thead>
<tr>
<th>Entry</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>driver_name</td>
<td>CSV</td>
<td>DB2</td>
</tr>
</tbody>
</table>

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


Table 89  Entries in [SQR Remote]

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HostName</td>
<td>Default host name.</td>
</tr>
<tr>
<td>UserName</td>
<td>Default username used to log into the host.</td>
</tr>
<tr>
<td>Entry</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| TimeOut | Timeout duration in seconds.  
Default = 30 |
| Passive | Whether the FTP transfer is in (#0) passive or (0) non-passive mode.  
Default = 0 |
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_______.DAT</td>
<td>Data files used by the LOADALL.Production Reporting programs</td>
</tr>
<tr>
<td>_______.MEM</td>
<td>Startup files to run tiny, medium, and big Production Reporting programs</td>
</tr>
<tr>
<td>APPEND.SQR</td>
<td>Demonstrates the append and fixed-nolf commands</td>
</tr>
<tr>
<td>APTDIARY.SQR</td>
<td>Demonstrates columns, text wrapping</td>
</tr>
<tr>
<td>AREA100.SQR</td>
<td>Demonstrates a 100% area chart</td>
</tr>
<tr>
<td>BAR100.SQR</td>
<td>Demonstrates a 100% bar chart</td>
</tr>
<tr>
<td>BARCODE.SQR</td>
<td>Demonstrates printing a bar code</td>
</tr>
<tr>
<td>CALENDAR.SQR</td>
<td>Demonstrates nondatabase formatting</td>
</tr>
<tr>
<td>COMP_FOR.SQR</td>
<td>Prints a graph of the forecasted and actual sales for a given employee</td>
</tr>
<tr>
<td>COMP_F_G.SQR</td>
<td>Prints a graph of the forecasted and actual sales for month or quarter</td>
</tr>
<tr>
<td>COMP_PLN.SQR</td>
<td>Prints a graph of the planned and actual sales for a given employee</td>
</tr>
<tr>
<td>COMP_P_G.SQR</td>
<td>Prints a graph of the planned and actual sales for month or quarter</td>
</tr>
<tr>
<td>COVLET02.SQR</td>
<td>Uses Production Reporting to input data from user, enter data in the database, and write a form letter using a DOCUMENT paragraph</td>
</tr>
<tr>
<td>CRUPSAL.SQR</td>
<td>(Oracle) Creates stored functions and procedures for Oracle Version 7</td>
</tr>
<tr>
<td>CUST.SQR</td>
<td>Prints a list of all of the customers bursted by page</td>
</tr>
<tr>
<td>CUSTLBLS.SQR</td>
<td>Demonstrates printing mailing labels within columns</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CUSTOMER.SQR</td>
<td>Demonstrates multiple detail lines, NEXT-LISTING command</td>
</tr>
<tr>
<td>CUSTOMR2.SQR</td>
<td>Demonstrates the use of the ON-BREAK argument to the PRINT command</td>
</tr>
<tr>
<td>CUSTOMR3.SQR</td>
<td>Demonstrates the use of the INPUT command to change report output</td>
</tr>
<tr>
<td>CUSTOMR4.SQR</td>
<td>Demonstrates the use of arrays</td>
</tr>
<tr>
<td>CUSTOMR5.SQR</td>
<td>Demonstrates dynamic queries to allow user to qualify a report as it runs</td>
</tr>
<tr>
<td>CUST_SUM.SQR</td>
<td>Prints and charts on a bar chart information about each customer in the customer table</td>
</tr>
<tr>
<td>CUSTTAPE.SQR</td>
<td>Demonstrates the flat file output for magnetic tape or other post-processing</td>
</tr>
<tr>
<td>DATAA.DAT</td>
<td>Needed for append.sqr</td>
</tr>
<tr>
<td>DATAB.DAT</td>
<td>Needed for append.sqr</td>
</tr>
<tr>
<td>DROPALL.SQR</td>
<td>Drops all the Production Reporting sample tables created by the LOADALL program</td>
</tr>
<tr>
<td>DROPPROC.SQR</td>
<td>(Sybase) Deletes leftover temporary stored procedures belonging to the user</td>
</tr>
<tr>
<td>DYNAMCOL.SQR</td>
<td>Demonstrates use of dynamic columns, dynamic tables and variables passed to ON-ERROR procedure</td>
</tr>
<tr>
<td>EMP.SQR</td>
<td>Prints a list of all of the employees bursted by page</td>
</tr>
<tr>
<td>EMP_COMM.SQR</td>
<td>Calculates each employee's commission based on sales</td>
</tr>
<tr>
<td>EMP_P_Q.SQR</td>
<td>List all employee quotas for a given month or quarter</td>
</tr>
<tr>
<td>ENVELOPE.SQR</td>
<td>Demonstrates use of printing envelope with proper bar code</td>
</tr>
<tr>
<td>EXPORT.SQR</td>
<td>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.</td>
</tr>
<tr>
<td>FLATFILE.SQR</td>
<td>Creates a Production Reporting report to extract a database table and place it into a flat file</td>
</tr>
<tr>
<td>FLOATBAR.SQR</td>
<td>Demonstrates a floating bar chart</td>
</tr>
<tr>
<td>FOR_CUST.SQR</td>
<td>Sales forecast for given customer grouped by month or quarter</td>
</tr>
<tr>
<td>FOR_EMP.SQR</td>
<td>Sales forecast for given employee grouped by month or quarter</td>
</tr>
<tr>
<td>FOR_PROD.SQR</td>
<td>Sales forecast for given product grouped by month or quarter</td>
</tr>
<tr>
<td>FOR_REG.SQR</td>
<td>Sales forecast for given region grouped by month or quarter</td>
</tr>
<tr>
<td>FOR_SUM.SQR</td>
<td>Creates a table of projected product sales with links to more information</td>
</tr>
<tr>
<td>FORMLETTR.SQR</td>
<td>Demonstrates form letters using a DOCUMENT paragraph</td>
</tr>
<tr>
<td>HILO.SQR</td>
<td>Demonstrates a high-low-close chart</td>
</tr>
<tr>
<td>HISTGRAM.SQR</td>
<td>Demonstrates a histogram chart</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INQUIRY.SQR</td>
<td>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.</td>
</tr>
<tr>
<td>INVOICE.SQR</td>
<td>Demonstrates multiple reports, printing invoices, and printing envelopes</td>
</tr>
</tbody>
</table>
| 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. |
<p>| 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 sqrlogo.sqr                                                        |
| 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 |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STCKAREA.SQR</td>
<td>Demonstrates a stacked area chart</td>
</tr>
<tr>
<td>SQR3DBAR.SQR</td>
<td>Demonstrates a 3D bar chart</td>
</tr>
<tr>
<td>SQRLASER.SQR</td>
<td>Demonstrates graphic and file I/O commands</td>
</tr>
<tr>
<td>SQRLINE.SQR</td>
<td>Demonstrates a line chart</td>
</tr>
<tr>
<td>SQRLOGO.SQR</td>
<td>Demonstrates printing images</td>
</tr>
<tr>
<td>SQRPIE.SQR</td>
<td>Demonstrates a pie chart</td>
</tr>
<tr>
<td>TABREP.SQR</td>
<td>Creates a tabular Production Reporting report for a table you choose</td>
</tr>
<tr>
<td>UPDATE.SQR</td>
<td>Generates a Production Reporting program that allows you to query and update database tables. The created program uses the SHOW command to simulate a menu interface.</td>
</tr>
<tr>
<td>UPDSAL.SQR</td>
<td>A sample report that demonstrates use of stored functions and procedures in Oracle</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out of memory.</td>
<td>Occurs when a call to the C routine 'malloc()' fails.</td>
</tr>
<tr>
<td></td>
<td>● PC – Use the -Mfile to reduce memory requirements. Remove unneeded TSRs</td>
</tr>
<tr>
<td></td>
<td>● UNIX – Increase the size of the system swap file.</td>
</tr>
<tr>
<td></td>
<td>● VAX – Increase the amount of memory allowed for that user.</td>
</tr>
<tr>
<td>No cursors defined.</td>
<td>From the -S command line flag. The Production Reporting program did not contain any commands that required a database cursor.</td>
</tr>
<tr>
<td>Not processed due to report errors.</td>
<td>From the -S command line flag. Production Reporting cannot provide information about the cursor due to errors in the program.</td>
</tr>
<tr>
<td>Enter <code>01</code>02</td>
<td>Type the value to assign to the specified variable.</td>
</tr>
<tr>
<td></td>
<td><code>01</code> = First character of the variable name</td>
</tr>
<tr>
<td></td>
<td><code>02</code> = Rest of the variable name</td>
</tr>
<tr>
<td>NOPROMPT used - Enter value below</td>
<td>(Windows) Appears when an INPUT command is defined with the NOPROMPT argument.</td>
</tr>
<tr>
<td>Enter <code>01</code></td>
<td>Type the value to assign to the specified substitution variable.</td>
</tr>
<tr>
<td></td>
<td><code>01</code> = Name of the substitution variable</td>
</tr>
<tr>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Enter this run's parameters:</td>
<td>Enter the values for the parameters defined in the program.</td>
</tr>
<tr>
<td>Error on line <code>01</code>: `02</td>
<td>Production Reporting detected an error while processing the report file. Correct the error and rerun. <code>01 = Source line number </code>02 = Source line</td>
</tr>
<tr>
<td>Error in include file &quot;<code>01&quot; on line </code>02: `03</td>
<td>Production Reporting detected an error while processing the report file. Correct the error and rerun. <code>01 = Name of the include file </code>02 = Source line number `03 = Source line</td>
</tr>
<tr>
<td>Warning on line <code>01</code>: `02</td>
<td>Production Reporting detected a non-fatal error while processing the report file. <code>01 = Source line number </code>02 = Source line</td>
</tr>
<tr>
<td>Warning in include file &quot;<code>01&quot; on line </code>02: `03</td>
<td>Production Reporting detected a nonfatal error while processing the report file. <code>01 = Name of the include file </code>02 = Source line number `03 = Source line</td>
</tr>
<tr>
<td>Type RETURN for more, C to continue w/o display, X to exit run:</td>
<td>Informational message used with the -D command line flag.</td>
</tr>
<tr>
<td>Loading Oracle DLL Failed!!!</td>
<td>(Oracle) Title for the dialog box that informs the user that Production Reporting could not load the Oracle DLL.</td>
</tr>
<tr>
<td>Errors were found in the program file.</td>
<td>Correct the errors and rerun.</td>
</tr>
<tr>
<td>Errors were found during the program run.</td>
<td>Correct the errors and rerun.</td>
</tr>
<tr>
<td>`01: End of Run.</td>
<td>Informational message. `01 = Image name (for example, SQR)</td>
</tr>
<tr>
<td>Enter report name:</td>
<td>Enter the name of the report (.SQR or .SQT) to run.</td>
</tr>
<tr>
<td>Enter database name:</td>
<td>Enter the name of the database.</td>
</tr>
<tr>
<td>Enter Username:</td>
<td>Enter the user name to log onto the database.</td>
</tr>
<tr>
<td>Enter Password:</td>
<td>Enter the password. For security reasons, the password is not echoed.</td>
</tr>
<tr>
<td>Customer ID:</td>
<td>Text message</td>
</tr>
<tr>
<td>Press Enter to close...</td>
<td>Text message</td>
</tr>
<tr>
<td>Enter Subsystem Name:</td>
<td>Enter the subsystem name.</td>
</tr>
<tr>
<td>`01: Program Aborting</td>
<td>Informational message. `01 = Image name (for example, SQR)</td>
</tr>
</tbody>
</table>
### 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>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>000001</td>
<td>Error while opening the message file: ``01<code> (</code>02): `03</td>
<td>Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. <code>01</code> = Name of the error message file <code>02</code> = System error code <code>03</code> = System error message</td>
</tr>
<tr>
<td>000002</td>
<td>Error while reading the message file. (<code>01): </code>02</td>
<td>Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. <code>01</code> = Name of the error message file <code>02</code> = System error code <code>03</code> = System error message</td>
</tr>
<tr>
<td>000004</td>
<td>Error while seeking the message file. (<code>01): </code>02</td>
<td>Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. <code>01</code> = Name of the error message file <code>02</code> = System error code <code>03</code> = System error message</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>000005</td>
<td>Corrupt message file: Invalid header information.</td>
<td>Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support.</td>
</tr>
<tr>
<td>000006</td>
<td>Corrupt message file: Invalid count (Got <code>01</code>, Should be <code>02</code>).</td>
<td>The header contains an invalid entry count. Ensure SQRDIR points to the correct directory. Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. <code>01 = The value read from the header </code>02 = What the value should be</td>
</tr>
<tr>
<td>000010</td>
<td>Invalid SEMCode encountered: `01.</td>
<td>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</td>
</tr>
<tr>
<td>000011</td>
<td>Unknown conversion type (<code>01) for code </code>02.</td>
<td>Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. <code>01 = Invalid type </code>02 = Internal error code</td>
</tr>
<tr>
<td>000012</td>
<td>Message <code>01</code> must be either Preload or BuiltIn.</td>
<td>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</td>
</tr>
<tr>
<td>000013</td>
<td>Cannot point to message `01.</td>
<td>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</td>
</tr>
<tr>
<td>000014</td>
<td>The required environment variable <code>01</code> has not been defined.</td>
<td>Define the named environment variable and restart Production Reporting. `01 = Environment variable name</td>
</tr>
<tr>
<td>000015</td>
<td>The Meta ESC characters do not match (Got <code>01</code>, Should be <code>02</code>).</td>
<td>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. <code>01 = What was found in the header </code>02 = What was expected to be found</td>
</tr>
<tr>
<td>000016</td>
<td><code>01() called to process (</code>02) and the message file is not open.</td>
<td>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. <code>01 = Name of the routine </code>02 = Error code</td>
</tr>
<tr>
<td>000017</td>
<td>Message <code>01</code> must be ReportParameters or CopyrightNotice.</td>
<td>Try reloading <code>sqrerr.dat</code> from the release media. If the error persists, contact technical support. `01 = Error code</td>
</tr>
<tr>
<td>000028</td>
<td>Cannot access the initialization file: <code>01 (</code>02): `03</td>
<td>The initialization file specified by the <code>-ZIF</code> command line flag cannot be accessed. <code>01 = Name of the file </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>000029</td>
<td>Unknown encoding name: `01</td>
<td>The encoding name specified by the <code>-ZEN</code> command line flag is not valid `01 = Encoding name</td>
</tr>
<tr>
<td>000202</td>
<td>DPUT: Bad field number.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000203</td>
<td>DARRAY: Unknown command number.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>000204</td>
<td><code>01: Cannot find </code>02 command.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. <code>01 = Name of the routine </code>02 = Name of the command</td>
</tr>
<tr>
<td>000205</td>
<td>DDO: DO arguments do not match procedure's.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000206</td>
<td>SDO: Bad params for DO command.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000207</td>
<td>SDO: Bad params for BEGIN-PROCEDURE command.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000209</td>
<td>SGOTO: Bad goto function parameters.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000210</td>
<td>SGOTO: Could not find beginning of section or paragraph.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000211</td>
<td>SGOTO: Bad label: from parameters.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000212</td>
<td>COMPAR: Unknown relational (numeric) operator.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000213</td>
<td>COMPAR: Unknown relational (string) operator.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000214</td>
<td>DONBRK: Unknown case for putlin.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000216</td>
<td>GARRAY: Unknown command number.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000217</td>
<td>GCMDS: No Gfunc found.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000218</td>
<td>GDOC: Unknown document type.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000219</td>
<td>GLET: Bad operator.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>000220</td>
<td>GLET: Stack incorrect for expression - arg `01.</td>
<td>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</td>
</tr>
<tr>
<td>000221</td>
<td>GLET: Unknown operator type.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000222</td>
<td>GLET: Unknown operator in expression.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000223</td>
<td>GPARS: Column not SCOL, TCOL, or NCOL type.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000224</td>
<td>GPARS: Bad parameter format: <code>01 = </code>02=</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. <code>01 = Internal command format string </code>02 = Bad format field found</td>
</tr>
<tr>
<td>000225</td>
<td>GPARS: No end of required word in parfmt: `01</td>
<td>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</td>
</tr>
<tr>
<td>000226</td>
<td>GPARS: Bad parfmt entry: `01</td>
<td>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</td>
</tr>
<tr>
<td>000227</td>
<td>GPARS: Bad parameter string.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000228</td>
<td>GPARS: Repeat count bad: `01</td>
<td>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</td>
</tr>
<tr>
<td>000229</td>
<td>GPARS: Only a,b,8,9 allowed for repeats: `01</td>
<td>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</td>
</tr>
<tr>
<td>000230</td>
<td>GPARS: Missing required x: `01</td>
<td>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</td>
</tr>
<tr>
<td>000231</td>
<td>GPARS: Bad type in 'ckvrpr()'.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000232</td>
<td>GPROC: No Gfunc found.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>000233</td>
<td>GRDWRT: Unknown command number.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000234</td>
<td>GSHOW: Unknown SHOW option.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000235</td>
<td>PGMPARS: `addvar()' passed maxlen but not column.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000238</td>
<td>PGMPARS: <code>01</code> passed invalid parameter number: <code>02</code>.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. <code>01 = Routine name </code>02 = Invalid parameter number</td>
</tr>
<tr>
<td>000239</td>
<td>PGMPARS: <code>txclrf()</code> encountered bad column reference type: <code>01</code>.</td>
<td>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</td>
</tr>
<tr>
<td>000240</td>
<td>PLCMNT: <code>getpic()</code> passed invalid element number: <code>01</code>.</td>
<td>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</td>
</tr>
<tr>
<td>000241</td>
<td>RDPGM: Command array size exceeded (change COMDMAX to at least `01).</td>
<td>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.</td>
</tr>
<tr>
<td>000242</td>
<td>RDPGM: Bad match adding internal variable: <code>01</code></td>
<td>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</td>
</tr>
<tr>
<td>000243</td>
<td>RDPGM: No cmdget function found for BEGIN_.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000244</td>
<td>Function <code>01</code> not included in run-time package.</td>
<td>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</td>
</tr>
<tr>
<td>000245</td>
<td>SETSQL: Could not find variable <code>01</code>, in Run Time.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable name</td>
</tr>
<tr>
<td>000249</td>
<td>SPINIT: Bad parameters.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000251</td>
<td>DBFFIX: DBDATLEN returned out of range status.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>000252</td>
<td>DPRPST: Error converting Sybase type for EXECUTE.</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000254</td>
<td>SETSQL: Could not find variable entry in list.</td>
<td>(Oracle) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000255</td>
<td>DBDESC: SQLD not = number of select columns.</td>
<td>(DB2 and Informix) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>000256</td>
<td>DBFETCH: Unknown variable dbtype encountered: <code>01 (</code>02)</td>
<td>(DB2 and Informix) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. <code>01 = Variable name </code>02 = Unknown database type</td>
</tr>
<tr>
<td>000257</td>
<td>WRITE_SPF: Unknown code encountered: `01</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td><code>01: Cannot find LOAD-LOOKUP table: </code>02</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. <code>01 = Name of the routine </code>02 = Name of the table</td>
</tr>
<tr>
<td>000259</td>
<td>PGMPARS: <code>01 called with wrong variable </code>02</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. <code>01 = Name of the routine </code>02 = Name of the variable</td>
</tr>
<tr>
<td>000261</td>
<td>MODIFYVAR: Attempt to change variable which is not xVAR (`01).</td>
<td>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</td>
</tr>
<tr>
<td>000262</td>
<td>MODIFYVAR: Incompatible variable types (<code>01) and (</code>02).</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. <code>01 = Variable type (from) </code>02 = Variable type (to)</td>
</tr>
</tbody>
</table>

Table 93  Numbered Messages  001000 to 001999

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>001201</td>
<td>Cannot open the argument file: <code>01</code>. (<code>02): </code>03</td>
<td>Depends on the system error message. <code>01 = Name of the file </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>001202</td>
<td>Cannot close the argument file. (<code>01): </code>02</td>
<td>Depends on the system error message. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>001203</td>
<td>Cannot open the -MFile: <code>01</code>. (<code>02): </code>03</td>
<td>Depends on the system error message. <code>01 = Name of the file </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>001204</td>
<td>Minimum value for <code>01' in the -MFile is </code>02.</td>
<td>Correct the -Mfile entry. <code>01 = Keyword in question </code>02 = Minimum value allowed</td>
</tr>
<tr>
<td>001205</td>
<td>Maximum value for <code>01' in the -MFile is </code>02.</td>
<td>Correct the -Mfile entry. <code>01 = Keyword in question </code>02 = Maximum value allowed</td>
</tr>
<tr>
<td>001206</td>
<td>Invalid -MFile entry: `01'.</td>
<td>Correct the -Mfile entry. `01 = The line from the -Mfile</td>
</tr>
<tr>
<td>001207</td>
<td>Cannot close the -MFile. (<code>01): </code>02</td>
<td>Depends on the system error message. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>001209</td>
<td>The minimum value for <code>01' (</code>02) is `03.</td>
<td>Value out of range. <code>01 = Entry name </code>02 = Specified value `03 = Minimum value</td>
</tr>
<tr>
<td>001210</td>
<td>The maximum value for <code>01' (</code>02) is `03.</td>
<td>Value out of range. <code>01 = Entry name </code>02 = Specified value `03 = Maximum value</td>
</tr>
<tr>
<td>001211</td>
<td>The value for <code>01' (</code>02) is not an integer number.</td>
<td>Value must be an integer value. <code>01 = Entry name </code>02 = Specified value</td>
</tr>
<tr>
<td>001300</td>
<td>Bind list does not match query (do not use '@__p' string).</td>
<td>Production Reporting reserves the variable names that start with '@__p' for internal use. Edit the source code and use different variable names.</td>
</tr>
<tr>
<td>001301</td>
<td>Forward references not permitted in select list bind variables.</td>
<td>Within the body of BEGIN-SQL paragraphs, forward references to &amp;column names are not permitted. Move the BEGIN-SQL paragraph after the &amp;column definition.</td>
</tr>
<tr>
<td>001303</td>
<td>Error in SQL (perhaps missing &amp;name after expression):</td>
<td>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.</td>
</tr>
<tr>
<td>001304</td>
<td>Check SELECT columns, expressions and 'where' clause for syntax.</td>
<td>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.</td>
</tr>
<tr>
<td>001305</td>
<td>CMPSQL: Unknown data type in database: `01.</td>
<td>Contact technical support with the version of the database you are connected to. `01 = Datatype in question</td>
</tr>
<tr>
<td>001307</td>
<td>CMPSQL: DBDEFN failed.</td>
<td>(ODBC, Oracle, Informix) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>001308</td>
<td><code>01: Could not bind column </code>02.</td>
<td>(ODBC, Oracle, Informix) Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. <code>01 = Name of the SQR routine </code>02 = Name of the column</td>
</tr>
<tr>
<td>001309</td>
<td>The type for <code>&amp;01' (</code>02) does not match the type from the database (`03).</td>
<td>Correct the source code. <code>01 = Name of the column/ expression pseudonym </code>02 = User specified type `03 = Database type</td>
</tr>
<tr>
<td>001400</td>
<td>Only numerics allowed for arithmetic.</td>
<td>Only #numeric variables, &amp;columns, and literals are permitted in the arithmetic commands. Correct the source code.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>001401</td>
<td>Optional qualifier is ROUND=n (0-`01).</td>
<td>Correct the syntax. `01 = Maximum value for ROUND=n.</td>
</tr>
<tr>
<td>001402</td>
<td>Optional qualifiers for DIVIDE are ON-ERROR=HIGH</td>
<td>ZER0 and ROUND=n.</td>
</tr>
<tr>
<td>001403</td>
<td>Attempting division by zero.</td>
<td>Use the ON-ERROR = HIGH</td>
</tr>
<tr>
<td>001404</td>
<td>Bad number of digits to ROUND or TRUNC (0-15).</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>001405</td>
<td>WARNING: The ROUND or TRUNC qualifier is greater than the number’s precision.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>001500</td>
<td>Array element out of range (0<code>01) for array </code>02 on line `03.</td>
<td>Correct the source logic. <code>01 = Element number passed </code>02 = Name of the array `03 = Program line number</td>
</tr>
<tr>
<td>001501</td>
<td>Field element out of range (0<code>01) for array </code>02, field <code>03, on line </code>04.</td>
<td>Correct the source logic. <code>01 = Element number passed </code>02 = Name of the array <code>03 = Name of the field </code>04 = Program line number</td>
</tr>
<tr>
<td>001502</td>
<td>WARNING: Attempting division by zero on line <code>01. Array field </code>02 unchanged. Run continuing...</td>
<td>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. <code>01 = Program line number </code>02 = Name of field</td>
</tr>
<tr>
<td>001601</td>
<td>‘FILL’ not appropriate for numeric data.</td>
<td>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.</td>
</tr>
<tr>
<td>001700</td>
<td>Report ‘01: Columns must be between 1 and the page width (‘02).</td>
<td>The specified value is wider than the width of the page. Correct the source line. <code>01 = Name of the current report </code>02 = Page width</td>
</tr>
<tr>
<td>001702</td>
<td>Report ‘01: GOTO-TOP=’02 must be between 0 and the page depth (‘03).</td>
<td>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. <code>01 = Name of the current report </code>02 = Goto-Top value `03 = Page width</td>
</tr>
<tr>
<td>001703</td>
<td>Report ‘01: ERASE-PAGE=’02 must be between 0 and the page depth (‘03).</td>
<td>The line number specified on the ERASE-PAGE argument of the NEXT-COLUMN command is greater than the page depth. Correct the source line. <code>01 = Name of the current report </code>02 = Erase-Page value `03 = Page width</td>
</tr>
<tr>
<td>001704</td>
<td>Report ‘01: The NEXT-COLUMN command is not legal in the ‘02 section with the qualifier AT-END=NEWPAGE.</td>
<td>Correct the source line. <code>01 = Name of the current report </code>02 = Name of the section</td>
</tr>
<tr>
<td>001705</td>
<td>Report ‘01: Column number ‘02 is not defined.</td>
<td>The column number specified with the USE-COLUMN command is greater than the highest column defined in the COLUMNS command. Correct the source line. <code>01 = Name of the current report </code>02 = Column number</td>
</tr>
<tr>
<td>001800</td>
<td>Format for CONNECT: username/password [ON-ERROR=procedure([arg1],[arg2],...)]</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>001801</td>
<td>Cannot use CONNECT while SQL statements are active.</td>
<td>Correct the program logic to ensure that all BEGIN-SELECT paragraphs have completed before executing the CONNECT command.</td>
</tr>
<tr>
<td>001802</td>
<td>Logoff failed prior to CONNECT.</td>
<td>The database server returned an error while trying to log off from the database. SQR aborts the program run since it cannot continue.</td>
</tr>
<tr>
<td>001803</td>
<td>CONNECT failed. Perhaps username/password incorrect.</td>
<td>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.</td>
</tr>
<tr>
<td>001804</td>
<td>Sybase extensions SET and SETUSER not permitted in SQR.</td>
<td>Remove SET and SETUSER from the source.</td>
</tr>
<tr>
<td>001805</td>
<td>USE allowed once in SETUP section only, not in BEGIN-SQL. Elsewhere, specify db.[user].table...</td>
<td>Correct the source.</td>
</tr>
<tr>
<td>001807</td>
<td>The requested database connection (`01) is already active.</td>
<td>The -Cnn value specified is being used by another BEGIN-SELECT paragraph that is currently selecting data. Use another connection number. `01 = Connection number</td>
</tr>
<tr>
<td>001808</td>
<td>Cannot find inactive database cursor. Program too large.</td>
<td>Too many BEGIN-SELECT and BEGIN-SQL paragraphs are active at the same time. Reduce the complexity of the program.</td>
</tr>
<tr>
<td>001809</td>
<td>Database commit failed.</td>
<td>(DB2, ODBC, Oracle) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>001810</td>
<td>Database rollback failed.</td>
<td>(DB2, ODBC, Oracle) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>001811</td>
<td>Cannot open database cursor.</td>
<td>(ODBC, Oracle) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>001901</td>
<td>Variable for date-time must begin with <code>&amp;</code>.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>001913</td>
<td>Format code must be SYYYY when specifying signed year.</td>
<td>Correct the edit mask.</td>
</tr>
<tr>
<td>001914</td>
<td>Bad input data (<code>01) for edit mask: </code>02`.</td>
<td>Correct the input. <code>01 = Data being converted </code>02 = Edit mask</td>
</tr>
<tr>
<td>001915</td>
<td>Year cannot be zero.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001916</td>
<td>Year must be between -4713 and 9999 inclusive.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001917</td>
<td>Ambiguous date-time.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001918</td>
<td><code>01</code> is not a valid date part.</td>
<td>Correct the date part. `01 = Date part.</td>
</tr>
<tr>
<td>001919</td>
<td>Invalid day of week.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001920</td>
<td>Format code cannot appear in date input format: <code>01</code>.</td>
<td>Correct the edit mask. `01 = Improper format characters.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>001921</td>
<td>Bad date mask starting at: ‘01’.</td>
<td>Correct the edit mask. ‘01 = Improper format characters.</td>
</tr>
<tr>
<td>001922</td>
<td>Seconds past midnight must be between 0 and 86399.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001923</td>
<td>Seconds must be between 0 and 59.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001924</td>
<td>Minutes must be between 0 and 59.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001925</td>
<td>Month must be between 1 and 12.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001926</td>
<td>Day must be between 1 and 01.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001927</td>
<td>Hour must be between 1 and 12.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001928</td>
<td>Hour must be between 0 to 23.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001929</td>
<td>HH24 precludes the use of meridian indicator.</td>
<td>Correct the edit mask.</td>
</tr>
<tr>
<td>001930</td>
<td>HH12 requires meridian indicator.</td>
<td>Correct the edit mask.</td>
</tr>
<tr>
<td>001931</td>
<td>Day of year must be between 1 and 365 (366 for leap year).</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001932</td>
<td>Date string too long.</td>
<td>Correct the date.</td>
</tr>
<tr>
<td>001933</td>
<td>The month (‘01) is not valid for the current locale or database.</td>
<td>Correct the date. ‘01 = Name of the month.</td>
</tr>
<tr>
<td>001934</td>
<td>The format mask must be a literal when the date-time is not loaded into a variable.</td>
<td>Correct the format mask. The format mask must be a literal when the date-time is not loaded into a variable.</td>
</tr>
<tr>
<td>001935</td>
<td>Date-time format too long.</td>
<td>Correct the format mask.</td>
</tr>
<tr>
<td>001936</td>
<td>Bad date-time format.</td>
<td>Correct the format mask.</td>
</tr>
<tr>
<td>001937</td>
<td>Bad SQL for default date-time. (Table DUAL required for syntax.)</td>
<td>(Oracle) Possibly the format mask needs to be corrected; otherwise, there is a problem with the database server.</td>
</tr>
<tr>
<td>001937</td>
<td>Bad SQL for default date-time. (Table DUAL required for syntax.)</td>
<td>(DB2) Possibly the format mask needs to be corrected; otherwise, there is a problem with the database server.</td>
</tr>
<tr>
<td>001938</td>
<td>Cannot recompile sql.</td>
<td>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.</td>
</tr>
<tr>
<td>001939</td>
<td>Problem executing cursor.</td>
<td>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.</td>
</tr>
<tr>
<td>001940</td>
<td>Error fetching row.</td>
<td>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.</td>
</tr>
<tr>
<td>001941</td>
<td>Cannot redefine variable addresses.</td>
<td>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.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>001942</td>
<td>The date '01' is not in the format: SYYYYMMDD[HH24[MI[SS[NNNNNN]]]].</td>
<td>When specifying an SQR date, at a minimum, the date must be specified; the time is optional. '01 = The invalid date.</td>
</tr>
<tr>
<td>001943</td>
<td>The date '01' is not in one of the accepted formats listed below: MM/DD/YYYY [BC</td>
<td>AD] [HH:MI:[SS:[NNNNNN]]] [AM</td>
</tr>
<tr>
<td>001944</td>
<td>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]]]].</td>
<td>(Oracle) The date was not in one of the expected formats for this database. '01 = The invalid date.</td>
</tr>
<tr>
<td>001944</td>
<td>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:[NNNN]]] [AM</td>
<td>PM] Mon DD YYYY [HH:MI:[SS:[NNNN]]] [AM</td>
</tr>
<tr>
<td>001944</td>
<td>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:[NNNN]]] [AM</td>
<td>PM] Mon DD YYYY [HH:MI:[SS:[NNNN]]] [AM</td>
</tr>
<tr>
<td>001944</td>
<td>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:[NNNN]]] [AM</td>
<td>PM] Mon DD YYYY [HH:MI:[SS:[NNNN]]] [AM</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>001944</td>
<td>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 YYYYMMDD[HH24][MI][SS][NNNNNN]]</td>
<td>(Informix) The date was not in one of the expected formats for this database. `01 = The invalid date.</td>
</tr>
<tr>
<td>001944</td>
<td>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.MM][SS][NNNNNN] MM/DD/YYYY DD.MM.YYYY YYYYMMDD[HH24][MI][SS][NNNNNN]]</td>
<td>(DB2) The date was not in one of the expected formats for this database. `01 = The invalid date.</td>
</tr>
<tr>
<td>001944</td>
<td>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 YYYYMMDD[HH24][MI][SS][NNNNNN]]</td>
<td>(Teradata) The date was not in the one of the expected formats for this database. `01 = The invalid date.</td>
</tr>
<tr>
<td>001946</td>
<td>The date variables are incompatible with each other.</td>
<td>The SQR function references two date variables which cannot be logically be used together (for example, DateDiff of 'date-only' and 'time-only' dates).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 94 Numbered Messages 002000 to 002999</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>002000</td>
</tr>
<tr>
<td>002001</td>
</tr>
<tr>
<td>002002</td>
</tr>
<tr>
<td>002003</td>
</tr>
<tr>
<td>002100</td>
</tr>
<tr>
<td>002101</td>
</tr>
<tr>
<td>002103</td>
</tr>
<tr>
<td>002104</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>002106</td>
</tr>
<tr>
<td>002107</td>
</tr>
<tr>
<td>002200</td>
</tr>
<tr>
<td>002300</td>
</tr>
<tr>
<td>002301</td>
</tr>
<tr>
<td>002400</td>
</tr>
<tr>
<td>002401</td>
</tr>
<tr>
<td>002500</td>
</tr>
<tr>
<td>002501</td>
</tr>
<tr>
<td>002502</td>
</tr>
<tr>
<td>002503</td>
</tr>
<tr>
<td>002506</td>
</tr>
<tr>
<td>002507</td>
</tr>
<tr>
<td>002508</td>
</tr>
<tr>
<td>002510</td>
</tr>
<tr>
<td>002511</td>
</tr>
<tr>
<td>002512</td>
</tr>
<tr>
<td>002513</td>
</tr>
<tr>
<td>002514</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>002515</td>
</tr>
<tr>
<td>002600</td>
</tr>
<tr>
<td>002601</td>
</tr>
<tr>
<td>002602</td>
</tr>
<tr>
<td>002603</td>
</tr>
<tr>
<td>002604</td>
</tr>
<tr>
<td>002605</td>
</tr>
<tr>
<td>002606</td>
</tr>
<tr>
<td>002607</td>
</tr>
<tr>
<td>002609</td>
</tr>
<tr>
<td>002610</td>
</tr>
<tr>
<td>002611</td>
</tr>
<tr>
<td>002613</td>
</tr>
<tr>
<td>002615</td>
</tr>
<tr>
<td>002616</td>
</tr>
<tr>
<td>002617</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>002618</td>
</tr>
<tr>
<td>002619</td>
</tr>
<tr>
<td>002620</td>
</tr>
<tr>
<td>002700</td>
</tr>
<tr>
<td>002800</td>
</tr>
<tr>
<td>002801</td>
</tr>
<tr>
<td>002802</td>
</tr>
<tr>
<td>002804</td>
</tr>
<tr>
<td>002805</td>
</tr>
<tr>
<td>002806</td>
</tr>
<tr>
<td>002900</td>
</tr>
<tr>
<td>002901</td>
</tr>
<tr>
<td>002902</td>
</tr>
<tr>
<td>002903</td>
</tr>
<tr>
<td>002904</td>
</tr>
<tr>
<td>002905</td>
</tr>
<tr>
<td>002906</td>
</tr>
<tr>
<td>002907</td>
</tr>
<tr>
<td>002908</td>
</tr>
<tr>
<td>002909</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>002910</td>
</tr>
<tr>
<td>002911</td>
</tr>
<tr>
<td>002912</td>
</tr>
<tr>
<td>002913</td>
</tr>
<tr>
<td>002914</td>
</tr>
<tr>
<td>002915</td>
</tr>
<tr>
<td>002916</td>
</tr>
<tr>
<td>002917</td>
</tr>
<tr>
<td>002918</td>
</tr>
<tr>
<td>002919</td>
</tr>
<tr>
<td>002920</td>
</tr>
<tr>
<td>002921</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>003000</td>
</tr>
<tr>
<td>003100</td>
</tr>
<tr>
<td>003101</td>
</tr>
<tr>
<td>003200</td>
</tr>
<tr>
<td>003201</td>
</tr>
<tr>
<td>003202</td>
</tr>
<tr>
<td>003203</td>
</tr>
<tr>
<td>003204</td>
</tr>
<tr>
<td>003205</td>
</tr>
<tr>
<td>003206</td>
</tr>
<tr>
<td>003207</td>
</tr>
<tr>
<td>003208</td>
</tr>
<tr>
<td>003209</td>
</tr>
<tr>
<td>003210</td>
</tr>
<tr>
<td>003211</td>
</tr>
<tr>
<td>003212</td>
</tr>
<tr>
<td>003214</td>
</tr>
<tr>
<td>003300</td>
</tr>
<tr>
<td>003301</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>003400</td>
</tr>
<tr>
<td>003401</td>
</tr>
<tr>
<td>003402</td>
</tr>
<tr>
<td>003403</td>
</tr>
<tr>
<td>003404</td>
</tr>
<tr>
<td>003405</td>
</tr>
<tr>
<td>003500</td>
</tr>
<tr>
<td>003501</td>
</tr>
<tr>
<td>003502</td>
</tr>
<tr>
<td>003503</td>
</tr>
<tr>
<td>003504</td>
</tr>
<tr>
<td>003505</td>
</tr>
<tr>
<td>003506</td>
</tr>
<tr>
<td>003507</td>
</tr>
<tr>
<td>003508</td>
</tr>
<tr>
<td>003509</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>003510</td>
</tr>
<tr>
<td>003511</td>
</tr>
<tr>
<td>003512</td>
</tr>
<tr>
<td>003513</td>
</tr>
<tr>
<td>003514</td>
</tr>
<tr>
<td>003515</td>
</tr>
<tr>
<td>003516</td>
</tr>
<tr>
<td>003517</td>
</tr>
<tr>
<td>003518</td>
</tr>
<tr>
<td>003519</td>
</tr>
<tr>
<td>003520</td>
</tr>
<tr>
<td>003521</td>
</tr>
<tr>
<td>003522</td>
</tr>
<tr>
<td>003523</td>
</tr>
<tr>
<td>003525</td>
</tr>
<tr>
<td>003526</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>003527</td>
</tr>
<tr>
<td>003529</td>
</tr>
<tr>
<td>003530</td>
</tr>
<tr>
<td>003600</td>
</tr>
<tr>
<td>003603</td>
</tr>
<tr>
<td>003605</td>
</tr>
<tr>
<td>003700</td>
</tr>
<tr>
<td>003701</td>
</tr>
<tr>
<td>003702</td>
</tr>
<tr>
<td>003703</td>
</tr>
<tr>
<td>003704</td>
</tr>
<tr>
<td>003705</td>
</tr>
<tr>
<td>003706</td>
</tr>
<tr>
<td>003707</td>
</tr>
<tr>
<td>003708</td>
</tr>
<tr>
<td>003709</td>
</tr>
<tr>
<td>003710</td>
</tr>
<tr>
<td>003711</td>
</tr>
<tr>
<td>003712</td>
</tr>
<tr>
<td>003713</td>
</tr>
<tr>
<td>003714</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>003715</td>
</tr>
<tr>
<td>003716</td>
</tr>
<tr>
<td>003717</td>
</tr>
<tr>
<td>003718</td>
</tr>
<tr>
<td>003719</td>
</tr>
<tr>
<td>003720</td>
</tr>
<tr>
<td>003721</td>
</tr>
<tr>
<td>003721</td>
</tr>
<tr>
<td>003721</td>
</tr>
<tr>
<td>003721</td>
</tr>
<tr>
<td>003721</td>
</tr>
<tr>
<td>003721</td>
</tr>
<tr>
<td>003721</td>
</tr>
<tr>
<td>003722</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>003723</td>
</tr>
<tr>
<td>003724</td>
</tr>
<tr>
<td>003725</td>
</tr>
<tr>
<td>003726</td>
</tr>
<tr>
<td>003727</td>
</tr>
<tr>
<td>003728</td>
</tr>
<tr>
<td>003729</td>
</tr>
<tr>
<td>003730</td>
</tr>
<tr>
<td>003730</td>
</tr>
<tr>
<td>003730</td>
</tr>
<tr>
<td>003730</td>
</tr>
<tr>
<td>003730</td>
</tr>
<tr>
<td>003730</td>
</tr>
<tr>
<td>003731</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>003732</td>
</tr>
<tr>
<td>003733</td>
</tr>
<tr>
<td>003734</td>
</tr>
<tr>
<td>003735</td>
</tr>
<tr>
<td>003736</td>
</tr>
<tr>
<td>003737</td>
</tr>
<tr>
<td>003738</td>
</tr>
<tr>
<td>003741</td>
</tr>
<tr>
<td>003742</td>
</tr>
<tr>
<td>003743</td>
</tr>
<tr>
<td>003745</td>
</tr>
<tr>
<td>003746</td>
</tr>
<tr>
<td>003747</td>
</tr>
<tr>
<td>003748</td>
</tr>
<tr>
<td>003749</td>
</tr>
<tr>
<td>003750</td>
</tr>
<tr>
<td>003751</td>
</tr>
<tr>
<td>003752</td>
</tr>
<tr>
<td>003755</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>003800</td>
</tr>
<tr>
<td>003801</td>
</tr>
<tr>
<td>003802</td>
</tr>
<tr>
<td>003803</td>
</tr>
<tr>
<td>003900</td>
</tr>
<tr>
<td>003901</td>
</tr>
<tr>
<td>003902</td>
</tr>
<tr>
<td>003902</td>
</tr>
<tr>
<td>003902</td>
</tr>
<tr>
<td>003902</td>
</tr>
<tr>
<td>003903</td>
</tr>
<tr>
<td>003904</td>
</tr>
<tr>
<td>003905</td>
</tr>
<tr>
<td>003906</td>
</tr>
<tr>
<td>003907</td>
</tr>
<tr>
<td>003908</td>
</tr>
<tr>
<td>003909</td>
</tr>
<tr>
<td>003910</td>
</tr>
<tr>
<td>003911</td>
</tr>
<tr>
<td>003912</td>
</tr>
<tr>
<td>003913</td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>003913</td>
</tr>
<tr>
<td>003913</td>
</tr>
<tr>
<td>003913</td>
</tr>
<tr>
<td>003914</td>
</tr>
<tr>
<td>003915</td>
</tr>
<tr>
<td>003918</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>003918</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>003920</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>003921</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>003921</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>003922</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Error Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
</tbody>
</table>
| 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

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>004000</td>
<td>Result #variabe or $variable or '=' missing in expression.</td>
<td>The LET command is not properly formatted. Correct the source line.</td>
</tr>
<tr>
<td>004001</td>
<td>Expression too complex.</td>
<td>The expression is either too long or is too deeply nested. Break the expression into smaller expressions.</td>
</tr>
<tr>
<td>004002</td>
<td>Parentheses unbalanced in expression.</td>
<td>A left or right parenthesis is missing. Correct the source line.</td>
</tr>
</tbody>
</table>
| 004003       | Too many variables; maximum is "`01".                                          | Break the expression into smaller expressions.  
`01 = Maximum number supported by this version of SQR |
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>004004</td>
<td>Empty expression.</td>
<td>The expression is invalid. Correct the source line.</td>
</tr>
<tr>
<td>004005</td>
<td>Extra comma in expression.</td>
<td>An argument is missing after a comma in the expression. Correct the source line.</td>
</tr>
<tr>
<td>004006</td>
<td>Unknown operator <code>01'. Do you mean </code>02`?</td>
<td>The concatenation operator is `</td>
</tr>
<tr>
<td>004007</td>
<td>Too many &amp;column forward references in expression; maximum is <code>01</code>.</td>
<td>The expression contains too many forward references. Break the expression into smaller expressions. `01 = Maximum number supported by this version of SQR</td>
</tr>
<tr>
<td>004008</td>
<td>Unknown function or variable in expression: `01</td>
<td>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</td>
</tr>
<tr>
<td>004009</td>
<td>Function `01' missing parentheses.</td>
<td>All functions in an expression must be followed by their arguments enclosed in parentheses. Correct the source line.</td>
</tr>
<tr>
<td>004010</td>
<td>Empty parentheses or expression.</td>
<td>A pair of parentheses were found with nothing inside them. Remove the () in question from the source line.</td>
</tr>
<tr>
<td>004011</td>
<td>User function `01' has incorrect number of arguments.</td>
<td>Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name</td>
</tr>
<tr>
<td>004012</td>
<td>Function `01' has incorrect number of arguments.</td>
<td>Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name</td>
</tr>
<tr>
<td>004013</td>
<td>Missing operator in expression.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>004014</td>
<td>Operator `01' missing argument.</td>
<td>Correct the syntax of the function. Functions are described under the LET command. `01 = Operator</td>
</tr>
<tr>
<td>004015</td>
<td>Function `01' missing argument.</td>
<td>Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name</td>
</tr>
<tr>
<td>004016</td>
<td>Function or operator `01' missing arguments.</td>
<td>Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name</td>
</tr>
<tr>
<td>004017</td>
<td>User function `01' requires character argument.</td>
<td>Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name</td>
</tr>
<tr>
<td>004018</td>
<td>User function `01' requires numeric argument.</td>
<td>Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name</td>
</tr>
<tr>
<td>004019</td>
<td>User function `01' requires $string variable.</td>
<td>Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name</td>
</tr>
<tr>
<td>004020</td>
<td>User function `01' requires #numeric variable.</td>
<td>Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>004021</td>
<td>User function <code>o1</code> has incorrect argument type list. Must be of: c,n,C,N</td>
<td>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. <code>o1</code> = User function name</td>
</tr>
<tr>
<td>004022</td>
<td>User function <code>o1</code> missing arguments.</td>
<td>Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. <code>o1</code> = User function name</td>
</tr>
<tr>
<td>004023</td>
<td>User function <code>o1</code> has incorrect return type. Must be c or n.</td>
<td>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. <code>o1</code> = User function name</td>
</tr>
<tr>
<td>004024</td>
<td><code>isnull</code> requires a &amp;column, $string, or $date argument.</td>
<td>#numeric variables cannot be NULL. Correct the source line.</td>
</tr>
<tr>
<td>004025</td>
<td><code>nvl</code> requires a &amp;column, $string, or $date as its first argument.</td>
<td>#numeric variables cannot be NULL. Correct the source line.</td>
</tr>
<tr>
<td>004026</td>
<td>Function or operator <code>o1</code> requires character argument.</td>
<td>Correct the source line. <code>o1</code> = Function or operator</td>
</tr>
<tr>
<td>004027</td>
<td>Function or operator <code>o1</code> requires numeric argument.</td>
<td>Correct the source line. <code>o1</code> = Function or operator</td>
</tr>
<tr>
<td>004028</td>
<td>IF or WHILE expression must return logical result.</td>
<td>The expression used must evaluate a statement that will be TRUE or FALSE. Correct the source line.</td>
</tr>
<tr>
<td>004029</td>
<td>Attempting division by zero in expression.</td>
<td>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).</td>
</tr>
<tr>
<td>004030</td>
<td>Attempting division by zero with <code>%</code>.</td>
<td>An attempt was made to divide a number using the &quot; %&quot; operator. Use the COND() function to check if the divisor is zero; then divide by something else (for example, 1).</td>
</tr>
<tr>
<td>004031</td>
<td>The number used with '%' (<code>o1</code>) is out of range.</td>
<td>The &quot;%&quot; operator works with integers only. Correct the program logic. <code>o1</code> = Maximum value allowed</td>
</tr>
<tr>
<td>004032</td>
<td>User function has unknown return type -- expecting n or c -- need to recompile Run-Time file?</td>
<td>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.</td>
</tr>
<tr>
<td>004033</td>
<td>In user function use C type with allocated string to change $variable.</td>
<td>SQR detected an error while processing a user defined function. Correct the UFUNC.C program file recompile UFUNC.C and recreate the SQR executable.</td>
</tr>
<tr>
<td>004034</td>
<td>Could not find array <code>o1</code> in ARRAY function.</td>
<td>Check for a misspelled array name. <code>o1</code> = Array name</td>
</tr>
<tr>
<td>004035</td>
<td>Could not find array field <code>o1</code> in ARRAY function.</td>
<td>Check for a misspelled array field name. <code>o1</code> = Array field name</td>
</tr>
<tr>
<td>004036</td>
<td>Math error in expression (usually over- or under-flow).</td>
<td>Most of the SQR mathematical built-in functions have a corresponding C library routine. One returned an error.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>004037</td>
<td>Error executing expression.</td>
<td>Break the expression into discrete expressions in order to identify the function that caused the error.</td>
</tr>
<tr>
<td>004038</td>
<td>Out of space while processing expression; Use -Mfile to increase EXPRESSIONSPACE.</td>
<td>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.</td>
</tr>
<tr>
<td>004039</td>
<td>`&quot;01' assumed to be a variable name, not an expression.</td>
<td>Warning message. `01 = Expression in question</td>
</tr>
<tr>
<td>004040</td>
<td>The array `&quot;01' has not been defined.</td>
<td>Define the array using the CREATE-ARRAY command. `01 = Array name</td>
</tr>
<tr>
<td>004041</td>
<td>The field <code>&quot;01' is not valid for array </code>&quot;02'.</td>
<td>Correct the source code. <code>01 = Field name </code>02 = Array name</td>
</tr>
<tr>
<td>004042</td>
<td>The array reference `&quot;01' has an incorrect number of parameters specified.</td>
<td>Correct the source code. `01 = Array name</td>
</tr>
<tr>
<td>004043</td>
<td>The array reference `&quot;01' requires numeric parameters for the element and occurs arguments.</td>
<td>Correct the source code. `01 = Array name</td>
</tr>
<tr>
<td>004045</td>
<td>Function or operator `&quot;01' requires date argument.</td>
<td>Correct the source code. `01 = Array name</td>
</tr>
<tr>
<td>004046</td>
<td>Incompatible types between expression and variable.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>004047</td>
<td>The field `&quot;01' is must be 'char' or 'float'.</td>
<td>Correct the source code. `01 = Field name</td>
</tr>
<tr>
<td>004048</td>
<td>Function or operator `&quot;01' must be a string or date argument.</td>
<td>Correct the source line. `01 = Function or operator</td>
</tr>
<tr>
<td>004049</td>
<td>Unknown transform value `&quot;01' in TRANSFORMATION function.</td>
<td>Check for a misspelled transform value. `01 = Transform value</td>
</tr>
<tr>
<td>004100</td>
<td>Use 'print' command to format data outside SELECT query.</td>
<td>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.</td>
</tr>
<tr>
<td>004101</td>
<td>Cannot find required parameter.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>004102</td>
<td>Bad number found.</td>
<td>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.</td>
</tr>
<tr>
<td>004103</td>
<td>Cannot find required numeric parameter.</td>
<td>Correct the syntax</td>
</tr>
<tr>
<td>004104</td>
<td>Cannot find placement parameters.</td>
<td>The position qualifier &quot;(Row,Col,Len)&quot; was not found. Check for a missing parentheses.</td>
</tr>
<tr>
<td>004105</td>
<td>Placement parameter incorrect.</td>
<td>The &quot;Row&quot;, &quot;Column&quot; or &quot;Length&quot; fields are invalid or ill-formed. Correct the source line.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>004106</td>
<td>Invalid second function on line.</td>
<td>An SQR command used as a qualifier for a primary command (for example, PRINT) is incorrect. Correct the source line.</td>
</tr>
<tr>
<td>004107</td>
<td>Second function must be FORMAT type.</td>
<td>The PRINT command may have format command qualifiers such as WRAP, CENTER, or FILL. Other qualifier commands are not permitted.</td>
</tr>
<tr>
<td>004108</td>
<td>Missing operator =, &lt;, &gt;, ...</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>004109</td>
<td>Invalid operator.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>004110</td>
<td>Missing variable.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>004111</td>
<td>Please give this expression a &amp;pseudonym.</td>
<td>Expressions selected in BEGIN-SELECT should be given an &amp;Name or be followed by a print position <em>(Row,Col,Len)</em>. Correct the source line.</td>
</tr>
<tr>
<td>004112</td>
<td>Wrong variable type.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>004113</td>
<td>Command incomplete, expected `01'.</td>
<td>Correct the syntax. `01 = What was expected</td>
</tr>
<tr>
<td>004114</td>
<td>Expecting '01', found '02'.</td>
<td>Correct the syntax. <code>01 = What was expected </code>02 = What was encountered</td>
</tr>
<tr>
<td>004115</td>
<td>Unknown command or extra parameters found (missing quotes?).</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>004116</td>
<td>Duplicate references to parameter `01'.</td>
<td>Correct the syntax. `01 = Duplicated parameter</td>
</tr>
<tr>
<td>004117</td>
<td>Unexpected equal sign found with `01'.</td>
<td>Correct the syntax. `01 = Parameter name</td>
</tr>
<tr>
<td>004118</td>
<td>Qualifier `01' cannot be used with the following qualifiers:</td>
<td>Correct the syntax. `01 = Qualifier name</td>
</tr>
<tr>
<td>004119</td>
<td>Expecting numeric column, found string column.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>004120</td>
<td>Date variables (`01) cannot be used with this command.</td>
<td>Correct the syntax. `01 = Parameter name</td>
</tr>
<tr>
<td>004200</td>
<td>Page width and depth must be &gt; 0 and &lt; 32767.</td>
<td>The values specified with PAGE-SIZE are out of bounds. Specify legal values.</td>
</tr>
<tr>
<td>004300</td>
<td>Missing end of placement (...) in SHOW.</td>
<td>The placement parameter is ill-formed. Correct the source line.</td>
</tr>
<tr>
<td>004301</td>
<td>Bad (...) location in SHOW.</td>
<td>Screen positions must be valid numbers. Correct the source line.</td>
</tr>
<tr>
<td>004302</td>
<td>Missing literal or variable name to EDIT in SHOW.</td>
<td>The literal or variable name must immediately precede the EDIT, NUMBER, MONEY, or DATE keywords.</td>
</tr>
<tr>
<td>004303</td>
<td>Missing edit mask in SHOW.</td>
<td>The word EDIT must be followed by a valid edit mask. Correct the source line.</td>
</tr>
<tr>
<td>004304</td>
<td>Only string variable allowed for dynamic edit mask.</td>
<td>Dynamic edit masks may only be stored in $Variables. Correct the line.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>004305</td>
<td>Unknown option for SHOW.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>004406</td>
<td>Number `01 not allowed.</td>
<td>Use a different value. `01 = Internal number</td>
</tr>
<tr>
<td>004407</td>
<td>Referenced variables not defined:</td>
<td>References were made to column variables (&amp;var) that were not defined in the program. The list of variable names follows this message.</td>
</tr>
<tr>
<td>004501</td>
<td>Use <code>+</code> and negate variable for reverse relative placement.</td>
<td>The use of &quot;+#Variable&quot; is not legal here. Negate the #Variable value and use &quot;+#Variable&quot;.</td>
</tr>
<tr>
<td>004503</td>
<td>Fixed line placement #variable must be &gt; 0. Use relative positioning, (+#line,10,0).</td>
<td>Correct the source line as indicated.</td>
</tr>
<tr>
<td>004504</td>
<td>Fixed column placement #variable must be &gt; 0. Use relative positioning, (5,+#col,0).</td>
<td>Correct the source line as indicated.</td>
</tr>
<tr>
<td>004505</td>
<td>Length placement #variable must be &gt;= 0.</td>
<td>The length field cannot be a negative value. Correct the source line.</td>
</tr>
<tr>
<td>004600</td>
<td>CODE not appropriate for numeric data.</td>
<td>The CODE qualifier in PRINT may only be used for text fields. Move &quot;+#Variable&quot; to &quot;$Variable&quot; first and then print &quot;$Variable&quot;.</td>
</tr>
<tr>
<td>004601</td>
<td>Unknown option for GRAPHIC command: BOX, HORZ-LINE, VERT-LINE or FONT</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>004602</td>
<td>GRAPHIC BOX out of bounds. Row: <code>01, Column: </code>02, Width: <code>03, Depth: </code>04</td>
<td>SQR aborts the program run. <code>01 = Row </code>02 = Column <code>03 = Width </code>04 = Depth</td>
</tr>
<tr>
<td>004603</td>
<td>GRAPHIC VERT-LINE out of bounds. Row: <code>01, Column: </code>02, Length: `03</td>
<td>SQR aborts the program run. <code>01 = Row </code>02 = Column `03 = Length</td>
</tr>
<tr>
<td>004604</td>
<td>GRAPHIC HORZ-LINE out of bounds. Row: <code>01, Column: </code>02, Length: `03</td>
<td>SQR aborts the program run. <code>01 = Row </code>02 = Column `03 = Length</td>
</tr>
<tr>
<td>004605</td>
<td>Cannot draw the box; values are out of bounds. Row: <code>01, Column: </code>02, Width: <code>03, Height: </code>04</td>
<td>SQR aborts the program run <code>01 = Row </code>02 = Column <code>03 = Width </code>04 = Height</td>
</tr>
<tr>
<td>004606</td>
<td>Cannot draw the vertical line; values are out of bounds. Row: <code>01, Column: </code>02, Height: `03</td>
<td>SQR aborts the program run <code>01 = Row </code>02 = Column `03 = Height</td>
</tr>
<tr>
<td>004607</td>
<td>Cannot draw the horizontal line; values are out of bounds. Row: <code>01, Column: </code>02, Width: `03</td>
<td>SQR aborts the program run <code>01 = Row </code>02 = Column `03 = Width</td>
</tr>
<tr>
<td>004700</td>
<td>Cannot open the program file: <code>01 (‘02): </code>03</td>
<td>Depends on the system error message. <code>01 = Name of the program file </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>004701</td>
<td>Cannot logon to the database.</td>
<td>Connectivity information is either incorrect or the database server is unavailable. Check connectivity information and the server availability.</td>
</tr>
<tr>
<td>004702</td>
<td>Line found outside paragraph.</td>
<td>All commands must be within BEGIN... END statements. Correct the source code.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>004703</td>
<td>Cannot close the program file. (<code>01): </code>02</td>
<td>Depends on the system error message. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>004704</td>
<td>#ENDIF not found for #IF.</td>
<td>Missing an #ENDIF to complete conditional compilation. Correct the source code.</td>
</tr>
<tr>
<td>004705</td>
<td>Program line too long; maximum is `01.</td>
<td>Break the program line into smaller lines. `01 = Maximum line length supported by this version of SQR</td>
</tr>
<tr>
<td>004706</td>
<td>Substitution variable (<code>01) would cause this line to exceed the maximum line length of </code>02 characters.</td>
<td>The substitution variable value would cause this line to exceed the maximum line size. Break the program line into smaller lines. <code>01 = Name of the substitution variable </code>02 = Maximum line length supported by this version of SQR</td>
</tr>
<tr>
<td>004707</td>
<td>No value found for substitution variable: (`01)</td>
<td>An empty value was found for the substitution variable. Check for a misspelled name. `01 = Name of the substitution variable</td>
</tr>
<tr>
<td>004708</td>
<td>#ELSE without preceding #IF.</td>
<td>Missing an #IF or #IFDEF or #IFNDEF to begin conditional compilation. Correct the source code.</td>
</tr>
<tr>
<td>004709</td>
<td>#ENDIF without preceding #IF.</td>
<td>Missing an #IF or #IFDEF or #IFNDEF to begin conditional compilation. Correct the source code.</td>
</tr>
<tr>
<td>004710</td>
<td>#IF's nested too deeply; maximum is `01.</td>
<td>Reduce the number of nested #IF directives. `01 = The maximum depth supported by this version of SQR</td>
</tr>
<tr>
<td>004711</td>
<td>#INCLUDE files nested too deeply; maximum is `01.</td>
<td>Reduce the number of nested #INCLUDE directives. `01 = The maximum depth supported by this version of SQR</td>
</tr>
<tr>
<td>004712</td>
<td>Include file name too long; Modify -I flag.</td>
<td>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.</td>
</tr>
<tr>
<td>004713</td>
<td>Cannot open the #INCLUDE file: (<code>01) (</code>02): `03</td>
<td><code>01 = Include file name </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>004714</td>
<td>Cannot close the #INCLUDE file: (<code>01) (</code>02): `03</td>
<td><code>01 = Include file name </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>004716</td>
<td>'BEGIN-PROGRAM' command not found in program.</td>
<td>This section is required for all reports. Correct the source code.</td>
</tr>
<tr>
<td>004717</td>
<td>Cannot open the report output file: (<code>01) (</code>02): `03</td>
<td><code>01 = Output file name </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>004719</td>
<td>Cannot logoff the database.</td>
<td>The database server returned an error while trying to log off from the database. SQR aborts the program run.</td>
</tr>
<tr>
<td>004720</td>
<td>Cannot open the run-time file: (<code>01) (</code>02): `03</td>
<td>SQR aborts the program run. <code>01 = Run-Time file name </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>004721</td>
<td>Cannot close the run-time file. (<code>01): </code>02</td>
<td>SQR aborts the program run. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>004722</td>
<td>Error reading the run-time file. (<code>01): </code>02</td>
<td>SQR aborts the program run. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>004723</td>
<td>Run time file must be recreated for this version of SQR.</td>
<td>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.</td>
</tr>
<tr>
<td>004724</td>
<td>The -XL option cannot be specified with this run-time file because access to the database is required.</td>
<td>Do not use the -XL option.</td>
</tr>
<tr>
<td>004725</td>
<td>Cannot open cursor.</td>
<td>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.</td>
</tr>
<tr>
<td>004726</td>
<td>Cannot create procedure for SQL statement.</td>
<td>(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.</td>
</tr>
<tr>
<td>004727</td>
<td>Error writing the run-time file. (<code>01): </code>02</td>
<td><code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>004729</td>
<td>Cannot find inactive database cursor. Program too large.</td>
<td>(DB2, Oracle) The program has too many concurrent database cursors. Reduce the complexity of the program.</td>
</tr>
<tr>
<td>004730</td>
<td>Run-time saved in file: `01</td>
<td>Informational message. `01 = Name of the .sqt file created</td>
</tr>
<tr>
<td>004735</td>
<td>Unknown variable type encountered in run-time file: `01</td>
<td>SQR aborts loading the run-time file. `01 = Variable type</td>
</tr>
<tr>
<td>004736</td>
<td>Unexpected End-Of-File while processing the run-time file.</td>
<td>SQR aborts loading the run-time file.</td>
</tr>
<tr>
<td>004737</td>
<td>Cannot load the run-time file because it was built for the <code>01database and </code>02 is built for the `03 database.</td>
<td>SQR aborts loading the run-time file. <code>01 = Database name from run-time file </code>02 = SQR image name `03 = Database that SQR is built for</td>
</tr>
<tr>
<td>004738</td>
<td>'END-REPORT' not paired with 'BEGIN-REPORT'.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>004739</td>
<td>'END-PROGRAM' not paired with 'BEGIN-PROGRAM'.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>004743</td>
<td>#INCLUDE filename must be enclosed in quotation marks.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>004744</td>
<td>#INCLUDE command format is: #Include 'filename'.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>004747</td>
<td>The SQT file is corrupted and cannot be processed.</td>
<td>SQR aborts loading the run-time file.</td>
</tr>
<tr>
<td>004748</td>
<td>The user function <code>01 needs to be defined as entry </code>02 in the user function table. It requires a definition of: Return Type = <code>03 Arg Count = </code>04 Arg Types = &quot;05&quot;</td>
<td>The SQT file requires that the specified user function be defined. <code>01 = User function name </code>02 = Entry in the user function table <code>03 = Return type </code>04 = Argument count `05 = Argument types</td>
</tr>
</tbody>
</table>
### Table 97 Numbered Messages 005000 to 005999

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>005000</td>
<td>Report <code>01</code> heading section size exceeds the page depth.</td>
<td>Reduce the size of the heading or increase the page depth.</td>
</tr>
<tr>
<td>005001</td>
<td>Report <code>01</code> footing location must be less than the page depth.</td>
<td>Reduce the size of the footing or increase the page depth.</td>
</tr>
<tr>
<td>005002</td>
<td>Check 'BEGIN-HEADING' commands: Discovered 2nd page-initialization while heading in progress.</td>
<td>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.</td>
</tr>
<tr>
<td>005003</td>
<td>Check 'BEGIN-FOOTING' commands; perhaps number of footing lines is too small. Discovered 2nd page-write while footing in progress.</td>
<td>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.</td>
</tr>
<tr>
<td>005004</td>
<td>Attempt to execute the <code>01</code> command while processing the <code>02</code> section.</td>
<td>Change the SQR program logic to prevent the command from executing while the specified section is active. <code>01</code> = Command name <code>02</code> = Section name</td>
</tr>
<tr>
<td>005005</td>
<td>Report <code>01</code> already has been assigned a <code>02</code> section.</td>
<td>Correct the source code. <code>01</code> = Report name <code>02</code> = Duplicated section name</td>
</tr>
<tr>
<td>005006</td>
<td>You cannot define more than one default <code>01</code> section.</td>
<td>Correct the source code. <code>01</code> = Duplicated section name</td>
</tr>
<tr>
<td>005007</td>
<td>Report <code>01</code> has overlapping heading and footing sections.</td>
<td>Correct the source code. <code>01</code> = Report name</td>
</tr>
<tr>
<td>005008</td>
<td>TOC <code>01</code> already has been assigned a <code>02</code> section.</td>
<td>Correct the source code. <code>01</code> = Table of Contents name <code>02</code> = Duplicated section name</td>
</tr>
<tr>
<td>005009</td>
<td>The name can only contain characters [0-9 A-Z _ ].</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>005010</td>
<td>The name cannot be the reserved names 'none' or 'default'.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>005011</td>
<td>This name has already be used.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>005012</td>
<td>The specified <code>01 (</code>02) does not exist.</td>
<td>Correct the source code. <code>01 = Heading or Footing </code>02 = Heading/Footing name</td>
</tr>
<tr>
<td>005013</td>
<td>FOR-REPORTS and FOR-TOC cannot be specified when NAME= is used.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>005014</td>
<td>TOC (`01) has already been defined as the default.</td>
<td>Correct the source code. `01 = Default TOC name</td>
</tr>
<tr>
<td>005100</td>
<td>'IF', 'WHILE', 'EVALUATE' commands nested too deeply; maximum is `01.</td>
<td>Reduce the nested commands. `01 = Maximum depth allowed by this version of SQR</td>
</tr>
<tr>
<td>005101</td>
<td>'BREAK' found outside 'WHILE' or 'EVALUATE' statement.</td>
<td>The BREAK command is valid only in the context of a WHILE or EVALUATE statement. Correct the source code.</td>
</tr>
<tr>
<td>005103</td>
<td>END-WHILE found without matching 'WHILE'.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>005104</td>
<td>'IF' or 'EVALUATE' command not completed before 'END-WHILE'.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>005105</td>
<td>'ELSE' found without matching 'IF'.</td>
<td>ELSE can be used only within the context of an IF command. Correct the source code.</td>
</tr>
<tr>
<td>005106</td>
<td>Single 'ELSE' found inside 'WHILE' or 'EVALUATE' statement.</td>
<td>ELSE can be used only within the context of an IF command. Correct the source code.</td>
</tr>
<tr>
<td>005107</td>
<td>Only one 'ELSE' allowed per 'IF'.</td>
<td>Rewrite the source code to use nested IF statements.</td>
</tr>
<tr>
<td>005108</td>
<td>Found 'END-IF' without matching 'IF'.</td>
<td>Each IF command must have a matching END-IF command. Correct the source code.</td>
</tr>
<tr>
<td>005109</td>
<td>'WHILE' or 'EVALUATE' command not completed before 'END-IF'.</td>
<td>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.</td>
</tr>
<tr>
<td>005110</td>
<td>EVALUATE statements nested too deep; maximum is `01.</td>
<td>Reduce the number of nested statements. `01 = Maximum depth supported by this version of SQR</td>
</tr>
<tr>
<td>005111</td>
<td>'WHEN' found outside 'EVALUATE' clause.</td>
<td>WHEN may be used only in the context of an EVALUATE clause. Correct the source code.</td>
</tr>
<tr>
<td>005112</td>
<td>'IF' or 'WHILE' not completed before 'WHEN' statement.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>005114</td>
<td>Incorrect types for comparison. Both must be of the same type (string, numeric or date).</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>005115</td>
<td>'When-other' found outside 'Evaluate' statement.</td>
<td>WHEN can be used only in the context of an EVALUATE statement. Correct the source code.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>005116</td>
<td>'IF' or 'WHILE' not ended before 'WHEN-OTHER' command.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>005117</td>
<td>Only one 'WHEN-OTHER' allowed per 'EVALUATE'.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>005118</td>
<td>Found 'END-EVALUATE' without matching 'EVALUATE'.</td>
<td>Each EVALUATE command must have a matching END-EVALUATE command. Correct the source code.</td>
</tr>
<tr>
<td>005119</td>
<td>'IF' or 'WHILE' command not completed before 'END-EVALUATE'.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>005120</td>
<td>'WHEN-OTHER' must be after all 'WHEN's.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>005121</td>
<td>No 'WHEN's found inside 'EVALUATE' statement.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>005122</td>
<td>'IF', 'EVALUATE' and 'WHILE' statements cannot cross sections or paragraphs.</td>
<td>These commands must be contained within a single section or paragraph. Correct the source code.</td>
</tr>
<tr>
<td>005123</td>
<td>'CONTINUE' found outside 'WHILE' statement.</td>
<td>The CONTINUE command is valid only in the context of a WHILE statement. Correct the source code.</td>
</tr>
<tr>
<td>005200</td>
<td>Did not find '&gt;' after &lt;....</td>
<td>A leading left angled bracket &quot;&lt;&quot; indicates that you are beginning an ASCII value, which must be ended by a right angled bracket &quot;&gt;&quot;. Correct the source line.</td>
</tr>
<tr>
<td>005201</td>
<td>Bad ascii character in &lt;....&gt;</td>
<td>Numbers in angled brackets &lt;&gt; must be between 1 and 255. Correct the source line.</td>
</tr>
<tr>
<td>005202</td>
<td>Bad ascii number in &lt;....&gt;</td>
<td>Numbers in angled brackets &lt;&gt; must be between 1 and 255. Correct the source line.</td>
</tr>
<tr>
<td>005203</td>
<td>&lt;....&gt; string is too long; maximum is `01 characters.</td>
<td>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</td>
</tr>
<tr>
<td>005300</td>
<td>Did not find <code>=&quot; after qualifier: </code>01</td>
<td>Correct the syntax. `01 = Qualifier name</td>
</tr>
<tr>
<td>005301</td>
<td>Qualifier <code> &quot;01</code> requires a numeric value.</td>
<td>Correct the syntax. `01 = Qualifier name</td>
</tr>
<tr>
<td>005302</td>
<td>Incorrect value for qualifier <code> &quot;01</code>. Valid values are:</td>
<td>Correct the source line. `01 = Qualifier name</td>
</tr>
<tr>
<td>005303</td>
<td>Invalid qualifier <code> &quot;01</code>. Valid qualifiers are:</td>
<td>Correct the source line. `01 = Qualifier name</td>
</tr>
<tr>
<td>005304</td>
<td>Qualifier <code> &quot;01</code> requires a numeric literal, variable, or column.</td>
<td>Correct the source line. `01 = Qualifier name</td>
</tr>
<tr>
<td>005305</td>
<td>Qualifier <code> &quot;01</code> references a numeric variable that has not been defined.</td>
<td>Correct the source line. `01 = Qualifier name</td>
</tr>
<tr>
<td>005306</td>
<td>Qualifier <code> &quot;01</code> requires a string literal, variable, or column.</td>
<td>Correct the source line. `01 = Qualifier name</td>
</tr>
<tr>
<td>005307</td>
<td>List not terminated.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>005308</td>
<td>Missing comma in list.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>005309</td>
<td>Required argument <code> &quot;01</code> was not specified.</td>
<td>Correct the source line. `01 = Qualifier name</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>005310</td>
<td>Qualifier `'01' has already been specified.</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005311</td>
<td>Qualifier `'01' requires a string literal.</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005312</td>
<td>Qualifier `'01' requires a list of values: (val [,val]...).</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005313</td>
<td>Qualifier `'01' requires an integer value.</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005314</td>
<td>Invalid character in variable name `'01'.</td>
<td>Correct the source line. <code>'01</code> = Invalid character</td>
</tr>
<tr>
<td>005315</td>
<td>Qualifier `'01' references a string variable that has not been defined.</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005316</td>
<td>Qualifier `'01' uses an invalid Unit-Of-Measure suffix. Valid suffixes are: dp pt mm cm in</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005317</td>
<td>Qualifier `'01' can only reference string literals or variables.</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005318</td>
<td>Qualifier `'01' can only reference string or numeric literals.</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005319</td>
<td>Qualifier `'01' requires a valid position value.</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005320</td>
<td>Qualifier `'01' is not allowed.</td>
<td>Correct the source line. <code>'01</code> = Qualifier name</td>
</tr>
<tr>
<td>005400</td>
<td>Second page write attempted while writing current page. Check BEFORE-PAGE, AFTER-PAGE procedures.</td>
<td>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.</td>
</tr>
<tr>
<td>005402</td>
<td>String cannot be placed on page: <code>'01</code> -- placement specified is out of range. (<code>02</code>, <code>03</code>, <code>04</code>)</td>
<td>Ensure the values are within the page limits. <code>01</code> = Text value <code>02</code> = Row <code>03</code> = Column <code>04</code> = Length</td>
</tr>
<tr>
<td>005403</td>
<td>Error writing the output file. (<code>01</code>): <code>02</code></td>
<td><code>01</code> = System error code <code>02</code> = System error message</td>
</tr>
<tr>
<td>005404</td>
<td>Cannot open the Postscript startup file: <code>01</code> (<code>02</code>): <code>03</code></td>
<td><code>01</code> = Name of the file <code>02</code> = System error code <code>03</code> = System error message</td>
</tr>
<tr>
<td>005405</td>
<td>SQR trial copy exiting after <code>01</code> pages.</td>
<td><code>01</code> = Number of pages.</td>
</tr>
<tr>
<td>005406</td>
<td>Exiting after requested number of test pages (<code>01</code>).</td>
<td><code>01</code> = Number of pages</td>
</tr>
<tr>
<td>005408</td>
<td>Program stopped by user request.</td>
<td>Informational message.</td>
</tr>
<tr>
<td>005500</td>
<td>Cannot set parse_only option.</td>
<td><code>(Sybase) The DB-Library routine dbsetopt() returned an error. This should never happen. Contact technical support. </code>01` = Stored procedure name</td>
</tr>
<tr>
<td>005501</td>
<td>Cannot reset parse_only option.</td>
<td><code>(Sybase) The DB-Library routine dbclropt() returned an error. This should never happen. Contact technical support. </code>01` = Stored procedure name</td>
</tr>
<tr>
<td>005502</td>
<td>Cannot drop SQR generated stored procedure: <code>01</code>.</td>
<td><code>(Sybase) See the database server error message that was also output. This should never happen. Contact technical support. </code>01` = Stored procedure name</td>
</tr>
<tr>
<td>005503</td>
<td>Cannot use <code>01</code> datatype as bind variable.</td>
<td><code>(Sybase) Use another database column. </code>01` = The database datatype.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>005504</td>
<td>Unknown datatype for bind variable: `01 Cannot create stored procedure.</td>
<td>(Sybase) Please contact technical support. `01 = Unknown database datatype</td>
</tr>
<tr>
<td>005505</td>
<td>SQL too large to create stored procedure.</td>
<td>(Sybase) The size of the SQL text needed to create the stored procedure is too large for SQR to handle. Add –XP to BEGIN-SQL or BEGIN-SELECT.</td>
</tr>
<tr>
<td>005506</td>
<td>SQR’s EXECUTE command not available for this version of Sybase.</td>
<td>(Sybase) Some early versions of Sybase SQL Server or Microsoft SQL Server do not support Remote Procedure Calls (RPCs). Update your database server.</td>
</tr>
<tr>
<td>005507</td>
<td>Could not add param to remote procedure call.</td>
<td>(Sybase) A DB-Library routine returned an unexpected error. See the error message from the database.</td>
</tr>
<tr>
<td>005508</td>
<td>The number of EXECUTE...INTO &amp;columns does not match the procedure.</td>
<td>(Sybase) Check the definition for the stored procedure you are referencing.</td>
</tr>
<tr>
<td>005509</td>
<td>Incorrect number of INTO &amp;columns defined in EXECUTE.</td>
<td>(Sybase) Check the definition for the stored procedure you are referencing.</td>
</tr>
<tr>
<td>005510</td>
<td>Error converting OUTPUT Sybase type for EXECUTE.</td>
<td>(Sybase) The DB-Library routine dbconvert() failed to convert the data from the stored procedure. Contact technical support.</td>
</tr>
<tr>
<td>005511</td>
<td>Number of OUTPUT parameters from EXECUTE is incorrect.</td>
<td>(Sybase) Check the definition for the stored procedure you are referencing.</td>
</tr>
<tr>
<td>005512</td>
<td>Missing default database name for USE.</td>
<td>(Sybase) Correct the syntax.</td>
</tr>
<tr>
<td>005512</td>
<td>Missing default database name for USE.</td>
<td>(ODBC) Could not connect to the specified datasource.</td>
</tr>
<tr>
<td>005513</td>
<td>You may only specify 'USE db' once, before any SQL statements are executed.</td>
<td>(Sybase) Only one USE command is allowed in a report. Place the SETUP section at the beginning of the SQR report.</td>
</tr>
<tr>
<td>005515</td>
<td>Undefined variable referenced in -DB flag: `01</td>
<td>(ODBC) Check for a misspelling. `01 = Variable name</td>
</tr>
<tr>
<td>005523</td>
<td>Database commit failed.</td>
<td>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.</td>
</tr>
<tr>
<td>005524</td>
<td>Cannot close database cursor.</td>
<td>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.</td>
</tr>
<tr>
<td>005528</td>
<td>DB2 SQL <code>01 error </code>02 in cursor `03:</td>
<td>(DB2) <code>01 = Routine name </code>02 = Error code `03 = SQR cursor number</td>
</tr>
<tr>
<td></td>
<td>INFORMIX SQL <code>01 error </code>02 (ISAM: <code>03) in cursor </code>04: `05</td>
<td>(Informix) <code>01 = Routine name </code>02 = Error code <code>03 = ISAM code </code>04 = SQR cursor number `05 = Error message from database</td>
</tr>
<tr>
<td></td>
<td>ODBC SQL <code>01 error </code>02 in cursor <code>03: </code>04</td>
<td>(ODBC) <code>01 = Routine name </code>02 = Error code <code>03 = SQR cursor number </code>04 = Error message from database</td>
</tr>
<tr>
<td></td>
<td>ODBC SQL <code>01 error </code>02 in cursor <code>03: </code>04</td>
<td>(DDO)</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>01 = Routine name  </code>02 = Error code  <code>03 = SQR cursor number  </code>04 = Error message from database</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORACLE <code>01 error </code>02 in cursor <code>03: </code>04</td>
<td>(Oracle) <code>01 = Routine name  </code>02 = Error code  <code>03 = SQR cursor number  </code>04 = Error message from database</td>
<td></td>
</tr>
<tr>
<td>Sybase <code>01 error in cursor </code>02: `03</td>
<td>(Sybase) <code>01 = Routine name  </code>02 = SQR cursor number  `03 = Error message from database</td>
<td></td>
</tr>
<tr>
<td>Teradata SQL <code>01 error </code>02 in cursor `03:</td>
<td>(Teradata) <code>01 = Routine name  </code>02 = Error code  `03 = SQR cursor number</td>
<td></td>
</tr>
<tr>
<td>005532</td>
<td>System 10 files are missing.</td>
<td>(Sybase) Contact your system administrator.</td>
</tr>
<tr>
<td>005533</td>
<td>Not a System 10 SQL Server.</td>
<td>(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.</td>
</tr>
<tr>
<td>005534</td>
<td>SQL too long for PREPARE/DECLARE; maximum `01 characters.</td>
<td>(Teradata) The SQL statement is too large. `01 = Maximum number of characters supported by this version of SQR</td>
</tr>
<tr>
<td>005536</td>
<td>Unknown error message number: `01.</td>
<td>(DB2) `01 = Error message number</td>
</tr>
<tr>
<td>005537</td>
<td>Empty error message returned from system for error number: `01.</td>
<td>(DB2) `01 = Error message number</td>
</tr>
<tr>
<td>005538</td>
<td>Invalid SELECT statement; COMPUTE clauses are not supported.</td>
<td>(Sybase) The select statement contains a COMPUTE clause that is not supported.</td>
</tr>
<tr>
<td>005539</td>
<td>Could not connect to datasource specified in -db variable: `01.</td>
<td>(ODBC) Could not connect to the specified datasource.</td>
</tr>
<tr>
<td>005540</td>
<td>Not connected to a database, database access is not allowed.</td>
<td>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.</td>
</tr>
<tr>
<td>005543</td>
<td>Specify the Oracle DLL name in SQRINI in [Environment:Oracle] section for ORACLE_DLL entry, such as ORACLE_DLL=orant71.dll</td>
<td>(Oracle) SQR was unable to load the Oracle DLL. By default, SQR looks first for <code>ociw32.dll</code> or the DLL specified by the ORACLE_DLL entry in the [Environment:Oracle] section of SQRINI. If that DLL could not be loaded, then SQR attempts to load <code>orant71.dll</code>.</td>
</tr>
<tr>
<td>005600</td>
<td>GETWRD: Word too long; maximum is `01.</td>
<td>Reduce the length of the &quot;word&quot;. `01 = Maximum size of a &quot;word&quot; supported by this version of SQR</td>
</tr>
<tr>
<td>005700</td>
<td>Cannot call SQR recursively.</td>
<td>SQR cannot be called recursively. This error can only occur if a User Function from either UFUNC.C or UCALL.C calls</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the <code>sqr()</code> routine. Do not call <code>sqr()</code> from a UFUNC.C or UCALL.C routine.</td>
</tr>
<tr>
<td>005701</td>
<td>Too many SQR command line arguments; maximum is <code>01</code></td>
<td>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.</td>
</tr>
<tr>
<td>005702</td>
<td>Log file name specified is too long.</td>
<td>Reduce the length of the log file name.</td>
</tr>
<tr>
<td>005703</td>
<td>Error opening the SQR log file: <code>01' (02): </code>03</td>
<td><code>01 = Name of the file </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>005704</td>
<td>Missing program name.</td>
<td>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.</td>
</tr>
<tr>
<td>005705</td>
<td>Program file name specified is too long.</td>
<td>Reduce the length of the program file name.</td>
</tr>
<tr>
<td>005707</td>
<td>Error opening the -E error file: <code>01' (02): </code>03</td>
<td><code>01 = Name of the file </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>005708</td>
<td>Cannot find `01 in SQRDIR, PATH or \SQR.</td>
<td>The specified file cannot be located in any of the directories pointed to by the mentioned environment variables or default directories. Make sure the &quot;file&quot; is present in one of the locations searched. `01 = File name</td>
</tr>
<tr>
<td>005709</td>
<td>`01 environment variable is not defined.</td>
<td>As of version 2.5, the environment variable SQRDIR must be defined. `01 = Name of the environment variable</td>
</tr>
<tr>
<td>005710</td>
<td>`01 path too long.</td>
<td>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</td>
</tr>
<tr>
<td>005711</td>
<td>Bad number in -T test flag.</td>
<td>The number specified must be &gt; zero. Correct the value.</td>
</tr>
<tr>
<td>005712</td>
<td>-G option requires arguments.</td>
<td>(VAX) The command line option is ill-formed. Correct the syntax.</td>
</tr>
<tr>
<td>005713</td>
<td>Too many arguments to -G option; maximum is `01.</td>
<td>(VAX) The command line option is ill-formed. Correct the syntax. `01 = Maximum number of arguments supported by this version of SQR</td>
</tr>
<tr>
<td>005714</td>
<td>-G attribute too long; maximum is `01.</td>
<td>(VAX) The command line option is ill-formed. Correct the syntax. `01 = Maximum number of each attribute supported by this version of SQR</td>
</tr>
<tr>
<td>005716</td>
<td>Unknown flag on command line: `01</td>
<td>Correct the syntax. `01 = Unknown command line flag</td>
</tr>
<tr>
<td>005717</td>
<td>Cannot open channel to TT; status = `01</td>
<td>(VAX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System status</td>
</tr>
<tr>
<td>005718</td>
<td>Cannot read from TT; status = `01</td>
<td>(VAX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System status</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>005719</td>
<td>Cannot close channel to TT; status = `01</td>
<td>(VAX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System status</td>
</tr>
<tr>
<td>005720</td>
<td>Error opening tty. (<code>01): </code>02</td>
<td>(DG, UNIX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>005721</td>
<td>Error with <code>ioctl()</code>. (<code>01): </code>02</td>
<td>(DG, UNIX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>005722</td>
<td>Error reading tty. (<code>01): </code>02</td>
<td>(DG, UNIX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>005723</td>
<td>Error closing tty. (<code>01): </code>02</td>
<td>(DG, UNIX) Should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>005724</td>
<td>Bad number in -B flag.</td>
<td>(Oracle, Sybase) The number specified must be &gt; zero. Correct the value.</td>
</tr>
<tr>
<td>005734</td>
<td>No program name given.</td>
<td>The report name must be the first command line argument.</td>
</tr>
<tr>
<td>005737</td>
<td>Unknown printer type specified with -PRINTER: switch.</td>
<td>The printer type can be EH, HT, LP, HP, PS, or WP. WP is valid only with PC/Windows.</td>
</tr>
<tr>
<td>005738</td>
<td>Database name needs to be included with -DB switch.</td>
<td>(ODBC) Could not connect to the specified datasource.</td>
</tr>
<tr>
<td>005738</td>
<td>Database name needs to be included with -DB switch.</td>
<td>(Sybase) Supply the database name.</td>
</tr>
<tr>
<td>005739</td>
<td>Too many -F switches; maximum is `01.</td>
<td>Reduce the number of -F switches. `01 = Maximum number allowed</td>
</tr>
<tr>
<td>005742</td>
<td>Attempt to invoke viewer (using WinExec) failed; error code = `01</td>
<td>(Windows) `01 = System error code</td>
</tr>
<tr>
<td>005743</td>
<td>Unknown numeric type specified with -DNT: switch.</td>
<td>Correct the command line.</td>
</tr>
<tr>
<td>005744</td>
<td>-DNT:Decimal precision (<code>01) is out of range (</code>02 - `03).</td>
<td>Correct the command line. <code>01 = Specified precision </code>02 = Minimum allowed `03 = Maximum allowed</td>
</tr>
<tr>
<td>005745</td>
<td>The specified default numeric type <code>01 = </code>02 is invalid.</td>
<td>Correct the SQR.INI file entry. <code>01 = Entry </code>02 = Value</td>
</tr>
<tr>
<td>005746</td>
<td>The decimal precision <code>01 = </code>02 is out of range (<code>03 - </code>04).</td>
<td>Correct the SQR.INI file entry. <code>01 = Entry </code>02 = Value <code>03 = Minimum allowed </code>04 = Maximum allowed</td>
</tr>
<tr>
<td>005747</td>
<td>The following error(s) occurred while processing the `01 section from SQR.INI.</td>
<td>See the error message(s) that follow. `01 = Name of the section</td>
</tr>
<tr>
<td>005750</td>
<td>The -Burst switch is not properly formatted.</td>
<td>The “Burst” command line flag is not properly formatted.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>005751</td>
<td>The -Burst switch cannot be used with the -NOLIS switch.</td>
<td>The &quot;Burst&quot; command line flag cannot be specified when the -NOLIS command line flag is also specified.</td>
</tr>
<tr>
<td>005752</td>
<td>The -Burst switch requires either the -Printer:HT or -Printer:EH switch to be specified.</td>
<td>The &quot;Burst&quot; command line flag is applicable only when HTML code is produced. You must specify either the -PRINTER:HT or -PRINTER:EH switch.</td>
</tr>
<tr>
<td>005754</td>
<td>The -Burst switch caused no output to be generated.</td>
<td>The &quot;Burst&quot; command line flag was specified with a set of page ranges that prevented any output to be created. Change the page ranges.</td>
</tr>
<tr>
<td>005755</td>
<td>The -Printer:HT switch does not support UTF-8 encoded data. Use the -Printer:EH switch instead.</td>
<td>Spf_ht.c can't handle UTF-8</td>
</tr>
<tr>
<td>005756</td>
<td>The -EH_FullHTML switch support the following values: 30, 32, and 40.</td>
<td>The 'EH_FullHTML' command line flag is not properly formatted.</td>
</tr>
<tr>
<td>005757</td>
<td>The -EH_Browser switch can be specified with one of the following values: Basic, Netscape, IE, or ALL.</td>
<td>The 'EH_Browser' command line flag is not properly formatted.</td>
</tr>
<tr>
<td>005758</td>
<td>The -EH_Language switch can be specified with one of the following values: English, French, German, Portuguese, Spanish, SCHinese, TChinese, or Japanese.</td>
<td>The 'EH_Language' command line flag is not properly formatted.</td>
</tr>
<tr>
<td>005781</td>
<td>An ASCII-based encoding (ASCII, CP1252, etc) must be specified in order to generate barcodes for HPLaserJet output.</td>
<td>An ASCII-based encoding must be specified for the ENCODING-REPORT-OUTPUT setting in the INI file when generating barcodes for HPLaserJet (-printer:hp)</td>
</tr>
<tr>
<td>005900</td>
<td>Bad number in -`01</td>
<td>(Windows) Specify a valid number. `01 = Command line option</td>
</tr>
<tr>
<td>005901</td>
<td>Bad filename in -`01</td>
<td>(Windows) Specify a valid file name. `01 = Command line option</td>
</tr>
<tr>
<td>005902</td>
<td>Bad directory in -`01</td>
<td>(Windows) Specify a valid directory path. `01 = Command line option</td>
</tr>
<tr>
<td>005903</td>
<td>Cannot access the @ parameter file (<code>01): </code>02</td>
<td>(Windows) Depends on the system error message. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>005904</td>
<td>The argument list is too long; maximum is `01.</td>
<td>(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.</td>
</tr>
<tr>
<td>005905</td>
<td>Cannot open the report file (<code>01): </code>02</td>
<td>(Windows) Depends on the system error message. <code>01 = System error code </code>02 = System error message</td>
</tr>
</tbody>
</table>

Table 98  Numbered Messages  006000 to 006999

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>006000</td>
<td>Error writing the printer file. (<code>01): </code>02</td>
<td>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. <code>01 = System error code </code>02 = System error message</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>006001</td>
<td>Error reading the printer file. (¨01): ¨02</td>
<td>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.</td>
</tr>
<tr>
<td>006002</td>
<td>Cannot open the printer file: ¨01 (¨02): ¨03</td>
<td>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.</td>
</tr>
<tr>
<td>006003</td>
<td>Unexpected End-Of-File while processing the printer file.</td>
<td>Possibly the file got corrupted. Try to recreate the .spf file. If the error persists, contact technical support.</td>
</tr>
<tr>
<td>006004</td>
<td>Encountered unknown SPF code (¨01) while reading the printer file.</td>
<td>Possibly the file got corrupted. Try to recreate the .spf file. If the error persists, contact technical support. ¨01 = Unknown SPF code.</td>
</tr>
<tr>
<td>006100</td>
<td>Duplicate chart (¨01).</td>
<td>Each chart must be given a unique name. ¨01 = Chart name.</td>
</tr>
<tr>
<td>006101</td>
<td>Unknown chart (¨01).</td>
<td>Chart could not be found. ¨01 = Chart name.</td>
</tr>
<tr>
<td>006104</td>
<td>Too many pie segments (¨01). Max is ¨02.</td>
<td>Correct the source code. ¨01 = Number of segments ¨02 = Maximum allowed segments.</td>
</tr>
<tr>
<td>006105</td>
<td>Chart module is not initialized.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006106</td>
<td>XY charts may have only numeric columns.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006107</td>
<td>The 3rd column in the data array must be a character column to specify USE-3RD-DATA-COLUMN.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006120</td>
<td>INTERNAL: Bad chart index from stack (¨01).</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ¨01 = Chart index.</td>
</tr>
<tr>
<td>006122</td>
<td>INTERNAL: Unsupported Grafsman chart type (¨01).</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ¨01 = Chart type.</td>
</tr>
<tr>
<td>006123</td>
<td>INTERNAL: Unsupported pie-explode setting (¨01).</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ¨01 = Setting value.</td>
</tr>
<tr>
<td>006124</td>
<td>INTERNAL: Unsupported tick-mark placement (¨01).</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ¨01 = Placement value.</td>
</tr>
<tr>
<td>006125</td>
<td>Grafsman interface message (¨01) not supported.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support. ¨01 = Message code.</td>
</tr>
<tr>
<td>006126</td>
<td>Unrecognized return code (¨01) from Grafsman command message (¨02).</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support. ¨01 = Return code ¨02 = Message code.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>006127</td>
<td>Cannot fit Chart/Image into the current page. Position: (`01, `02) Size: `03, `04</td>
<td>Correct the source code. Production Reporting aborts the program run. `01 = Row `02 = Column `03 = Width `04 = Depth</td>
</tr>
<tr>
<td>006140</td>
<td>Duplicate image (`01).</td>
<td>Images must be given unique names. `01 = Image name</td>
</tr>
<tr>
<td>006141</td>
<td>Unknown image (`01).</td>
<td>Image name could not be found. `01 = Image name</td>
</tr>
<tr>
<td>006142</td>
<td>Cannot open image file (`01). (`02): `03</td>
<td>`01 = Name of the file `02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>006143</td>
<td>Unknown or missing image type (`01).</td>
<td>Enter a valid image type. `01 = Image type</td>
</tr>
<tr>
<td>006144</td>
<td>Unknown or missing printer type (`01).</td>
<td>Enter a valid Printer type. `01 = Printer type</td>
</tr>
<tr>
<td>006145</td>
<td>Duplicate FOR-PRINTER entries for printer (`01).</td>
<td>Only a single FOR-PRINTER can be specified for a printer type. `01 = Printer type</td>
</tr>
<tr>
<td>006146</td>
<td>The image type (`01) is not supported by printer type (`02).</td>
<td>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</td>
</tr>
<tr>
<td>006147</td>
<td>Invalid number of items in FOR-PRINTER list.</td>
<td>Too few or too many items in the FOR-PRINTER list. Correct the syntax.</td>
</tr>
<tr>
<td>006150</td>
<td>INTERNAL: Bad image index from stack (`01).</td>
<td>Internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Image name</td>
</tr>
<tr>
<td>006200</td>
<td>This report has already been defined.</td>
<td>Each report must be given a unique name.</td>
</tr>
<tr>
<td>006201</td>
<td>This layout has already been defined.</td>
<td>Each layout must be given a unique name.</td>
</tr>
<tr>
<td>006202</td>
<td>This printer has already been defined.</td>
<td>Each printer must be given a unique name.</td>
</tr>
<tr>
<td>006203</td>
<td>The values for ``01'' must be &gt; 0.</td>
<td>Correct the syntax. `01 = Qualifier name</td>
</tr>
<tr>
<td>006204</td>
<td>Qualifiers `01'' and `02'' are mutually exclusive.</td>
<td>Correct the syntax. `01 = Qualifier name `02 = Qualifier name</td>
</tr>
<tr>
<td>006205</td>
<td>Qualifier ``01'' is not applicable with a 'default' printer.</td>
<td>Correct the syntax. `01 = Qualifier name</td>
</tr>
<tr>
<td>006206</td>
<td>The list must contain report names or ALL.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006207</td>
<td>'ALL' must be specified by itself.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006208</td>
<td>No report name was specified.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006209</td>
<td>No layout name was specified.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>006210</td>
<td>No printer name was specified.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006211</td>
<td>The name cannot be 'ALL'.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006212</td>
<td>The name can only contain characters [0-9 A-Z _ -].</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006213</td>
<td>Report <code>01</code> is referenced by multiple <code>02</code> printers.</td>
<td>Correct the syntax. <code>01 = Report name </code>02 = Printer type</td>
</tr>
<tr>
<td>006214</td>
<td>Qualifier <code>01</code> is not allowed with a <code>02</code> printer.</td>
<td>Correct the syntax. <code>01 = Qualifier name </code>02 = Printer type</td>
</tr>
<tr>
<td>006215</td>
<td>The value for <code>01</code> must be <code>02</code> or 0.</td>
<td>Correct the syntax. <code>01 = Qualifier name </code>02 = Relation to zero (&lt;,&lt;=,=,&gt;=,&gt;)</td>
</tr>
<tr>
<td>006216</td>
<td>Report <code>01</code> does not exist.</td>
<td>Correct the syntax. `01 = Report name</td>
</tr>
<tr>
<td>006217</td>
<td>The report name can be a string literal, variable, or column.</td>
<td>Correct the syntax. `01 = Report name</td>
</tr>
<tr>
<td>006218</td>
<td>Referenced layouts not defined:</td>
<td>A list of undefined layouts follows this message.</td>
</tr>
<tr>
<td>006219</td>
<td>Referenced reports not defined:</td>
<td>A list of undefined reports follows this message.</td>
</tr>
<tr>
<td>006220</td>
<td>Referenced printers not defined:</td>
<td>A list of undefined printers follows this message.</td>
</tr>
<tr>
<td>006221</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>006224</td>
<td>No printer type was specified.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006225</td>
<td>Incorrect value for printer type. Valid values are:</td>
<td>Correct the syntax. `01 = Report name</td>
</tr>
<tr>
<td>006226</td>
<td>Attempt to execute the <code>01 command while processing the </code>02 procedure.</td>
<td>SQR aborts the program run. <code>01 = SQR command </code>02 = Procedure name</td>
</tr>
<tr>
<td>006227</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>006228</td>
<td>Referenced TOC (Table Of Contents) not defined:</td>
<td>A list of undefined Table of Contents follows this message.</td>
</tr>
<tr>
<td>006229</td>
<td>This TOC (Table Of Contents) has already been defined.</td>
<td>Each Table of Contents must be given a unique name.</td>
</tr>
<tr>
<td>006230</td>
<td>The list must contain TOC (Table of Contents) names or ALL.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>006231</td>
<td>The TOC (Table Of Contents) entry cannot be positioned given the LEVEL ( <code>01) and INDENTATION ( </code>02) values. The Table of Contents entry will not fit given the specified level and current indentation values. <code>01 = Specified LEVEL= value </code>02 = Current INDENTATION= value</td>
<td></td>
</tr>
<tr>
<td>006232</td>
<td>`01 command not allowed while generating the Table of Contents.</td>
<td>The specified command cannot be used while the Table of Contents is being generated. `01 = SQR command</td>
</tr>
<tr>
<td>006233</td>
<td>The TOC (Table of Contents) entry &quot;A&quot; cannot be processed because the existing entry &quot;B&quot; is positioned below it. A: Line = <code>01, Level = </code>02, Text = <code>03 B: Line = </code>04, Level = <code>05, Text = </code>06 Correct the program logic to eliminate the conflict between the two TOC (Table of Contents) entries. <code>01 = A: Line number </code>02 = A: Level value <code>03 = A: Text value </code>04 = B: Line number <code>05 = B: Level value </code>06 = B: Text value</td>
<td></td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>006303</td>
<td>Parameter (`01) is required, but has not been specified.</td>
<td>Correct the syntax. `01 = Parameter name</td>
</tr>
<tr>
<td>006304</td>
<td>Parameter (`01) already specified.</td>
<td>Correct the syntax. `01 = Parameter name</td>
</tr>
<tr>
<td>006308</td>
<td>Missing part of specification for parameter (`01).</td>
<td>Correct the syntax. `01 = Parameter name</td>
</tr>
<tr>
<td>006309</td>
<td>Parameter (`01) requires literal.</td>
<td>Correct the syntax. `01 = Parameter name</td>
</tr>
<tr>
<td>006352</td>
<td>INTERNAL: Unsupported option/request (`01) in (`02).</td>
<td>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</td>
</tr>
<tr>
<td>006400</td>
<td>Unsupported background color.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006401</td>
<td>Unsupported border color.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006402</td>
<td>Border width out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006403</td>
<td>X position out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006404</td>
<td>Y position out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006405</td>
<td>X size out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006406</td>
<td>Y size out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006407</td>
<td>Unsupported font.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006408</td>
<td>Unsupported font style.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006409</td>
<td>Unsupported font color.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006410</td>
<td>Unsupported horizontal text justification value.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006411</td>
<td>Unsupported vertical text justification value.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006412</td>
<td>Unsupported font path.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006413</td>
<td>Unsupported font rotation.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006414</td>
<td>Font size out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>006415</td>
<td>Text line id# out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006416</td>
<td>Unsupported chart type.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006417</td>
<td>Unsupported chart sub-type.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006418</td>
<td>Unsupported chart orientation (not H or V).</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006419</td>
<td>Unsupported perspective (not 2D or 3D).</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006420</td>
<td>Unsupported axis (not X or Y).</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006421</td>
<td>Unsupported axis label data type.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006422</td>
<td>Dataset id# out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006423</td>
<td>Unsupported dataset type.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006424</td>
<td>Unsupported dataset color.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006425</td>
<td>Unsupported dataset line style.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006426</td>
<td>Unsupported dataset fill pattern.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006427</td>
<td>Unsupported dataset marker.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006428</td>
<td>Chart type does not support Y-axis datasets.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006429</td>
<td>Pie-chart segment id# is out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006430</td>
<td>Unsupported pie-segment color.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006431</td>
<td>Unsupported pie-segment border color.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006432</td>
<td>Unsupported pie-segment pattern.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006433</td>
<td>Unsupported pie-segment explode setting.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>006434</td>
<td>Command only valid for charts of type 'pie'.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006435</td>
<td>Pie-chart radius out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006436</td>
<td>Pie-chart starting angle out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006437</td>
<td>Unsupported pie-chart fill direction. Must be clockwise or counter-clockwise.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006438</td>
<td>Unsupported pie-segment label position.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006439</td>
<td>Unsupported pie-segment quantity display position.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006440</td>
<td>Unsupported pie-segment per-cent display position.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006441</td>
<td>Unsupported legend style.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006442</td>
<td>Unsupported legend horizontal position.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006443</td>
<td>Unsupported legend vertical position.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006444</td>
<td>Text charts do not support legend.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006445</td>
<td>Number of datasets specified does not match data.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006446</td>
<td>Unsupported axis label position.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006447</td>
<td>Unsupported axis type (not LINEAR or LOG).</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006448</td>
<td>Pie and text charts do not support axis control.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006449</td>
<td>Unsupported axis min scaling.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006450</td>
<td>Unsupported axis max scaling.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006451</td>
<td>Unsupported axis max scaling.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006452</td>
<td>Beginning of tickmarks is after end.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>006453</td>
<td>Unsupported tickmark type.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006454</td>
<td>Unsupported grid type.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006455</td>
<td>Unsupported grid color.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006456</td>
<td>Grid line width out of range.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006457</td>
<td>Unable to open grafcap file.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006458</td>
<td>Unsupported grafcap device.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006459</td>
<td>Error in grafcap entry specification.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006460</td>
<td>Unable to open chart output destination.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006461</td>
<td>Internal error during ggDraw.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006462</td>
<td>Improper parameters passed to gscale.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006463</td>
<td>The shared library specified in the grafcap file could not be found.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006464</td>
<td>A function called from the shared library specified in the grafcap file could not be found.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006500</td>
<td>The bar code could not be positioned on the page. Row: <code>01, Column: </code>02, Height: `03</td>
<td>Correct the source code. <code>01 = Row </code>02 = Column `03 = Height</td>
</tr>
<tr>
<td>006501</td>
<td>Unknown BCL error (`01) encountered.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = BCL error code</td>
</tr>
<tr>
<td>006502</td>
<td>Invalid bar code type (`01): Valid values are from 1 to 15.</td>
<td>Correct the source code. `01 = Bar code type.</td>
</tr>
<tr>
<td>006503</td>
<td>The length of the bar code text '01' must be between 1 and 30 characters.</td>
<td>Correct the source code. `01 = Bar code text</td>
</tr>
<tr>
<td>006504</td>
<td>The length of the caption text '01' must be between 1 and 30 characters.</td>
<td>Correct the source code. `01 = Caption text</td>
</tr>
<tr>
<td>006505</td>
<td>Invalid printer type (`01): Valid values are from 0 to 13.</td>
<td>Correct the source code. `01 = Printer type</td>
</tr>
<tr>
<td>006506</td>
<td>Invalid offset: Valid values are from 0 to 250.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>006507</td>
<td>Invalid height (`01): Valid values are from 0.1 to 2.0 inches.</td>
<td>Correct the source code. `01 = Height</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>006508</td>
<td>Invalid checksum: Valid values are from 0 to 2.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>006509</td>
<td>Invalid pass: Valid values are from 1 to 6.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>006510</td>
<td>The bar code text '01' is not valid for the type of bar code ('02) selected.</td>
<td>Correct the source code. 01 = Bar code text 02 = Bar code type</td>
</tr>
<tr>
<td>006511</td>
<td>Internal error: Could not generate the bar code.</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006512</td>
<td>Internal error: Bar code buffer required too large (&gt;32K).</td>
<td>Should never occur during normal operations. Record the steps leading up to the error and contact technical support.</td>
</tr>
<tr>
<td>006601</td>
<td>Cannot allocate the device context for the default printer.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006602</td>
<td>Failed to start printing the document.</td>
<td>(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.</td>
</tr>
<tr>
<td>006603</td>
<td>New-page (start) failed on page <code>01</code>.</td>
<td>(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</td>
</tr>
<tr>
<td>006604</td>
<td>New-page (end) failed on page <code>01</code>.</td>
<td>(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</td>
</tr>
<tr>
<td>006605</td>
<td>End document failed.</td>
<td>(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.</td>
</tr>
<tr>
<td>006606</td>
<td>Error reading font information from the [Fonts] section in SQR.INI. Using the default font.</td>
<td>(Windows) Correct the [Fonts] section in SQR.INI.</td>
</tr>
<tr>
<td>006607</td>
<td>Failed to create a brush for shading.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006608</td>
<td>Failed to select font <code>01</code>.</td>
<td>(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</td>
</tr>
<tr>
<td>006609</td>
<td>Failed to modify font <code>01</code>.</td>
<td>(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</td>
</tr>
<tr>
<td>006610</td>
<td>Failed to create a pen that was required to draw a box.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006612</td>
<td>Failed to create a pen that was required to draw a vertical line.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>006613</td>
<td>Failed to open the image bitmap file (<code>01). (</code>02): `03</td>
<td>(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. <code>01 = Name of the file </code>02 = System error code `03 = System error message</td>
</tr>
<tr>
<td>006614</td>
<td>The file (`01) does not contain a valid bitmap.</td>
<td>(Windows) Specify a valid bitmap file. `01 = Name of the file</td>
</tr>
<tr>
<td>006615</td>
<td>Failed to create the palette for image (`01).</td>
<td>(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</td>
</tr>
<tr>
<td>006616</td>
<td>Failed to load RLE into memory for image (`01).</td>
<td>(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</td>
</tr>
<tr>
<td>006617</td>
<td>Failed to convert DIB to DDB for image (`01).</td>
<td>(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</td>
</tr>
<tr>
<td>006618</td>
<td>Failed to draw the bitmap image (`01).</td>
<td>(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</td>
</tr>
<tr>
<td>006619</td>
<td>Cannot access the default printer's driver.</td>
<td>(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.</td>
</tr>
<tr>
<td>006620</td>
<td>Cannot select the charting clip area onto the printers DC.</td>
<td>(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.</td>
</tr>
<tr>
<td>006621</td>
<td>Cannot select create a metafile required for business graphics.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006622</td>
<td>Cannot create a region required for business graphics.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006623</td>
<td>Cannot create a DC required for business graphics.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006624</td>
<td>Cannot create a bitmap required for business graphics.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006625</td>
<td>Business graphics failed while setting up the device (ggWinDevice).</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006626</td>
<td>Cannot draw business graphics.</td>
<td>(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.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>006700</td>
<td>SQRDIR is not defined.</td>
<td>Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006701</td>
<td>Could not allocate memory while attempting to register the .spf filename extension.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006704</td>
<td>Cannot open or read file (`01) (`02): `03</td>
<td>(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</td>
</tr>
<tr>
<td>006705</td>
<td>File (`01) is not in SPF packet format.</td>
<td>(Windows) The file was not produced by SQR or it has been corrupted. `01 = Name of the file</td>
</tr>
<tr>
<td>006706</td>
<td>Failed to identify the start of the report (`01).</td>
<td>(Windows) The file was not produced by SQR or it has been corrupted. `01 = Name of the file</td>
</tr>
<tr>
<td>006707</td>
<td>An invalid seek was made for page `01.</td>
<td>(Windows) Internal error which should not occur under normal operations. Contact technical support. `01 = Page number</td>
</tr>
<tr>
<td>006708</td>
<td>Too many errors were encountered while processing the file. Processing has been stopped.</td>
<td>(Windows) Can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.</td>
</tr>
<tr>
<td>006709</td>
<td>Failed to open the image bitmap file (`01), (`02): `03 This message is displayed only once per SPF file.</td>
<td>(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</td>
</tr>
<tr>
<td>006800</td>
<td>`01: Detected internal program error.</td>
<td>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</td>
</tr>
<tr>
<td>006801</td>
<td>`01: Null Operand Passed as input.</td>
<td>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</td>
</tr>
<tr>
<td>006802</td>
<td>`01: Decimal Exponent Under/Overflow.</td>
<td>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</td>
</tr>
<tr>
<td>006803</td>
<td>`01: Decimal to Integer Conversion Under/Overflow.</td>
<td>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</td>
</tr>
</tbody>
</table>
| 006805       | \`01: Decimal Precision Under/Overflow. | Decimal Precision Under/Overflow: Attempt made to initialize decimal object with an invalid precision. Check the
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>006806</td>
<td>`01: String to Decimal Object Conversion Error.</td>
<td>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</td>
</tr>
<tr>
<td>006807</td>
<td>`01: Truncation/Rounding Error - Outside Valid Range for Decimal Object.</td>
<td>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</td>
</tr>
<tr>
<td>006808</td>
<td>`01: Decimal Error: Cannot Divide by Zero.</td>
<td>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</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>007000</td>
<td>The locale <code>01</code> is not defined in SQR.INI.</td>
<td>Check for a misspelled locale name and/or the SQR.INI file. `01 = Locale name</td>
</tr>
<tr>
<td>007001</td>
<td>At least one qualifier must be specified.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007002</td>
<td>The value for <code>01</code> must be a list of <code>02</code> string literals, variables or columns.</td>
<td>Correct the source code. <code>01 = Qualifier </code>02 = Number of entities in list</td>
</tr>
<tr>
<td>007003</td>
<td>The values for <code>01</code> and <code>02</code> cannot be the same.</td>
<td>Correct the source code. <code>01 = Qualifier </code>02 = Qualifier</td>
</tr>
<tr>
<td>007004</td>
<td>The value for <code>01</code> (<code>02</code>) must be a single character which is not in the list: &quot;03&quot;.</td>
<td>Correct the source code. <code>01 = Qualifier </code>02 = Value `03 = List of invalid characters</td>
</tr>
<tr>
<td>007005</td>
<td>The value for <code>01</code> (<code>02</code>) is invalid. Valid values are:</td>
<td>Correct the source code. <code>01 = Qualifier </code>02 = Value</td>
</tr>
<tr>
<td>007006</td>
<td>The last character of the <code>01</code> value (<code>02</code>) cannot be a digit or the minus sign or the same as either of the separators.</td>
<td>Correct the source code. <code>01 = Qualifier </code>02 = Invalid character</td>
</tr>
<tr>
<td>007007</td>
<td>The first character of the <code>01</code> value (<code>02</code>) cannot be a digit or the minus sign or the same as either of the separators.</td>
<td>Correct the source code. <code>01 = Qualifier </code>02 = Invalid character</td>
</tr>
<tr>
<td>007008</td>
<td>The following errors occurred while processing the (`01) locale from SQR.INI.</td>
<td>This message precedes error messages encountered while processing the SQR.INI file. `01 = Locale name</td>
</tr>
<tr>
<td>007009</td>
<td>The value for <code>01</code> cannot be 'DEFAULT' or 'SYSTEM'.</td>
<td>Correct the syntax. `01 = Qualifier</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>007010</td>
<td>The value for <code>01</code> (<code>02</code>) is not properly formatted: Did not find the '&gt;' for the <code>&lt;nmm&gt;</code> construct.</td>
<td>Correct the syntax. <code>01 = Qualifier </code>02 = Value</td>
</tr>
<tr>
<td>007011</td>
<td>The value for <code>01</code> (<code>02</code>) is not properly formatted: The value of an <code>&lt;nmm&gt;</code> construct must be from 1 to 255.</td>
<td>Correct the syntax. <code>01 = Qualifier </code>02 = Value</td>
</tr>
<tr>
<td>007012</td>
<td>The default locale (<code>01</code>) specified in the [`02] section of SQR.INI has not been defined.</td>
<td>Correct the syntax. <code>01 = Locale name </code>02 = Section name</td>
</tr>
<tr>
<td>007013</td>
<td>The value for <code>01</code> (<code>02</code>) must be a list of `03 quoted string literals.</td>
<td>Correct the syntax. <code>01 = Qualifier </code>02 = Value `03 = Number of entities in list</td>
</tr>
<tr>
<td>007014</td>
<td>The entry (<code>01 = </code>02) is not valid.</td>
<td>Correct the SQR.INI entry. <code>01 = Qualifier from the SQR.INI file </code>02 = Qualifier's value</td>
</tr>
<tr>
<td>007100</td>
<td>The use of an edit mask or the keywords NUMBER, MONEY, or DATE is not legal when storing numeric variables.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007101</td>
<td>The last keyword is not <code>01</code>.</td>
<td>Correct the source code. `01 = Keyword</td>
</tr>
<tr>
<td>007102</td>
<td>Incompatible source and destination variable types.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007103</td>
<td>The keyword (<code>01) is not compatible with the variable (</code>02).</td>
<td>Correct the source code. <code>01 = Keyword </code>02 = Variable name</td>
</tr>
<tr>
<td>007104</td>
<td>The use of an edit mask or the keyword DATE is not legal if both variables are date variables.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007200</td>
<td>The specified precision (<code>01) is out of range (</code>02 - `03).</td>
<td>Correct the source code. <code>01 = Specified precision </code>02 = Minimum precision `03 = Maximum precision</td>
</tr>
<tr>
<td>007201</td>
<td>The precision is specified by a value from <code>01 to </code>02 surrounded by parentheses.</td>
<td>Correct the source code. <code>01 = Minimum precision </code>02 = Maximum precision</td>
</tr>
<tr>
<td>007202</td>
<td>Variable (`01) is not a decimal variable and cannot have a precision associated with it.</td>
<td>Correct the source code. `01 = Variable name</td>
</tr>
<tr>
<td>007203</td>
<td>A string variable name is required here.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007204</td>
<td>A numeric variable name is required here.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007205</td>
<td>The variable (<code>01) has already been defined as </code>02` and may not be redefined.</td>
<td>Correct the source code. <code>01 = Variable name </code>02 = Variable type</td>
</tr>
<tr>
<td>007206</td>
<td>The variable type has not been specified.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007207</td>
<td>This command is only allowed within local procedures.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007208</td>
<td>This command must be before all other commands in the procedure.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007209</td>
<td>Only string ($) and numeric (#) variables may be declared.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>007210</td>
<td>Invalid variable name specified.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007211</td>
<td>You cannot declare a global variable from within a procedure.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007400</td>
<td>The specified character is invalid in the current character set.</td>
<td>Correct the program logic.</td>
</tr>
<tr>
<td>007401</td>
<td><code>01</code> is not a valid value for the ENCODING environment variable.</td>
<td>The specified encoding scheme is not known by SQR. <code>01</code> = ENCODING environment variable setting.</td>
</tr>
<tr>
<td>007403</td>
<td>The Double-Byte SQR command <code>01</code> is not supported in this version of SQR.</td>
<td>The SQT file contains a reference to an SQR command, which is not supported by this version of SQR. <code>01</code> = SQR command name</td>
</tr>
<tr>
<td>007405</td>
<td>The barcode text <code>01</code> cannot contain double-byte characters.</td>
<td>Correct the source code. <code>01</code> = Bar code text</td>
</tr>
<tr>
<td>007501</td>
<td>Using <code>01</code> edit mask from (<code>02) against (</code>03)</td>
<td>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. <code>01</code> = Edit mask element <code>02</code> = Edit mask being used <code>03</code> = Value being applied to the edit mask</td>
</tr>
<tr>
<td>007601</td>
<td>Cannot access the Java file (<code>01) (</code>02): `03</td>
<td>SQR cannot access the required file. <code>01</code> = Name of the file <code>02</code> = System error code <code>03</code> = System error message</td>
</tr>
<tr>
<td>007602</td>
<td>-EH_Scale: value (<code>01) is out of range (</code>02 - `03).</td>
<td>Correct the command line. <code>01</code> = Specified scale <code>02</code> = Minimum allowed <code>03</code> = Maximum allowed</td>
</tr>
<tr>
<td>007603</td>
<td>-Printer:EH functionality is not available on this platform.</td>
<td>Enhanced HTML functionality is not available on this platform.</td>
</tr>
<tr>
<td>007604</td>
<td>-Printer:PD functionality is not available on this platform.</td>
<td>PDF functionality is not available on this platform.</td>
</tr>
<tr>
<td>007605</td>
<td>Cannot support Unicode internally. Please reset the UseUnicodeInternal setting in SQR.INI to FALSE.</td>
<td>Cannot support Unicode internally. Reset the UseUnicodeInternal setting in SQR.INI to FALSE.</td>
</tr>
<tr>
<td>007702</td>
<td>Invalid entry for keyword, <code>01= </code>02'</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007703</td>
<td>May only specify either PROCEDURE=, or COMMAND=, or GETDATA=, exclusive.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007704</td>
<td>Must specify a SCHEMA.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007705</td>
<td>Must specify either a PROCEDURE, COMMAND, or GETDATA.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007706</td>
<td>CONNECTION <code>01</code> not found. No such connection.</td>
<td>Correct the source code.</td>
</tr>
<tr>
<td>007707</td>
<td>The returned set of Procedure parameters (INOUT and OUT) (length `01 items) did not include one or more of the specified items.</td>
<td>Stored procedure error.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>007708</td>
<td>Encountered a parameter of type '01'. Valid types are either IN, OUT, or INOUT. If no type is entered, the type defaults to IN.</td>
<td>Stored procedure error.</td>
</tr>
<tr>
<td>007709</td>
<td>The datasource failed to provide the expected return status value. Verify the query metadata.</td>
<td>Datasource error.</td>
</tr>
<tr>
<td>007711</td>
<td>Failed to login to the requested datasource (Connection='01', username='02'). DETAILS: '03</td>
<td>Logon failed.</td>
</tr>
<tr>
<td>007712</td>
<td>The requested rowset ('01) was not available. Verify the query metadata.</td>
<td>Not enough rowsets.</td>
</tr>
<tr>
<td>007713</td>
<td>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.</td>
<td>Incorrect environment setup.</td>
</tr>
<tr>
<td>007714</td>
<td>The datasource ('01) does not support the requested capability ('02'). Check the capabilities list for the datasource, located in the Properties folder.</td>
<td>Invalid query for datasource.</td>
</tr>
<tr>
<td>007715</td>
<td>Failed to start the Java Virtual Machine (JVM). Possible causes are: missing or invalid jdk files, incorrect CLASSPATH, or insufficient resources.</td>
<td>Incorrect environment setup.</td>
</tr>
<tr>
<td>007717</td>
<td>The query failed. DETAILS: `01</td>
<td>Query failed.</td>
</tr>
<tr>
<td>007718</td>
<td>Failure setting property <code>01. DETAILS: </code>02</td>
<td>Property-set failed.</td>
</tr>
<tr>
<td>007721</td>
<td>Parameter <code>01 (</code>02) was passed to the PROCEDURE as data type <code>03; expected (</code>04) type `05. Verify the query metadata.</td>
<td>A failure occurred during row fetch.</td>
</tr>
<tr>
<td>007722</td>
<td>Invalid query parameter. Reason: `01</td>
<td>Bad procedure parameter.</td>
</tr>
<tr>
<td>007723</td>
<td>Too many parameters (= `01) were supplied to the query. Verify the query metadata.</td>
<td>Bad procedure parameter.</td>
</tr>
<tr>
<td>007724</td>
<td>Parameter <code>01 (</code>02) was passed to the PROCEDURE as type <code>03; expected type </code>04. Verify the query metadata.</td>
<td>Bad procedure parameter.</td>
</tr>
<tr>
<td>007725</td>
<td>Parameter <code>01 (</code>02', JDO-type `03), specified 'NULL', is a required-parameter. Specify a value or variable name.</td>
<td>Bad procedure parameter.</td>
</tr>
<tr>
<td>007727</td>
<td>Unable to retrieve metadata for Procedure=<code>01, Schema=</code>02. DETAILS: `03</td>
<td>Metadata check failed.</td>
</tr>
<tr>
<td>007728</td>
<td>Parameter list type mismatch (# <code>01, SQR type = </code>02). The datasource expected a parameter of type `03. Verify the query metadata.</td>
<td>Parameter list mismatch.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>007729</td>
<td>List size mismatch detected while fetching data of type ROW, <code>01 items, into SQR list-variable, </code>02 items. Fetching will proceed to the smaller size.</td>
<td>List size mismatch.</td>
</tr>
<tr>
<td>007730</td>
<td>Incorrect syntax for BEGIN-SELECT ... FROM. Options are: FROM ROWSETS=... FROM PARAMETER= $strvar</td>
<td>Bad begin-select syntax.</td>
</tr>
<tr>
<td>007732</td>
<td>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.</td>
<td>Bad proc parameter.</td>
</tr>
<tr>
<td>007733</td>
<td>The list of keywords entered to the PARAMETERS keyword must be terminated with a semicolon.</td>
<td>Bad proc parameter. Correct the source code.</td>
</tr>
<tr>
<td>007734</td>
<td>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.</td>
<td>Bad proc parameter. Correct the source code.</td>
</tr>
<tr>
<td>007735</td>
<td>Missing one or more DDO (fname) .jar files. Verify the location of the original-installation files, and that they are accessible. Error code: <code>01. Classpath: </code>02.</td>
<td>Bad environment.</td>
</tr>
<tr>
<td>007736</td>
<td>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.</td>
<td>Bad environment.</td>
</tr>
<tr>
<td>007738</td>
<td>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=<code> 01', Proc=</code> 02'.</td>
<td>Bad environment.</td>
</tr>
<tr>
<td>007739</td>
<td>Unable to locate query object <code>01 in the specified schema (</code>02). DETAILS: `03.</td>
<td>Bad environment.</td>
</tr>
<tr>
<td>007740</td>
<td>Invalid &amp;pseudonym or 'TYPE=' data-type specified for a begin-select column-variable. Valid types are: CHAR, TEXT, DATE, NUMBER, BINARY.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>007741</td>
<td>Illegal attempt to fetch a non-scalar field into a column variable. Correct the query.</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>007742</td>
<td>The output parameter specified in 'Begin-Select ... From Parameter = <code>01' is not available. Available parameters: </code>02.</td>
<td>Bad command.</td>
</tr>
<tr>
<td>007743</td>
<td>The output parameter specified in 'Begin-Select ... From Parameter = `01' is not of type ROWSET. Verify the query metadata.</td>
<td>Bad command.</td>
</tr>
<tr>
<td>007744</td>
<td>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.</td>
<td>Bad var assignment.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>007745</td>
<td>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.</td>
<td>Bad var assignment.</td>
</tr>
<tr>
<td>007746</td>
<td>Failed to locate the requested Rowset ('01) while processing the query. The last available Rowset number is `02. Verify the query metadata.</td>
<td>Not enough RowSets.</td>
</tr>
<tr>
<td>007747</td>
<td>The query raised a DDO exception. DETAILS: `01.</td>
<td>Bad query.</td>
</tr>
<tr>
<td>007748</td>
<td>A BEGIN-SELECT paragraph was coded, but the query returned no Rows.</td>
<td>No data warning.</td>
</tr>
<tr>
<td>007749</td>
<td>Invalid syntax for PARAMETERS=(...) statement. Use: PARAMETERS=( %v</td>
<td>$v</td>
</tr>
<tr>
<td>007750</td>
<td>FATAL: Failure creating Java object.</td>
<td>General failure.</td>
</tr>
<tr>
<td>007751</td>
<td>Attempt to create a List variable of size greater than the maximum size of `01 items.</td>
<td>General failure.</td>
</tr>
<tr>
<td>007753</td>
<td>Attempt to access List-row (<code>01) beyond the List size (</code>02 rows).</td>
<td>Bad list assignment/setup.</td>
</tr>
<tr>
<td>007754</td>
<td>Attempt to assign/modify a List row is not compatible with the List definition.</td>
<td>Bad list assignment/setup.</td>
</tr>
<tr>
<td>007755</td>
<td>Attempt to assign a row to a non-existant List variable. Define the List first, using the syntax: let %lname[size] = list(NUMBER</td>
<td>DATE</td>
</tr>
<tr>
<td>007756</td>
<td>Incorrect syntax for List-variable reference. Use: let [$</td>
<td>#]var = %listname$nlit</td>
</tr>
<tr>
<td>007757</td>
<td>Alter-connection statement missing 'DSN='...'.</td>
<td>Improper alter-conn.</td>
</tr>
<tr>
<td>007758</td>
<td>List-definition size specifier must be literal.</td>
<td>Improper alter-conn.</td>
</tr>
<tr>
<td>007759</td>
<td>Attempt to access a non-existent List-column ('01').</td>
<td>No such list column name.</td>
</tr>
<tr>
<td>007760</td>
<td>Must specify one of the keywords, FROM-ROWSETS or FROM_PARAMETER.</td>
<td>Incorrect syntax for Load-lookup.</td>
</tr>
<tr>
<td>007761</td>
<td>Incorrect syntax to Load-lookup 'PARAMETERS=' keyword. Use: PARAMETERS=(slit</td>
<td>nlit</td>
</tr>
<tr>
<td>007762</td>
<td>Too many parameters ( <code>02) entered to Load-Lookup command. Max parameters is </code>01.</td>
<td>Incorrect syntax for Load-lookup.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>007763</td>
<td>Problem executing the cursor for LOAD-LOOKUP table <code>\01</code>. DETAILS: <code>\02</code>.</td>
<td>The database server returned an error while trying to execute the SQL statement needed to process the LOAD-LOOKUP command. <code>\01</code> = Load lookup table name</td>
</tr>
<tr>
<td>007764</td>
<td>Bad return fetching row from database in LOAD-LOOKUP table <code>\01</code>. DETAILS: <code>\02</code></td>
<td>The database server returned an error while fetching the data. <code>\01</code> = Load lookup table name</td>
</tr>
<tr>
<td>007765</td>
<td>DC, DI sort options not supported with this SQR version. To sort, use SORT=SC or SORT=SI.</td>
<td>Database sort not supported for Load-Lookup with DDO.</td>
</tr>
<tr>
<td>007766</td>
<td>Must specify a query keyword; PROCEDURE=, COMMAND= or GETDATA=.</td>
<td>Incorrect syntax for Load-lookup. Specify a keyword representing the query.</td>
</tr>
<tr>
<td>007767</td>
<td>Unknown column variable type.</td>
<td>Unknown data type returned by the server.</td>
</tr>
<tr>
<td>007768</td>
<td>The property <code>\01</code> was not found in the property sheet for the specified datasource (<code>\02</code>). Available property names are: <code>\03</code>. The datasource property sheet does not include the named property.</td>
<td>Verify the metadata and correct the syntax.</td>
</tr>
<tr>
<td>007771</td>
<td>Did not find value after <code>\01 =</code></td>
<td>The code specified a Connection = keyword, but no matching literal. Correct the syntax.</td>
</tr>
<tr>
<td>007774</td>
<td>Invalid attempt to establish a second connection to datasource <code>\02</code>, using Connection <code>\01</code>. The Connection <code>\01</code> is declared to allow only one active login (no-duplicate=TRUE).</td>
<td>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</td>
</tr>
<tr>
<td>007775</td>
<td>Bad value (<code>\01</code>) for Alter-connection keyword (<code>\02</code>). Valid values are: <code>\03</code>.</td>
<td>Bad keyword value. Refer to the language reference and correct the syntax.</td>
</tr>
<tr>
<td>007778</td>
<td>Datasource login not available.</td>
<td>Connection non-existent. Correct the source. Possible causes: No BEGIN-EXECUTE statement.</td>
</tr>
<tr>
<td>007779</td>
<td>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 <code>type=&lt;datatype&gt;</code>'.</td>
<td>Can't verify colvar types with variable entry; use literals or define column types. Correct the source.</td>
</tr>
<tr>
<td>007780</td>
<td>Unable to verify ResultSet column types. Must specify column types using `type=CHAR</td>
<td>NUMBER</td>
</tr>
<tr>
<td>007781</td>
<td>Unable to log onto datasource to obtain query metadata. Specify the Connection for this query using a complete Declare-Connection statement, or specify type=&lt;datatype&gt; for each column variable in the Begin-Select.</td>
<td>Must declare a complete connection if use early binding of select column variables.</td>
</tr>
<tr>
<td>007783</td>
<td>Could not execute SQL. DETAILS: <code>\01</code></td>
<td>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.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>007784</td>
<td>Bad CONNECTION specification (&quot;01&quot;). Possible causes: Syntax error, Dimension name not found, Dimension attributes not found, Dimension name not found in Begin-Select list. DETAILS: `02</td>
<td>The OLAP-related members of the named CONNECTION could not be processed, either due to syntax or no such name.</td>
</tr>
<tr>
<td>007785</td>
<td>The column specified (&quot;01&quot;) is ambiguous. It appears more than once in the data.</td>
<td>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</td>
</tr>
<tr>
<td>007786</td>
<td>Column (&quot;01&quot;) not found.</td>
<td>A column specified in the query was not found in the data. `01 = The column name</td>
</tr>
<tr>
<td>007787</td>
<td>Unsupported datatype (&quot;02&quot;) found in column (&quot;01&quot;). Only Text, Numeric, and Date are currently supported.</td>
<td>A column specified in the query contained values which are not currently supported. Change the query to use a different column. <code>01 = The column name </code>02 = The unsupported datatype</td>
</tr>
<tr>
<td>007789</td>
<td>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.</td>
<td>The columns specified in the query are found in different branches of a hierarchical 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.</td>
</tr>
<tr>
<td>007790</td>
<td>Must specify PROCEDURE= to use PARAMETERS= in BEGIN-EXECUTE.</td>
<td>Parameters may only be specified for PROCEDURE queries, not for COMMAND or GETDATA queries. Remove the PARAMETERS= line, or specify PROCEDURE= instead.</td>
</tr>
<tr>
<td>007791</td>
<td>Unable to locate one or more JAVA classes in the DDO JAR file.</td>
<td>Verify that the original installation files are not corrupted.</td>
</tr>
<tr>
<td>007792</td>
<td>Unable to locate one or more JAVA methods in the DDO JAR file.</td>
<td>Verify that the original installation files are not corrupted.</td>
</tr>
<tr>
<td>007793</td>
<td>An incompatible version of the DDO JAR file was found.</td>
<td>Verify that the original installation files are not corrupted.</td>
</tr>
</tbody>
</table>

Table 100  Numbered Messages  008000 to 009999

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>008000</td>
<td>Delay not appropriate for database columns or literals.</td>
<td>The DELAY argument to the PRINT command can only be used with SQR #variables or $variables.</td>
</tr>
<tr>
<td>008001</td>
<td>The width must also be specified when DELAY is used.</td>
<td>The DELAY argument to the PRINT command requires that the width argument be specified.</td>
</tr>
<tr>
<td>008003</td>
<td>The SET-PRINT-DELAY command cannot find a pending PRINT DELAY statement.</td>
<td>An attempt was made to process an SET-DELAY-PRINT command against an SQR variable for which there was no pending PRINT DELAY statement.</td>
</tr>
<tr>
<td>008004</td>
<td>The PRINT DELAY statement did not have an SET-PRINT-DELAY command executed against it.</td>
<td>This PRINT DELAY statement did not have an SET-DELAY-PRINT command executed against it when SQR ended its run.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>008005</td>
<td>The variable (<code>01</code> <code>02</code>) was referenced by a PRINT DELAY statement but the SQR program does not contain a matching SET-PRINT-DELAY command.</td>
<td>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.</td>
</tr>
<tr>
<td>008006</td>
<td>The variable (<code>01</code> <code>02</code>) was referenced by a SET-PRINT-DELAY command but the SQR program does not contain a matching PRINT DELAY statement.</td>
<td>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.</td>
</tr>
<tr>
<td>008101</td>
<td>The specified MODE value <code>01</code> is not legal. Legal values are 'ON' or 'OFF'.</td>
<td>The MODE= qualifier values are ON or OFF. `01 = Invalid value</td>
</tr>
<tr>
<td>008102</td>
<td>At least one qualifier must be specified.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>008200</td>
<td>The specified color (<code>01</code>) is not defined.</td>
<td>The specified color is not defined in the color map. `01 = Undefined color</td>
</tr>
<tr>
<td>008201</td>
<td>Qualifier <code>01</code> has a malformed color reference.</td>
<td>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.</td>
</tr>
<tr>
<td>008202</td>
<td>Qualifier <code>01</code> must reference SQR variables only.</td>
<td>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.</td>
</tr>
<tr>
<td>008203</td>
<td>Invalid RGB value (<code>01</code>, <code>02</code>, <code>03</code>)</td>
<td>The RGB values are out of range. Each value can be from 0 to 255. <code>01 = Red value </code>02 = Green value `03 = Blue value</td>
</tr>
<tr>
<td>008204</td>
<td>At least one qualifier must be specified.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>008205</td>
<td>The Declare-Color-Map entry is not properly defined.</td>
<td>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)</td>
</tr>
<tr>
<td>008206</td>
<td>Duplicate palette name: <code>01</code></td>
<td>Change the name of the palette. `01 = Palette name in question</td>
</tr>
<tr>
<td>008207</td>
<td>The name can only contain characters [0-9 A-Z _ -].</td>
<td>Correct the syntax.</td>
</tr>
<tr>
<td>008208</td>
<td>The palette cannot have gaps. All colors up to the highest one defined (<code>01</code>) must be specified.</td>
<td>An SQR palette cannot have gaps. Correct the source code. `01 = Highest color defined for this palette</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>008209</td>
<td>The specified palette (<code>01</code>) does not exist.</td>
<td>Change the name of the palette. <code>01</code> = Palette name in question</td>
</tr>
<tr>
<td>008300</td>
<td>For font (<code>01</code>) the specified typeface (<code>02</code>) is not legal.</td>
<td>Correct the name of the CJK typeface <code>01</code> = Font id <code>02</code> = Typeface name</td>
</tr>
<tr>
<td>008301</td>
<td>For font (<code>01</code>) the specified character map (<code>02</code>) is not legal.</td>
<td>Correct the name of the CJK character map <code>01</code> = Font id <code>02</code> = Character map name</td>
</tr>
<tr>
<td>008302</td>
<td>For font (<code>01</code>) both a typeface and character map must be specified.</td>
<td>If a CJK typeface is specified with a font then a character map must also be specified. <code>01</code> = Font id</td>
</tr>
</tbody>
</table>
| 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:  
  - Simplified Chinese: EUC-CN, GBK (CP936), UCS-2  
  - Traditional Chinese: EUC-TW, BIG5, USC-2  
<p>| 008305       | For font (<code>01</code>) the encoding (<code>02</code>) is incompatible with the report output encoding (<code>03</code>). | The encoding for the current font is incompatible with the encoding used for the PDF file. <code>01</code> = Font id <code>02</code> = Font encoding <code>03</code> = Report output encoding |</p>
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>008407</td>
<td>When ACTION=INFO only the ROW and COUNT parameters are allowed. At least one must be specified.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>008408</td>
<td>When ACTION=GET only the following combination of parameters are allowed:</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td></td>
<td>● ARRAY with optional ROW and COUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● VALUES with optional ROW</td>
<td></td>
</tr>
<tr>
<td>008409</td>
<td>When ACTION=REPLACE only the following combination of parameters are allowed:</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td></td>
<td>● ARRAY with optional FIRST, ROW and COUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● BLANK with optional ROW and COUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● VALUES with optional ROW</td>
<td></td>
</tr>
<tr>
<td>008410</td>
<td>When ACTION=INSERT only the following combination of parameters are allowed:</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td></td>
<td>● ARRAY with optional MODE, FIRST, ROW and COUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● BLANK with optional MODE, ROW and COUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● VALUES with optional MODE and ROW</td>
<td></td>
</tr>
<tr>
<td>008411</td>
<td>When ACTION=DELETE only the ROW and COUNT parameters are allowed.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>008412</td>
<td>When ACTION=APPEND only the following combination of parameters are allowed:</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td></td>
<td>● ARRAY with optional FIRST, and COUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● BLANK with optional COUNT VALUES</td>
<td></td>
</tr>
<tr>
<td>008413</td>
<td>Qualifier `01' requires a numeric variable.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>008414</td>
<td>The value for <code>01</code> (<code>02) must be </code>03 <code>04 and </code>05 `06.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td></td>
<td>`01 = Qualifier name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`02 = Value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`03 = Minimum value relation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`04 = Minimum value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`05 = Maximum value relation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`06 = Maximum value</td>
<td></td>
</tr>
<tr>
<td>008415</td>
<td>The first argument in <code>01' must be the column number (&gt;= 0 and &lt;= </code>02).</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td></td>
<td>`01 = Qualifier name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`02 = Maximum value</td>
<td></td>
</tr>
<tr>
<td>008416</td>
<td>The specified column number (`01) exceeds the number of columns for this table as specified by the COLUMN-COUNT qualifier.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td></td>
<td>`01 = Column number</td>
<td></td>
</tr>
<tr>
<td>008417</td>
<td>The specified column number (<code>01) has already been defined by another </code>02 qualifier.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>`01 = Column number</td>
<td>`02 = Qualifier name</td>
</tr>
<tr>
<td>008418</td>
<td>Incorrect value (<code>01) for qualifier </code>02'. Valid values are:</td>
<td>Correct the source line. <code>01 = Value </code>02 = Qualifier name</td>
</tr>
<tr>
<td></td>
<td>`01 = Value</td>
<td>`02 = Qualifier name</td>
</tr>
<tr>
<td>008419</td>
<td>Unknown keyword (<code>01) for qualifier </code>02'. Valid keywords are:</td>
<td>Correct the source line. <code>01 = Value </code>02 = Qualifier name</td>
</tr>
<tr>
<td></td>
<td>`01 = Value</td>
<td>`02 = Qualifier name</td>
</tr>
<tr>
<td>008420</td>
<td>Incorrect value (<code>01) for qualifier </code>02'. It must be an integer value &gt; 0.</td>
<td>Correct the source line. <code>01 = Value </code>02 = Qualifier name</td>
</tr>
<tr>
<td></td>
<td>`01 = Value</td>
<td>`02 = Qualifier name</td>
</tr>
<tr>
<td>008421</td>
<td>Incorrect value (<code>01) for qualifier </code>02'. It must be a numeric value &gt; 0.</td>
<td>Correct the source line. <code>01 = Value </code>02 = Qualifier name</td>
</tr>
<tr>
<td></td>
<td>`01 = Value</td>
<td>`02 = Qualifier name</td>
</tr>
<tr>
<td>008422</td>
<td>Incorrect value (<code>01) for qualifier </code>02'. It must be a string literal.</td>
<td>Correct the source line. <code>01 = Value </code>02 = Qualifier name</td>
</tr>
<tr>
<td>008423</td>
<td>Incorrect value (<code>01) for qualifier </code>02'. It must be either YES or NO.</td>
<td>Correct the source line. <code>01 = Value </code>02 = Qualifier name</td>
</tr>
<tr>
<td>008424</td>
<td>Incorrect value (<code>01) for qualifier </code>02'. It must be either YES, NO, or the</td>
<td>Correct the source line. <code>01 = Value </code>02 = Qualifier name</td>
</tr>
<tr>
<td></td>
<td>number of lines.</td>
<td></td>
</tr>
<tr>
<td>008425</td>
<td>Qualifier `01' has already been defined.</td>
<td>Correct the source line. <code>01 = Qualifier name </code>02 = List Keyword</td>
</tr>
<tr>
<td>008426</td>
<td>There are TABLE-FORMAT entries for column numbers which exceed the number of</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td></td>
<td>columns as defined by the COLUMN-COUNT qualifier.</td>
<td></td>
</tr>
<tr>
<td>008427</td>
<td>The value for SIZE= (`01) must be &gt; 0.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>008428</td>
<td>The value for EXTENT= (`01) must be &gt; 0.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>008429</td>
<td>Encountered an invalid VALUES= argument. Legal arguments types are $Variable</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td></td>
<td>and #Variable.</td>
<td></td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>008430</td>
<td>Encountered an invalid (`01) VALUES= argument. Legal argument types are $Variable, #Variable, &amp;Variable, 'Literal', Numerics, and the word NULL.</td>
<td>Correct the source line. `01 = List contents</td>
</tr>
<tr>
<td>008431</td>
<td>The type of VALUES= parameter (<code>01) is not compatible with the corresponding table column (</code>02) defined as `03'.</td>
<td>The VALUES= parameter type is not compatible with the corresponding TABLE column type. <code>01 = VALUES parameter </code>02 = Table column number `03 = Table column type</td>
</tr>
<tr>
<td>008432</td>
<td>The number of parameters in the VALUES= list exceeds the number of columns (<code>01) for table (</code>02).</td>
<td>The number of parameters in the VALUES= list exceeds the number of columns defined in the table. <code>01 = Table column count </code>02 = Table name</td>
</tr>
<tr>
<td>008433</td>
<td>The value for MODE= (`01) must be either BEFORE or AFTER.</td>
<td>Correct the source line.</td>
</tr>
<tr>
<td>008434</td>
<td>Cannot perform GET because the specified table (`01) is empty.</td>
<td>The ACTION=GET option cannot be used against an empty table. `01 = Table name</td>
</tr>
<tr>
<td>008435</td>
<td>The number of columns (<code>01) defined for array (</code>02) does not match the number of columns (<code>03) defined for table (</code>04).</td>
<td>The number of columns defined for the specified array must be the same as the number of columns defined for the specified table. <code>01 = Array column count </code>02 = Array name <code>03 = Table column count </code>04 = Table name</td>
</tr>
<tr>
<td>008436</td>
<td>Column type mismatch in column (<code>01): Array (</code>02) is defined as (<code>03) and Table (</code>04) is defined as (`05)</td>
<td>The column types for the specified array must be compatible with the columns types for the specified table. <code>01 = Column number </code>02 = Array name <code>03 = Array column type </code>04 = Table name `05 = Table column type</td>
</tr>
<tr>
<td>008437</td>
<td>Cannot perform GET from table (<code>01) for </code>02 rows because the array (<code>03) can only support (</code>04) rows.</td>
<td>The ACTION=GET option cannot be used to increase the size of array. <code>01 = Table name </code>02 = Table rows <code>03 = Array name </code>04 = Array size</td>
</tr>
<tr>
<td>8600</td>
<td>Binary variables (`01) cannot be used with this command.</td>
<td>Correct the source syntax. `01 = Variable name</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Message</td>
<td>Suggestion/Interpretation</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>8601</td>
<td>The use of an edit mask or the keywords NUMBER, MONEY or DATE is not legal when storing numeric or binary variables.</td>
<td>Correct the source code.</td>
</tr>
</tbody>
</table>
| 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’.                                                 | SQR aborts the program run.  
  (‘02): ‘03  
  ‘01 = Image file name  
  ‘02 = System error code  
  ‘03 = System error message |
| 8617        | Error closing the image: ‘01’.                                                  | SQR aborts the program run.  
  (‘02): ‘03  
  ‘01 = Image file name |
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Message</th>
<th>Suggestion/Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>02</code></td>
<td>System error code <code>01</code></td>
<td><code>02</code> = System error code <code>01</code> = System error message</td>
</tr>
<tr>
<td>8618</td>
<td>Error writing the image: <code>01</code>. (<code>02): </code>03`</td>
<td>SQR aborts the program run. <code>01</code> = Image file name <code>02</code> = System error code <code>03</code> = System error message</td>
</tr>
<tr>
<td>8619</td>
<td>Error opening the image: <code>01</code>. (<code>02): </code>03`</td>
<td>SQR aborts the program run. <code>01</code> = Image file name <code>02</code> = System error code <code>03</code> = System error message</td>
</tr>
<tr>
<td>8620</td>
<td>Error reading the image: <code>01</code>. (<code>02): </code>03`</td>
<td>SQR aborts the program run. <code>01</code> = Image file name <code>02</code> = System error code <code>03</code> = System error message</td>
</tr>
<tr>
<td>8621</td>
<td>The SQR compression logic failed, reason <code>01</code>.</td>
<td>The SQR compression logic failed while compressing an image. <code>01</code> = Reason code</td>
</tr>
<tr>
<td>8622</td>
<td>The SQR decompression logic failed, reason <code>01</code>.</td>
<td>The SQR decompression logic failed while decompressing an image. <code>01</code> = Reason code</td>
</tr>
<tr>
<td>8623</td>
<td>The embedded BMP image is not valid, reason <code>01</code>.</td>
<td>(Windows) The embedded bitmap image is not valid. <code>01</code> = Reason code</td>
</tr>
<tr>
<td>009999</td>
<td>The printer (<code>01</code>) specified with the -Printer:WP command line flag is invalid.</td>
<td>(Windows) The specified printer is not valid. <code>01</code> = Reason code</td>
</tr>
</tbody>
</table>
### Production Reporting Commands

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>{src_num_lit</td>
</tr>
<tr>
<td>ALTER-COLOR-MAP</td>
<td>NAME=(color_name_lit</td>
</tr>
<tr>
<td>ALTER-CONNECTION</td>
<td>NAME=connection_name [DSN=or[USER=or[PARAMETERS=keyword_str=attr_str;</td>
</tr>
<tr>
<td></td>
<td>[keyword_str=attr_str;...]]] [NO-DUPLICATE=TRUE</td>
</tr>
<tr>
<td></td>
<td>SET-GENERATIONS=([[dimension1, hierarchy1] [,dimensioni, hierarchyi]...])</td>
</tr>
<tr>
<td></td>
<td>SET-LEVELS=([[dimension1, level1] [,dimensioni, leveli]...])</td>
</tr>
<tr>
<td></td>
<td>SET-MEMBERS=([[dimension1, level1] [,dimensioni, leveli]...])</td>
</tr>
<tr>
<td>ALTER-LOCALE</td>
<td>[LOCALE=or[NUMBER-EDIT-MASK=or[MONEY-EDIT-MASK=</td>
</tr>
<tr>
<td></td>
<td>[DATE-EDIT-MASK=or[INPUT-DATE-EDIT-MASK=or[MONEY-SIGN=</td>
</tr>
<tr>
<td></td>
<td>{txt_lit</td>
</tr>
<tr>
<td></td>
<td>[THOUSAND-SEPARATOR=or[DECIMAL-SEPARATOR=</td>
</tr>
<tr>
<td></td>
<td>[DATE-SEPARATOR=or[TIME-SEPARATOR=</td>
</tr>
<tr>
<td></td>
<td>[EDIT-OPTION-NA,AM,PM,BC,AD=</td>
</tr>
<tr>
<td></td>
<td>{txt_lit</td>
</tr>
<tr>
<td></td>
<td>{txt_var</td>
</tr>
<tr>
<td></td>
<td>[DAY-OF-WEEK-SHORT=</td>
</tr>
<tr>
<td></td>
<td>{txt_lit1</td>
</tr>
<tr>
<td></td>
<td>{txt_lit1</td>
</tr>
<tr>
<td>ALTER-PRINTER</td>
<td>[POINT-SIZE={point_size_num_lit</td>
</tr>
</tbody>
</table>
| 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]
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER-TABLE</td>
<td>NAME=table_name_var</td>
</tr>
<tr>
<td></td>
<td>[COUNT=count_var</td>
</tr>
<tr>
<td>ARRAY-ADD</td>
<td>{src_num_lit</td>
</tr>
<tr>
<td>ARRAY-DIVIDE</td>
<td>{src_num_lit</td>
</tr>
<tr>
<td>ARRAY-MULTIPLY</td>
<td>{src_num_lit</td>
</tr>
<tr>
<td>ARRAY-SUBTRACT</td>
<td>{src_num_lit</td>
</tr>
<tr>
<td>ASK</td>
<td>substitution_variable [prompt]</td>
</tr>
<tr>
<td>BEGIN-DOCUMENT</td>
<td>(position) END-DOCUMENT</td>
</tr>
</tbody>
</table>
| 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))]}
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM PARAMETER={txt_lit</td>
<td>_var})</td>
</tr>
<tr>
<td><strong>BEGIN-FOOTING</strong></td>
<td>footing_lines_int_lit</td>
</tr>
<tr>
<td><strong>BEGIN-HEADING</strong></td>
<td>heading_lines_int_lit</td>
</tr>
<tr>
<td><strong>BEGIN-PROCEDURE</strong></td>
<td>procedure_name [LOCAL</td>
</tr>
<tr>
<td><strong>BEGIN-PROGRAM</strong></td>
<td>END-PROGRAM</td>
</tr>
<tr>
<td><strong>BEGIN-SELECT</strong></td>
<td>[DISTINCT]</td>
</tr>
<tr>
<td><strong>BEGIN-SETUP</strong></td>
<td>END-SETUP</td>
</tr>
<tr>
<td><strong>BEGIN-SQL</strong></td>
<td>[-Cnn][-XP][-NR][-SORTnn]</td>
</tr>
<tr>
<td><strong>BREAK</strong></td>
<td>BREAK</td>
</tr>
<tr>
<td><strong>CALL</strong></td>
<td>subroutine USING</td>
</tr>
<tr>
<td><strong>CALL SYSTEM</strong></td>
<td>Same as CALL</td>
</tr>
<tr>
<td><strong>CLEAR-ARRAY</strong></td>
<td>NAME=array_name</td>
</tr>
<tr>
<td><strong>CLOSE</strong></td>
<td>{filenum_lit</td>
</tr>
<tr>
<td>COMMAND</td>
<td>SYNTAX</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>CLOSE-RS</td>
<td>NAME=row_set_name_var</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>{int_lit</td>
</tr>
<tr>
<td>COMMIT</td>
<td>COMMIT</td>
</tr>
<tr>
<td>CONCAT</td>
<td>(src_any_lit</td>
</tr>
<tr>
<td>CONNECT</td>
<td>{txt_lit</td>
</tr>
<tr>
<td>CREATE-ARRAY</td>
<td>NAME=array_name SIZE=nn</td>
</tr>
<tr>
<td></td>
<td>[EXTENT=nn]</td>
</tr>
<tr>
<td></td>
<td>{FIELD=name:type[:occurs]</td>
</tr>
<tr>
<td></td>
<td>[=init_value_txt_lit</td>
</tr>
<tr>
<td>CREATE-COLOR-PALLETTE</td>
<td>NAME={palette_name_txt_lit}</td>
</tr>
<tr>
<td></td>
<td>COLOR_1={(rgb)</td>
</tr>
<tr>
<td></td>
<td>COLOR_2={(rgb)</td>
</tr>
<tr>
<td></td>
<td>[COLOR_n]={rgb}</td>
</tr>
<tr>
<td>CREATE-LIST</td>
<td>NAME=list_name_txt_lit</td>
</tr>
<tr>
<td></td>
<td>LIST=(value_lit</td>
</tr>
<tr>
<td>CREATE-TABLE</td>
<td>NAME=table_name_var</td>
</tr>
<tr>
<td></td>
<td>USING=table_template_var</td>
</tr>
<tr>
<td></td>
<td>[COLUMN-COUNT=number_of_columns_var</td>
</tr>
</tbody>
</table>
|             | [COLUMN-ATTRIBUTES=({column—number},{keyword1},{value1}, ...,
<p>|             | {keywordn},{valuen})]                                               |
|             | [ROW-ATTRIBUTES=({keyword1},{value1}, ..., {keywordn},{valuen})]   |
|             | [TABLE-ATTRIBUTES=({keyword1},{value1}, ...,{keywordn},{valuen})]  |
| #DEBUG      | [x...] sqr_command                                                 |
| 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]                                   |</p>
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ITEM-COLOR={chart_item_keyword_lit, color_value_lit</td>
<td>(r,g,b)}]</td>
</tr>
<tr>
<td>[LEGEND=legend_lit]</td>
<td>[LEGEND-TITLE=legend_title_txt_lit]</td>
</tr>
<tr>
<td>[LEGEND-PRESENTATION= legend_presentation_lit]</td>
<td>[PIE-SEGMENT-QUANTITY-DISPLAY=pie_segment_quantity_display_lit]</td>
</tr>
<tr>
<td>[PIE-SEGMENT-PERCENT-DISPLAY=pie_segment_percent_display_lit]</td>
<td>[PIE-SEGMENT-EXPLODE=pie_segment_explode_lit]</td>
</tr>
<tr>
<td>[X-AXIS-GRID=x_axis_grid_lit]</td>
<td>[X-AXIS-LABEL=x_axis_label_txt_lit]</td>
</tr>
<tr>
<td>[X-AXIS-MIN-VALUE={x_axis_min_value_lit</td>
<td>_num_lit}]</td>
</tr>
<tr>
<td>[X-AXIS-MAJOR-INCREMENT={x_axis_major_increment_lit</td>
<td>_num_lit}]</td>
</tr>
<tr>
<td>[X-AXIS-MAJOR-TICK-MARKS=x_axis_major_tick_marks_lit]</td>
<td>[X-AXIS-MINOR-TICK-MARKS=x_axis_minor_tick_marks_lit]</td>
</tr>
<tr>
<td>[X-AXIS-ROTATE=x_rotate_num_lit]</td>
<td>[X-AXIS-SCALE=x_axis_scale_lit]</td>
</tr>
<tr>
<td>[Y-AXIS-GRID=y_axis_grid_lit]</td>
<td>[Y-AXIS-LABEL=y_axis_label_lit]</td>
</tr>
<tr>
<td>[Y-AXIS-MIN-VALUE={y_axis_min_value_lit</td>
<td>_num_lit}]</td>
</tr>
<tr>
<td>[Y-AXIS-MAJOR-INCREMENT={y_axis_major_increment_lit</td>
<td>_num_lit}]</td>
</tr>
<tr>
<td>[Y-AXIS-MAJOR-TICK-MARKS=y_axis_major_tick_marks_lit]</td>
<td>[Y-AXIS-MINOR-TICK-MARKS=y_axis_minor_tick_marks_lit]</td>
</tr>
<tr>
<td>[Y-AXIS-SCALE=y_axis_scale_lit]</td>
<td>[Y-AXIS-TICK-MARK-PLACEMENT=y_axis_tick_mark_placement_lit]</td>
</tr>
<tr>
<td>[Y2-AXIS-LABEL=y2_axis_label_lit]</td>
<td>[Y2-AXIS-MASK=mask_txt_lit]</td>
</tr>
<tr>
<td>[Y2-AXIS-MIN-VALUE={y2_axis_min_value_lit</td>
<td>_num_lit}]</td>
</tr>
<tr>
<td>[Y2-AXIS-MAJOR-INCREMENT={y2_axis_major_increment_lit</td>
<td>_num_lit}]</td>
</tr>
<tr>
<td>[Y2-AXIS-MAJOR-TICK-MARKS=y2_axis_major_tick_marks_lit]</td>
<td>[Y2-AXIS-MINOR-TICK-MARKS=y2_axis_minor_tick_marks_lit]</td>
</tr>
<tr>
<td>[Y2-AXIS-SCALE=y2_axis_scale_lit]</td>
<td>[Y2-AXIS-COLOR-PALETTE=color_palette_lit]</td>
</tr>
<tr>
<td>[Y2-DATA-ARRAY=array_name]</td>
<td>[Y2-DATA-ARRAY-ROW-COUNT=row_count_num_lit]</td>
</tr>
<tr>
<td>[Y2-DATA-ARRAY-COLUMN-COUNT=column_count_num_lit]</td>
<td>[Y2-DATA-ARRAY-COLUMN-LABELS={NONE</td>
</tr>
<tr>
<td>[Y2-TYPE=chart_type_lit]</td>
<td>END-DECLARE</td>
</tr>
</tbody>
</table>

| DECLARE-COLOR-MAP | color_name=({rgb})
| DECLARE-COLOR-MAP | color_name=({rgb})
| DECLARE-COLOR-MAP | .
| DECLARE-COLOR-MAP | .
<p>| DECLARE-COLOR-MAP | END-DECLARE |</p>
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SYNTAX</th>
</tr>
</thead>
</table>
| DECLARE-CONNECTION | `connection_name
DSN=(uq_txt_lit)
[USER= or [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` |
| 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` |
| DECLARE-LAYOUT   | `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]]
[LINELINE=width_num_lit[uom]]
[MAX-COLUMNS=columns_int_lit]
[BOTTOM-MARGIN= bottom_margin_num_lit[uom]]
[PAGEDEPTH=page_depth_num_lit[uom]]
[MAX-LINES=lines_int_lit]
[CHAR-WIDTH=char_width_num_lit[uom]]
[LINELINE=height_num_lit[uom]]
END-DECLARE` |
| DECLARE-PRINTER  | `printer_name
[[[TYPE=printer_type_lit]FOR-REPORTS=(report_name1
[,report_namei]...))]
[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]
[S]YMBOl-SET=symbol_set_id_lit]
[STARTUP-FILE=file_name_txt_lit]
[PITCH=pitch_num_lit]
[APT=font_int_lit]
[BEFORE-BOLD=before_bold_string_txt_lit]
[AFTER-BOLD=after_bold_string_txt_lit]
END-DECLARE` |
| DECLARE-PROCEDURE| `[FOR-REPORTS=(report_name1[,report_namei]...])
[BEFORE-REPORT=procedure_name[(arg1[,argi]...))]` |
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFTER-REPORT=procedure_name[(arg1[,argi]...)]</td>
<td>BEFORE-PAGE=procedure_name[(arg1[,argi]...)]</td>
</tr>
<tr>
<td>END-DECLARE</td>
<td>DECLARE-REPORT</td>
</tr>
<tr>
<td>report_name</td>
<td>TOC=toc_name</td>
</tr>
<tr>
<td></td>
<td>LAYOUT=layout_name</td>
</tr>
<tr>
<td></td>
<td>PRINTER-TYPE=printer_type</td>
</tr>
<tr>
<td>END-DECLARE</td>
<td>DECLARE-TABLE</td>
</tr>
<tr>
<td></td>
<td>NAME=table_template_name</td>
</tr>
<tr>
<td></td>
<td>COLUMN-COUNT=number_of_columns</td>
</tr>
<tr>
<td></td>
<td>[COLUMN-ATTRIBUTES=({column number},{keyword1},{value1}, ..., {keywordn},{valuen})]</td>
</tr>
<tr>
<td></td>
<td>[ROW-ATTRIBUTES=({keyword1},{value1}, ..., {keywordn},{valuen})]</td>
</tr>
<tr>
<td></td>
<td>[TABLE-ATTRIBUTES=({keyword1},{value1}, ..., {keywordn},{valuen})]</td>
</tr>
<tr>
<td>DECLARE-TOC</td>
<td></td>
</tr>
<tr>
<td>toc_name</td>
<td>[FOR-REPORTS=(report_name1[,report_namei]...)]</td>
</tr>
<tr>
<td></td>
<td>[DOT-LEADER=YES</td>
</tr>
<tr>
<td></td>
<td>[INDENTATION=position_count_num_lit]</td>
</tr>
<tr>
<td></td>
<td>[BEFORE-TOC=procedure_name[(arg1[,argi]...)]]</td>
</tr>
<tr>
<td></td>
<td>[AFTER-TOC=procedure_name[(arg1[,argi]...)]]</td>
</tr>
<tr>
<td></td>
<td>[BEFORE-PAGE=procedure_name[(arg1[,argi]...)]</td>
</tr>
<tr>
<td></td>
<td>[AFTER-PAGE=procedure_name[(arg1[,argi]...)]</td>
</tr>
<tr>
<td></td>
<td>[ENTRY=procedure-name [(argi [,argi] ...)]</td>
</tr>
<tr>
<td>END-DECLARE</td>
<td>DECLARE-VARIABLE</td>
</tr>
<tr>
<td></td>
<td>[DEFAULT-NUMERIC= {DECIMAL[(prec_lit)]</td>
</tr>
<tr>
<td></td>
<td>[DECIMAL[(prec_lit)]num_var[(prec_lit)][num_var[(prec_lit)]...]]</td>
</tr>
<tr>
<td></td>
<td>[FLOAT num_var[num_var]...]]</td>
</tr>
<tr>
<td></td>
<td>[DATE date_var[date_var]...]]</td>
</tr>
<tr>
<td></td>
<td>[INTEGER num_var[num_var]...]]</td>
</tr>
<tr>
<td></td>
<td>[TEXT string_var[string_var]...]]</td>
</tr>
<tr>
<td></td>
<td>[BINARY binary_var[binary_var]...]]</td>
</tr>
<tr>
<td>END-DECLARE</td>
<td>#DEFINE</td>
</tr>
<tr>
<td></td>
<td>substitution_variable value</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>{any_lit</td>
</tr>
<tr>
<td></td>
<td>[:$]edit_mask</td>
</tr>
<tr>
<td>DIVIDE</td>
<td>{src_num_lit</td>
</tr>
<tr>
<td></td>
<td>[ON-ERROR={HIGH</td>
</tr>
<tr>
<td>DO</td>
<td>procedure_name[(arg1[, argi]...)]</td>
</tr>
<tr>
<td>DRAW</td>
<td>DRAW {position}</td>
</tr>
<tr>
<td></td>
<td>TYPE={type_lit</td>
</tr>
<tr>
<td></td>
<td>[HEIGHT={height_lit</td>
</tr>
<tr>
<td></td>
<td>[WIDTH={width_lit</td>
</tr>
<tr>
<td></td>
<td>[RULE={rule_lit</td>
</tr>
<tr>
<td></td>
<td>[FILL COLOR=({color_name_lit</td>
</tr>
<tr>
<td></td>
<td>[LINE COLOR= {color_name_lit</td>
</tr>
<tr>
<td>COMMAND</td>
<td>SYNTAX</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>COMMAND</td>
<td>[CAP={cap_lit</td>
</tr>
<tr>
<td>DUMP-TABLE</td>
<td>NAME=table_name_var</td>
</tr>
<tr>
<td>#ELSE</td>
<td>#ELSE</td>
</tr>
<tr>
<td>ELSE</td>
<td>ELSE</td>
</tr>
<tr>
<td>ENCODE</td>
<td>src_code_string_lit INTO dst_txt_var</td>
</tr>
<tr>
<td>END-DECLARE</td>
<td>END-DECLARE</td>
</tr>
<tr>
<td>END-DOCUMENT</td>
<td>END-DOCUMENT</td>
</tr>
<tr>
<td>END-EVALUATE</td>
<td>END-EVALUATE</td>
</tr>
<tr>
<td>END-EXECUTE</td>
<td>END-EXECUTE</td>
</tr>
<tr>
<td>END-FOOTING</td>
<td>END-FOOTING</td>
</tr>
<tr>
<td>END-HEADING</td>
<td>END-HEADING</td>
</tr>
<tr>
<td>#ENDIF</td>
<td>#END-IF</td>
</tr>
<tr>
<td>#ENDIF</td>
<td>#ENDIF</td>
</tr>
<tr>
<td>END-IF</td>
<td>END-IF</td>
</tr>
<tr>
<td>END-PROCEDURE</td>
<td>END-PROCEDURE</td>
</tr>
<tr>
<td>END-PROGRAM</td>
<td>END-PROGRAM</td>
</tr>
<tr>
<td>END-SELECT</td>
<td>END-SELECT</td>
</tr>
<tr>
<td>END-SETUP</td>
<td>END-SETUP</td>
</tr>
<tr>
<td>END-SQL</td>
<td>END-SQL</td>
</tr>
<tr>
<td>END-WHILE</td>
<td>END-WHILE</td>
</tr>
<tr>
<td>EVALUATE</td>
<td>{any_lit</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>[-XC][ON-ERROR=procedure[(arg1[,argi]...)]] [DO=procedure[(arg1[,argi]...)]] {{@#status_var=}stored_procedure_name}</td>
</tr>
<tr>
<td>COMMAND</td>
<td>SYNTAX</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>COMMAND</td>
<td>[[@param=]{any_col</td>
</tr>
<tr>
<td>EXIT-SELECT</td>
<td>EXIT-SELECT</td>
</tr>
<tr>
<td>EXTRACT</td>
<td>{dst_txt_var</td>
</tr>
<tr>
<td></td>
<td>{(src_txt_lit</td>
</tr>
<tr>
<td>FILL-TABLE</td>
<td>NAME=table_name_var</td>
</tr>
<tr>
<td></td>
<td>VALUE=value_var</td>
</tr>
<tr>
<td></td>
<td>LOCATION=(row_var</td>
</tr>
<tr>
<td></td>
<td>[ATTRIBUTES=(keyword1), (value1), ..., (keywordn), (valuen)]</td>
</tr>
<tr>
<td>FIND</td>
<td>{(obj_txt_lit</td>
</tr>
<tr>
<td></td>
<td>{(src_txt_var</td>
</tr>
<tr>
<td>GET</td>
<td>dst_any_var...FROM src_array_name(element)[field[(occurs)]]...</td>
</tr>
<tr>
<td>GET-COLOR</td>
<td>[PRINT-TEXT-FOREGROUND=({color_name_var})]</td>
</tr>
<tr>
<td></td>
<td>[PRINT-TEXT-BACKGROUND=({color_name_var})]</td>
</tr>
<tr>
<td></td>
<td>[PRINT-PAGE-BACKGROUND=({color_name_var})]</td>
</tr>
<tr>
<td></td>
<td>[LINE-COLOR=({color_name_var})]</td>
</tr>
<tr>
<td></td>
<td>[FILL-COLOR=({color_name_var})]</td>
</tr>
<tr>
<td>GOTO</td>
<td>label</td>
</tr>
<tr>
<td>#IF</td>
<td>{txt_lit</td>
</tr>
<tr>
<td>IF</td>
<td>logical_expression</td>
</tr>
<tr>
<td></td>
<td>sqr_commands...</td>
</tr>
<tr>
<td></td>
<td>[ELSE</td>
</tr>
<tr>
<td></td>
<td>sqr_commands...</td>
</tr>
<tr>
<td></td>
<td>END-IF</td>
</tr>
<tr>
<td>#IFDEF</td>
<td>substitution_variable</td>
</tr>
<tr>
<td>#IFNDEF</td>
<td></td>
</tr>
<tr>
<td>#INCLUDE</td>
<td>filename_lit</td>
</tr>
<tr>
<td>INPUT</td>
<td>input_var[MAXLEN=nn][prompt]</td>
</tr>
<tr>
<td></td>
<td>[TYPE={CHAR</td>
</tr>
<tr>
<td></td>
<td>[STATUS=num_var][NOPROMPT][BATCH-MODE]</td>
</tr>
<tr>
<td></td>
<td>[FORMAT={txt_lit</td>
</tr>
<tr>
<td>LAST-PAGE</td>
<td>position[pre_txt_lit[post_txt_lit]]</td>
</tr>
<tr>
<td>LET</td>
<td>dst_var=expression</td>
</tr>
<tr>
<td>LOAD-LOOKUP</td>
<td>In the SETUP section:</td>
</tr>
<tr>
<td></td>
<td>NAME=lookup_table_name</td>
</tr>
<tr>
<td></td>
<td>TABLE=database_table_name</td>
</tr>
<tr>
<td></td>
<td>KEY=key_column_name</td>
</tr>
<tr>
<td></td>
<td>RETURN_VALUE=return_column_name</td>
</tr>
<tr>
<td>COMMAND</td>
<td>SYNTAX</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>[ROWS=initial_row_estimate_int_lit]</td>
</tr>
<tr>
<td></td>
<td>[EXTENT=size_to_grow_by_int_lit]</td>
</tr>
<tr>
<td></td>
<td>[WHERE=where_clause_txt_lit]</td>
</tr>
<tr>
<td></td>
<td>[SORT=sort_mode]</td>
</tr>
<tr>
<td></td>
<td>[QUIET]</td>
</tr>
<tr>
<td></td>
<td>[SCHEMA=schema_txt_lit]</td>
</tr>
<tr>
<td></td>
<td>[PROCEDURE=proc_txt_lit]</td>
</tr>
<tr>
<td></td>
<td>[PARAMETERS=({{arg1 [IN</td>
</tr>
<tr>
<td></td>
<td>(or)</td>
</tr>
<tr>
<td></td>
<td>COMMAND=command_txt_lit</td>
</tr>
<tr>
<td></td>
<td>(or)</td>
</tr>
<tr>
<td></td>
<td>GETDATA=getdata_txt_lit</td>
</tr>
<tr>
<td></td>
<td>[(FROM-ROWSETS=({m</td>
</tr>
<tr>
<td></td>
<td>{FROM-PARAMETER=parameter_txt_lit}</td>
</tr>
</tbody>
</table>

In the body of the report:
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} [,argn [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}}

**LOOKUP**
lookup_table_name {key_any_lit|_var|_col} {ret_txt_var|_date|_var}

**LOWERCASE**
txt_var

**MBTOSBS**
(txt_var)

**MOVE**
{src_any_lit|_var|_col} TO dst_any_var
   [:]$format_mask|NUMBER|MONEY|DATE]

**MULTIPLY**
(src_num_lit|_var|_col) TIMES dst_num_var
   [ROUND=nn]

**NEW-PAGE**
[erase_from_line_num_lit|_var|_col]

**NEW-REPORT**
{report_filename_txt_lit|_var|_col}
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NEXT-COLUMN</strong></td>
<td></td>
</tr>
</tbody>
</table>
  [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|decl_val_list_name_lit|var|_col|decl_val_lit|_var|_col)\} |
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>_col,...)</td>
<td></td>
</tr>
<tr>
<td>PALETTE:(color_palette_lit[_var</td>
<td>_col]),...))</td>
</tr>
<tr>
<td>[DATA-ARRAY=array_name]</td>
<td></td>
</tr>
<tr>
<td>[DATA-ARRAY-ROW-COUNT={x_num_lit[_var</td>
<td>_col}]</td>
</tr>
<tr>
<td>[DATA-ARRAY-COLUMN-COUNT={x_num_lit[_var</td>
<td>_col}]</td>
</tr>
<tr>
<td>[DATA-ARRAY-COLUMN-LABELS={NONE</td>
<td>array_name{(txt_lit[var</td>
</tr>
<tr>
<td>[DATA-LABELS={data_labels_txt_lit[_var</td>
<td>_col}]</td>
</tr>
<tr>
<td>[FOOTER-TEXT=NONE</td>
<td>text_lit[_var</td>
</tr>
<tr>
<td>[SUB-FOOTER-TEXT=NONE</td>
<td>text_lit[_var</td>
</tr>
<tr>
<td>[ITEM-COLOR=(item_color_keyword_lit[_var</td>
<td>_col],(color_txt_lit_var[_col]</td>
</tr>
<tr>
<td>[ITEM-SIZE=(item_size_keyword_lit[_var</td>
<td>_col], item_size_num_lit[_var</td>
</tr>
<tr>
<td>[LEGEND={legend_txt_lit[_var</td>
<td>_col}]</td>
</tr>
<tr>
<td>[LEGEND-TITLE={legend_title_txt_lit[_var</td>
<td>_col}]</td>
</tr>
<tr>
<td>[LEGEND-PRESENTATION={legend_presentation_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[PIE-SEGMENT-QUANTITY-DISPLAY={pie_segment_quantity_display_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[PIE-SEGMENT-PERCENT_DISPLAY={pie_segment_percent_display_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[PIE-SEGMENT-EXPLODE={pie_segment_explode_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-GRID={x_axis_grid_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-MIN-VALUE={x_axis_min_value_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-MAX-VALUE={x_axis_max_value_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-MAJOR_INCREMENT={x_axis_major_increment_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-MINOR_INCREMENT={x_axis_minor_increment_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-MAJOR-TICK_MARKS={x_axis_major_tick_marks_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-MINOR-TICK-MARKS={x_axis_minor_tick_marks_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-TICK-MARK-PLACEMENT={x_axis_tick_mark_placement_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-ROTATE={x_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[X-AXIS-SCALE={x_axis_scale_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y-AXIS-GRID={y_axis_grid_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y-AXIS-MIN-VALUE={y_axis_min_value_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y-AXIS-MAX-VALUE={y_axis_max_value_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y-AXIS-MAJOR_INCREMENT={y_axis_major_increment_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y-AXIS-MINOR_INCREMENT={y_axis_minor_increment_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y-AXIS-MAJOR-TICK_MARKS={y_axis_major_tick_marks_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y-AXIS-MINOR-TICK-MARKS={y_axis_minor_tick_marks_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y-AXIS-TICK-MARK-PLACEMENT={y_axis_tick_mark_placement_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y-AXIS-SCALE={y_axis_scale_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y2-AXIS-LABEL={y2_axis_label_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y2-AXIS-MASK={mask_txt_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y2-AXIS-MIN-VALUE={y2_axis_min_value_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>[Y2-AXIS-MAX-VALUE={y2_axis_max_value_num_lit[_num_lit[_var</td>
<td>_col]}</td>
</tr>
<tr>
<td>COMMAND</td>
<td>SYNTAX</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>[Y2-AXIS-MAJOR-INCREMENT={(y2_axis_major_increment_num_lit</td>
<td>_var</td>
</tr>
<tr>
<td>[Y2-AXIS-MINOR-INCREMENT={(y2_axis_minor_increment_num_lit</td>
<td>_var</td>
</tr>
<tr>
<td>[Y2-AXIS-MAJOR-TICK-MARKS={(y2_axis_major_tick_marks_txt_lit</td>
<td>_var</td>
</tr>
<tr>
<td>[Y2-AXIS-MINOR-TICK-MARKS={(y2_axis_minor_tick_marks_txt_lit</td>
<td>_var</td>
</tr>
<tr>
<td>[Y2-AXIS-SCALE={(y2_axis_scale_txt_lit</td>
<td>_var</td>
</tr>
<tr>
<td>[Y2-COLOR-PALETTE=color_palette_lit</td>
<td>_var</td>
</tr>
<tr>
<td>[Y2-DATA-ARRAY=array_name]</td>
<td></td>
</tr>
<tr>
<td>[Y2-DATA-ARRAY-ROW-COUNT={(x_num_lit</td>
<td>_var</td>
</tr>
<tr>
<td>[Y2-DATA-ARRAY-COLUMN-COUNT={(x_num_lit</td>
<td>_var</td>
</tr>
<tr>
<td>[Y2-DATA-ARRAY-COLUMN-LABELS={(NONE</td>
<td>array_name</td>
</tr>
<tr>
<td>[Y2-TYPE={(chart_type_txt_lit</td>
<td>_var</td>
</tr>
<tr>
<td>PRINT-DIRECT</td>
<td>[NOLF]</td>
</tr>
<tr>
<td>[PRINTER={LINEPRINTER</td>
<td>POSTSCRIPT</td>
</tr>
<tr>
<td>PRINT-IMAGE</td>
<td>{image_name]position</td>
</tr>
<tr>
<td></td>
<td>[TYPE={(image_type_lit</td>
</tr>
<tr>
<td></td>
<td>[IMAGE-SIZE= (width_num_lit</td>
</tr>
<tr>
<td></td>
<td>[SOURCE= {file_name_lit</td>
</tr>
<tr>
<td></td>
<td>[(FOR-PRINTER=((POSTSCRIPT</td>
</tr>
<tr>
<td>PRINT-TABLE</td>
<td>NAME=table_name_var</td>
</tr>
<tr>
<td></td>
<td>[CONTINUATION=continuation_var</td>
</tr>
<tr>
<td>PUT</td>
<td>{src_any_lit</td>
</tr>
<tr>
<td>READ</td>
<td>{filenum_lit</td>
</tr>
<tr>
<td></td>
<td>[STATUS=status _num_var]</td>
</tr>
<tr>
<td>ROLLBACK</td>
<td>ROLLBACK</td>
</tr>
<tr>
<td>SBTOMBS</td>
<td>(txt_var)</td>
</tr>
<tr>
<td>SECURITY</td>
<td>[SET={(sid [,sid]...)}]</td>
</tr>
<tr>
<td></td>
<td>[APPEND={(sid [,sid]...)}]</td>
</tr>
<tr>
<td></td>
<td>[REMOVE={(sid [,sid]...)}]</td>
</tr>
<tr>
<td></td>
<td>[MODE=mode]</td>
</tr>
<tr>
<td>SET-COLOR</td>
<td>[PRINT-TEXT-FOREGROUND={(color_name_lit</td>
</tr>
<tr>
<td></td>
<td>[PRINT-TEXT-BACKGROUND={(color_name_lit</td>
</tr>
<tr>
<td></td>
<td>[PRINT-PAGE-BACKGROUND={(color_name_lit</td>
</tr>
<tr>
<td></td>
<td>[LINE-COLOR={(color_name_lit</td>
</tr>
<tr>
<td></td>
<td>[FILL-COLOR={(color_name_lit</td>
</tr>
<tr>
<td>SET-DELAY-PRINT</td>
<td>delay_var WITH {src_lit</td>
</tr>
</tbody>
</table>

433
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHOW</td>
<td>[cursor_position</td>
</tr>
<tr>
<td></td>
<td>[EDIT edit_mask</td>
</tr>
<tr>
<td></td>
<td>[UNDERLINE][REVERSE][NORMAL][BEEP][NOLINE]...</td>
</tr>
<tr>
<td>STOP</td>
<td>[QUIET]</td>
</tr>
<tr>
<td>STRING</td>
<td>{src_any_lit</td>
</tr>
<tr>
<td></td>
<td>BY {delim_txt_lit</td>
</tr>
<tr>
<td></td>
<td>INTO dst_txt_var</td>
</tr>
<tr>
<td>SUBTRACT</td>
<td>SUBTRACT {src_num_lit</td>
</tr>
<tr>
<td>TOC-ENTRY</td>
<td>TEXT={src_txt_lit</td>
</tr>
<tr>
<td></td>
<td>[LEVEL={level_num_lit</td>
</tr>
<tr>
<td>UNSTRING</td>
<td>{{src_txt_lit</td>
</tr>
<tr>
<td></td>
<td>BY {delim_txt_lit</td>
</tr>
<tr>
<td></td>
<td>INTO dst_txt_var...</td>
</tr>
<tr>
<td>UPPERCASE</td>
<td>txt_var</td>
</tr>
<tr>
<td>USE</td>
<td>database</td>
</tr>
<tr>
<td>USE-COLUMN</td>
<td>{column_number_int_lit</td>
</tr>
<tr>
<td>USE-PRINTER-TYPE</td>
<td>printer-type</td>
</tr>
<tr>
<td></td>
<td>See DECLARE-PRINTER for valid types.</td>
</tr>
<tr>
<td>USE-PROCEDURE</td>
<td>{FOR-REPORTS=(report_name1[,report_namei]...)}</td>
</tr>
<tr>
<td></td>
<td>{BEFORE-REPORT=procedure_name[(arg1[,argi]...)]}</td>
</tr>
<tr>
<td></td>
<td>{AFTER-REPORT=procedure_name[(arg1[,argi]...)]}</td>
</tr>
<tr>
<td></td>
<td>{BEFORE-PAGE=procedure_name[(arg1[,argi]...)]}</td>
</tr>
<tr>
<td></td>
<td>{AFTER-PAGE=procedure_name[(arg1[,argi]...)]}</td>
</tr>
<tr>
<td>USE-REPORT</td>
<td>{report_name_lit</td>
</tr>
<tr>
<td>WHILE</td>
<td>logical_expression</td>
</tr>
<tr>
<td></td>
<td>sqr_commands...</td>
</tr>
<tr>
<td></td>
<td>[BREAK]</td>
</tr>
<tr>
<td></td>
<td>[CONTINUE]</td>
</tr>
<tr>
<td></td>
<td>sqr_commands...</td>
</tr>
<tr>
<td></td>
<td>END-WHILE</td>
</tr>
<tr>
<td>WRITE</td>
<td>{filenum_lit</td>
</tr>
<tr>
<td></td>
<td>{{{txt_lit</td>
</tr>
<tr>
<td></td>
<td>[:len_int_lit</td>
</tr>
<tr>
<td></td>
<td>[STATUS=status_num_var]</td>
</tr>
<tr>
<td>WRITE-RS</td>
<td>NAME=row_set_name_var</td>
</tr>
<tr>
<td></td>
<td>VALUE=((name_var</td>
</tr>
</tbody>
</table>
# Deprecated Production Reporting Command-line Flags

## Table 102 Deprecated Production Reporting Command-line Flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-EH_FULLHTML:xx</td>
<td>Specifies the level of HTML that your browser supports so appropriate Enhanced HTML code is generated. Acceptable values for this flag are:</td>
</tr>
<tr>
<td></td>
<td>- 40—Generates XHTML 1.1</td>
</tr>
<tr>
<td></td>
<td>- 32—Generates HTML 3.2</td>
</tr>
<tr>
<td></td>
<td>The following values are deprecated (no longer valid but still accepted for upward compatibility):</td>
</tr>
<tr>
<td></td>
<td>- 30—Was used to specify HTML 3.0</td>
</tr>
<tr>
<td></td>
<td>- TRUE—Was used to specify HTML 3.2</td>
</tr>
</tbody>
</table>
Flag | Description
---|---
● FALSE—Was used to specify HTML 3.0  
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.

-Mfile | Defines a startup file containing sizes to be assigned to various internal parameters—extremely small, large, or complex reports.  
-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.)

-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 for more information.)

<table>
<thead>
<tr>
<th>Entry</th>
<th>Deprecated Values</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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. | Specifies the level of HTML that the browser supports so appropriate Enhanced HTML code is generated.  
Note: See FullHTML under the “[Enhanced-HTML] Section” on page 343 for more information on current values. |

### [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>
<thead>
<tr>
<th>Table 104</th>
<th>Entries for [Processing-Limits] Section</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entry</strong></td>
<td><strong>Old Default</strong></td>
</tr>
<tr>
<td>BREAKS</td>
<td>100</td>
</tr>
<tr>
<td>DYNAMICARGS</td>
<td>70</td>
</tr>
<tr>
<td>EXPRESSIONSPACE</td>
<td>8192</td>
</tr>
<tr>
<td>FORWARDREFS</td>
<td>200</td>
</tr>
<tr>
<td>ONBREAKS</td>
<td>30</td>
</tr>
<tr>
<td>POSITIONS</td>
<td>1800</td>
</tr>
<tr>
<td>PROGLINEPARS</td>
<td>18000</td>
</tr>
<tr>
<td>PROGLINES</td>
<td>5000</td>
</tr>
<tr>
<td>QUERIES</td>
<td>60</td>
</tr>
<tr>
<td>QUERYARGS</td>
<td>240</td>
</tr>
<tr>
<td>SQLSIZE</td>
<td>4000</td>
</tr>
<tr>
<td>STRINGSPACE</td>
<td>15000</td>
</tr>
<tr>
<td>SUBVARS</td>
<td>100</td>
</tr>
<tr>
<td>VARIABLES</td>
<td>1500</td>
</tr>
<tr>
<td>WHENS</td>
<td>70</td>
</tr>
</tbody>
</table>

**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


Table 105  Deprecated Entries in the [Default-Settings] Section

<table>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFCompressionText</td>
<td>0 - 9</td>
<td>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.</td>
</tr>
<tr>
<td>PDFCompressionGraphics</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Deprecated Transforms

Table 106  Deprecated Transforms

<table>
<thead>
<tr>
<th>Transform</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToCanonical</td>
<td>Transforms Unicode &quot;compatibility characters&quot; to their standard equivalents.</td>
</tr>
<tr>
<td>ToTraditionalChinese</td>
<td>Converts all Simplified Chinese characters to their Traditional Chinese equivalent.</td>
</tr>
<tr>
<td>ToSimplifiedChinese</td>
<td>Converts all Traditional Chinese characters to their Simplified Chinese equivalent.</td>
</tr>
</tbody>
</table>

Note:

A transform is a function of the LET command. See “Unicode Functions” on page 210 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

<table>
<thead>
<tr>
<th>Old Commands</th>
<th>Use Instead</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN-REPORT (END-REPORT)</td>
<td>BEGIN-PROGRAM (END-PROGRAM)</td>
</tr>
<tr>
<td>DATE-TIME</td>
<td>datenow function</td>
</tr>
<tr>
<td>Old Commands</td>
<td>Use Instead</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>DECLARE PRINTER</td>
<td>DECLARE-PRINTER</td>
</tr>
<tr>
<td>DECLARE PROCEDURE</td>
<td>DECLARE-PROCEDURE</td>
</tr>
<tr>
<td>DOLLAR-SYMBOL</td>
<td>ALTER-LOCALE</td>
</tr>
<tr>
<td>GRAPHIC FONT</td>
<td>ALTER-PRINTER</td>
</tr>
<tr>
<td>GRAPHIC-BOX</td>
<td>DRAW</td>
</tr>
<tr>
<td>GRAPHIC HORZ-LINE</td>
<td>DRAW</td>
</tr>
<tr>
<td>GRAPHIC VERT-LINE</td>
<td>DRAW</td>
</tr>
<tr>
<td>MONEY-SYMBOL</td>
<td>ALTER-LOCALE</td>
</tr>
<tr>
<td>NO-FORMFEED</td>
<td>DECLARE-LAYOUT</td>
</tr>
<tr>
<td>PAGE-SIZE</td>
<td>DECLARE-LAYOUT</td>
</tr>
<tr>
<td>PRINTER-DEINIT</td>
<td>DECLARE-PRINTER</td>
</tr>
<tr>
<td>PRINTER-INIT</td>
<td>DECLARE-PRINTER</td>
</tr>
<tr>
<td>PRINT...CODE</td>
<td>PRINT...CODE-PRINTER</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Database</th>
<th>Default Date-Time Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>YYYY-MM-DD-HH:MI</td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD-HH:MI:SS.NNNNN</td>
</tr>
<tr>
<td>Informix</td>
<td>YYYY-MM-DD HH:MI</td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS.NNN</td>
</tr>
<tr>
<td>Oracle</td>
<td>DD-Mon-YYYY HH:MI PM</td>
</tr>
<tr>
<td>Sybase</td>
<td>DD-MON-YYYY HH:MI</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Argument</th>
<th>Choice or Measure</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>LINEPRINTER, POSTSCRIPT, HPLASERJET</td>
<td>LINEPRINTER</td>
<td>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.</td>
</tr>
<tr>
<td>ORIENTATION</td>
<td>PORTRAIT, LANDSCAPE</td>
<td>PORTRAIT</td>
<td>Portrait pages are printed vertically. Landscape pages are printed horizontally. Printing in landscape on HP Laserjet printers requires landscape fonts.</td>
</tr>
<tr>
<td>LEFT-MARGIN</td>
<td>inches</td>
<td>0.5</td>
<td>This argument does not apply to LINEPRINTER printers. This is the amount of blank space to leave at the left side of the page.</td>
</tr>
<tr>
<td>TOP-MARGIN</td>
<td>inches</td>
<td>0.5</td>
<td>This argument does not apply to LINEPRINTER printers. This is the amount of blank space to leave at the top of the page.</td>
</tr>
<tr>
<td>LINE-SIZE</td>
<td>points</td>
<td>12</td>
<td>This argument does not apply to LINEPRINTER printers.</td>
</tr>
<tr>
<td>Argument</td>
<td>Choice or Measure</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>LINE-SIZE</td>
<td>points</td>
<td>7.2</td>
<td>This is the size of each Production Reporting line on the page. There are 72 points per inch. 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.</td>
</tr>
<tr>
<td>CHAR-SIZE</td>
<td>points</td>
<td>7.2</td>
<td>This argument does not apply to LINEPRINTER printers. 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.</td>
</tr>
<tr>
<td>LINES-INCH</td>
<td>lines</td>
<td>6</td>
<td>This argument does not apply to Lineprinter printers. This is an alternate way of indicating the line size, in lines per inch, rather than in points for the LINE-SIZE.</td>
</tr>
<tr>
<td>CHARS-INCH</td>
<td>characters</td>
<td>10</td>
<td>This argument does not apply to LINEPRINTER printers. 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.</td>
</tr>
<tr>
<td>POINT-SIZE</td>
<td>points</td>
<td>12</td>
<td>This argument does not apply to Lineprinter printers. This is the beginning size of the selected font.</td>
</tr>
<tr>
<td>FONT-TYPE</td>
<td>PROPORTIONAL, FIXED</td>
<td>Depends on the font</td>
<td>This argument applies only to HP Laserjet printers and needs to be specified only for font types not defined in Table 33, &quot;Fonts Available for HP Laserjet Printers in Production Reporting,&quot; on page 140.</td>
</tr>
<tr>
<td>SYMBOL-SET</td>
<td>HP defined sets</td>
<td>0U</td>
<td>This argument applies only to HP Laserjet printers. The default value of &quot;0U&quot; is for the ASCII symbol set. For a complete list of the symbol sets, see the HP Laserjet Technical Reference Manual.</td>
</tr>
<tr>
<td>STARTUP-FILE</td>
<td>filename</td>
<td>POSTSCRI.STR</td>
<td>This argument applies only to PostScript printers. 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.</td>
</tr>
<tr>
<td>FONT</td>
<td>font_number</td>
<td>3</td>
<td>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 HP Laserjet Technical Reference Manual. For PostScript printers, Production Reporting supplies a list of fonts and arbitrary font number assignments in the file</td>
</tr>
</tbody>
</table>
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

---

**Table 33, “Fonts Available for HP LaserJet Printers in Production Reporting,” on page 140**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Choice or Measure</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSTSCRI.STR.</td>
<td>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. Table 33, “Fonts Available for HP LaserJet Printers in Production Reporting,” on page 140 lists the fonts available in Production Reporting internally. Table 34, “Fonts Available for PostScript Printers,” on page 141 lists the fonts available in the Production Reporting POSTSCRI.STR file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEFORE-BOLD</td>
<td>any string</td>
<td>(none)</td>
<td>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: BEFORE-BOLD=&lt;27&gt;[r ! Turn on bold AFTER-BOLD=&lt;27&gt;[u ! Turn it off These arguments work in conjunction with the BOLD argument of the PRINT command.</td>
</tr>
<tr>
<td>AFTER-BOLD</td>
<td>any string</td>
<td>(none)</td>
<td>See BEFORE-BOLD.</td>
</tr>
</tbody>
</table>
Production Reporting maps its line and column positions on the page by using a grid determined by the \texttt{LINE-SIZE} and \texttt{CHAR-SIZE} (or \texttt{LINES-INCH} and \texttt{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 \texttt{PRINT} statement or indicate a larger \texttt{CHAR-SIZE} in the \texttt{DECLARE PRINTER} command.

\section*{DECLARE PROCEDURE}

\textbf{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 \texttt{DECLARE-PROCEDURE}.

\textbf{Function}

Defines specific event procedures.

\textbf{Syntax}

\begin{verbatim}
DECLARE PROCEDURE
[BEBFORE-REPORT=procedure_name]
[AFTER-REPORT=procedure_name]
[BEBFORE-PAGE=procedure_name]
[AFTER-PAGE=procedure_name]
\end{verbatim}

\textbf{Arguments}

\begin{description}
\item[BEFORE-REPORT]
A procedure to execute at the time of the first \texttt{PRINT} command. It may be used, for example, to create a report heading.
\item[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.
\item[BEFORE-PAGE]
A procedure to execute at the beginning of every page, just before the first \texttt{PRINT} command for the page. It can be used, for example, to set up page totals.
\item[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.
\end{description}
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>
<thead>
<tr>
<th>Type</th>
<th>Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers</td>
<td>0, 8, 9</td>
</tr>
<tr>
<td>Alphabetical</td>
<td>b, B</td>
</tr>
<tr>
<td></td>
<td>e, E</td>
</tr>
<tr>
<td></td>
<td>n, N</td>
</tr>
<tr>
<td></td>
<td>r, R</td>
</tr>
<tr>
<td></td>
<td>v, V</td>
</tr>
<tr>
<td>Symbols</td>
<td>., ,</td>
</tr>
<tr>
<td></td>
<td>-, +</td>
</tr>
<tr>
<td></td>
<td>!, *</td>
</tr>
<tr>
<td></td>
<td>- ,</td>
</tr>
<tr>
<td></td>
<td>&lt;, &gt;</td>
</tr>
<tr>
<td></td>
<td>(, )</td>
</tr>
</tbody>
</table>

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)
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:

```plaintext
begin-setup
  dollar-symbol £    ! Define £ as the currency symbol
end-setup
begin-procedure    ! If #Amount=1234.56
  ...              ! If #Amount=1234.56
  money-symbol £
  print #Amount () Edit £££,999.99    ! Prints as: £1,234.56
  ...              ! If #Amount=1234.56
  money-symbol $
  print #Amount () Edit £££,999.99    ! Prints as: $1,234.56
  ...              ! If #Amount=1234.56
  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

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

```plaintext
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:

```plaintext
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**

```plaintext
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
#ifndef, 187
#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-dbc, 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_DEBUG, 24
-EH_CS, 23
-EH_CS, 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
-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
custom 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
DECLARE-PRINT, 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
gettext, 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
HORIZ-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
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
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
MONTHS-FULL argument, ALTER-LOCALE command, 44, 336
MONTHS-SHORT argument, ALTER-LOCALE command, 44
MULTIPLY, 228
MOVE, 224
multiple reports, 49
NEED argument, NEXT-LISTING command, 233
nesting
  arguments, 197
  IF command, 186
NAME, 39
NAME argument, 85
  CREATE-ARRAY command, 92
  LOAD-LOOKUP command, 219
natural log base e raised to x power, 199
NEED base e raised to x power, 199
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
nvl, 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
INDEX

A B C D E F G H I K L M N O P Q R S T V W Z

PRINT-IMAGE, 271
position, PRINT-IMAGE command, 271
PRINT-IMAGE, 139
position, PRINT-IMAGE command, 271
PRINT-IMAGE, 270

powers, 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
#sql-count, 17
#return-status, 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

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
  ascii, 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
translate, 208
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

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