

Hyperion® Performance Suite™

Release 8.3



Hyperion SQR Language Reference



Hyperion®

Hyperion Solutions Corporation

D840183000

Copyright © 1994–2004 Hyperion Solutions Corporation.

All rights reserved.

“Hyperion” and Hyperion's product names are trademarks of Hyperion. References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

No portion of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without the express written permission of Hyperion.

The information contained in this manual is subject to change without notice. Hyperion shall not be liable for errors contained herein or consequential damages in connection with the furnishing, performance, or use of this material.

This software described in this manual is licensed exclusively subject to the conditions set forth in the Hyperion license agreement. Please read and agree to all terms before using this software.

GOVERNMENT RIGHTS LEGEND: Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Hyperion license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14, as applicable.

Hyperion Solutions Corporation
1344 Crossman Avenue
Sunnyvale, California 94089

Printed in the U.S.A.

Contents in Brief

Preface

CHAPTER 1 Introduction

CHAPTER 2 SQR Command Reference

CHAPTER 3 HTML Procedures

CHAPTER 4 Encoding in SQR

CHAPTER 5 SQR.INI

CHAPTER 6 SQR Samples

CHAPTER 7 SQR Messages

CHAPTER 8 Deprecated Information

Index

Contents

Preface

Purpose	xiii
Audience	xiii
Document Structure	xiv
Related Documents	xiv
Where to Find Documentation	xv
Conventions	xvi
Additional Support	xvii
Documentation Feedback	xviii

CHAPTER 1 Introduction

About the SQR Language	1-2
SQR Language Program Structure	1-3
SQR Lanugage Syntax Conventions	1-4
SQR Lanugage Syntax Abbreviation Conventions	1-5
Rules for Entering SQR Commands	1-6
SQR Data Elements	1-7
Columns	1-7
Literals	1-7
Variables	1-7
Variable Rules	1-8
List Variables	1-8
SQR Reserved Variables	1-10
SQR Command Line	1-13
SQR Command-line Arguments	1-13
SQR Command-line Flags	1-15

CHAPTER 2 SQR Command Reference

ADD	2-2
ALTER-COLOR-MAP	2-3
ALTER-CONNECTION	2-5
ALTER-LOCALE	2-9
ALTER-PRINTER	2-18
ALTER-REPORT	2-20
ARRAY-ADD	2-28
ARRAY-DIVIDE	2-28
ARRAY-MULTIPLY	2-28
ARRAY-SUBTRACT	2-28
ASK	2-31
BEGIN-DOCUMENT	2-32
BEGIN-EXECUTE	2-33
BEGIN-FOOTING	2-39
BEGIN-HEADING	2-41
BEGIN-PROCEDURE	2-43
BEGIN-PROGRAM	2-46
BEGIN-SELECT	2-47
BEGIN-SETUP	2-52
BEGIN-SQL	2-54
BREAK	2-58
CALL	2-59
CALL SYSTEM	2-59
CLEAR-ARRAY	2-66
CLOSE	2-67
COLUMNS	2-68
COMMIT	2-69
CONCAT	2-71
CONNECT	2-73
CREATE-ARRAY	2-75
CREATE-COLOR-PALETTE	2-78
CREATE-LIST	2-80
#DEBUG	2-81

DECLARE-CHART	2-83
Attributes Argument	2-98
Selector/Sub-Selector Keywords	2-99
Declaration Keywords	2-100
DECLARE-COLOR-MAP	2-107
DECLARE-CONNECTION	2-109
DECLARE-IMAGE	2-112
DECLARE-LAYOUT	2-115
DECLARE-PRINTER	2-123
DECLARE-PROCEDURE	2-130
DECLARE-REPORT	2-132
DECLARE-TOC	2-134
DECLARE-VARIABLE	2-136
#DEFINE	2-140
DISPLAY	2-142
DIVIDE	2-146
DO	2-148
#ELSE	2-150
ELSE	2-151
ENCODE	2-152
END-DECLARE	2-153
END-DOCUMENT	2-153
END-EVALUATE	2-153
END-FOOTING	2-153
END-HEADING	2-153
#END-IF	2-155
#ENDIF	2-155
END-IF	2-156
END-PROCEDURE	2-157
END-PROGRAM	2-157
END-SELECT	2-157
END-SETUP	2-157
END-SQL	2-157
END-WHILE	2-157
END-EXECUTE	2-157

EVALUATE	2-159
EXECUTE	2-162
EXIT-SELECT	2-168
EXTRACT	2-169
FIND	2-171
GET	2-173
GET-COLOR	2-175
GOTO	2-177
GRAPHIC BOX	2-178
GRAPHIC HORZ-LINE	2-179
GRAPHIC VERT-LINE	2-180
#IF	2-181
IF	2-183
#IFDEF	2-185
#IFNDEF	2-186
#INCLUDE	2-187
INPUT	2-188
LAST-PAGE	2-192
LET	2-193
Operands	2-194
Operators	2-195
Bit-Wise Operators	2-196
Functions	2-197
Numeric Functions	2-199
File-Related Functions	2-206
String Functions	2-207
Date Functions	2-214
Unicode Functions	2-216
Miscellaneous Functions	2-220
Writing Custom Functions	2-225
LOAD-LOOKUP	2-228
LOOKUP	2-235
LOWERCASE	2-236
MBTOSBS	2-237
MOVE	2-238
MULTIPLY	2-243

NEW-PAGE	2-244
NEW-REPORT	2-245
NEXT-COLUMN	2-246
NEXT-LISTING	2-248
OPEN	2-250
PAGE-NUMBER	2-253
POSITION	2-254
PRINT	2-255
BOLD	2-257
BOX	2-257
CENTER	2-257
CODE-PRINTER	2-258
DATE	2-259
DELAY	2-259
EDIT	2-260
Text Edit Format Characters	2-260
Numeric Edit Format Characters	2-260
Date Edit Format Characters	2-262
Edit Masks	2-264
Sample Edit Masks	2-266
Uses of Edit Masks	2-267
Edit Masks with Specified Width Value	2-268
Default Formats	2-270
FILL	2-271
FONT	2-271
FOREGROUND/BACKGROUND	2-271
MATCH	2-272
MONEY	2-273
NOP	2-273
NUMBER	2-273
ON-BREAK	2-273
POINT-SIZE	2-277
SHADE	2-277
UNDERLINE	2-277
URL	2-277
URL-TARGET	2-278
WRAP	2-278
PRINT-BAR-CODE	2-281

PRINT-CHART	2-284
PRINT-DIRECT	2-294
PRINT-IMAGE	2-295
PUT	2-298
READ	2-300
ROLLBACK	2-303
SBTOMBS	2-304
SECURITY	2-305
SET-COLOR	2-307
SET-DELAY-PRINT	2-309
SHOW	2-310
STOP	2-316
STRING	2-317
SUBTRACT	2-319
TOC-ENTRY	2-320
UNSTRING	2-321
UPPERCASE	2-323
USE	2-324
USE-COLUMN	2-325
USE-PRINTER-TYPE	2-326
USE-PROCEDURE	2-327
USE-REPORT	2-329
WHILE	2-330
WRITE	2-332

CHAPTER 3 HTML Procedures

HTML General Purpose Procedures	3-2
HTML Heading Procedures	3-5
HTML Highlighting Procedures	3-7
HTML Hypertext Link Procedures	3-10
HTML List Procedures	3-11
HTML Table Procedures	3-15

CHAPTER 4 Encoding in SQR

Encoding Methods	4-2
Encoding Keys in the SQR.INI File	4-2
Encoding Keys in the [Default-Settings] Section	4-2
UseUnicodeInternal Key	4-3
AutoDetectUnicodeFiles Key	4-3
Substitution-Character Key	4-3
Encoding Keys in the [Environment] Section	4-4
Encodings Supported without Using Unicode Internally	4-6
Encodings Supported in SQR	4-8

CHAPTER 5 SQR.INI

Installation of SQR.INI	5-2
For the Windows Platforms Only	5-2
For All Other Platforms	5-3
[Default-Settings] Section	5-4
[Environment: environment] Section	5-9
Using the Java Virtual Machine	5-9
DDO Variables in the [Environment] Section	5-10
Encoding Keys in the [Environment] Section	5-10
[SQR Extension] Section	5-11
[Locale:local-name] Section	5-12
[Fonts] Section	5-14
Adding [Fonts] Entries	5-14
Specifying Character Sets in Windows	5-15
[PDF Fonts] Section	5-16
Embedding Fonts	5-16
Available Fonts	5-16
[PDF Settings] Section	5-18
[HTML:Images] Section	5-20
[Enhanced-HTML] Section	5-21
[Color Map] Section	5-22
[MAP-ODBC-DB] Section	5-23
[MAP-DDO-DB] Section	5-23
[SQR Remote] Section	5-24

CHAPTER 6 SQR Samples**CHAPTER 7 SQR Messages**

Unnumbered Messages	7-2
Numbered Messages	7-5

CHAPTER 8 Deprecated Information

Deprecated SQR Command-line Flags	8-2
Depreceted SQR.INI Entries	8-2
Values for the FullHTML Keyword in the [Enhanced-HTML] Section	8-2
[Processing-Limits] Section	8-3
Values for PDFCompressionText and PDFCompressionGraphics in the [Default-Settings] Section	8-4
Deprecated Transforms	8-5
Deprecated SQR Commands	8-6
BEGIN-REPORT	8-7
DATE-TIME	8-8
DECLARE PRINTER	8-10
DECLARE PROCEDURE	8-14
DOLLAR-SYMBOL	8-16
GRAPHIC FONT	8-18
MONEY-SYMBOL	8-19
NO-FORMFEED	8-21
PAGE-SIZE	8-22
PRINT ...CODE	8-24
PRINTER-DEINIT	8-25
PRINTER-INIT	8-26

Index

Preface

Welcome to the *Hyperion SQR Language Reference*. This preface discusses the following topics:

- “Purpose” on page xiii
- “Audience” on page xiii
- “Document Structure” on page xiv
- “Related Documents” on page xiv
- “Where to Find Documentation” on page xv
- “Conventions” on page xvi
- “Additional Support” on page xvii
- “Documentation Feedback” on page xviii

Purpose

This guide describes the structure, command set, and syntax of the SQR language. It also discusses HTML procedures and encoding in SQR, provides a directory to the library of sample SQR programs, gives an overview of the SQR initialization file, and provides a listing of SQR messages.

Audience

This guide is intended for SQR and SQL developers who must report on data from a wide range of enterprise datasources. Before using this reference, familiarize yourself with the datasources you will be reporting from as well as the connectivity between those datasources and your operating system(s).

Document Structure

This document contains the following information:

Chapter 1, “Introduction” introduces you to the SQR language and describes the SQR command-line arguments, command-line flags, and data elements.

Chapter 2, “SQR Command Reference” describes the SQR commands, and provides examples of how to use the commands.

Chapter 3, “HTML Procedures” describes how to generate HTML output with SQR.

Chapter 4, “Encoding in SQR” describes the encoding methods and lists the encodings used by SQR to support many languages, including Japanese, Chinese, and Korean.

Chapter 5, “SQR.INI” provides a detailed description of the SQR.INI file and how to modify it for custom SQR applications.

Chapter 6, “SQR Samples” provides a directory to the library of sample SQR programs and output included on your installation media.

Chapter 7, “SQR Messages” provides two tables of SQR-generated messages, their interpretations, and suggestions for appropriate action to take.

Chapter 8, “Deprecated Information” provides a list of commands, command-line flags, and sections in the SQR.INI file that have been phased out of SQR. Where appropriate, suitable alternatives are discussed.

Index contains a list of Hyperion SQR terms and their page references.

Related Documents

In addition to the *Hyperion SQR Language Reference*, Hyperion SQR includes the following documentation:

Hyperion SQR Getting Started Guide – Gives a complete overview of the Hyperion SQR product, and provides procedures and code examples to help you develop SQR reports. This guide provides the framework for mastering Hyperion SQR’s capabilities.

Designing Reports with Hyperion SQR Developer – Explains how to create SQR reports on Windows systems using Hyperion SQR Developer, Hyperion SQR’s GUI report and design layout tool.

Using Hyperion SQR Activator – Explains how to integrate SQR into a business application using an application development environment (such as Visual Basic) that supports Microsoft ActiveX Controls to satisfy production and enterprise reporting needs.

Accessing Data with Hyperion SQR for DDO – Provides examples and instructions for accessing data from a DDO application. This guide discusses DDO basics, creating DDO applications, writing a DDO driver, programming considerations, and DDO common utilities.

Hyperion SQR Installation Guide – Contains installation instructions and configuration information specific to your platform.

New Features in Hyperion SQR Release 8.3 – Describes updates, enhancements, and new features.

Online Help – Provides information on the SQR commands, command-line flags, command-line arguments, functions, variables, html procedures, error messages, and the initialization file.

Where to Find Documentation

All Hyperion SQR documentation is accessible from the following locations:

- Online help is accessible from the product that it documents. Start the product and click the Help button or menu command.
- The Brio Software Web site is located at <http://www.hyperion.com>.
- Access to the Hyperion Download Center is through <http://hyperion.subscribenet.com>.

To access documentation from the Brio Software Web site:

- 1** Log on to <http://www.hyperion.com>.
- 2** Select the Support link and type your username and password to log on.



Note New users must register to receive a username and password.

- 3** Follow the on-screen instructions.

To access documentation from the Hyperion Download Center:

- 1** Log on to <http://hyperion.subscribenet.com>.
- 2** In the Login ID and Password text boxes, enter your assigned login ID name and password. Then click Login.
- 3** If you are a member on multiple Hyperion Download Center accounts, select the account that you want to use for the current session.
- 4** Follow the on-screen instructions.

To order printed documentation:

- 1** Visit the Brio Software Web site at <http://www.hyperion.com>.
- 2** In the United States, call Brio Software Customer Support at 877-901-4975.
- 3** From outside the United States, including Canada, call Brio Software Customer Support at 203-703-3600. Clients who are not serviced by support from North America should call their local support centers.

Conventions

The following table shows the conventions that are used in this document:

Table 1 Conventions Used in This Document

Item	Meaning
►	Arrows indicate one-step procedures.
Bold	Bold highlights options, buttons, or tabs that you need to choose and text that you need to type.
CAPITAL LETTERS	Capital letters denote commands and various IDs. For example: CLEARBLOCK command
[Ctrl + 0]	Keystroke combinations shown with the plus sign (+) indicate that you should press the first key and hold it while you press the next key. Do not type the plus sign.

Table 1 Conventions Used in This Document (*Continued*)

Item	Meaning
Example text	Courier font indicates that the example text is code or syntax. Courier font is also used for file names, directory/folder names, and path names.
	For example: Sample1.bqy is located in the \\HYPERION_HOME\Samples directory.
<i>Courier italics</i>	Courier italic text indicates a variable field in command syntax. Substitute a value in place of the variable shown in Courier italics.
Mouse orientation	This document provides examples and procedures using a right-handed mouse. If you use a left-handed mouse, adjust the procedures accordingly.
Menu options	Options in menus are shown in the following format. Substitute the appropriate option names in the placeholders, as indicated. Menu name→Menu command→Extended menu command For example: 1. Select File→Desktop→Accounts .

Additional Support

In addition to providing documentation and online help, Hyperion offers the following product information and support. For details on education, consulting, or support options, visit Hyperion's Web site at <http://www.hyperion.com>.

Education Services

Hyperion offers instructor-led training, custom training, and eTraining covering all Hyperion applications and technologies. Training is geared to administrators, end users, and information systems (IS) professionals.

Consulting Services

Experienced Hyperion consultants and partners implement software solutions tailored to clients' particular reporting, analysis, modeling, and planning requirements. Hyperion also offers specialized consulting packages, technical assessments, and integration solutions.

Technical Support

Hyperion provides enhanced electronic-based and telephone support to clients to resolve product issues quickly and accurately. This support is available for all Hyperion products at no additional cost to clients with current maintenance agreements.

Documentation Feedback

Hyperion strives to provide complete and accurate documentation. We value your opinions on this documentation and want to hear from you. Give us your comments by going to http://www.hyperion.com/services/support_programs/doc_survey/index.cfm.

1

Introduction

The *Hyperion SQR Language Reference* describes all the elements of the SQR language that allow you to create clear, elegant reports from complex arrays of information systems.

This chapter gives you an overview of the SQR language, its program structure, syntax conventions, and the rules for entering SQR commands. It also discusses SQR data elements and the SQR command line.

In This Chapter	About the SQR Language	1-2
	SQR Data Elements	1-7
	SQR Command Line	1-13

About the SQR Language

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

The SQR language has several key benefits:

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

With the SQR language, you can design custom reports by defining the page size, headers, footers, and layout. The SQR 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 SQR language provides enable you to add procedural logic and control to datasource calls. You can use the SQR language to write other types of applications, such as database manipulation and maintenance, table load and unload, and interactive query and display.

SQR Language Program Structure

The SQR language processes source code from a standard text file and generates your report. The text file containing source code consists of a simple set of sections that you delimit with `BEGIN-section` and `END-section` commands. The following examples show the general structure of the SQR 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
```

SQR Lanugage Syntax Conventions

The SQR language employs a set of conventions for clearly conveying its syntax. Table 1-1 describes the SQR language syntax conventions.

Table 1-1 SQR Language Syntax Conventions

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

SQR Lanugage Syntax Abbreviation Conventions

Table 1-2 describes the SQR language syntax abbreviation conventions.

Table 1-2 Syntax Abbreviation Conventions

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

Rules for Entering SQR Commands

Use the following command rules as you develop SQR programs:

- SQR commands are *not* case sensitive. Common practice among many SQR programmers is to use upper case for SQR commands, but SQR ignores case as it compiles your 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. (SQR ignores hyphens and carriage returns within commands.)
- Begin each comment line with an exclamation point (!).

To display the ! or ' symbols within your 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.

SQR Data Elements

SQR 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 that you define and manipulate.

- \$ 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 for printing inside a BEGIN-DOCUMENT paragraph.

Variable Rules

The following rules govern the use of variables in SQR:

- Variables can be almost any name of almost any length. For example, \$state_name or #total_cost. For exceptions, see “SQR Reserved Variables” on page 1-10.
- 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.
- SQR 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 one of three types: FLOAT, INTEGER, or DECIMAL. See the command, “DECLARE-VARIABLE” on page 2-136 for a full explanation.
- 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 the command, “BEGIN-PROCEDURE” on page 2-43 for a full explanation.

List Variables

List variables contain an ordered collection of SQR 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 SQR for DDO only.

You can perform the following actions with list variables:

- **Define a list variable** – You can use a list variable to 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** – To 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** – To 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 there are no brackets, 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 SQR 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.

SQR Reserved Variables

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



Note

All of SQR's reserved variables are global variables. Within a local procedure you must reference them with a leading underscore. For example, `#_sqlstatus` or `$_sql-error`. See "BEGIN-PROCEDURE" on page 2-43 for information on defining local procedures.

Table 1-3 describes the SQR reserved variables.

Table 1-3 SQR Reserved Variables

Variable	Description
<code>#current-column</code>	The current column on the page.
<code>\$current-date</code>	The current date-time on the local machine when SQR starts running the program.
<code>#current-line</code>	The current line on the page. This value is the physical line on the page, not the line in the report body. See <i>Hyperion SQR Getting Started Guide</i> .
	Line numbers are referenced in PRINT and other SQR commands used for positioning data on the page. Optional page headers and footers, defined with BEGIN-HEADING and BEGIN-FOOTING commands, have their own line sequences. Line 2 of the heading is different from line 2 of the report body or footing.
<code>#end-file</code>	Set to one (1) if end of file occurs when reading a flat file; otherwise, it is set to zero (0). Your program should check this variable after each READ command. (See "READ" on page 2-300 for more information.)
<code>#page-count</code>	The current page number.
<code>#return-status</code>	Value to be returned to the operating system when SQR exits. Can be set in your report. <code>#return-status</code> is initialized to the "success" return value for the operating system.
<code>#sql-count</code>	The count of rows affected by a DML statement (INSERT, UPDATE, or DELETE). This is equivalent to ROWCOUNT in Oracle and Sybase.

Table 1-3 SQR Reserved Variables (*Continued*)

Variable	Description
<code>\$sql-error</code>	The text message from the database explaining an error. This variable is rewritten when a new error is encountered.
<code>#sql-status</code>	The value of #SQL-STATUS is set whenever a BEGIN-SELECT command 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.
<code>\$sql-text</code>	The last SQL statement sent to the database by SQR.
	The variable contents are valid after SQR processes a BEGIN-SQL, a BEGIN-SELECT, or a LOAD-LOOKUP command, 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.
<code>\$sqr-connected-db</code> <code>{sqr-connected-db}</code>	The class of the backend database from the initial connection or from the CONNECT command. For ODBC, the class is defined to be: ODBC, Sybase, Oracle, Informix, Redbrick, Teradata, or DB2. For DDO, the class is defined to be: XML, CSV, SAP, MSOLAP, Essbase, SQLServer, Sybase, Oracle, Informix, or DB2.
	The information is derived from <code>\$sqr-connected-db-name</code> / <code>{sqr-connected-db-name}</code> .
<code>\$sqr-connected-db-name</code> <code>{sqr-connected-db-name}</code>	The name of the database driver from the initial connection or from the CONNECT command. For ODBC, the value is derived from the ODBC Driver Manager. For DDO, the value is derived from the driver being used. For all other databases, the value is the same as the value in <code>\$sqr-database</code> / <code>{sqr-database}</code> .
<code>\$sqr-database</code> <code>{sqr-database}</code>	The database type for which SQR was compiled. Valid values are: DB2, ODBC, Sybase, Informix, and Oracle.
<code>\$sqr-dbcs</code> <code>{sqr-dbcs}</code>	Specifies whether SQR recognizes double-byte character strings. The value can be either YES or NO.
<code>\$sqr-encoding</code> <code>{sqr-encoding}</code>	The name of the default encoding as defined by the ENCODING environment variable when SQR is invoked.

Table 1-3 SQR Reserved Variables (*Continued*)

Variable	Description
<code>\$sqr-encoding-console</code> <code>{sqr-encoding-console}</code>	Name of encoding for character data written to the log file or console.
<code>\$sqr-encoding-database</code> <code>{sqr-encoding-database}</code>	The character data retrieved from and inserted into the database.
<code>\$sqr-encoding-file-input</code> <code>{sqr-encoding-file-input}</code>	Name of encoding for character data read from files used with the OPEN command.
<code>\$sqr-encoding-file-output</code> <code>{sqr-encoding-file-output}</code>	Name of encoding for character data written to files used with the OPEN command.
<code>\$sqr-encoding-report-output</code> <code>{sqr-encoding-report-output}</code>	The report generated by SQR (for example, an LIS file or a Post-Script file).
<code>\$sqr-encoding-source</code> <code>{sqr-encoding-source}</code>	Name of encoding for SQR source files and include files.
<code>\$sqr-hostname</code> <code>{sqr-hostname}</code>	The name of the computer on which SQR is currently executing.
<code>\$sqr-locale</code>	The name of the current locale being used. A + at the end of the name indicates an argument used in the locale has changed.
<code>#sqr-max-columns</code>	The 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.
<code>#sqr-max-lines</code>	The 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.
<code>#sqr-pid</code>	The process ID of the current SQR process. #sqr-pid is unique for each run of SQR. This variable is useful in creating unique temporary names.
<code>\$sqr-platform</code> <code>{sqr-platform}</code>	The hardware/operating system type for which SQR was compiled. Valid values are WINDOWS and UNIX.
<code>\$sqr-program</code>	The name of the SQR program file.
<code>\$sqr-report</code>	The 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 NEW-REPORT command).
<code>\$sqr-ver</code>	The text string shown with the -ID flag. SQR version.
<code>\$username</code>	The database user name specified on the command line.

SQR Command Line

SQR includes command-line arguments and command-line flags. Review Table 1-4 for detailed information on SQR's command-line arguments. Review Table 1-5 for detailed information on SQR's command-line flags.

SQR Command-line Arguments

Table 1-4 describes the SQR command-line arguments.

Table 1-4 SQR Command-line Arguments

Argument	Description
<i>program</i>	The name of the text file containing source code. The default file type or extension is .sqr. If entered as “?” or omitted, SQR prompts you for the report program name. On UNIX-based systems, if your shell uses the question character as a WILD CARD character, you must precede it with a backslash (\).
<i>flags</i>	Any of the flags listed in Table 1-5.
<i>args...</i>	Arguments used by SQR while the program is running. Arguments listed here are used by the ASK and INPUT commands rather than prompting the user. Arguments must be entered on the command line in the same sequence they are expected by the program – first all ASK arguments in order and then INPUT arguments.
<i>@file...</i>	File containing program arguments, one argument per line. Arguments listed in the file are processed one at a time – first all ASK arguments in order and then INPUT arguments. The command line arguments program, connectivity, and args can be specified in this file for non-Windows platforms.
<i>connectivity</i>	The information needed by SQR to connect to the database. If entered as “?” or omitted, SQR prompts you for it.
AS400/DB2	<i>Ssname/SQLid</i> <ul style="list-style-type: none">■ <i>Ssname</i> - The subsystem name.■ <i>SQLid</i> - The SQL authorization ID to use.
Unix/Windows/DB2	[<i>Database</i>]/[<i>Username</i>]/ [<i>Password</i>] <ul style="list-style-type: none">■ <i>Database</i> - The name of the database to use.■ <i>Username</i> - Your user name for the database.■ <i>Password</i> - Your password for the database.

Table 1-4 SQR Command-line Arguments (*Continued*)

Argument	Description
Informix	<i>Database[/username/password]</i> <ul style="list-style-type: none">■ <i>Database</i> – The name of the database to use.■ <i>Username</i> – Your user name for the database.■ <i>Password</i> – Your password for the database.
ODBC	<i>Data_Source_Name/[Username]/[Password]</i> <ul style="list-style-type: none">■ <i>Data_Source_Name</i> – The name you give to the ODBC driver when you setup the driver.■ <i>Username</i> – Your user name for the database.■ <i>Password</i> – Your password for the database.
	This port has been certified against Microsoft SQL Server.
Oracle	<i>[Username]/[Password[@Database]]</i> <ul style="list-style-type: none">■ <i>Username</i> – Your user name for the database.■ <i>Password</i> – Your password for the database.■ <i>Database</i> – Optionally, you can specify the connection string for the database (for example, @sales.2cme.com).
Sybase	<i>Username/[Password]</i> <ul style="list-style-type: none">■ <i>Username</i> – Your user name for the database.■ <i>Password</i> – Your password for the database.
Teradata	<i>[tdpid/]username[,password]</i> <ul style="list-style-type: none">■ <i>tdpid</i> – Your Teradata tdp name.■ <i>username</i> – Your user name for the database.■ <i>password</i> – Your password for the database.

SQR Command-line Flags

SQR supports several command-line flags. Each flag begins with a dash (-). When a flag takes an argument, the argument must follow the flag with no intervening space.

Table 1-5 describes the SQR command-line flags.

Table 1-5 SQR Command-line Flags

Flag	Description	Program	Database
-A	Causes the output to be appended to an existing output file carrying the same name as the output's source. If the file does not exist, a new one is created. This is useful when you want to run the same report more than once but only want to create a single output file. Note: The -A flag has the following restrictions: <ul style="list-style-type: none">■ It only works with LIS files. It does not work with SPF files.■ It only applies to reports for -PRINTER:LP. It is ignored for all other printer types and output formats.■ In non-Windows environments, it can only be used with SQR and SQRP. It does not work with SQRWP.	SQR SQR Execute SQR Print	All
-Bnn	Indicates the number of rows to buffer each time SQR retrieves data from the database. The default is 10 rows. Regardless of the setting, all rows are retrieved. When used on the command line, -B controls the setting for all BEGIN-SELECT commands. Within a program, each BEGIN-SELECT command can also have its own -B flag for further optimization.	SQR SQR Execute	ODBC Oracle Sybase

Table 1-5 SQR Command-line Flags (*Continued*)

Flag	Description	Program	Database
-BURST:{xx}	<p>Specifies the type of bursting to be performed. (See Chapter 31, “Working with HTML” in <i>Hyperion SQR Getting Started Guide</i> for additional information.)</p> <p>-BURST:T generates the Table of Contents file only.</p> <p>-BURST:S generates the report output according to the symbolic Table of Contents entries set in the program with the TOC-ENTRY command’s <i>level</i> argument. In -BURST:S[{l}], {l} is the level at which to burst upon. The setting -BURST:S is equivalent to -BURST:S1.</p> <p>-BURST:P generates the report output by report page numbers. In -BURST:P[{l} , {s} [, {s}....]] , {l} is the number of logical report pages that each HTM file will contain and {s} is the page selection: {n}, {n}-{m}, -{m}, or {n}-. The setting -BURST:P is equivalent to -BURST:P0,1- when using -PRINTER:HT or -BURST:P1 when using -PRINTER:EH.</p> <p>See ““Bursting” and Demand Paging” on page 31-20 in <i>Hyperion SQR Getting Started Guide</i> for more information.</p> <p>Note: -BURST:P and -BURST:S require -PRINTER:EH or -PRINTER:HT. The Page range selection feature of -BURST:P requires -PRINTER:HT. -BURST:T requires -PRINTER:HT.</p>	SQR SQR Execute SQR Print	All
-C	(Windows) Specifies that the Cancel dialog box appears while the program runs so you can easily terminate the program execution.	SQR SQR Execute	All
-CB	(Windows, Callable SQR) Forces the communication box to be used.	SQR	All
-Dnn	(non-Windows) Causes SQR to display the report output on the terminal at the same time it is written to the output file. <i>nn</i> is the maximum number of lines to display before pausing. If no number is entered after -D, the display scrolls continuously.	SQR SQR Execute SQR Print	All
	<p>Note: The printer type must be <i>LP</i> or the display is ignored. If the program produces more than one report, the display is for the first report only.</p>		
-DB <i>database</i>	Causes the SQR program to use the specified database, which overrides any USE command in the SQR program.	SQR SQR Execute	Sybase
-DEBUG[xxx]	Causes lines preceded by #DEBUG to be compiled. Without this flag, SQR ignores these lines. See “#DEBUG” on page 2-81 for more information.	SQR	All

Table 1-5 SQR Command-line Flags (*Continued*)

Flag	Description	Program	Database
-DNT:{xx}	Specifies 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 2-136 for a detailed explanation. If used, the DEFAULT argument in the DECLARE-VARIABLE command takes precedence.	SQR	All
-E[file]	Causes SQR to direct error messages to the named file, or to the default file <i>program.err</i> . If no errors occur, no file is created.	SQR SQR Execute SQR Print	All
-EH_APPLETS: <i>dir</i>	Specifies the directory location of the Enhanced HTML applets. If you include an applet, SQR needs to know where it resides. SQR usually looks for the applet in a default directory. The default directory for these applets is IMAGES. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_BQD	Generates a {report}.bqd file from the report data. Also associates a BQD (Brio Query Format File) icon with {report}.bqd in the navigation bar. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_BQD: <i>file</i>	Associates the BQD icon with the specified file. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_BROWSER:xx	Specifies the browser and generates the appropriate HTML. Acceptable values include: <ul style="list-style-type: none">■ BASIC – SQR generates HTML suitable for all browsers.■ IE – SQR generates HTML designed for Internet Explorer.■ NETSCAPE – SQR generates HTML designed for Netscape.■ ALL – If necessary, SQR generates Basic, IE, and Netscape HTML files. Report_frm.htm contains Javascript to “sense” the browser on the user’s machine and displays the appropriate version. (In this case, the user’s machine is the machine of the person reading the report, not the person writing it.) Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified. In addition, the flag is only recognized when combined with the -EH_FULLHTML flag.	SQR SQR Execute SQR Print	All

Table 1-5 SQR Command-line Flags (*Continued*)

Flag	Description	Program	Database
-EH_CSV	Generates a {report}.csv file from the report data. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_CSV:file	Associates the CSV icon with the specified file. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_CCSVONLY	Creates a CSV file but does not create an HTML file. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_DEBUG[:opts]	Produces a DBG output file that contains compiler and internal error messages. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_FULLHTML:xx	Specifies the level of HTML that your browser supports so appropriate Enhanced HTML code is generated. Acceptable values for this flag are: ■ 40 - Generates XHTML 1.1 ■ 32 - Generates HTML 3.2 The following values are deprecated (no longer valid but still accepted for upward compatibility): ■ 30 - Was used to specify HTML 3.0 ■ TRUE - Was used to specify HTML 3.2 ■ 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.	SQR SQR Execute SQR Print	All
-EH_ICONS:dir	Specifies the directory where HTML should look for the referenced icons. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All

Table 1-5 SQR Command-line Flags (*Continued*)

Flag	Description	Program	Database
-EH_IMAGES: <i>dir</i>	Specifies the directory path for the GIF files used by the Navigation Bar. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_KEEP	Copies (does not move) the files when used in conjunction with -EH_ZIP. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_LANGUAGE: <i>xx</i>	Sets the language used for the HTML navigation bar. You can specify English, French, German, Portuguese, Spanish, Japanese, or Simplified Chinese. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_PDF	Associates a PDF icon with {report}.pdf in the navigation bar. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_SCALE:{ <i>nn</i> }	Sets the scaling factor from 50 to 200. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_XIMG	Specifies to not remove the directory path from the IMAGE reference. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_XML: <i>file</i>	Associates the XML icon with the specified file. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All
-EH_ZIP[: <i>file</i>]	Moves the generated files to the specified file or {report}.zip if {file} is not specified. Note: This flag is only applicable when either the -PRINTER:EH or the -PRINTER:EP flag is specified.	SQR SQR Execute SQR Print	All

Table 1-5 SQR Command-line Flags (*Continued*)

Flag	Description	Program	Database									
-F[file directory]	Overrides the default output file name, <i>program.lis</i> . The default action places the <i>program.lis</i> file in the same directory as the <i>program.sqr</i> file. To use the current directory, specify -F without an argument. To change the name of the output file, specify -F with the new name. If the new name does not specify a directory, the file is created in the current directory. The output file is not created until data actually prints on the page. If no data prints, no output file is created. The following table shows how to specify the file name and directory for different operating systems.	SQR SQR Execute SQR Print	All									
	<table><thead><tr><th>Operating System</th><th>Directory Character</th><th>Example</th></tr></thead><tbody><tr><td>UNIX</td><td>/</td><td>-F\$HOME/reports/</td></tr><tr><td>Windows</td><td>\</td><td>-FC:\Brio\Files\</td></tr></tbody></table>	Operating System	Directory Character	Example	UNIX	/	-F\$HOME/reports/	Windows	\	-FC:\Brio\Files\		
Operating System	Directory Character	Example										
UNIX	/	-F\$HOME/reports/										
Windows	\	-FC:\Brio\Files\										
-ID	(non-Windows) Displays the copyright banner on the console.	SQR SQR Execute SQR Print	All									
-KEEP	Creates SPF file output in addition to LIS files for each report that your program generates. See Chapter 28, “Printing Issues,” in <i>Hyperion SQR Getting Started Guide</i> for more information on LIS and SPF files.	SQR SQR Execute	All									
-ldir_list	Specifies the list of directories that SQR searches when processing the #INCLUDE directive if the include file does not exist in the current directory and no path was specified for the file. The directory names must be separated by either commas (,) or semicolons (;). For UNIX-based systems, if your shell uses semicolons as command delimiters, you must precede each semicolon with a backslash (\). Always append the directory character to the end of each directory name. See the -F flag for a list of directory characters by operating system. For example, under UNIX: sqr myreport sammy/baker - I/home/sqr/inc/,/usr/sqr/incl/	SQR	All									
-LL{s d}{c i}	LOAD-LOOKUP: s = SQR, d = DB, c = case-sensitive, i = case-insensitive See the command, “LOAD-LOOKUP” on page 2-228 for more information.	SQR	All									
-NOLIS	Prevents the creation of all SQR output file types. SPF output is created instead.	SQR SQR Execute	All									

Table 1-5 SQR Command-line Flags (*Continued*)

Flag	Description			Program	Database			
-O[file]	Directs log messages to the specified file or to <i>program.log</i> if no file is specified. By default, the file <i>sqr.log</i> is used in the current working directory.			SQR SQR Execute	All			
-PB	Causes column data to retain trailing blanks.			SQR SQR Execute	Informix			
-PRINTER:xx	Causes printer type xx to be used when creating output files.			SQR SQR Execute SQR Print	All			
xx Printer Type Example								
EH Enhanced HTML -PRINTER:EH								
EP Enhanced HTML/PDF -PRINTER:EP								
HP HP LaserJet -PRINTER:HP								
HT HTML 2.0 -PRINTER:HT								
LP Line Printer -PRINTER:LP								
PD PDF -PRINTER:PD								
PS PostScript -PRINTER:PS								
WP Windows -PRINTER:WP								

Table 1-5 SQR Command-line Flags (*Continued*)

Flag	Description	Program	Database
Notes:			
	<ul style="list-style-type: none">■ Types LP, HP, and PS produce files with the .lis extension.■ Types EH and HT produce .htm file output.<ul style="list-style-type: none">□ HT is controlled by the PrinterHT setting in the [Default-Settings] section of the SQR.INI file. If the PrinterHT setting is set to <i>standard</i>, HT produces version 2.0 HTML files with the report content inside of <PRE></PRE> tags. If the PrinterHT setting is set to <i>enhanced</i>, HT is mapped to EH. (See “PrinterHT” on page 5-8 for additional information.)□ EH produces reports in which content is fully formatted with version 1.1 XHTML tags.■ Type PD produces a PDF 1.3 compliant file. When -PRINTER:PD is used, PRINT-DIRECT, PRINT ...Code, and Print with CODE-PRINTER commands are processed but ignored.■ On Windows systems, the WP extension sends the output to the default Windows printer. To specify a non-default Windows printer, use the following format: -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 <i>NewPrinter</i>, you could use -PRINTER:WP:<i>NewPrinter</i>. If your printer name has spaces, enclose the entire command in double quotes.		
-RS	Saves the program in a run-time file. The program is scanned, compiled, and checked for correct syntax. Queries are validated and compiled. Then, the executable version is saved in a file with the name <i>program.sqt</i> . Note that SQR does not prompt for ASK variables after compilation.	SQR	All
-RT	Uses the run-time file saved with the -RS flag. Skips all syntax and query checking and begins processing immediately. Note that SQR does not prompt for ASK variables after compilation.	SQR	All
-S	Requests that the status of all cursors display 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.	SQR SQR Execute	All

Table 1-5 SQR Command-line Flags (*Continued*)

Flag	Description	Program	Database
-Tnn	Specifies that you want to test your report for <i>nn</i> pages. To save time during testing, SQR ignores all ORDER BY clauses in SELECT statements. If the program produces more than one report, SQR stops after the specified number of pages defined for the first report have been output.	SQR SQR Execute	All
-T{B}	Trims trailing blanks from database character columns.	SQR SQR Execute	DB2 Sybase ODBC Teradata
-T{Z}	Trims trailing zeros from the decimal portion of numeric columns.	SQR SQR Execute	DB2 Teradata
-Vserver	Uses the named server.	SQR SQR Execute	Sybase
-XB	(non-Windows) Suppresses the SQR banner and the “SQR... End of Run” message.	SQR SQR Execute SQR Print	All
-XC	(Callable SQR) Suppresses the database commit when the report has finished running.	SQR	All
-XCB	(Windows) Do not use the communication box. Requests for input are made in Windows dialog boxes.	SQR SQR Execute	All
-XI	Prevents user interaction during a program run. If an ASK or INPUT command requires user input, an error displays and the program ends.	SQR SQR Execute	All
-XL	Prevents SQR from logging onto the database. Programs run in this mode cannot contain any SQL statements. -XL lets you run SQR without accessing the database. You still must supply at least an empty slash (/) on the command line as a placeholder for the connectivity information. For example: <code>sqr myprog / -xl</code>	SQR SQR Execute	All
-XLFF	Prevents trailing form feed.	SQR SQR Execute	All
-XMB	(Windows) Disables the error message display so that you can run a program without interruption by error messages/warnings generated by SQR, or by user generated messages (SHOW/DISPLAY). All messages will still go to their designated output files (SQR.ERR or -E{filename} / SQR.LOG or -O{filename})	SQR	All

Table 1-5 SQR Command-line Flags (*Continued*)

Flag	Description	Program	Database
-XNAV	Prevents SQR from creating the <i>Navigation Bar</i> in HTM files generated with -PRINTER:HT. This occurs when only a single HTM file is produced. Multiple HTM files generated from a single report always contain the <i>Navigation Bar</i> .	SQR SQR Execute SQR Print	All
-XP	Prevents SQR from creating temporary stored procedures. See "BEGIN-SELECT" on page 2-47 for more information.	SQR SQR Execute	Sybase
-XTB	Preserves the trailing blanks in a LIS file at the end of a line.	SQR SQR Execute SQR Print	All
-XTOC	Prevents SQR from generating the Table of Contents for the report. This flag is ignored when -PRINTER:EH, -PRINTER:HT, or PRINTER:EP is also specified. -XTOC only applies to HTML files.	SQR SQR Execute SQR Print	All
-ZEN{name}	Sets the default encoding name.	SQR SQR Execute SQR Print	All
-ZIF{file}	Sets the full path and name of the SQR initialization file, SQR.INI.	SQR SQR Execute SQR Print	All
-ZIV	Invokes the SPF Viewer after generating the <i>program.spf</i> file. This flag implicitly invokes the -KEEP flag to create <i>program.spf</i> . In case of multiple output files, only the first report file passes to the SQR Viewer.	SQR SQR Execute	All
-ZMF{file}	Specifies the full path and name of the SQR error message file, <i>sqr-err.dat</i> .	SQR SQR Execute SQR Print	All
-ZRF{file}	(DDO) Sets the full path and name of an alternate <i>registry.propertiesZ</i> file. The following path is a common default path to the <i>registry.properties</i> file on a Windows system: <i>c:\program files\brio\properties\registry.properties</i> The <i>registry.properties</i> file lists datasources that SQR can access. The information in the <i>registry.properties</i> file makes it possible for SQR to access datasources for which DDO drivers have been loaded and configured.	SQR SQR Execute	All

2

SQR Command Reference

This chapter describes and demonstrates each command in the SQR lexicon. The commands in this chapter follow the conventions listed in Table 1-1, “SQR Language Syntax Conventions,” on page 1-4 and use the abbreviations described in Table 1-2, “Syntax Abbreviation Conventions,” on page 1-5.

Caution! If you copy codes directly from the examples in the PDF file, make sure to change the slanted quotes to regular quotes or else you will get an error message.



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	<code>ADD {src_num_lit _var _col} TO dst_num_var [ROUND=nn]</code>
Arguments	<p><i>src_num_lit _var _col</i> A numeric source value added to the contents of <i>dst_num_var</i>.</p> <p><i>dst_num_var</i> A numeric destination variable that contains the result after execution.</p> <p>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.</p>
Description	<p>The source value is added to the destination variable and the result is placed in the destination. The source is always first and the destination is always second.</p> <p>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 adding many numbers in succession. These inaccuracies can appear due to the way floating point numbers are represented by different hardware and software implementations and also due to inaccuracies that can be introduced when converting between floating point and decimal.</p>
Examples	<pre>add 10 to #counter add #counter to #new_count add &price to #total round=2</pre>
See Also	The LET command for information on complex arithmetic expressions.

ALTER-COLOR-MAP

Function Dynamically alters a defined color.

Syntax

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

Arguments

NAME
Defines the name of the color to alter. For example, *light blue*.

VALUE
Defines the RGB value of the color to alter. For example, (193, 233, 230).

{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 SQR 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

The ALTER-COLOR-MAP command is allowed wherever the PRINT command is allowed. This command allows you to dynamically alter a defined color. You cannot use this command to define a new 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

The DECLARE-COLOR-MAP, SET-COLOR, and GET-COLOR commands.

ALTER-CONNECTION

Function	Alters the datasource logon parameters prior to logon. Can be used to override the default connection logon parameters.
-----------------	---



Note The ALTER-CONNECTION command is specific to SQR/DDO ports only.

Syntax

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

Arguments

NAME

A user-defined name for describing a datasource connection.

DSN

The logical datasource name as recorded in the *Registry.properties* file.

USER , PASSWORD

Traditional logon semantics.

PARAMETERS

Defines a 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 in its property file.

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

This optional keyword prevents SQR 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 SQR, allowing a single CONNECTION declaration to be used 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

Specifies the dimension hierarchy for the previously-declared dimension.

Consider the following example:

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

In this example, SET-GENERATIONS:

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

To clear values previously established with SET-GENERATIONS, use the following command:

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

SET-LEVELS

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

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

In this example:

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

To clear values previously established with SET-LEVELS, use the following command:

```
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 the SET-MEMBERS command can be a *literal* value only. Consider the following example:

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

In this example, SET-MEMBERS:

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

To clear values previously established with SET-MEMBERS, use the following command:

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

Examples

```
alter-connection  
  name=SAPR3-1  
  password=psswd  
parameters=logon.client=600;logon.ashost=starfish;logon.sysnr=  
0;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

The DECLARE-CONNECTION command.

ALTER-LOCALE

Function	Selects a locale or changes locale parameters used for printing date, numeric, and money data and for data accepted by the INPUT command. A locale is a set of preferences for language, currency, and the presentation of charts and numbers.
-----------------	--

Syntax

```
ALTER-LOCALE  
[ LOCALE={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ NUMBER-EDIT-MASK={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ MONEY-EDIT-MASK={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ DATE-EDIT-MASK={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ INPUT-DATE-EDIT-MASK={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ MONEY-SIGN={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ MONEY-SIGN-LOCATION={txt_var|DEFAULT|SYSTEM|LEFT  
|RIGHT} ]  
[ THOUSAND-SEPARATOR={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ DECIMAL-SEPARATOR={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ DATE-SEPARATOR={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ TIME-SEPARATOR={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ EDIT-OPTION-NA={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ EDIT-OPTION-AM={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ EDIT-OPTION-PM={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ EDIT-OPTION-BC={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ EDIT-OPTION-AD={txt_lit |_var|DEFAULT|SYSTEM} ]  
[ DAY-OF-WEEK-CASE={txt_var|DEFAULT|SYSTEM|UPPER|LOWER  
|EDIT|NO-CHANGE} ]  
[ DAY-OF-WEEK-FULL=({txt_lit1|_var1}...{txt_lit7  
|_var7}) ]  
[ DAY-OF-WEEK-SHORT=({txt_lit1|_var1}...{txt_lit7  
|_var7}) ]  
[ MONTHS-CASE={txt_var|DEFAULT|SYSTEM|UPPER|LOWER|EDIT  
|NO-CHANGE} ]  
[ MONTHS-FULL=({txt_lit1|_var1}...{txt_lit12|_var12}) ]  
[ MONTHS-SHORT=({txt_lit1|_var1}...{txt_lit12|_var12}) ]
```

 **Note**

Many of the settings can have a value of DEFAULT or SYSTEM. For a given setting, specifying DEFAULT retrieves the value from the corresponding setting of the *default* locale as identified in the [Default-Settings] section of the SQR.INI file. Similarly, specifying the keyword SYSTEM retrieves the value from the corresponding setting of the *system* locale. You can alter the system locale using the ALTER-LOCALE command; however, you cannot define it in the SQR.INI file.

LOCALE

Specifies the name of the locale to use. This name must be defined in the SQR.INI file. If this field is omitted, then the current locale is used. The locale name is case-insensitive and is limited to the following character set:
A-Z, 0-9, underscore, or hyphen. The current locale can be determined by printing the reserved variable \$sqr-locale.

NUMBER-EDIT-MASK

Specifies the numeric edit mask to use with the keyword NUMBER in a PRINT, MOVE, SHOW, or DISPLAY command.

MONEY-EDIT-MASK

Specifies the numeric edit mask to use with the keyword MONEY in a PRINT, MOVE, SHOW, or DISPLAY command.

DATE-EDIT-MASK

The default date edit mask to use with the keyword DATE in the PRINT, MOVE, SHOW, or DISPLAY command, or the LET functions datetostr() or strtodate().

INPUT-DATE-EDIT-MASK

The 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.

For more information on Edit Masks, see “PRINT” on page 2-255.

MONEY-SIGN

Specifies the character(s) that replace the \$ or other currency symbol used in edit masks.

MONEY-SIGN-LOCATION

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

THOUSAND-SEPARATOR

Specifies the character to replace the ',' edit character.

DECIMAL-SEPARATOR

Specifies the character to replace the '.' edit character.

DATE-SEPARATOR

Specifies the character to replace the '/' character.

TIME-SEPARATOR

Specifies the character to replace the ':' character.

EDIT-OPTION-NA

Specifies the character(s) to use with the 'na' option.

EDIT-OPTION-AM

Specifies the character(s) to replace 'AM'.

EDIT-OPTION-PM

Specifies the character(s) which to replace 'PM'.

EDIT-OPTION-BC

Specifies the character(s) to replace 'BC'.

EDIT-OPTION-AD

Specifies the character(s) to replace 'AD'.

DAY-OF-WEEK-CASE

Specifies 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 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 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 explicitly listed in the DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT entries.

DAY-OF-WEEK-FULL

Specifies the full names for the days of the week. SQR considers the first day of the week to be Sunday. You must specify all seven days.

DAY-OF-WEEK-SHORT

Specifies the abbreviated names for the days of the week. SQR considers the first day of the week to be Sunday. You must specify all seven abbreviations.

MONTHS-CASE

Specifies how the case for the MONTHS-FULL or MONTHS-SHORT entries are 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 specified with the format code in the edit mask. Use NO-CHANGE to ignore the case of the format code and output the month explicitly listed in the MONTHS-FULL or MONTHS-SHORT entries.

MONTHS-FULL

Specifies the full names for the months of the year. SQR considers the first month of the year to be January. You must specify all 12 months.

MONTHS-SHORT

Specifies the abbreviated names for the months of the year. SQR considers the first month of the year to be January. You must specify all 12 abbreviations.

Description

The SYSTEM locale represents the behavior of older versions of SQR prior to Version 4.0. When you install SQR Version 4.0 or later, the default locale is set to SYSTEM. This provides upwards compatibility for older SQR programs. The SYSTEM locale settings are described in Table 2-1.

Table 2-1 SYSTEM Locale Settings

Keyword	Value
NUMBER-EDIT-MASK	The PRINT command prints two digits to the right of the decimal point and left justifies the number in the field. The MOVE, SHOW, and DISPLAY commands format the number with six digits to the right of the decimal point and left justifies the number.
MONEY-EDIT-MASK	SQR uses the same default as the NUMBER-EDIT-MASK keyword.
DATE-EDIT-MASK	SQR uses the default database date formats in Table 2-45, "Default Formats by Database," on page 2-270.

Table 2-1 SYSTEM Locale Settings (*Continued*)

Keyword	Value
INPUT-DATE-EDIT-MASK	SQR uses a default date edit mask with the INPUT command. See Table 2-43, “Sample Date Edit Masks,” on page 2-264 for a listing of the date edit masks.
MONEY-SIGN	'\$'
MONEY-SIGN-LOCATION	LEFT
THOUSAND-SEPARATOR	','
DECIMAL-SEPARATOR	'.'
DATE-SEPARATOR	'/'
TIME-SEPARATOR	'.'
EDIT-OPTION-NA	'n/a'
EDIT-OPTION-AM	'am'
EDIT-OPTION-PM	'pm'
EDIT-OPTION-BC	'bc'
EDIT-OPTION-AD	'ad'
DAY-OF-WEEK-CASE	EDIT
DAY-OF-WEEK-FULL	('Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday')
DAY-OF-WEEK-SHORT	('Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat')
MONTHS-CASE	EDIT
MONTHS-FULL	('January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December')
MONTHS-SHORT	('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')

Examples

The following code:

```
!
! The following program segments will illustrate the various
! ALTER-LOCALE features.
!
begin-setup
    declare-variable
        date $date $date1 $date2 $date3
    end-declare
end-setup

!
! Set default masks
!
alter-locale
    number-edit-mask = '9,999,999.99'
    money-edit-mask  = '$999,999,999.99'
    date-edit-mask   = 'Mon DD, YYYY'

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

Produces the following output:

```
With NUMBER option    #Value = 123,456.00
With MONEY  option    #Value = $123,456.00
Without NUMBER option #Value = 123456.000000
With DATE   option    $Date  = Jan 01, 1995
Without DATE  option  $Date  = 01-JAN-95
```

The following code:

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

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 SQR defaults and flip the thousand and
! decimal separator characters.
!
alter-locale
    locale = 'System'
    thousand-separator = '.'
    decimal-separator = ','

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

Produces the following output:

```
123.456,00
```

The following code:

```
!
! Reset locale to SQR 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 SQR defaults and change the text used with
! the edit options 'na', 'am', 'pm', 'bc', 'ad'
!
alter-locale
    locale = 'System'
    edit-option-na = 'Not/Applicable'
    edit-option-am = 'a.m.'
    edit-option-pm = 'p.m.'
    edit-option-bc = 'b.c.'
    edit-option-ad = 'a.d.'

let $value = ''
let $edit = 'Mon DD YYYY HH:MI'
let $date1 = strtodate('Jan 01 1995 11:59', $edit)
let $date2 = strtodate('Feb 28 1995 12:01', $edit)
show $value edit '999,999,999,999.99Na'
show $date1 edit 'Mon DD YYYY HH:MI:SS PM'
show $date2 edit 'Mon DD YYYY HH:MI:SS pm'
```

Produces the following output:

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

The following code:

```
!
! Input some dates using the 'system' locale and
! output using other locales from the SQR.INI file.
!
alter-locale
    locale = 'System'
let $date1 = strtodate('Jan 01 1995', 'Mon DD YYYY')
let $date2 = strtodate('Feb 28 1995', 'Mon DD YYYY')
let $date3 = strtodate('Mar 15 1995', 'Mon DD YYYY')
show 'System:'
show
show $date1 edit 'Month DD YYYY' ' is ' $date1 edit 'Day'
show $date2 edit 'Month DD YYYY' ' is ' $date2 edit 'Day'
show $date3 edit 'Month DD YYYY' ' is ' $date3 edit 'Day'
alter-locale
    locale = 'German'
show
show 'German:'
show
show $date1 edit 'DD Month YYYY' ' ist ' $date1 edit 'Day'
show $date2 edit 'DD Month YYYY' ' ist ' $date2 edit 'Day'
show $date3 edit 'DD Month YYYY' ' ist ' $date3 edit 'Day'
alter-locale
    locale = 'Spanish'
```

```
show
show 'Spanish:'
show
show $date1 edit 'DD Month YYYY' ' es ' $date1 edit 'Day'
show $date2 edit 'DD Month YYYY' ' es ' $date2 edit 'Day'
show $date3 edit 'DD Month YYYY' ' es ' $date3 edit 'Day'
```

Produces the following output:

System:

```
January 01 1995 is Sunday
February 28 1995 is Tuesday
March 15 1995 is Wednesday
```

German:

```
01 Januar 1995 ist Sonntag
28 Februar 1995 ist Dienstag
15 März 1995 ist Mittwoch
```

Spanish:

```
01 enero 1995 es domingo
28 febrero 1995 es martes
15 marzo 1995 es miércoles
```

See Also

- The DISPLAY, LET, MOVE, PRINT, and SHOW commands.
- Chapter 5, “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
Specifies the new font point size.

FONT-TYPE
Specifies the new font type. Enter PROPORTIONAL or FIXED.

SYMBOL-SET
Specifies the new symbol set identifier.

FONT
Specifies the new font as a number. (For example, 3 = Courier and 4 = Helvetica.)

PITCH
Specifies the new pitch in characters per inch.



Note See Table 2-21, “DECLARE-PRINTER Command Arguments,” on page 2-124 for more information on the ALTER-PRINTER arguments.

Description You can place the ALTER-PRINTER command in any part of an SQR program except the SETUP section.

ALTER-PRINTER attempts to change the attributes of the *current* printer for the *current* report. If an attribute does not apply to the *current* printer, it is ignored. For example, ALTER-PRINTER is ignored if it specifies proportional fonts for a report printed on a line printer. When your 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, the SYMBOL-SET argument is ignored; however, if the .SPF file is kept (see the -KEEP command line flag) and later printed on an HP LaserJet, the symbol set 12U can be used.

See Also

The DECLARE-PRINTER command.

ALTER-REPORT

Function	Alters some of the report-specific functionality.
Syntax	<pre>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]...)]</pre>
Arguments	<p>HEADING Specifies the name of the BEGIN-HEADING section to use.</p> <p>HEADING-SIZE Specifies the amount of space the BEGIN-HEADING section will occupy in the page.</p> <p>FOOTING Specifies the name of the BEGIN-FOOTING section to use.</p> <p>FOOTING-SIZE Specifies the amount of space the BEGIN-FOOTING section will occupy in the page.</p>

PDF-APPEARANCE

Specifies the appearance of the document when you open it. Table 2-2 describes the valid appearance values.

Table 2-2 Appearance Values

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

For example:

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

PDF-INFORMATION

Specifies 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 2-3, 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 2-3 Standard PDF Information Names

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

For example:

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

PDF-OPEN-ACTION

Specifies the action the PDF viewer takes upon opening the file, the name of the additional value, and the value to apply. Table 2-4 describes the valid open actions. Table 2-5 describes the valid additional values.

Table 2-4 Open Actions

Open Action	Description
Fixed	Use a fixed destination view specified by three values (<i>Zoom</i> , <i>Left</i> , <i>Top</i>).
Window	Fit the complete page to the window.
Width	Fit the page width to the window specified with one value (<i>Top</i>).
Height	Fit the page height to the window specified with one value (<i>Left</i>).
Rectangle	Fit the rectangle specified by specified by four values (<i>Left</i> , <i>Bottom</i> , <i>Right</i> , <i>Top</i>).
Visible	Fit the visible contents of the page to the window.
VisibleWidth	Fit the visible contents of the page to the window with one value (<i>Top</i>).
VisibleHeight	Fit the visible contents of the page to the window with one value (<i>Left</i>).

Table 2-5 Additional Values

Value	Description
Page	The initial page to display. The default value is 1.
Zoom	The zoom factor to be used when the page displays. The default value is 100.
Left	The column number of the page to position at the left edge of the window. The default value is 1.

Table 2-5 Additional Values (*Continued*)

Value	Description
Top	The line number of the page that will be positioned at the top edge of the window. The default value is 1.
Bottom	The last line of the page to be displayed. The default value is <i>Entire page</i> .
Right	The last column of the page to be displayed. The default value is <i>Entire page</i> .

For example:

```
ALTER-REPORT  
PDF-OPEN-ACTION= ('Fixed', 'Zoomv, 150, 'Page', 2)
```

PDF-PAGE-TRANSITION

Specifies the page transition to use for the current and future pages and the duration (in seconds) for the transaction. Table 2-6 describes the valid transition values.

Table 2-6 Transitions

Transition	Description
Split	Two lines sweeping across the screen reveal the page.
Blinds	Multiple lines sweeping across the screen reveal the page.
Box	A box reveals the page.
Wipe	A single line sweeping across the screen reveals the page.
Dissolve	The old page dissolves to reveal the new page.
Glitter	The dissolve effect moves from one screen edge to another.
Replace	The old page is replaced by the new page. (This is the default value.)

For example:

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

PDF-SECURITY

Specifies the security parameter to address and the value to apply to the security parameter. Table 2-7 describes the valid security parameters.

Table 2-7 Security Parameters

Security	Description
User-Password	Specifies the user level password to apply to the generated PDF document. The password can be up to 32 characters in length. You need this value to open the document.
Master-Password	Specifies the master level password to apply to the generated PDF document. The password can be up to 32 characters in length. You can use this value to open the document or to override the permissions.
Permissions	Specifies the list of permissions to apply to the generated PDF document. This is a string of space-delimited values that dictate to the PDF viewer the permissions associated with this document. Valid permissions include: <ul style="list-style-type: none">■ NoPrint – Prevents printing the file.■ NoModify – Prevents users from adding form fields or making any other changes.■ NoCopy – Prevents copying and extracting text and graphics and disables the accessibility interface.■ NoAnnots – Prevents adding or changing comments or form fields.■ NoForms – Prevents form field filling, even if NoAnnots is not specified. (Requires Acrobat 5 or higher compatibility)■ NoAccessible – Prevents extracting text or graphics for accessibility purposes. For example, a screen reader program. (Requires Acrobat 5 or higher compatibility)■ NoAssemble – Prevents inserting, deleting, or rotating pages and creating bookmarks and thumbnails, even if NoModify is no specified. (Requires Acrobat 5 or higher compatibility)■ NoHiResPrint – Prevents high-resolution printing. If NoPrint is not specified printing this setting is restricted to the <i>Print As Image</i> feature, which prints a low-resolution rendition of the page. (Requires Acrobat 5 or higher compatibility)



Note The default document that is generated has no passwords and all permissions possible.

For example:

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

PDF-VIEWER-PREFERENCE

Specifies the viewer preference to address and the value to apply to the preference. Table 2-8 describes the valid viewer preferences.

Table 2-8 Viewer Preferences

Preference	Description
Toolbar	Specify <i>True</i> to hide Acrobat's tool bar. The default value is <i>False</i> .
MenuBar	Specify <i>True</i> to hide Acrobat's menu bar. The default value is <i>False</i> .
WindowUI	Specify <i>True</i> to hide Acrobat's windows controls. The default value is <i>False</i> .
FitWindow	Specify <i>True</i> to resize the document's window to the size of the first page. The default value is <i>False</i> .
CenterWindow	Specify <i>True</i> to position the document's windows in the center of the screen. The default value is <i>False</i> .
DisplayDocTitle	Specify <i>True</i> to display the document information field in Acrobat's title bar or <i>False</i> to display the file name. The default value is <i>False</i> .
NonFullscreen-Page-Mode	Defines how to display the document on exiting full-screen mode. Valid values include: <ul style="list-style-type: none">■ UseOutlines – Displays page and document outlines.■ UseThumbs – Displays page and thumbnails.■ UseNone – Displays neither document outlines nor thumbnails. The default value is <i>UseNone</i> .
Direction	Defines the reading order of the document. This preference affects the scroll ordering in double-page view. Valid values include: <ul style="list-style-type: none">■ L2R – Left to right■ R2L – Right to left (including vertical writing systems). The default value is <i>L2R</i> .

For example:

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

Description

This command gives you the ability to dynamically change those heading and/or footing sections that are active for the current report and change how much space the heading and/or footing sections occupy. For PDF reports, you can also add information about the report, control the display and appearance of the report, and control security settings.

If the **HEADING** or **FOOTING** value is set to ‘NONE’ then that section is disabled for the current report.

If the **HEADING** or **FOOTING** value is set to ‘DEFAULT’ then that section will revert to whatever was in effect when the report was initiated.

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

When the **HEADING**, **HEADING-SIZE**, **FOOTING**, or **FOOTING-SIZE** parameters are specified and the command was not invoked from within a **BEGIN-HEADING** and/or **BEGIN-FOOTING** section and the page has not been written to, then the assignment takes effect immediately; otherwise it will take effect for the next page.

Examples

```
begin-footing 2 name=confidential
    print 'Company Confidential' (1,1,0) center
    page-number (2,37,0)
end-footing

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

begin-program
.
.
.

alter-report
    footing = 'Proprietary'
    footing-size = 6      ! Increase depth

.
.
.

end-program
```

See Also

- The BEGIN-FOOTING command.
- The BEGIN-HEADING command.

ARRAY-ADD

ARRAY-DIVIDE

ARRAY-MULTIPLY

ARRAY-SUBTRACT

Function These four commands perform arithmetic on one or more elements in an array.

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 to, divided into, multiplied times, or subtracted from the respective destination array fields. All variables must be numeric in type.

*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 in type.

Description	The following information applies to the array arithmetic commands:
	<ul style="list-style-type: none"> ■ The array must first be created using the CREATE-ARRAY command. <p>The four array arithmetic commands perform on one or more source numbers, placing the result into the corresponding field in the array.</p> <ul style="list-style-type: none"> ■ 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 SQR continues executing.
Examples	<p>The following example adds &salary and #comm to the first two fields defined in the emps array. The #j'th element of the array is used.</p> <pre>array-add &salary #comm to emps(#j)</pre> <p>The following example subtracts #lost, #count, and 1 from the fields loses, tot and sequence of the #j2'th element of the stats array.</p> <pre>array-subtract #lost #count 1 from stats(#j2) loses tot sequence</pre> <p>The following example multiplies occurrences 0 through 2 of the field p in the #i'th element of the percentages array by 2.</p> <pre>array-multiply 2 2 2 times percentages(#i) p(0) p(1) p(2)</pre> <p>The following example divides the #i2'th occurrence of the salesman field of the #j'th element of the commissions array by 100.</p> <pre>array-divide 100 into commissions(#j) salesman(#i2)</pre>

The following example uses the ARRAY-ADD command in an SQR program.

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

begin-program
do Main
end-program

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

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

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

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

See Also

- The CREATE-ARRAY command for information on creating an array.
- The CLEAR-ARRAY command for information on clearing or initializing an array.
- The GET, PUT, and LET commands for information on using arrays.

ASK

Function	Retrieves values for a compile-time substitution variable. The retrieval can be by user input, command-line arguments, or as entries in the @file on the command line. See “SQR Command-line Flags” on page 1-15 for more information.
Syntax	<code>ASK <i>substitution_variable</i> [<i>prompt</i>]</code>
Arguments	<i>substitution_variable</i> The variable to be used as the substitution variable. <i>prompt</i> An 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 the ASK command across program lines.
Examples	In the following example, state takes the value entered by the user in response to the prompt <i>Enter state for this report</i> . <pre>begin-setup ask state 'Enter state for this report' end-setup ... begin-select name, city, state, zip from customers where state = '{state}' end-select</pre>
See Also	<ul style="list-style-type: none">■ The INPUT command for information on input at run time.■ Chapter 28, “Compiling Programs and Using SQR Execute” in the <i>Hyperion SQR Getting Started Guide</i>.

BEGIN-DOCUMENT

Function	Begins a DOCUMENT paragraph. A document paragraph allows you to write free-form text to create form letters, invoices, and so on.
Syntax	<code>BEGIN-DOCUMENT <i>position</i></code> <code>END-DOCUMENT</code>
Arguments	<i>position</i> The location on the page where the document begins. The position can be fixed or relative to the current position.
Description	Database columns, SQR variables, and document markers can be referenced within a document. Their location in the document determines where they are printed on the page. You should not use tabs inside a document paragraph. To indent text or field use the spacebar. Note also that if the variables being printed inside a document paragraph are variable in length, you might want to manipulate the variable outside the DOCUMENT paragraph.

**Note**

A document must be executed before you can reference its document markers. Since documents can be printed at relative positions on the page, the actual location of a document marker may not be known by SQR until the document itself has been executed.

Examples

```
begin-document (1,1)
.b
Dear $firstname
...
end-document
```

See Also

- The END-DOCUMENT command.
- Chapter 11, “Creating Form Letters” in the *Hyperion SQR Getting Started Guide* for a full example of the BEGIN-DOCUMENT command).

BEGIN-EXECUTE



Note The BEGIN-EXECUTE command is specific to SQR/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. (See the tip on page 2-38 for more information.) In a BEGIN-EXECUTE paragraph, the syntax of BEGIN-SELECT varies as shown below.

Syntax

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

Arguments

CONNECTION

Identifies a name previously specified using the DECLARE-CONNECTION construct. If you do not specify a name, SQR uses the default connection. The default connection is defined by the command-line entries for datasource (DSN), username (USER), and password (PASSWORD). Name is not case-sensitive.

ON-ERROR

Declares a procedure to execute if an error occurs.

RSV

Row Set Variable. A global SQR variable containing the row set being retrieved.

STATUS

Identifies a list or scalar variable that receives the status of the stored procedure.

PROPERTIES

Specifies a (set of) keyword/value pair(s) that represent modifications to be made to the Properties of the datasource (specified by the CONNECTION= statement). An arbitrary number of such pairs can be specified.

SCHEMA

Identifies the location in the datasource of the object being queried. You can enter the following options under SCHEMA:

- PROCEDURE – The name of the datasource-stored procedure to be executed. 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|num_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 – A text string you pass to the datasource without modification by SQR. This string can include embedded SQR variables.
- GETDATA – Supports the Java (DDO) GetData paradigm for data access.

BEFORE/AFTER

Names an SQR 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 the BEGIN-SELECT 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. If you specify more than one row set, 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 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

Special case addition to the BEGIN-SELECT syntax. Available only for SAP R/3 datasources. Use only in conjunction with PROCEDURE keyword. This argument names an output parameter containing one or more rows from which the column variables are to be retrieved.



Note

This is a similar concept to the PARAMETERS= statement in DECLARE- and ALTER- CONNECTION, with the minor difference being 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)

FROM {table_name}

The relational datasource 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 as specified in the *Registry.properties* file. For example, if your *Registry.properties* file contains:

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

Then your SQR should look similar to:

```
begin-setup
declare-connection default
    DSN=XML_DATA           ! use the same name as specified in the
    Registry.properties file. Case sensitive!
end-declare
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 SQR produces the following output:

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



Tip BEGIN-EXECUTE is only required when additional information about the DDO datasource or query is needed, such as 'Connection', 'Schema', 'Command', 'GetData', 'Procedure', or 'Parameters'. The following example does not require a BEGIN-EXECUTE command 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)xxxx-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

The EXECUTE command.

BEGIN-FOOTING

Function	Begins the FOOTING section.
Syntax	<pre>BEGIN-FOOTING <i>footing_lines_int_lit</i> [FOR-REPORTS=(<i>report_name1</i>[,<i>report_namei</i>]...)] [FOR-TOCS=(<i>toc_name1</i>[,<i>toc_namei</i>]...)] [NAME={<i>footing_name</i>}] END-FOOTING</pre>
Arguments	<p><i>footing_lines_int_lit</i> The number of lines to be reserved at the bottom of each page.</p> <p>FOR-REPORTS Specifies the reports to which this footing applies. 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.</p> <p>FOR-TOCS Specifies the Table of Contents to which this heading applies.</p> <p>NAME Specifies the name to be associated with this footing section. This is used in conjunction with the ALTER-REPORT command. The name cannot be NONE or DEFAULT.</p>
Description	<p>The FOOTING section defines and controls information to be printed at the bottom of each page.</p> <p>You must define the <i>report_name</i> in a DECLARE-REPORT paragraph. If you do not use DECLARE-REPORT, the footing is applied to all reports. You can also specify FOR-REPORTS= (ALL). Note that the parentheses are required.</p> <p>You can specify more than one BEGIN-FOOTING section; however, there can be only one for each report. A BEGIN-FOOTING section with FOR-REPORTS= (ALL) can be followed by other BEGIN-FOOTING sections for specific reports, which override the ALL setting.</p>

You must define the *toc_name* in a DECLARE-TOC paragraph. You can also specify FOR-TOCS=(ALL). Note that the parentheses are required.

You can specify more than one BEGIN-FOOTING section; however, there can be only one 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 the ALL setting.

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

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

Examples

```
begin-footing 2 for-reports=(customer, summary)
    print 'Company Confidential' (1,1,0) center
    page-number (2,37,0)
end-footing
begin-footing 2                      ! For all reports
    print 'Division Report' (1,1,0) center
    page-number (2,37,0)
end-footing
begin-footing 2 for-tocts=(all)
    print 'Table of Contents' (2,1)
    let $page = roman(#page-count)! ROMAN numerals
    print $page (,64)
end-footing
```

See Also

- The ALTER-REPORT command for more information about dynamic headings/footings.
- The DECLARE-LAYOUT command for more information on page layout.
- The DECLARE-REPORT command for more information on programs with multiple reports.
- The DECLARE-TOC command for more information on Table of Contents.
- The END-FOOTING command.

BEGIN-HEADING

Function	Begins a HEADING section.
Syntax	<pre>BEGIN-HEADING <i>heading_lines_int_lit</i> [FOR-REPORTS=(<i>report_name1</i>[, <i>report_namei</i>]...)] [FOR-TOCS=(<i>toc_name1</i>[, <i>toc_namei</i>]...)] [NAME={<i>heading_name</i>}] END-HEADING</pre>
Arguments	<p><i>heading_lines_int_lit</i> The number of lines to be reserved at the top of each page.</p> <p>FOR-REPORTS Specifies the reports to which this heading applies. This 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.</p> <p>FOR-TOCS Specifies the Table of Contents to which this heading applies.</p> <p>NAME Specifies the name to be associated with this heading section. This option cannot be used if FOR-REPORTS or FOR-TOCS is also specified. This is used in conjunction with the ALTER-REPORT command. The name cannot be NONE or DEFAULT.</p>
Description	<p>The HEADING section defines and controls information to be printed at the top of each page.</p> <p>You must define the <i>report_name</i> in a DECLARE-REPORT paragraph. If you do not use DECLARE-REPORT, the heading is applied to all reports. You can also specify FOR-REPORTS= (ALL). Note that the parentheses are required.</p> <p>You can specify more than one BEGIN-HEADING section; however, there can be only one 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 the ALL setting.</p> <p>You must define the <i>toc_name</i> in a DECLARE-TOC paragraph. You can also specify FOR-TOCS= (ALL). Note that the parentheses are required.</p>

You can specify more than one BEGIN-HEADING section; however, there can be only one for each Table of Contents. A BEGIN-HEADING section with FOR-TOCS= (ALL) can be specified followed by other BEGIN-HEADING sections for specific Table of Contents, which override the ALL setting.

The BEGIN-HEADING section can be shared between reports and Table of Contents.

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

Examples

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

See Also

- The ALTER-REPORT command for more information about dynamic headings/footings.
- The DECLARE-LAYOUT command for more information on page layout.
- The DECLARE-REPORT command for more information on programs with multiple reports.
- The DECLARE-TOC command for more information on Table of Contents.
- The END-HEADING command.

BEGIN-PROCEDURE

Function	Begins a procedure. A procedure is one of the most powerful parts of the SQR language. It modularizes functions and provides standard execution control.
Syntax	<pre>BEGIN-PROCEDURE <i>procedure_name</i> [LOCAL (<i>arg1</i> [, <i>argi</i>]...)] END-PROCEDURE</pre>
Arguments	<p><i>procedure_name</i> Specifies a unique name for this procedure. Procedure names are not case-sensitive.</p> <p>LOCAL Specifies that this is a local procedure.</p> <p><i>arg1</i> [, <i>argi</i>]... Specifies arguments to be passed to or returned from the procedure. Arguments can be either string variables (\$<i>arg</i>), numeric variables (#<i>arg</i>), or date variables (\$<i>arg</i>). If you want to return a value passed back to the calling DO command, place a colon (:) before the variable name. The arguments of the BEGIN-PROCEDURE and DO commands must match in number, order, and type.</p>
Description	<p>The procedure name must be unique. The name is referenced in DO commands. Procedures contain other commands and paragraphs (for example, SELECT, SQL, DOCUMENT).</p> <p>By default, procedures are global. That is, variables or columns defined within a procedure are known and can be referenced outside the procedure.</p> <p>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. In addition, any query defined in a local procedure will have local database column variable names assigned that do not conflict with similarly named columns defined in queries in other procedures.</p> <p>SQR procedure can be called recursively. However, unlike C or Pascal, SQR only maintains one copy of the local variables and they are persistent.</p>

Arguments passed by a DO command 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 the SQR_DB_DATE_FORMAT setting, or a database dependent format (see Table 2-45, “Default Formats by Database,” on page 2-270), or the database-independent format SYYYMMDD [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. SQR 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 SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.
 - For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
 - For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

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

 **Note**

All the SQR reserved variables, such as #sql-status and \$sql-error, are global variables. Within a local procedure, 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 calling 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

The DO and END-PROCEDURE commands.

BEGIN-PROGRAM

Function Begins the program section of an SQR program.

Syntax

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

Description After processing the commands in the SETUP section, if any, SQR starts program execution at the BEGIN-PROGRAM 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 SQR program.

Examples

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

See Also The BEGIN-SETUP and END-PROGRAM commands.

BEGIN-SELECT

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

Syntax

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

Arguments



Note The 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

Specifies that you want to eliminate duplicate rows from your query.

-Cnn

(Oracle) Sets the context area size (buffer size for query) to larger or smaller than the default; this option is rarely needed.

-Bnn

(ODBC, Oracle, Sybase) Sets the number of rows to retrieve at one time. This is 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 SQR command line with -B, which can be overridden by a separate -B flag on each BEGIN-SELECT command.

-XP

(Sybase) Prevents the creation of a stored procedure for the SELECT paragraph. When -XP is specified, SQR generates a new SQL statement using the current value of any bind variables each time the BEGIN-SELECT is executed. This disables the potential performance optimization created by stored procedures. If you change the variables frequently during execution and you do not want SQR to automatically create stored procedure, use this flag. You can also use -XP if the username/password to your program does not have permission to create stored procedures. However, if you do not change variables frequently during execution, the use of stored procedures may optimize your program's performance. In that case, do not use this argument. Note also that -XP is available as a command-line flag.

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

-DBconnectionstring

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

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

This option allows you to combine data from multiple databases in one program. For example, a connection string for an Oracle database named "ora8" might look like the following:

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

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

LOOPS

Specifies the number of rows to retrieve. After the specified number has been processed, the SELECT loop exits.

ON-ERROR

Declares a procedure to execute if an error occurs due to incorrect SQL syntax. Error trapping should be used in conjunction with dynamic query variables. SELECT paragraphs without dynamic variables are checked for errors before the program is processed and therefore do not require a special error procedure.

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



Note

SQR invokes the ON-ERROR procedure when it safely can. If SQR can recover from a database error, the user is given the chance to fix the error. If SQR cannot recover from a database error, it will exit from the program.

Description

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



Note

In **SQR for DDO**, you can name datasource-specific aggregation functions in place of column names from within a BEGIN-SELECT block. This shifts the processing burden from the SQR system to the datasource 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 **SQR for DDO-SAP**, the `TYPE=datatype` qualifier used in a BEGIN-SELECT block is optional. When you report on datasources that provide adequate metadata (such as SAP), withholding the TYPE qualifier allows SQR 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 SQR 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, make sure 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 that follows 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 Chapter 5, “Column Variables”Chapter 5, “Column Variables” in *Hyperion SQR Getting Started Guide*.

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

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

```
begin-select          loops=100
[$col_var_char]      &col1=char
[$col_var_num]       &col2=number
cust_id              &id=number
from [$table_name]
[$where clause]
[$order_by_clause]
end-select
```

See Also

- Chapter 4, “Selecting Data from the Database” and Chapter 17, “Dynamic SQL and Error Checking” in the *Hyperion SQR Getting Started Guide*.
- The END-SELECT and EXIT-SELECT commands.

BEGIN-SETUP

Function	Begins a SETUP section. This section is optional, but if included, it is processed prior to the BEGIN-PROGRAM, BEGIN-HEADING, and BEGIN-FOOTING section.
Syntax	<pre>BEGIN-SETUP END-SETUP</pre>
Description	<p>The SETUP section should be the first section in the program.</p> <p>The SETUP section contains commands that determine the overall characteristics of the program. The commands used in the SETUP section cannot be used elsewhere unless specified. The SETUP section can include the following commands:</p> <p>ASK</p> <p>BEGIN-SQL (This command can also be used in BEGIN-PROCEDURE paragraphs.)</p> <p>CREATE-ARRAY (This command can also be used in the other sections of an SQR program.)</p> <p>DECLARE-CHART</p> <p>DECLARE-IMAGE</p> <p>DECLARE-LAYOUT</p> <p>DECLARE-PRINTER</p> <p>DECLARE-PROCEDURE</p> <p>DECLARE-REPORT</p> <p>DECLARE-TOC</p> <p>DECLARE-VARIABLE (This command can also be used in LOCAL procedures.)</p> <p>LOAD-LOOKUP (This command can also be used in the other sections of an SQR program.)</p> <p>USE (Sybase and ODBC only)</p>

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

The ASK, BEGIN-SQL, CREATE-ARRAY, LOAD-LOOKUP, and USE commands.

BEGIN-SQL

Function	Begins an SQL paragraph. This paragraph can reside in a BEGIN-PROCEDURE, BEGIN-SETUP, or BEGIN-PROGRAM section.
Syntax	<pre>BEGIN-SQL [-Cnn] [-XP] [-NR] [-SORTnn] [-LOCK{RR CS RO RL XX}] [-DB<i>database</i>] [-DB<i>connectionstring</i>] [ON-ERROR=<i>procedure</i>[(<i>arg1</i>[,<i>argi</i>]...)] (non-setup) [ON-ERROR={STOP WARN SKIP}] (SETUP) END-SQL</pre>
Arguments	<p>-Cnn (Oracle) Sets the context area size (buffer size for query) to larger or smaller than the default; this option is rarely needed.</p> <p>-XP (Sybase) Prevents the creation of a stored procedure for the SQL paragraph. When -XP is specified, SQR generates a new SQL statement using the current value of the bind variables each time the BEGIN-SQL is executed. This disables the performance optimization created by stored procedures. If you change the variables frequently during execution and you do not want SQR to automatically create stored procedures, use this flag. You may also use it if your program does not have permission to create stored procedures. However, if you do not change variables frequently during execution, the use of stored procedures optimizes the performance of the program. In that case, do not use this argument.</p> <p>-XP improves performance when you use 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.</p> <p>-DB<i>connectionstring</i> (ODBC) Specifies the ODBC connection string for this SQL paragraph only. A connection string has the following syntax:</p> <pre>DSN=<i>data_source_name</i>[;<i>keyword=value</i>[;<i>keyword=value</i>[;<i>...</i>]]]</pre>

This option allows you to combine data from multiple databases in one program. For example, a connection string for an Oracle 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.

```
Connection=connstr
```

This option is used with SQR for DDO. It specifies the name of a datasource previously declared using the DECLARE-CONNECTION construct. If it is not specified, the default connection is used. (See BEGIN-EXECUTE for the behavior of the default connection.)

ON-ERROR

Declares a procedure to execute if an error occurs due to incorrect SQL syntax except when executed in a BEGIN-SETUP section. By default, SQR 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 be passed to the ON-ERROR procedure. Arguments can be any variable, column, or literal.

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

STOP

Do not run the program.

WARN

Run the program but with a warning message.

SKIP

Ignore any errors and run the program.

**Note**

SQR invokes the ON-ERROR procedure when it safely can. If SQR can recover from a database error, the user is given the chance to fix the error. If SQR 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, each statement except, of course, the last must be terminated 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). Columns and variables can be referenced in the 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, SQR supports stored procedures with the EXECUTE command. For Oracle, stored procedures are implemented using PL/SQL in the BEGIN-SQL paragraph.

For some databases such as ORACLE, using DDL statements within a BEGIN-SQL paragraph causes a commit of outstanding inserts, updates, and deletes and releases cursors. For this reason, one must be sure 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. See the following Oracle example.

For Oracle PL/SQL:

```
begin-sql
declare
    varpl varchar2 (25);;
    var2 number (8,2);;
begin
varpl :='abcdefg';;
$v1 :=varpl;;
$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

- Chapter 17, “Dynamic SQL and Error Checking” and Chapter 20, “Using DML and DDL” in the *Hyperion SQR Getting Started Guide*.
- The END-SQL, BEGIN-PROCEDURE, and EXECUTE commands.
- The -S command-line flag.

BREAK

<i>Function</i>	Causes an exit from within an EVALUATE or WHILE command. Execution then continues to the command immediately following the END-WHILE or END-EVALUATE.
Syntax	<code>BREAK</code>
Description	This command is used inside a <code>WHILE . . . END-WHILE</code> loop or within an EVALUATE command.
See Also	The WHILE and EVALUATE commands.

CALL

CALL SYSTEM

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

 Note	CALL is available in all SQR environments except Hyperion SQR Developer. With Hyperion SQR Developer, use CALL SYSTEM instead.
---	--

Syntax	<pre>CALL <i>subroutine</i> USING {<i>src_txt_lit _var _col</i>} {<i>src_num_lit _var _col</i>} {<i>dst_txt_var _num_var</i>} [<i>param</i>]</pre>
	To issue operating system commands from within an SQR program, use the following syntax:
	<pre>CALL SYSTEM USING <i>command status</i> [WAIT NOWAIT]</pre>
Arguments	<p><i>subroutine</i> Specifies the name of your subroutine.</p> <p><i>src_txt_lit _var _col</i> Specifies a text column, variable, or literal which is to be input to the called subroutine.</p> <p><i>src_num_lit _var _col</i> Specifies a numeric column, variable (decimal, float, or integer), or literal that is to be input to the called subroutine.</p> <p><i>dst_txt_var _num_var</i> Specifies a text or numeric variable (decimal, float, or integer) into which the called subroutine is to place the return result. See UCALL Subroutine ArgumentsTable 2-10 on page 2-61 for more information.</p>

param

Specifies an optional alphanumeric string of characters to be passed as a parameter to the subroutine.

SYSTEM

Specifies that this CALL command issues an operating system command.

command

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

status

Contains the 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 2-9.

Table 2-9 Operating System Status Values for the CALL Command

System	Value Returned
OS/2, and UNIX	Zero (0) indicates success. Any other value is the system error code.
PC/Windows	A value less than 32 indicates an error.

WAIT | NOWAIT

(Windows) - WAIT specifies that SQR suspend its execution until the CALL SYSTEM command has finished processing. NOWAIT specifies that SQR start the CALL SYSTEM command but continue its own processing while that command is in progress.

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 SQR. Subroutines can be written in any language.

Caution! We recommend that the *ucall* function not use any database calls as it may cause erroneous results.

Used in an SQR program, CALL has the following format:

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

The CALL SYSTEM is a special subroutine which is provided as part of SQR 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 your subroutine. Upon return from the subroutine, a value is placed in the destination variable.

You must write the subroutine and call it in one of the supplied ucall routines. (Optionally, you could rewrite ucall in another language instead).

The source file UCALL.C contains sample subroutines written in C. The ucall function takes the following arguments:

Table 2-10 UCALL Subroutine Arguments

Argument	Description	How Passed
callname	Name of the subroutine.	By reference with a maximum of 31 characters, null terminated.
strsrc	Source string.	By reference with a maximum of 255 characters, null terminated.
strdes	Destination string.	By reference with a maximum of 255 characters.
dblsrc	Source double floating point.	By reference.
dbldes	Destination double floating point.	By reference.
param	Subroutine parameter string. It must be a literal.	By reference with a maximum of 80 characters, null terminated.

When you use the CALL command, arguments are handled as follows:

- Calling arguments are copied into the variables depending on the type of argument. Strings are placed into *strsrc*, and numerics are placed into *dblsrc*.
- Return values are placed into *strdes* or *dbldes* depending on whether your destination argument for CALL is a string or numeric variable.

The destination arguments can also be used to pass values to your subroutine.

To access your subroutine, add a reference to it in UCALL, passing along the arguments you need.

You must relink SQR to CALL after compiling a user defined function that becomes another SQR function.

You must add your subroutine to the link command file—in UNIX it is called SQRMAKE, in Windows it is called SQREXT.MAK—if you have created a new object file. (Alternatively, you could add your routine to the bottom of the UCALL source module that is already included in the link).

Your subroutine and calling SQR program are responsible for passing the correct string or numeric variables and optional parameter string to the subroutine. No checking is performed.

Examples

See these sample subroutines included in the UCALL source file:

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

The following code calls these subroutines:

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

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

```
! Unix  (Type 'exit' to return to SQR)
!
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 SQR)
!
let $comspec = getenv('COMSPEC')
let $cmd = comspec || '/c' || $comspec || ' /k'
call system using $cmd #win_status wait
```

The following step-by-step example shows how to add a user-defined subroutine to SQR so that it can be invoked from SQR using the CALL command. 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 the initcap function is applied. The second is the resultant string.

To add initcap function to SQR, the following modifications are needed to the UCALL.C file, which was provided with SQR:

1. Add the prototype for the initcap function:

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

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

```
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 the UCALL.C file, add the initcap routine listed in the following example. The routine name must be lower case; however, in your SQR program it can be referenced either upper or lower case.

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

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



Note The CC_ARG macros are defined in the UCALL.C source module. The macros give the programmer the ability to define a fully prototyped function without having to worry if the C compiler supports the feature.

After these modifications, recompile UCALL.C and relink SQR. See Chapter 25, “Interoperability” in *Hyperion SQR Getting Started Guide* for details.

Finally, the following is an example of a simple SQR program which uses the initcap function:

```
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

The LET command 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 each field of an array to its initial value.

Syntax CLEAR-ARRAY NAME=*array_name*

Arguments NAME
Specifies the name of the array to be cleared.

Description The CLEAR-ARRAY command resets each field of the named array to the initial value specified for that field in the CREATE-ARRAY command. 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 also releases all memory used by the specified array and returns it to its pristine state.

Examples clear-array name=custs

See Also The CREATE-ARRAY command.

CLOSE

Function	Closes a file, specified by its file number.
Syntax	CLOSE { <i>filenum_lit _var _col</i> }
Arguments	<i>filenum_lit _var _col</i> Specifies the number assigned to the file in the OPEN command.
Description	Closes a flat file that has been previously opened using the OPEN command.
Examples	close 5 close #j
See Also	The OPEN, READ, and WRITE commands.

COLUMNS

Function	Defines logical columns to use for PRINT commands.
Syntax	COLUMNS {int_lit _var _col} [int_lit _var _col] ...
Arguments	<i>int_lit _var _col</i> Specifies the left margin position of each column.
Description	<p>COLUMNS defines the left-most position of one or more columns within the current page layout. It sets the first column as current.</p> <p>COLUMNS can be used for printing data either down the page or across the page, depending on how you use the NEXT-COLUMN and USE-COLUMN commands.</p> <p>The COLUMNS command applies only to the current report. If you want to print columns in more than one report, you must specify the COLUMNS command for each report.</p> <p>The USE-COLUMN 0 turns off columns. See USE-COLUMN.</p>
See Also	The NEXT-COLUMN, NEXT-LISTING, NEW-PAGE, USE-COLUMN, and USE-REPORT commands.

COMMIT

Function	Causes a database commit.
Syntax	COMMIT
Description	<p>COMMIT is useful when you are doing many inserts, updates, or deletes in an SQL paragraph. A database commit releases the locks on the records that have been inserted, updated, or deleted. Used with some databases, it also has other effects. For this reason, it should not be used within the scope of an active SELECT paragraph or unpredictable results may occur.</p> <p>When the application completes, a commit is performed automatically unless a ROLLBACK was done or, for callable SQR, the -XC flag was set.</p> <p>Other commands or options, such as the CONNECT command and the use of DDL statements for some databases with a BEGIN-SQL paragraph, can also cause the database to do a commit.</p> <p>COMMIT is an SQR command and <i>should not</i> be used within an SQL paragraph. If used in an SQL paragraph, unpredictable errors can occur.</p>
 Note	The COMMIT command can be used with DB2, ODBC, DDO, Teradata, and Oracle. For Sybase, use BEGIN TRANSACTION and COMMIT TRANSACTION within SQL paragraphs as in the following code segment.

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

For Sybase:

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

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

Caution! Any data being 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 a variable, column, or literal with a string variable.
Syntax	<code>CONCAT {src_any_lit _var _col} WITH dst_txt_var[[:\$]edit_mask]</code>
Arguments	<p><i>src_any_lit _var _col</i> Specifies the source field to be concatenated with the <i>dst_txt_var</i> field.</p> <p><i>dst_txt_var</i> Contains the result after execution.</p> <p><i>edit_mask</i> Specifies an optional edit mask.</p>
Description	<p>The contents of the source field are appended to the end of the destination field.</p> <p>CONCAT can optionally edit the source field before appending it. Edit masks can be changed dynamically by placing them in a string variable and referencing the variable name preceded by a colon (:). See the command, “PRINT” on page 2-255 for information on the use of edit masks.</p> <p>Also, 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:</p> <ul style="list-style-type: none">■ For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.■ For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.■ For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

Examples

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

See Also

- The PRINT command for information on edit masks.
- The LET command for string functions.
- The STRING and UNSTRING commands.

CONNECT

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

Syntax CONNECT {*txt_lit|_var|_col*} [ON-ERROR=*procedure*[(*arg1*, *arg2*)...]]

Arguments *txt_lit|_var|_col*
Specifies a user name and password for the logon.

ON-ERROR

Specifies a procedure to be executed if the logon fails. If no ON-ERROR procedure is specified and the logon fails, SQR halts with an error message.

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



Note The argument to the CONNECT command is the same as the {connectivity} portion of the SQR command line as follows:

DB2: DB[/username/password]

DDO: DSN[/username/password]

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

ODBC: DSN[/username/password]

ORACLE: [username/password][@OracleServer]

TERADATA: [TDPID/]username[,password]

Description	The new connectivity information can be stored in a string variable, column, or literal.
	After each CONNECT, the reserved variable <code>\$username</code> is set to the new username.
	All database cursors or logons are closed before the CONNECT occurs. You should not issue a CONNECT within a SELECT or an SQL paragraph while a query is actively fetching or manipulating data from the database.

Examples	<pre>connect \$new-user on-error=bad-logon(\$new_user) connect 'sqr/test'</pre>
-----------------	---

Caution ! The connectivity information is not encrypted, so beware of security issues.

CREATE-ARRAY

Function	Creates an array of fields to store and process data.
Syntax	<pre>CREATE-ARRAY NAME=array_name SIZE=nn [EXTENT=nn] {FIELD=name:type[:occurs] [={init_value_txt_lit _num_lit}]}</pre> ...
Arguments	NAME Names the array. The name is referenced in other array commands. SIZE Defines the number of elements in the array. EXTENT Identifies a specific number of array elements used to incrementally extend the size of the arry beyond the initial allocation defined in SIZE. The value entered for EXTENT must be a <i>numeric literal</i> . FIELD Defines each field or column in the array. Each field must be defined as type: <ul style="list-style-type: none">■ 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 the DECLARE-VARIABLE command.■ 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 the CLEAR-ARRAY command 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 multiply occurring field are set to the same value. For dates, the initialization string must be formatted as 'SYYYMMDD [HH24 [MI [SS [NNNNNN]]]]'. See Table 2-41 on page 2-262 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.

SQR creates arrays before a program starts to execute. The CREATE-ARRAY command can be used in any section of a program.

Commands to process arrays include the following:

```
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 2-1 is a representation of an array *emps* with three fields. The following CREATE-ARRAY command defines the array:

```
create-array name=emps size=10  
field=name:char='Unknown'  
    field=rate:number:2=10.50  
    field=phone:char='None'
```

The *name* is a simple field (one occurrence), *rate* has two occurrences, and *phone* is a simple field. Both array elements and field occurrences are referenced beginning with zero (0). The *rate* is referenced by *rate(0)* or *rate(1)*. The *emps* array will contain 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.

Emps(0)	Name	Rate(0)	Rate(1)	Phone
Emps(1)				
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
Em(9)				

Figure 2-1 Sample Array

Examples

```
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
```

See Also

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

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
Specifies the name of the color palette.

COLOR_1
Specifies the first color in the palette.

COLOR_2
Specifies the second color in the palette.

COLOR_n
Specifies the n'th color in the palette. You can specify up to 64 colors in the palette.

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

Description
This command gives you the ability to create 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

- The DECLARE-CHART command.
- The PRINT-CHART command.

CREATE-LIST

Function Create a named list.

Syntax

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

Arguments

NAME
Specifies the name of the named list.

LIST
Specifies the values to be included in the named list.

Description

This command gives you the ability to create named lists of items. This command may be used anywhere within an SQR 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
```

#DEBUG

Function	Causes the current command to be processed during a debugging session.
Syntax	<code>#DEBUG [x...] sqr_command</code>
Arguments	<code>x</code> Any letter or digit.
Description	A <code>-DEBUG[xx]</code> flag in the SQR command line allows conditional compilation of SQR commands. When this flag is used, any command (including other compiler directives) preceded by the word <code>#DEBUG</code> is processed; otherwise, the command is ignored. This is useful for placing <code>DISPLAY</code> , <code>SHOW</code> , <code>PRINT</code> or other commands in your program for testing and for deactivating them when the report goes into production. The <code>-DEBUG</code> flag can be suffixed by up to 10 letters or digits. These characters are used to match any letters or digits appended to the <code>#DEBUG</code> preprocess command inside the program. <code>#DEBUG</code> commands with one or more matching suffix characters are processed; other commands are ignored. Commands without any suffix always match. In addition, for each <code>-DEBUGxx</code> letter, a substitution variable is defined. For example, if the flag <code>-DEBUGab</code> is used on the command line, three substitution variables are defined: <code>debug</code> , <code>debuga</code> , and <code>debugb</code> . These variables can be referenced in <code>#IFDEF</code> commands to turn whole sections of code on or off for debugging.
Examples	The following SQR command line contains the <code>-DEBUG</code> flag with no suffixes: <code>sqr myprog sammy/baker -debug</code> The following <code>SHOW</code> command in the program executes if invoked with the previous command line because the <code>-DEBUG</code> flag was used: <code>#debug show 'The total is ' #grand-tot 999,999,999</code>

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

The #IF, #IFDEF, and #IFNDEF commands.

DECLARE-CHART

Function Defines the attributes of a chart that can later be displayed using PRINT-CHART.

Syntax

```
DECLARE-CHART chart_name
[TYPE=chart_type_lit]
[CHART-SIZE=(chart_width_int_lit,chart_depth_int_lit)]
[TITLE=title_txt_lit]
[SUB-TITLE=subtitle_txt_lit]
[FILL=fill_lit]
[3D-EFFECTS=3d_effects_lit]
[BORDER=border_lit]
[COLOR-PALETTE=color_palette_lit]
[POINT-MARKERS=point_markers_lit]
[ATTRIBUTES={selector_lit|
  LIST:{selector_list_name_lit| (selector_lit,...)}},
   {decl_key_lit,{decl_value_lit|
     LIST:{decl_val_list_name_lit| (decl_val_lit,...)}|
       PALETTE:{color_palette_lit}}},...}]]
[DATA-ARRAY=array_name]
[DATA-ARRAY-ROW-COUNT=row_count_num_lit]
[DATA-ARRAY-COLUMN-COUNT=column_count_num_lit]
[DATA-ARRAY-COLUMN-LABELS={NONE|array_name|
  (txt_lit,...)}]
[DATA-LABELS=data_labels_lit]
[FOOTER-TEXT=NONE|text_lit]
[SUB-FOOTER-TEXT=NONE|text_lit]
[ITEM-COLOR=(item_color_keyword_lit,{color_value_lit|
  (r,g,b))}]
[ITEM-SIZE=(item_size_keyword_lit,item_size_num_lit)]
[LEGEND=legend_lit]
[LEGEND-TITLE=legend_title_txt_lit]
[LEGEND-PLACEMENT=legend_placement_lit]
[LEGEND-PRESENTATION=legend_presentation_lit]
[PIE-SEGMENT-QUANTITY-DISPLAY=pie_segment_quantity_
  display_lit]
[PIE-SEGMENT-PERCENT-DISPLAY=pie_segment_percent_
  display_lit]
[PIE-SEGMENT-EXPLODE=pie_segment_explode_lit]
[X-AXIS-GRID=x_axis_grid_lit]
[X-AXIS-LABEL=x_axis_label_txt_lit]
```

```

[X-AXIS-MIN-VALUE={x_axis_min_value_lit|_num_lit}]
[X-AXIS-MAX-VALUE={x_axis_max_value_lit|_num_lit}]
[X-AXIS-MAJOR-INCREMENT={x_axis_major_increment_lit|
    _num_lit}]
[X-AXIS-MINOR-INCREMENT=x_axis_minor_increment_num_lit]
[X-AXIS-MAJOR-TICK-MARKS=x_axis_major_tick_marks_lit]
[X-AXIS-MINOR-TICK-MARKS=x_axis_minor_tick_marks_lit]
[X-AXIS-TICK-MARK-PLACEMENT=x_axis_tick_mark_placement
    _lit]
[X-AXIS-ROTATE=x_axis_rotate_num_lit]
[X-AXIS-SCALE=x_axis_scale_lit]
[Y-AXIS-GRID=y_axis_grid_lit]
[Y-AXIS-LABEL=y_axis_label_lit]
[Y-AXIS-MASK=mask_txt_lit]
[Y-AXIS-MIN-VALUE={y_axis_min_value_lit|_num_lit}]
[Y-AXIS-MAX-VALUE={y_axis_max_value_lit|_num_lit}]
[Y-AXIS-MAJOR-INCREMENT={y_axis_major_increment_lit|
    _num_lit}]
[Y-AXIS-MINOR-INCREMENT=y_axis_minor_increment_num_lit]
[Y-AXIS-MAJOR-TICK-MARKS=y_axis_major_tick_marks_lit]
[Y-AXIS-MINOR-TICK-MARKS=y_axis_minor_tick_marks_lit]
[Y-AXIS-TICK-MARK-PLACEMENT=y_axis_tick_mark_placement
    _lit]
[Y-AXIS-SCALE=y_axis_scale_lit]
[Y2-AXIS-LABEL=y2_axis_label_lit]
[Y2-AXIS-MASK=mask_txt_lit]
[Y2-AXIS-MIN-VALUE={y2_axis_min_value_lit|_num_lit}]
[Y2-AXIS-MAX-VALUE={y2_axis_max_value_lit|_num_lit}]
[Y2-AXIS-MAJOR-INCREMENT={y2_axis_major_increment_lit|_
    num_lit}]
[Y2-AXIS-MINOR-INCREMENT=y2_axis_minor_increment_
    num_lit]
[Y2-AXIS-MAJOR-TICK-MARKS=y2_axis_major_tick_marks_lit]
[Y2-AXIS-MINOR-TICK-MARKS=y2_axis_minor_tick_marks_lit]
[Y2-AXIS-SCALE=y2_axis_scale_lit]
[Y2-COLOR-PALETTE=color_palette_lit]
[Y2-DATA-ARRAY=array_name]
[Y2-DATA-ARRAY-ROW-COUNT=row_count_num_lit]
[Y2-DATA-ARRAY-COLUMN-COUNT=column_count_num_lit]
[Y2-DATA-ARRAY-COLUMN-LABELS={NONE|array_name|
    (txt_lit,...)}]
[Y2-TYPE=chart_type_lit]
END-DECLARE

```

**Note**

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

Arguments

Table 2-11 describes the arguments for the DECLARE-CHART command (These arguments are also valid for the PRINT-CHART command). The default values are underlined.

**Note**

Several of the arguments in Table 2-11 refer to NewGraphics. To invoke NewGraphics, change the NewGraphics entry in the [Default-Settings] section of the SQR.INI file to TRUE (NewGraphics=True). (See NewGraphics under “[Default-Settings] Section” on page 5-4 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 2-11 **DECLARE-CHART Command Arguments**

Argument	Choices	Description
3D-EFFECTS	YES NO	<p>Gives your chart depth if set to YES. If set to NO, the chart is displayed in the default two-dimensional mode.</p> <p>This parameter must be set to YES for other 3D parameters to function.</p>
ATTRIBUTES	See “Attributes Argument” on page 2-98 for information on the valid choices.	<p>Defines the appearance of a chart.</p> <p>SQR reads and processes the keywords in the ATTRIBUTES argument left-to-right and first-to-last. As a result, a subsequent value setting overrides a previously-established value. Values assigned with DECLARE-CHART are overridden by values assigned with PRINT-CHART.</p> <p>Setting the ATTRIBUTES for the ALL selector establishes a default value for all property values within a chart. Any subsequent entry for a specific area, such as Header, overrides the value previously established by the ALL selector. Any invalid combination of selectors, sub-selectors, or declarations produce an error.</p> <p>Note: Some of the keywords in the ATTRIBUTES argument replace the functionality previously provided by the ITEM-COLOR, ITEM-SIZE, LEGEND-PLACEMENT, FILL, and COLOR-PALETTE arguments. SQR processes all old style keyword, value parameters prior to the new ATTRIBUTES argument. This may result in the new ATTRIBUTES argument overriding a value previously established with the old style keyword, value parameter pairs.</p>
BORDER	YES NO	If YES, then a border is drawn around the chart. If NO, then no border is displayed around the chart.
chart_name	User defined chart name.	A unique name to be used for referencing a chart.
CHART-SIZE	User defined chart size.	The size of the chart frame in standard SQR coordinate units.

Table 2-11 **DECLARE-CHART** Command Arguments (*Continued*)

Argument	Choices	Description
COLOR-PALETTE	palette_name	Sets the color palette to be used to color the individual data points in each chart (ex: Bar, slice, point). Use CREATE-COLOR-PALETTE to define a valid SQR color palette to use COLOR-PALETTE. Note: The defined color palette is only valid when NewGraphics=TRUE. Note: The FOREGROUND and BACKGROUND declaration keywords in the ATTRIBUTES argument (see “BACKGROUND” on page 2-100 and “FOREGROUND” on page 2-102) replace the functionality provided by COLOR-PALETTE. As a result, a value set with FOREGROUND or BACKGROUND overrides a value set with COLOR-PALETTE.
DATA-ARRAY Y2-DATA-ARRAY	array_name	Specifies the name of the array containing the data to be plotted. This must be the name of an array defined with CREATE-ARRAY. Use the DATA-ARRAY argument to define the data array for the Y-Axis. Use the Y2-DATA-ARRAY argument to define the data array for the Y2-Axis. (Y2-Axis values are ignored for pie charts.) Y2-DATA-ARRAY is only available with NewGraphics.
DATA-ARRAY-ROW-COUNT	row_count	Specifies the number of rows or sets of data to be used from the DATA-ARRAY. If the DATA-ARRAY has a greater number of rows, only DATA-ARRAY-ROW-COUNT is included in the chart.
Y2-DATA-ARRAY-ROW-COUNT	row_count	(NewGraphics) Specifies the number of rows or sets of data to be used from the Y2-DATA-ARRAY. If the Y2-DATA-ARRAY has a greater number of rows, only Y2-DATA-ARRAY-ROW-COUNT is included in the chart.
DATA-ARRAY-COLUMN-COUNT	column_count	Specifies the number of columns to be used from the DATA-ARRAY. If the DATA-ARRAY has a greater number of columns, only DATA-ARRAY-COLUMN-COUNT is included in the chart.

Table 2-11 **DECLARE-CHART** Command Arguments (*Continued*)

Argument	Choices	Description
Y2-DATA-ARRAY-COLUMN-COUNT	column_count	(New Graphics) Specifies the number of columns to be used from the Y2-DATA-ARRAY. If the Y2-DATA-ARRAY has a greater number of columns, only Y2-DATA-ARRAY-COLUMN-COUNT is included in the chart.
DATA-ARRAY-COLUMN-LABELS	<u>NONE</u> array_name (label1,label2, ...)	Specifies labels for each Y-Axis value of the data set (fields) in DATA-ARRAY. These labels are displayed in the legend box. Column labels are ignored for pie charts. See Table 2-49 on page 2-289 for applicable fields for each type of chart.
Y2-DATA-ARRAY-COLUMN-LABELS	<u>NONE</u> array_name (label1,label2, ...)	(New Graphics) Specifies labels for each Y2-Axis value of the data set (fields) in Y2-DATA-ARRAY. These labels are displayed in the legend box. Column labels are ignored for pie charts. See Table 2-49 on page 2-289 for applicable fields for each type of chart.
DATA-LABELS	YES <u>NO</u>	(New Graphics) If YES, SQR prints the numeric value above the individual data points. If NO, no numeric values are displayed.
FILL	<u>GRAYSCALE</u> COLOR CROSSHATCH NONE	Specifies the type of filling applied to the shapes (bars, pie-segments, and so on) that represent the data loaded in the chart. GRAYSCALE varies the density of black dots. COLOR sends color instructions to the current printer. If the current printer does not actually support color, then it could appear in a GRAYSCALE fashion. CROSSHATCH uses patterns to fill the shapes representing each data set. With NONE all graph shapes are filled with white. Note: The STYLE declaration keyword in the ATTRIBUTES argument (see "STYLE" on page 2-103) replaces the functionality provided by FILL. As a result, a value set with STYLE overrides a value set with FILL.
FOOTER-TEXT	<u>NONE</u> text	Specifies the footer text for the chart. The text is placed at the bottom of the chart. Default value is NONE.

Table 2-11 DECLARE-CHART Command Arguments (*Continued*)

Argument	Choices	Description
ITEM-COLOR	ChartBackground – Background color of entire chart area. ChartForeground – Text and Line color of chart area. HeaderBackground – Area within the text box specified for the title and sub-title. HeaderForeground – Text color of the Title and sub-title. LegendBackground – Area within the box defining the legend. LegendForeground – Text and Outline color of the legend. ChartAreaBackground – Area that includes the body of the chart. ChartAreaForeground – Text and Line colors of the chart area. PlotAreaBackground – Area within the X and Y Axis of a chart. Note: For more information see, “Changing Colors with New Graphics” on page 14-11 in the book titled <i>Hyperion SQR Getting Started Guide</i> .	(NewGraphics) Sets the color for an individual item in a chart. Specify a chart item and a valid (r,g,b) color to set the color of the chart item. Note: The COLOR declaration keyword in the ATTRIBUTES argument (see “COLOR” on page 2-101) replaces the functionality provided by ITEM-COLOR. As a result, a value set with COLOR overrides a value set with ITEM-COLOR.

Table 2-11 **DECLARE-CHART Command Arguments (Continued)**

Argument	Choices	Description
ITEM-SIZE	Title SubTitle XAxisLabel XAxisMarkers YAxisLabel YAxisMarkers Y2AxisLabel Y2AxisMarkers LegendText LegendTitle	<p>(NewGraphics) Sets the size of the following chart objects. The value is based on HTML text sizes. See the note on page 2-85.</p> <ul style="list-style-type: none"> ■ Title - Chart title ITEM-SIZE= ('Title',value) ■ SubTitle - Chart subtitle ITEM-SIZE= ('SubTitle',value) ■ XAxisLabel - Label below the X Axis of the chart ITEM-SIZE= ('XAxisLabel',value) ■ XAxisMarkers - Point labels on the X Axis ITEM-SIZE= ('XAxisMarkers',value) ■ YAxisLabel - Rotated label to the left of the chart ITEM-SIZE= ('YAxisLabel',value) ■ YAxisMarkers - Point labels on the Y Axis ITEM-SIZE= ('YAxisMarkers',value) ■ Y2AxisLabel - Rotated Label to the right of the chart ITEM-SIZE= ('Y2AxisLabel',value) ■ Y2AxisMarkers - Point labels on the Y Axis ITEM-SIZE= ('Y2AxisMarkers',value) ■ LegendText - Legend text ITEM-SIZE= ('LegendText',value) ■ LegendTitle - Legend title ITEM-SIZE= ('LegendTitle',value) <p>Note: If you do not define an ITEM-SIZE value, SQR uses the HTML text value of 3 (normal size).</p> <p>Note: The POINT-SIZE declaration keyword in the ATTRIBUTES argument (see “POINT-SIZE” on page 2-102) replaces the functionality provided by ITEM-SIZE. As a result, a value set with POINT-SIZE overrides a value set with ITEM-SIZE.</p>
LEGEND	<u>YES</u> NO	YES, displays a legend. NO does not display a legend.

Table 2-11 DECLARE-CHART Command Arguments (*Continued*)

Argument	Choices	Description
LEGEND-PLACEMENT	CENTER-RIGHT CENTER-LEFT UPPER-RIGHT UPPER-LEFT UPPER-CENTER LOWER-RIGHT LOWER-LEFT LOWER-CENTER	Places the legend in the specified location on the chart. The first portion of the placement parameter (CENTER, UPPER, or LOWER) is the vertical position, and the second portion is the horizontal. Note: The LOCATION declaration keyword in the ATTRIBUTES argument (see “LOCATION” on page 2-102) replaces the functionality provided by LEGEND-PLACEMENT. As a result, a value set with LOCATION overrides a value set with LEGEND-PLACEMENT.
LEGEND-PRESENTATION	INSIDE OUTSIDE	If INSIDE, then the legend is presented inside the area defined by the two axes. If OUTSIDE, then the legend is presented within the chart border, but outside of the region represented by the two axes.
LEGEND-TITLE	<u>NONE</u> text	Specifies the title for the legend. If NONE, then no title is displayed in the legend box.
PIE-SEGMENT-EXPLODE	<u>NONE</u> MAX MIN USE-3RD-DATA-COLUMN	Controls what pie segments are exploded (selected) within the pie chart. MAX selects the largest segment. MIN selects the smallest segment. If USE-3RD-DATA-COLUMN, then the third field in the DATA-ARRAY is used to determine which pie segments are exploded. This third field should be a CHAR and values of 'YES' or 'Y' indicate that the segment should be exploded.
PIE-SEGMENT-PERCENT-DISPLAY	<u>YES</u> NO	If YES, then percent-of-total figures is presented for each pie segment.
PIE-SEGMENT-QUANTITY-DISPLAY	<u>YES</u> NO	If YES, then the quantity is presented for each pie segment.
POINT-MARKERS	<u>YES</u> NO	If YES, then point markers are displayed for line charts. If NO, then point markers are not displayed.
SUB-FOOTER-TEXT	<u>NONE</u> text	Specifies the sub-footer text for the chart. The text is placed below the footer regardless of whether the FOOTER-TEXT is specified.
SUB-TITLE	<u>NONE</u> text	Default value is NONE. Specifies a subtitle for the chart. Text is placed below the title regardless of whether or not TITLE is specified.

Table 2-11 **DECLARE-CHART Command Arguments (Continued)**

Argument	Choices	Description
TITLE	<u>NONE</u> text	Specifies a title for the chart. Text is placed at the top of the chart.
TYPE Y2-TYPE (no pie charts)	LINE PIE <u>BAR</u> STACKED-BAR 100%-BAR OVER-LAPPED-BAR FLOATING-BAR HISTOGRAM AREA STACKED-AREA 100%-AREA XY-SCATTER-PLOT HIGH-LOW-CLOSE BUBBLE NONE	Specifies the type of chart. See Chapter 14, “Business Charts” in <i>Hyperion SQR Getting Started Guide</i> . (Y2-TYPE is only available with NewGraphics.)
	Note: NONE is only valid for Y2-TYPE.	
Y2-AXIS-COLOR-PALETTE	palette_name	(NewGraphics) Set the color palette to be used to color the individual data points in each chart (ex: Bar, slice, point). A valid SQR color-palette must have been defined using the CREATE-COLOR-PALETTE command.
X-AXIS-GRID Y-AXIS-GRID Y2-AXIS-GRID	YES <u>NO</u> <u>YES</u> NO YES <u>NO</u>	If YES, then a dashed grid line is drawn for each major tick-mark on the axis. If NO, then no grid line is drawn on this axis.
X-AXIS-LABEL Y-AXIS-LABEL Y2-AXIS-LABEL	<u>NONE</u> text	Specifies a line of text to be displayed below (or alongside) the tick-mark labels on the axis. (Y2-AXIS-LABEL is only available with NewGraphics.)
X-AXIS-MAX-VALUE	<u>AUTOSCALE</u> number	The maximum value on the axis. If there are data values that are greater than X-AXIS-MAX-VALUE, then they are not displayed. AUTOSCALE directs SQR to calculate an appropriate maximum value.
Y-AXIS-MAX-VALUE	<u>AUTOSCALE</u> number	The maximum value on the axis. If there are data values that are greater than Y-AXIS-MAX-VALUE, then they are not displayed. AUTOSCALE directs SQR to calculate an appropriate maximum value.
Y2-AXIS-MAX-VALUE	<u>AUTOSCALE</u> number	(NewGraphics) The maximum value on the axis. If there are data values that are greater than Y2-AXIS-MAX-VALUE, then they are not displayed. AUTOSCALE directs SQR to calculate an appropriate maximum value.

Table 2-11 DECLARE-CHART Command Arguments (*Continued*)

Argument	Choices	Description
X-AXIS-MIN-VALUE	<u>AUTOSCALE</u> number	The minimum value on the axis. If there are data values that are less than X-AXIS-MIN-VALUE, then they are not displayed. AUTOSCALE directs SQR to calculate an appropriate minimum value.
Y-AXIS-MIN-VALUE	<u>AUTOSCALE</u> number	The minimum value on the axis. If there are data values that are less than Y-AXIS-MIN-VALUE, then they are not displayed. AUTOSCALE directs SQR to calculate an appropriate minimum value.
Y2-AXIS-MIN-VALUE	<u>AUTOSCALE</u> number	(NewGraphics) The minimum value on the axis. If there are data values that are less than Y2-AXIS-MIN-VALUE, then they are not displayed. AUTOSCALE directs SQR to calculate an appropriate minimum value.
X-AXIS-MAJOR-TICK-MARKS	<u>YES</u> NO	YES specifies the display tick-marks along the axis between X-AXIS-MIN-VALUE and X-AXIS-MAX-VALUE, according to the X-AXIS-SCALE setting spaced by X-AXIS-MAJOR-INCREMENT.
Y-AXIS-MAJOR-TICK-MARKS	<u>YES</u> NO	YES displays the tick-marks along the axis between Y-AXIS-MIN-VALUE and Y-AXIS-MAX-VALUE, according to the Y-AXIS-SCALE setting spaced by Y-AXIS-MAJOR-INCREMENT.
Y2-AXIS-MAJOR-TICK-MARKS	<u>YES</u> NO	(NewGraphics) YES displays the tick-marks along the axis between Y2-AXIS-MIN-VALUE and Y2-AXIS-MAX-VALUE, according to the Y2-AXIS-SCALE setting spaced by Y2-AXIS-MAJOR-INCREMENT.
Y-AXIS-MASK	'\$999,999.99'	Specifies the numeric mask to be used to format the Y Axis. Follows the edit mask rules as defined in Table 2-40.
Y2-AXIS-MASK	'099999'	(NewGraphics) Specifies the numeric mask to be used to format the Y2 Axis. Follows the edit mask rules as defined in Table 2-40.
X-AXIS-MINOR-TICK-MARKS	YES <u>NO</u>	YES displays the tick-marks along the axis between X-AXIS-MIN-VALUE and X-AXIS-MAX-VALUE, according to the X-AXIS-SCALE setting spaced by X-AXIS-MINOR-INCREMENT.

Table 2-11 DECLARE-CHART Command Arguments (*Continued*)

Argument	Choices	Description
Y-AXIS-MINOR-TICK-MARKS	YES NO	YES displays the tick-marks along the axis between Y-AXIS-MIN-VALUE and Y-AXIS-MAX-VALUE, according to the Y-AXIS-SCALE setting spaced by Y-AXIS-MINOR-INCREMENT.
Y2-AXIS-MINOR-TICK-MARKS	YES NO	(NewGraphics) YES displays the tick-marks along the axis between Y2-AXIS-MIN-VALUE and Y2-AXIS-MAX-VALUE, according to the Y2-AXIS-SCALE setting spaced by Y2-AXIS-MINOR-INCREMENT.
X-AXIS-MAJOR-INCREMENT Y-AXIS-MAJOR-INCREMENT Y2-AXIS-MAJOR-INCREMENT	AUTOSCALE number	Specifies for SQR the increment used for spacing the major tick-marks on the axis. AUTOSCALE directs SQR to determine an appropriate increment. (Y2-AXIS-MAJOR-INCREMENT is only available with NewGraphics)
X-AXIS-MINOR-INCREMENT Y-AXIS-MINOR-INCREMENT Y2-AXIS-MINOR-INCREMENT	number	Specifies for SQR the increment used for spacing the minor tick-marks on the axis. These must be set for the X-AXIS-MINOR-TICK-MARKS, the Y-AXIS-MINOR-TICK-MARKS, or the Y2-AXIS-MINOR-TICK-MARKS to be displayed. (Y2-AXIS-MINOR-INCREMENT is only available with NewGraphics)
X-AXIS-ROTATE	X-Axis-Rotate = 0 No Rotation X-Axis-Rotate = 1 Always Rotate X-Axis-Rotate = n Rotate Labels if: Data-Array-Row-Count > n	Sets the rule for X Axis rotation of Markers. The default value is 5. These arguments have no meaning in NewGraphics.
X-AXIS-TICK-MARK-PLACEMENT Y-AXIS-TICK-MARK-PLACEMENT	INSIDE OUTSIDE BOTH	INSIDE (or OUTSIDE) directs SQR to place the tick marks on the inside (or outside) of the axis only. BOTH directs SQR to draw the tick-marks such that they appear on both sides of the axis. These arguments have no meaning in NewGraphics.
X-AXIS-SCALE Y-AXIS-SCALE Y2-AXIS-SCALE	LOG LINEAR	LOG specifies a logarithmic scale for the axis. Otherwise, the scale is LINEAR. (Y2-AXIS-SCALE is only available with NewGraphics)

Description

The DECLARE-CHART command 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. The DECLARE-CHART command can only appear in the SETUP section.

A chart defined with DECLARE-CHART prints by referencing its name in the PRINT-CHART command. You can override some or all of the chart attributes at run-time with the PRINT-CHART command. 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 SQR 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 SQR program. The following example illustrates this functionality.

```
Begin-Setup
    Declare-Chart Default-Chart
        Attributes= ('All','Font',31,'Font-Style','Plain',
                      'Point-Size',12)
        Attributes= ('Header','Font',31,'Font-Style','Bold',
                      'Point-Size',18)
        Attributes= ('chart1','start-angle',0,
                      'threshold-method','percent','threshold-value',20)
        Attributes= ('chart1','3d-depth',5,'3d-rotation',65,
                      '3d-elevation',85)
        Attributes= ('chart1','cluster-width',75,'cluster-overlap',65)
        Attributes= ('legend','location','lower-center')
        Attributes= ('chart1.fill','style',
                      LIST:('25per','50per','75per'))
        Attributes= ('chart1.line','style',LIST:'linestyle')
        Attributes= ('all.line','size',2)
        Attributes= ('all.line','color',
                      LIST:((('red'),('green'),('blue'))))
        Attributes= ('chart1','sort-order','Largest')
        Attributes= ('chart1','Background',(230,230,255))
        Attributes= ('all-axis.marker','foreground',(230,0,100))
    End-Declare

    Create-Array
        Name = emp_sales
        Size = 20
        Field = col_name:char:1
        Field = sales:number:3
    End-Setup
```

```

Begin-Report
  Create-List
    name='linestyle'
    list=('longdash','shortdash','dashdot','longshort')

    Put  'Madeline' 10 12 12
    Into emp_sales(0)  col_name(0)
          sales(0) sales(1) sales (2)
    Put  'Jacob' 25 35 45
    Into emp_sales(1)  col_name(0)
          sales(0) sales(1) sales (2)
    Put  'Evan' 18 28 38
    Into emp_sales(2)  col_name(0)
          sales(0) sales(1) sales (2)
    Put  'Claire' 60 70 80
    Into emp_sales(3)  col_name(0)
          sales(0) sales(1) sales (2)
  Print-Chart Default-Chart (4,1)
    Chart-Size = (50,50)
    Title = 'Employee Sales'
    Type = overlapped-bar
    3D-Effects = yes
    X-Axis-Label = 'Employees'
    Y-Axis-Label = 'Sales (in thousands)'
    Sub-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 FILL specification in the DECLARE-PRINTER command can influence the appearance of the chart. Table 2-12 lists the final appearance of the chart with a combination of values for PRINTER.COLOR and CHART.FILL options.

Table 2-12 PRINTER.COLOR setting effect on CHART.FILL

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

Examples

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

```
begin-setup

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

end-setup

begin-program

print-chart base_sales_chart
sub-title          = 'Region I'
data-array          = reg1_sales
data-array-row-count = #rows_reg1
data-array-column-count = 2
y-axis-max-value   = #max_of_all_regions
y-axis-min-value   = #min_of_all_regions
legend             = no

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

end-program

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

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

end-procedure
```

Attributes Argument

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

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

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

```
ATTRIBUTES= ('All', 'Point-Size', 12)
```

If desired, you can override a specific chart element property. 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 the CREATE-LIST command) and multiple declarations. For example, the following statement sets the point-size and foreground text color for Title and Sub-Title.

```
ATTRIBUTES= (LIST:('Title','Sub-Title'),  
            'Point-size',16,foreground, ('red'))
```

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

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

Possible sub-selector values include: MARKER, LABEL, LINE, GRID, SYMBOL, and FILL.



Note For Combination charts, use CHART1 and CHART2 selectors to identify chart elements for the primary and secondary charts, respectively. In the following example, the grid color for the primary chart is set to red:

```
ATTRIBUTES= ('Chart1.Grid', 'Color', ('Red'))
```

Table 2-13 shows the valid selector/sub-selector combinations.

Table 2-13 Valid Selector/Sub-selector Combinations

	MARKER	LABEL	LINE	GRID	SYMBOL	FILL
ALL	●	●	●	●	●	●
CHART1	●	●	●	●	●	●
CHART2	●	●	●	●	●	●
ALL-AXIS	●	●				
X-AXIS	●	●				
Y-AXIS	●	●				
Y2-AXIS	●	●				

Some examples of valid combinations using the selectors/sub-selectors in Table 2-13 include Chart1.Line, X-Axis.Marker, and Y2-Axis.Label.

Declaration Keywords

A *declaration* identifies a chart property followed by its value. Table 14 lists the declaration keywords available for the ATTRIBUTES argument.

Table 14 ATTRIBUTES Declaration Keywords

Declaration Keyword	Choices	Description
3D-DEPTH	Valid values are between 0 and 100.	Controls the depth of the three-dimensional display in percent. The default value is 10 percent. Note: You must set the 3D-EFFECTS argument to YES for this keyword to work.
3D-ELEVATION	Valid values are between 0 and 180.	Controls the elevation of the three-dimensional display in degrees. The default value is 45 degrees. Note: You must set the 3D-EFFECTS argument to YES for this keyword to work.
3D-ROTATION	Valid values are between 0 and 90.	Controls the rotation of the three-dimensional display in degrees. The default value is 40 degrees. Note: You must set the 3D-EFFECTS argument to YES for this keyword to work.
BACKGROUND	Named color or value in the range of RGB. Refer to “DECLARE-COLOR-MAP” on page 2-107 for an explanation of RGB values.	Sets the background color of a selected chart area. The default value is Transparent. Note: Along with FOREGROUND, BACKGROUND replaces the functionality provided by the COLOR-PALETTE (see page 2-87) command argument.
CLUSTER-WIDTH	Valid values are between 0 and 100.	Controls the percentage of available space between each bar cluster. The default value is 80%.

Table 14 ATTRIBUTES Declaration Keywords (*Continued*)

Declaration Keyword	Choices	Description
CLUSTER-OVERLAP	Valid values are between -100 and 100.	Controls the percentage of bar overlap. Negative values add space between bars. Positive values cause bar to overlap. The default value is 0%.
COLOR	Use the list generated from the CREATE LIST command, or enter 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_platte, 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.
FONT	Fonts are specified in the [Fonts] section of the SQR.INI file. If this section is not defined or a fontnum is not found, Times New Roman is used. Each entry consists of a font number assigned to a named font. For example, 3 may represent Courier.	COLOR can reference a color palette or a single color. For example: <pre>ATTRIBUTES= ('Chart2.Fill', 'Color', PALETTE: 'my_colors') ATTRIBUTES= ('Chart1.grid', 'Color', ('Black'))</pre> Note: COLOR replaces the functionality provided by the ITEM-COLOR (see page 2-89) command argument. Defines the font for all text and/or for specific text areas within a chart image. The default font is Times New Roman.
FONT-STYLE	PLAIN BOLD ITALIC	Controls the style of the font for all text and/or for specific text areas within a chart image. If you specify multiple values, separate each value with a space. The default value is PLAIN.

Table 14 ATTRIBUTES Declaration Keywords (*Continued*)

Declaration Keyword	Choices	Description
FOREGROUND	Named color or values in the range of RGB. Refer to “DECLARE-COLOR-MAP” on page 2-107 in the for an explanation of RGB values.	Sets the text, outline, and line color of a selected chart area. The default value is Black. Note: Along with BACKGROUND, FOREGROUND replaces the functionality provided by the COLOR-PALETTE (see page 2-87) command argument.
LOCATION	LOWER-RIGHT CENTER-RIGHT UPPER-RIGHT LOWER-LEFT CENTER-LEFT UPPER-LEFT LOWER-CENTER UPPER-CENTER Note: The CENTER-CENTER Location is reserved for the ChartArea.	Positions the main chart elements of the Header, Footer, and Legend in pixels, or controls the specific location for the Legend. Each positional object is considered a rectangle, and the (x,y) pixel location of (1,1) relates to the upper left-hand coordinate of an image. The charting application positions the upper left-hand corner of the rectangle at the pixel location specified. Note: LOCATION replaces the functionality provided by the LEGEND-PLACEMENT (see page 2-91) command argument.
POINT-SIZE	Any point size.	Controls the font size in points for all areas within a chart image. Control can be for all text on a chart and/or for specific text areas on a chart. The default value is 12. Note: POINT-SIZE replaces the functionality provided by the ITEM-SIZE (see page 2-90) command argument.
SIZE	Value ranges depend on the SUB-SELECTOR used: <ul style="list-style-type: none">■ LINE – 1 to 10 pixels■ SYMBOL – 1 to 100 pixels■ GRID – 1 to 10 pixels	Defines the size of the lines, symbols, and grids for an individual 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.

Table 14 ATTRIBUTES Declaration Keywords (*Continued*)

Declaration Keyword	Choices	Description
SORT-ORDER	LARGEST SMALLEST DATAORDER	<p>Specifies whether to display slices largest-to-smallest, smallest-to-largest, or the order they appear in the data.</p> <ul style="list-style-type: none">■ LARGEST – Largest-to-smallest■ SMALLEST – Smallest-to-largest■ DATAORDER – Order they appear in the data. <p>The default value is DATAORDER.</p>
START-ANGLE	Ranges of values are 0 to 359 degrees.	<p>Provides control of the position in the pie chart where the first pie slice is drawn.</p> <p>A value of zero degrees represents a horizontal line from the center of the pie to the right-hand side of the pie chart.</p> <p>A value of 90 degrees represents a vertical line from the center to the top of the pie.</p> <p>Slices are drawn clockwise from the specified angle.</p> <p>The default position for the first pie segment is 135 degrees.</p>
STYLE	Values depend on the SUB-SELECTOR used: <ul style="list-style-type: none">■ LINE – Solid LongDash ShortDash LongShort DashDot■ SYMBOL – None Dot Box Triangle Diamond Star VerticalLine HorizontalLine Cross Circle Square■ FILL – None Solid 25Per 50Per 75Per HorizontalStripe VerticalStripe 45Stripe 135Stripe DiagonalHatch CrossHatch	<p>Defines a group of styles for an individual chart element.</p> <p>This keyword can use the list generated from the CREATE-LIST command, or you can enter an in-line list of values. When using a named list, the keyword LIST: must prefix the name of the list. An error occurs if the contents of the list do not match the data type expected.</p> <p>STYLE values are used in presentation order and are reused once the current list is exhausted. The STYLE keyword overrides the default values established for the style property of a chart.</p> <p>The default line and fill style is Solid. The default symbol style is Dot.</p> <p>Note: STYLE replaces the functionality provided by the FILL (see page 2-88) command argument.</p>

Table 14 ATTRIBUTES Declaration Keywords (*Continued*)

Declaration Keyword	Choices	Description
THRESHOLD-METHOD	VALUE PERCENT	<p>Selects the grouping method to use for the Other slice.</p> <ul style="list-style-type: none">■ VALUE – Use when you know the data value to group into the Other slice. Value places only those items that are less than the Threshold-Value into the Other slice.■ PERCENT – Use when you want to devote a certain percentage of the pie to the Other slice. Percent places only those items that are accumulatively less than the Threshold-Value into the Other slice. <p>The default is VALUE.</p>
THRESHOLD-VALUE	<p>Valid values depend on the THRESHOLD-METHOD keyword:</p> <ul style="list-style-type: none">■ When THRESHOLD-METHOD is set to <i>Value</i>, valid values are numbers greater than or equal to 0.■ When the THRESHOLD-METHOD is set to <i>Percent</i>, valid values are from 0 to 100.	<p>Sets the data value to group into the Other slice.</p> <p>The default value is 0 (No Other Slice).</p>

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

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

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

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

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

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

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

Table 2-15 and Table 2-16 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.) SQR does not allow invalid combinations of selectors, sub-selectors, and declarations.

Table 2-15 Valid Selector/Sub-selector - Declaration Keyword Combinations

	Color	Style	Size	Font Font Style Point Size	Sort Order	Background Colors	ForeGround Colors
Selectors							
ALL				●	●	●	●
CHART1				●	●	●	●
CHART2				●	●	●	●
HEADER				●		●	●
TITLE				●			●
SUBTITLE				●			●
FOOTER				●		●	●
FOOTER-TEXT				●			●
SUB-FOOTER-TEXT				●			●
CHART-AREA				●		●	●
PLOT-AREA				●		●	●
LEGEND				●		●	●
LEGEND-TITLE				●			●
ALL-AXIS				●			●
X-AXIS				●			●
Y-AXIS				●			●
Y2-AXIS				●			●
Sub-selectors							
MARKER				●			●

Table 2-15 Valid Selector/Sub-selector - Declaration Keyword Combinations (Continued)

	Color	Style	Size	Font Font Style Point Size	Sort Order	Background Colors	ForeGround Colors
LABEL				●			●
LINE	●	●	●				
GRID	●		●				
SYMBOL	●	●	●				
FILL	●	●					

Table 2-16 Valid Selector/Sub-selector - Declaration Keyword Combinations

	Start Angle	3d-Depth 3d-Elevation 3d-Rotation	Cluster-Width Cluster-Overlap	Threshold Value Threshold Method	Location
ALL	●	●	●	●	
CHART1	●	●	●	●	
CHART2	●	●	●	●	
HEADER					●
FOOTER					●
LEGEND					●

See Also

The PRINT-CHART command.

DECLARE-COLOR-MAP

Function Defines colors in an SQR report.

Syntax In the SETUP section:

```
DECLARE-COLOR-MAP  
color_name=({rgb})  
color_name=({rgb})  
.  
. .  
END-DECLARE
```

Arguments *color_name*
A *color_name* is composed of the alphanumeric characters (A-Z, 0-9), the underscore (_) character, and the dash (-) character. It must start with an alpha (A-Z) character. It is case insensitive. The name 'none' is reserved and cannot be assigned a value. A name in the format (RGBredgreenblue) cannot be assigned a value. The name 'default' is reserved and can be assigned a value. 'Default' is used during execution when a referenced color is not defined in the runtime environment.

{rgb}
red_lit|var|col, *green_lit_var|col*, *blue_lit_var|col* where each component is a value in the range of 000 to 255. In the BEGIN-SETUP section, only literal values are allowed.

The default colors implicitly installed with SQR 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

The DECLARE-COLOR-MAP command in the BEGIN-SETUP section defines or redefines colors in an SQR 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

The ALTER-COLOR-MAP, GET-COLOR, and SET-COLOR commands.

DECLARE-CONNECTION

Function	Defines the datasource logon parameters prior to logon. Can be used to override the default connection logon parameters.
-----------------	--



Note The DECLARE-CONNECTION command is specific to SQR/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

A user-defined name for describing a datasource connection.

DSN

The logical datasource name as recorded in the DDO Registry (Registry.properties file).

USER, PASSWORD

Traditional logon semantics.

PARAMETERS=*keyword_str=attr_str*;

Defines a 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)

This optional keyword prevents SQR 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 SQR, which allows a single CONNECTION declaration to be used 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

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

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

In this example, SET-GENERATIONS:

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

Refer to “SET-GENERATIONS” on page C-10 of *Accessing Data with SQR for DDO* for detailed examples of the SET-GENERATIONS command.

SET-LEVELS

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

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

In this example:

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

Refer to “SET-LEVELS” on page C-12 of *Accessing Data with SQR for DDO* for detailed examples of the SET-LEVELS command.

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 the SET-MEMBERS command can be a *literal* value only. Consider the following example:

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

In this example, SET-MEMBERS:

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

Refer to “SET-MEMBERS” on page C-8 of *Accessing Data with Hyperion SQR for DDO* for detailed examples of the SET-MEMBERS command.

Examples

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

See Also

The ALTER-CONNECTION command.

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

Specifies a unique name for referencing the image declaration.

TYPE

Specifies the image type. Types can be EPS-FILE, HPGL-FILE, GIF-FILE, JPEG-FILE, BMP-FILE, or PNG-FILE.

IMAGE-SIZE

Specifies the width and height of the image in SQR coordinates.

SOURCE

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

FOR-PRINTER

Specifies a 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

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

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 2-17 illustrates the valid relationships between image type and report output type.

Table 2-17 Valid Images Types

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

Examples

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

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

```
Begin-setup
    Declare-Image hyperion_logo
        type=GIF-FILE
        Image-size=(40,10)
        source=hyperion.gif
        for-printer=(PS, EPS-FILE, 'hyperion.eps')
        for-printer=(HP, HPGL-FILE, 'hyperion.hpgl')
    End-Declare
End-Setup

Begin-Report
    Move 'hyperion.bmp' to $image_src
    print-Image hyperion_logo (10,15)
        for-printer=(WP, BMP-FILE, $image_src)
End-Report
```

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

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

See Also

The PRINT-IMAGE command.

“Adding Graphics” on page 13-4 in the *Hyperion SQR Getting Started Guide*.

DECLARE-LAYOUT

Function Defines the attributes for the layout of an output file.

Syntax

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

Arguments

layout_name
A unique layout name to be used for referencing the layout and its attributes.

uom

An optional suffix which denotes the unit of measure to apply to the preceding value. Table 2-18 lists valid *uom* suffixes.

Table 2-18 Valid *uom* Suffixes

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

paper_name

An option of PAPER-SIZE. This name is associated with predefined dimensions. Table 2-19 lists valid paper names.

Table 2-19 Valid Paper Names

Name	Width	Depth	Orientation
Letter	8.5 in	11 in	Portrait
Legal	8.5 in	14 in	Portrait
A4	8.27 in	11.69 in	Portrait
Executive	7.25 in	10.5 in	Portrait
B5	7.17 in	10.12 in	Portrait
Com-10	4.125 in	9.5 in	Landscape
Monarch	3.875 in	7.5 in	Landscape
DL	4.33 in	8.66 in	Landscape
C5	6.378 in	9.016 in	Landscape

Table 2-20 describes the other arguments of the DECLARE-LAYOUT command.

Table 2-20 DECLARE-LAYOUT Command Arguments

Argument	Choice or Default uom	Default Value	Description
PAPER-SIZE	inches	8.5 in, 11 in	Physical size of the page. The first parameter is the width of the page. The second parameter is the depth or length. It may also be a predefined name. (See Table 2-19.) Note: When ORIENTATION= LANDSCAPE the default values are 11 in, 8.5 in. See the note on page 2-119.
FORMFEED	YES, NO	YES	Specifies whether formfeeds are to be written at the end of each page.
ORIENTATION	PORTRAIT, LANDSCAPE	PORTRAIT	Portrait pages are printed vertically. Landscape pages are printed horizontally. Printing in landscape for the printer type HPLASERJET requires landscape fonts.

Table 2-20 DECLARE-LAYOUT Command Arguments (*Continued*)

Argument	Choice or Default uom	Default Value	Description
LEFT-MARGIN	inches	0.5 in	Amount of blank space to leave at the left side of the page.
TOP-MARGIN	inches	0.5 in	Amount of blank space to leave at the top of the page.
RIGHT-MARGIN	inches	0.5 in	Amount of blank space to leave at the right side of the page. If you specify LINE-WIDTH or MAX-COLUMNS, you cannot use this parameter.
LINE-WIDTH	inches	7.5 in	Length of the line. If you specify RIGHT-MARGIN or MAX-COLUMNS, you cannot use this parameter.
MAX-COLUMNS		75	Maximum number of columns in a line. If you specify RIGHT-MARGIN or LINE-WIDTH, you cannot use this parameter.
BOTTOM-MARGIN	inches	0.5 in	Amount of blank space to leave at the bottom of the page. If you specify PAGE-DEPTH or MAX-LINES, you cannot use this parameter.
PAGE-DEPTH	inches	10 in	Depth of the page. If you specify BOTTOM-MARGIN or MAX-LINES, you cannot use this parameter.
MAX-LINES		60	Maximum number of lines printed on the page. If you specify PAGE-DEPTH or BOTTOM-MARGIN, you cannot use this parameter.
LINE-HEIGHT	points	12 pt	Size of each SQR 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).
CHAR-WIDTH	points	7.2 pt	Size of each SQR 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

The DECLARE-LAYOUT command describes the characteristics of a layout to be used 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.

SQR maps its line and column positions on the page by using a grid determined by the LINE-HEIGHT and CHAR-WIDTH arguments. That is, SQR 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 bolded 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 the DECLARE-LAYOUT command.

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 , then SQR 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, then the ORIENTATION value overrides the PAPER-SIZE value.
- When you specify PAPER-SIZE= (*page_width, page_depth*) is specified then SQR *does not* swap the page width and page depth if ORIENTATION=LANDSCAPE.

**Note**

If none of the following commands are present in an SQR report, none of the default values in Table 2-20 take effect, and the report is created with 62 lines by 132 characters.

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

Examples

The following example illustrates the ability to specify the ! parameters using a different measurement system (that is, metric).

```
!
declare-layout my-layout! Results in:

paper-size=(a4) ! paper-size=(210mm, 297mm)

left-margin=12.7 mm ! top-margin=12.7mm

right-margin=25.4 mm ! left-margin=12.7mm
end-declare           ! right-margin=25.4mm

! bottom-margin=12.7mm

! orientation=portrait

! columns=67

! lines=64
```

The following example changes the page dimensions and also changes the left and right margins to be 1 inch.

```
!
declare-layout large-paper! Results in:

paper-size=(14, 11) paper-size=(14in, 11in)

left-margin=1           ! top-margin=0.5in

right-margin=1           ! left-margin=1.0in
end-declare             ! right-margin=1.0in

! bottom-margin=0.5in

! orientation=portrait

! columns=120

! lines=60
```

The following example retains the default page dimensions and changes the left and right margins to be 1 inch.

```
declare-layout default! Results in:  
  
left-margin=1      ! paper-size=(8.5in, 11in)  
  
right-margin=1      ! top-margin=0.5in  
end-declare         ! left-margin=1.0in  
  
                      ! right-margin=1.0in  
                      ! bottom-margin=0.5in  
                      ! orientation=portrait  
                      ! columns=65  
                      ! lines=60
```

The following example changes the orientation to landscape. The default page dimensions of (8.5in and 11in) are swapped. The columns and rows are recalculated. All other values remain the same.

```
declare-layout default! Results in:  
  
orientation=landscape! paper-size=(11in, 8.5in)  
end-declare          ! top-margin=0.5in  
  
                      ! left-margin=0.5in  
                      ! right-margin=0.5in  
                      ! bottom-margin=0.5in  
                      ! columns=100  
                      ! lines=45
```

The following example changes the orientation to landscape. The default page dimensions of (8.5in and 11in) are swapped. In addition the top margin is set to 1 inch.

```
declare-layout my_landscape! Results in:  
orientation=landscape ! paper-size=(11in, 8.5in)  
  
top-margin=1      ! top-margin=1.0in  
end-declare       ! left-margin=0.5in  
  
                      ! right-margin=0.5in  
                      ! bottom-margin=0.5in  
                      ! orientation=landscape  
                      ! columns=100  
                      ! lines=43
```

The following example illustrates the ability to specify the page dimensions using one of the predefined names. Note that the orientation has also changed since this example is an envelope.

```
declare-layout envelope! Results in:  
paper-size=(com-10) ! paper-size=(4.125in, 9.5in)  
end-declare          ! top-margin=0.5in  
  
                      ! left-margin=0.5in  
                      ! right-margin=0.5in  
                      ! bottom-margin=0.5in  
                      ! orientation=landscape  
                      ! columns=85  
                      ! lines=18
```

See Also

The DECLARE-REPORT command.

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*

A unique name to be used for referencing a printer definition and its attributes.

Table 2-21 describes the other arguments of the DECLARE-PRINTER command. The table lists the options, default values, and description of each of the arguments.

Description Each printer has a set of defaults as listed in Table 2-21. The DECLARE-PRINTER command overrides these defaults.

Use the DECLARE-PRINTER command in the SETUP section to define the characteristics of the printer or printers to be used. If you need to change some of the arguments depending on the run-time environment, you can use the ALTER-PRINTER command 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 2-21 describes each of the arguments, the possible choices, and the default values.

Table 2-21 DECLARE-PRINTER Command Arguments

Argument	Choice or Measure	Default	Description
AFTER-BOLD	any string	(none)	See BEFORE-BOLD.
BEFORE-BOLD	any string	(none)	The BEFORE-BOLD and AFTER-BOLD arguments are for LINE-PRINTER 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=<27>[r ! Turn on bold AFTER-BOLD=<27>[u ! Turn it off
			These arguments work in conjunction with the BOLD argument of the PRINT command.
COLOR	Yes, No	No	Specifies whether or not this printer can print in color.
FONT	font_number	3	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 <i>Technical Reference Manual</i> . For POSTSCRIPT printers, SQR 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, 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 2-22 lists the fonts available in SQR internally. This table lists the fonts available in the SQR POSTSCRI.STR file.
FONT-TYPE	PROPORTIONAL, FIXED	Depends on the font	This argument applies only to HP LASERJET printers and needs to be specified only for font types not defined in on Table 2-22.

Table 2-21 DECLARE-PRINTER Command Arguments (*Continued*)

Argument	Choice or Measure	Default	Description
FOR-REPORTS		ALL	The name of the reports that use this printer definition. The default is ALL reports. This argument is required only for a program with multiple reports. If you are writing a program that produces only a single report, you can ignore this argument.
INIT-STRING		(none)	Sends control or other characters to the printer at the beginning of the report. This parameter is designed primarily for the LINE-PRINTER and has limited use with other printer types. Specify non-display characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character.
PITCH	characters/inch	10	This argument is required for HPLASERJET printers and the SPF Viewer. If you specify a fixed pitch font, you should also indicate the pitch.
POINT-SIZE	points	12	This argument does not apply to LINEPRINTER printers. This is the beginning size of the selected font.
RESET-STRING		(none)	Sends control or other characters to the printer at the end of the report. This parameter is designed primarily for the LINEPRINTER and has limited use with other printer types. Specify non-display characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character.
STARTUP-FILE	filename	POSTSCRI.STR	This argument applies only to POSTSCRIPT printers. This argument is used to specify an alternate startup file. Unless otherwise specified, the default startup file is located in the directory pointed to by the environment variable SQRDIR.
SYMBOL-SET	HP defined sets	OU	This argument applies only to HP LASERJET printers. The default value of "OU" is for the ASCII symbol set. For a complete list of the symbol sets, see the <i>HP LaserJet Technical Reference Manual</i> .
TYPE	LINEPRINTER, POSTSCRIPT, HPLASERJET, HTML, LP, PS, HP, HT	LP	<p>SQR creates output specific to each printer.</p> <ul style="list-style-type: none">■ LINEPRINTER (LP) files generally consist of ASCII characters and can be viewed by a text editor.■ POSTSCRIPT (PS) files consist of ASCII characters, but you need to know PostScript to understand what will be 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 2-22 lists the fonts available in SQR for use with the FONT argument for HPLaserJet printer types.

Table 2-22 Fonts Available for HP LaserJet Printers in SQR

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

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

For fonts not listed in Table 2-22, you must indicate the font style using the FONT-TYPE argument, or the correct typeface cannot be selected by the printer.

Table 2-23 lists the fonts available in SQR 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 2-23 Fonts Available for PostScript Printers

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

Table 2-23 Fonts Available for PostScript Printers (*Continued*)

Value	Typeface	Bold
40	Bookman Demi	
41	Bookman Light Italic	
42	Bookman Demi Italic	

Other type faces can be added to the POSTSCRI.STR file.

Table 2-24 lists the fonts available in SQR for Windows when printing on Windows printer drivers (using the -PRINTER:WP command-line flag). When you use the -PRINTER:WP flag, your report is sent directly to the default Windows printer. To specify a non-default Windows printer, use the following format: -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 FONT qualifier of the ALTER-PRINTER command by their number.

Table 2-24 Fonts Available for Windows Printers

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

Table 2-24 Fonts Available for Windows Printers (*Continued*)

Value	Windows Font/Name	Style
8	Palatino	Proportional
800	Palatino	Bold
11	Symbol	Proportional

Note

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

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

You can add more fonts by editing the appropriate [Fonts] section in the SQR.INI file. See the [Fonts] section in the SQR.INI file for more information.

Examples

```
declare-printer HP-definition! Default HP definition
type=HP! for all reports
font=4! Helvetica
symbol-set=12U! PC-850 Multilingual
end-declare
declare-printer PS-Sales! PS definition
for-reports=(sales)! for the Sales report
type=PS
font=5! Times-Roman
end-declare
```

See Also

The ALTER-PRINTER and DECLARE-REPORT commands.

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

Specifies one or more reports that use the given procedures. This argument is required only for a program with multiple reports. If you are writing a program that produces only a single report, you can ignore this argument.

BEFORE-REPORT

Specifies a procedure to be 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

Specifies a procedure to be 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

Specifies a procedure to be 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

Specifies a procedure to be 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 be passed to any of the procedures. Arguments can be any variable, column, or literal.

Description	The DECLARE-PROCEDURE command can be used to define SQR procedures that are to be invoked before or after a report is printed or before the beginning or end of each page.
	Issue the 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 be used at run-time. To turn a procedure off, specify NONE in the USE-PROCEDURE statement.

Examples

```

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

```

See Also

The USE-PROCEDURE command.

DECLARE-REPORT

Function	Defines reports and their attributes.
Syntax	<pre>DECLARE-REPORT <i>report_name</i> [TOC=<i>toc_name</i>] [LAYOUT=<i>layout_name</i>] [PRINTER-TYPE=<i>printer_type</i>] END-DECLARE</pre>
Arguments	<p><i>report_name</i> Specifies the name of the report.</p> <p>TOC Specifies the name of the Table of Contents for this report.</p> <p>LAYOUT Specifies the name of the layout for this report. If none is specified, the default layout is used.</p> <p>PRINTER-TYPE Specifies the type of printer to be used for this report. 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.</p>
Description	<p>Issue the DECLARE-REPORT in the SETUP section.</p> <p>You can use the DECLARE-REPORT command to declare one or more reports to be produced in the application.</p> <p>You must use this command when developing applications to produce more than one report.</p> <p>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.</p> <p>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 <i>program.lis</i>, where <i>program</i> is the application name.</p>

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

The USE-REPORT, DECLARE-LAYOUT, DECLARE-PRINTER, and DECLARE-TOC commands.

DECLARE-TOC

Function Defines the Table of Contents and its attributes.

Syntax

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

Arguments

toc_name

Specifies the name of the Table of Contents.

FOR-REPORTS

Specifies one or more reports that uses this Table of Contents.

DOT-LEADER

Specifies whether or not a dot leader precedes the page number. The default setting is NO.

INDENTATION

Specifies the number of spaces by which each level is indented. The default setting is 4.

BEFORE-TOC

Specifies a procedure to be executed before generating the Table of Contents. If no Table of Contents is generated, the procedure does not execute.

AFTER-TOC

Specifies a procedure to be executed after generating the Table of Contents. If no Table of Contents is generated, the procedure does not execute.

BEFORE-PAGE

Specifies a procedure to be executed at the start of every page.

AFTER-PAGE

Specifies a procedure to be executed at the end of each page.

ENTRY

Specifies a procedure that is executed to process each Table of Contents entry (instead of SQR doing it for you). When this procedure is invoked, the following SQR-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-toct-page). These three SQR-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 command to declare one or more Table of Contents for the application.

A Table of Contents can be shared between reports.

Example

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

See Also

The BEGIN-FOOTING, BEGIN-HEADING, DECLARE-REPORT, and TOC-ENTRY commands.

DECLARE-VARIABLE

Function Explicitly declares a variable type.

Syntax

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

Arguments

DEFAULT-NUMERIC

Specifies the 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 the SQR.INI file. If `-DNT` was not specified on the command line and the SQR.INI file entry has no `DEFAULT-NUMERIC` entry, then the default numeric type is `FLOAT`.

DECIMAL

Specifies 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 -9.99999999999999999999999999999999E±4096 to +9.99999999999999999999999999999999E±4096

FLOAT

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

DATE

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

INTEGER

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

TEXT

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

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 the **SQR.INI** file. However, the setting in the **DECLARE-VARIABLE** command 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 the **SQR.INI** file.

Besides **FLOAT**, **INTEGER**, or **DECIMAL**, the **DEFAULT-NUMERIC** setting in the **SQR.INI** file 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 the **DECLARE-VARIABLE** command.

The **DECLARE-VARIABLE** command 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.

The **DECLARE-VARIABLE** command, the **-DNT** command-line flag, and the **DEFAULT-NUMERIC** setting in the **SQR.INI** file 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** command, the **-DNT** command-line flag, and the **DEFAULT-NUMERIC** setting in the **SQR.INI** file is the same as specifying **V30**.



Note In SQR for 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 the SQR.INI file, described in Chapter 5, “SQR.INI.”

#DEFINE

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

Syntax #DEFINE *substitution_variable* *value*

Arguments *substitution_variable*

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

value

The value to be substituted.

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

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

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

The #DEFINE command cannot be broken across program lines.

Examples

The following code defines several constants:

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

The following excerpt from a report uses the preceding definitions:

```
begin-setup

declare-printer contacts

type=hp

paper-size={({page_width}, {page_depth})

end-declare
end-setup
begin-heading 5
    print 'Company Contacts' (1,1) center
    print 'Sort: {order_by}' (2,1) center
    print 'Company' (4,{col1})
    print 'Contact' (4,{col2})
    print 'Phone' (4,{col3})
end-heading
begin-procedure main
begin-select
company (1,{col1})
    print '{bold}' (0,{col2}) ! Print contact in boldface.
contact ()
    print '{light}' () ! Back to lightface.
phone (0,{col3}) ! Note: There must be enough
    next-listing ! space between col2
from customers ! and col3 for both
order by {order_by} ! font changes and the
end-select ! contact field.
end-procedure
```

See Also

The ASK command.

DISPLAY

Function	Displays the specified column, variable, or literal.
Syntax	<code>DISPLAY {any_lit _var _col} [[:\$]edit_mask NUMBER MONEY DATE] [NOLINE]</code>
Arguments	<i>any_lit _var _col</i> The text, number, or date to be displayed. <i>edit_mask</i> Causes the field to be edited before being displayed. For additional information regarding edit masks, see the command, “PRINT” on page 2-255. <i>NUMBER</i> Indicates that <i>any_lit _var _col</i> is to be formatted using the NUMBER-EDIT-MASK of the current locale. This option is not legal with date variables. <i>MONEY</i> Indicates that <i>any_lit _var _col</i> is to be formatted using the MONEY-EDIT-MASK of the current locale. This option is not legal with date variables. <i>DATE</i> Indicates that <i>any_lit _var _col</i> is to be formatted using 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 2-45 on page 2-270). <i>NOLINE</i> Suppresses the carriage return after the field is displayed.
Description	The DISPLAY command 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 SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

When displaying a date in a string literal, column, or variable using EDIT or DATE, the string uses the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats as listed in Table 2-45 on page 2-270, or the database-independent format SYYYMMDD [HH24 [MI [SS [NNNNNN]]]].

If you require more control over the display, use the SHOW command.

Examples

The following segments illustrate the various features of the DISPLAY command:

The following code:

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

Produces the following output:

```
123-45-6789
```

The following code:

```
!
! Display a number using an edit mask
!
display 1234567.89 999,999,999.99
```

Produces the following output:

```
1,234,567.89
```

The following code:

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

Produces the following output:

```
123.780000
```

The following code:

```
!
! Display a number using the locale default numeric edit mask
!
alter-locale number-edit-mask = '99,999,999.99'
display 123456.78 number
```

Produces the following output:

```
123,456.78
```

The following code:

```
!
! Display a number using the locale default money edit mask
!
alter-locale money-edit-mask = '$$, $$, $$9.99'
display 123456.78 money
```

Produces the following output:

```
$123,456.78
```

The following code:

```
!
! Display a date column using the locale default date edit mask
!
begin-select
dcol
    from tables
end-select
alter-locale date-edit-mask = 'DD-Mon-YYYY'
display &dcol date
```

Produces the following output:

```
01-Jan-1999
```

The following code:

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

Produces the following output:

```
Hello World
```

The following code:

```
!
! Display two values on the same line with editing of the values
!
alter-locale money-edit-mask = '$$, $$, $$9.99'
let #taxes = 123456.78
display 'You owe ' noline
display #taxes money noline
display ' in back taxes.'
```

Produces the following output:

```
You owe $123,456.78 in back taxes.
```

See Also

- The SHOW command for more information on screen control.
- The LET command for information on copying, editing, or converting fields.
- The EDIT parameter of the PRINT command for a description of the edit masks.
- The ALTER-LOCALE command for a description of the arguments NUMBER-EDIT-MASK, MONEY-EDIT-MASK, and DATE-EDIT-MASK.

DIVIDE

Function	Divides one number into another.
Syntax	<pre>DIVIDE {src_num_lit _var _col} INTO dst_num_var [ON-ERROR={HIGH ZERO}] [ROUND=nn]</pre>
Arguments	<p><i>src_num_lit _var _col</i> Divided into the contents of <i>dst_num_var</i>.</p> <p><i>dst_num_var</i> Contains the result after execution.</p> <p>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, SQR halts with an error message.</p> <p>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.</p>
Description	<p>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.</p> <p>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.</p>
Examples	<pre>divide 37.5 into #price ! #price / 37.5 divide &rate into #tot on-error=high divide #j into #subtot on-error=zero</pre>

**Note**

High in the preceding example is the “Maximum Value,” while zero is the “Lowest Value.”

See Also

- The ADD command for more information.
- The LET command for a discussion of complex arithmetic expressions.

DO

Function	Invokes the specified procedure.
Syntax	<code>DO procedure_name[(arg1[, argi]...)]</code>
Arguments	<i>procedure_name</i> Specifies the name of the procedure to be executed. <i>arg1 [, argi]</i> Specifies the arguments to be passed to the procedure. Arguments can be any type of variable or constant value.
Description	When the procedure ends, processing continues with the command following the DO command. You can use arguments to send values to or receive values from a procedure. Arguments passed by a DO command to a procedure must match in number: <ul style="list-style-type: none">■ 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. SQR 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 a DO command 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 SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

Examples

```
do get_names  
do add_to_list ($name)  
do print_list ('A', #total, &co_name, $name)
```

See Also

The BEGIN-PROCEDURE command for more information about passing arguments.

#ELSE

Function

Compiles the code following the #ELSE command when a preceding #IF, #IFDEF, or #IFNDEF command is FALSE. (#ELSE is a compiler directive that works with the #IF, #IFDEF, and #IFNDEF compiler directives.)

Syntax

#ELSE

See Also

The #IF, #IFDEF, and #IFNDEF commands for a description of each compiler directive.

ELSE

Function ELSE is an optional command in an IF command.

Syntax ELSE

See Also The IF command for a description and example.

ENCODE

Function	Assigns a non-display or display character to a string variable.
Syntax	<code>ENCODE <i>src_code_string_lit</i> INTO <i>dst_txt_var</i></code>
Arguments	<p><i>src_code_string_lit</i> Specifies a string of characters to be encoded and placed in <i>dst_txt_var</i>.</p> <p><i>dst_txt_var</i> Contains the result after execution.</p>
Description	<p>The ENCODE command can define nondisplay characters or escape sequences sent to an output device. These characters or sequences can perform complex output device manipulations. The ENCODE command 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.</p> <p>The encode characters can be included in a report at the appropriate location using a PRINT or PRINT-DIRECT command.</p> <p>Only values <001> to <255> can be defined in the ENCODE command.</p>
Examples	<pre>encode '<27>L11233' into \$bold! Code sequence to turn bold on. print \$bold () code-printer=lp</pre>
See Also	<ul style="list-style-type: none">■ The chr function described in Table 2-36 on page 2-220 under the LET command.■ The PRINT and PRINT-DIRECT commands.■ The section titled “Encode Variables” on page 2-16 in <i>Designing Reports with Hyperion SQR Developer</i>.

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 The END-DECLARE command 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

- The DECLARE-paragraph command.
- The BEGIN-*section* command.

#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
#endif
```

See Also The #IF, #IFDEF, and #IFNDEF commands for a description of each compiler directive.

END-IF

Function Ends an IF command.

Syntax END-IF

See Also The IF command for a description and example.

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 <code>END-section</code> command completes the corresponding <code>BEGIN-section</code> command:
	<ul style="list-style-type: none">■ <code>BEGIN-PROCEDURE</code>■ <code>BEGIN-PROGRAM</code>■ <code>BEGIN-SELECT</code>■ <code>BEGIN-SETUP</code>■ <code>BEGIN-SQL</code>■ <code>WHILE</code>■ <code>BEGIN-EXECUTE</code>
	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'. See the tip on page 2-38 for an example.
Examples	<pre>begin-program do main end-program</pre>
See Also	<ul style="list-style-type: none">■ The <code>BEGIN-section</code> command.■ The <code>WHILE</code> command.

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 an EVALUATE command is the following:

```
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`
Specifies a text or numeric column; a text, numeric, or date variable; or a text or numeric literal to be used in the evaluation. In short, an evaluation argument.

`comparison_operator`

Any valid comparison operator. See comparison operators in Table 2-29 on page 2-195.

`WHEN`

Specifies the evaluation expression. The evaluation argument is compared with the argument, beginning from the first WHEN. If the expression is TRUE, SQR processes the commands after the WHEN. If the expression is FALSE, SQR 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`

Causes an immediate exit of the EVALUATE command. Use BREAK at the end of a set of commands.

WHEN-OTHER

Signifies the start of default commands to be processed if all other WHEN expressions are FALSE. WHEN-OTHER must appear after all other WHEN expressions.

Description

The EVALUATE command 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 SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats (see Table 2-45, “Default Formats by Database,” on page 2-270), or the database-independent format 'SYYYYMMDD [HH24 [MI [SS [NNNNNN]]]].'
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

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

The IF and LET commands for comparison operators.

EXECUTE

Function	Executes a stored procedure. The EXECUTE command is available with the DB2, ODBC, Oracle, and Sybase versions of SQR. (For DB2, SQR does not support overloaded stored procedures.)
Syntax	<pre>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_col data_type[(length_int_lit)] , . . .] [WITH RECOMPILE]</pre>
	The syntax of this command roughly follows that of the Sybase Transact-SQL EXECUTE command, with the exception of optional arguments and the INTO argument.
Arguments	<p>-XC (Sybase only) Specifies that the EXECUTE command shares the same connection as the DO= procedure it can invoke. This argument is required to share Sybase temporary tables.</p> <p>ON-ERROR Declares an SQR procedure to execute if an error occurs. If ON-ERROR is omitted and an error occurs, SQR halts with an error message. For severe errors (for example, passing too few arguments) SQR halts, even if an error procedure is specified.</p> <p>You can specify arguments to be passed to the ON-ERROR procedure. Arguments can be any variable, column, or literal.</p> <p>DO Specifies an SQR procedure to execute for each row selected in the query. Processing continues until all rows are retrieved.</p> <p>You can specify arguments to pass to the procedure. Arguments can be any variable, column, or literal.</p>

@#status_var

Returns the procedure's status in the specified *numeric* variable. The status is returned only after selected rows are retrieved.

@\$return_var

(Oracle only) Returns 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

Names the stored procedure or function to execute.

@param

Names the parameter to pass 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

Specifies the 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 SQR 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

Indicates 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 2-25 lists the valid data types for each database.

Table 2-25 Valid Data Types

Database	Valid Data Types
Oracle	CHAR[(n)] DATE DECIMAL[(p[,s])] FLOAT[(b)] INTEGER LONG NCHAR[(n)] NVARCHAR2[(n)] NUMBER[(p[,s])] NUMERIC[(p[,s])] REAL ROWID SMALLINT VARCHAR[(n)] VARCHAR2[(n)]
ODBC	BIT TINYINT SMALLINT INT CHAR[(n)] NCHAR[(n)] VARCHAR[(n)] NVARCHAR[(n)] TEXT REAL FLOAT[(b)] IMAGE SMALLMONEY MONEY DECIMAL[(p[,s])] NUMERIC [(p[,s])] SYSNAME SMALLDATETIME DATETIME TIMESTAMP BINARY VARBINARY

Table 2-25 Valid Data Types (*Continued*)

Database	Valid Data Types
Sybase	BIT TINYINT SMALLINT INT CHAR[(n)] NCHAR[(n)] VARCHAR[(n)] NVARCHAR[(n)] TEXT REAL FLOAT[(b)] IMAGE SMALLMONEY MONEY DECIMAL[(p[,s])] NUMERIC [(p[,s])] SYSNAME SMALLDATETIME DATETIME TIMESTAMP BINARY VARBINARY UNICHAR[(n)] UNIVARCHAR[(n)]
DB2	CHAR[(n)] VARCHAR[(n)] DATE TIME TIMESTAMP FLOAT DOUBLE NUMERIC DECIMAL[(p[,s])] INTEGER GRAPHIC[(n)] VARGRAPHIC[(n)]

If the stored procedure contains more than one result set, only the first query is described with the `INTO` argument. Rows from subsequent queries are ignored.

WITH RECOMPILE

(Sybase and ODBC only) Causes the query to recompile each time it executes rather than using the plan stored with the procedure. Normally, this is not required or recommended.

Description

If the stored procedure specified in `stored_procedure_name` contains a `SELECT` query, the `EXECUTE` command 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, SQR only processes the *first* Result Set returned from a stored procedure or function. After processing the first Result Set, all other Result Sets are ignored.
 - When Oracle or DB2 encounters an `INTO` clause, an *implied* Result-Set handle is created. The implied Result-Set handle processes an open cursor returned from a stored procedure or function. The procedure or function is “described” to ensure that the stored object returns a handle to a result set. The data returned from the “describe” is then used to validate the data types declared for each column contained in the `INTO` clause of the `EXECUTE` command.
-

Examples

The following example invokes the stored procedure `get_total` with two parameters: a string literal and a string variable. The result from the stored procedure is stored in the variable `#total`.

```
execute get_total 'S. Q. Reporter' $State #Total Output
```

The following example invokes the stored procedure `get_products` with two parameters. The stored procedure selects data into five column variables. The SQR procedure `print_products` is called for each row retrieved. The return status from the stored procedure is placed in the variable `#proc_return_status`.

```
execute do=print_products
  @#proc_return_status=
  get_products
  @prodcode=&code, @max=#maximum
  INTO &prod_code int,
       &description char(45),
       &discount float,
       &restock char,
       &expire_date datetime
begin-procedure print_products
print &prod_code(+1,1)
print &description(+5,45)
print &discount(+5) edit 99.99
print &restock(+5) match Y 0 5 Yes N 0 5 No
print &expire_date(+5,) edit 'Month dd, yyyy'
end-procedure
```

EXIT-SELECT

Function	Exits a SELECT paragraph immediately.
Syntax	EXIT-SELECT
Description	EXIT-SELECT causes SQR to jump to the command immediately following the END-SELECT command.
	Use EXIT-SELECT when you need to end a query before all rows have been retrieved.
Examples	<pre>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</pre>
See Also	The BEGIN-SELECT command.

EXTRACT

Function	Copies a portion of a string into a string variable.
Syntax	<pre>EXTRACT {dst_txt_var date_var} FROM {{src_txt_lit _var _col} {src_date_var _col}} {start_num_lit _var}{length_num_lit _var}</pre>
Arguments	<p><i>dst_txt_var date_var</i> Specifies a text or date variable into which the extracted string is placed.</p> <p><i>{src_txt_lit _var _col} {src_date_var _col}</i> Specifies a text or date variable, column, or literal from which the string is to be extracted.</p> <p><i>start_num_lit _var</i> Specifies starting location of the string to be extracted.</p> <p><i>length_num_lit _var</i> Specifies length of the string to be extracted.</p>
Description	<p>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.</p> <p>If the source is a date variable or column, it is converted to a string before the extraction according to the following rules:</p> <ul style="list-style-type: none">■ For DATETIME columns and SQR DATE variables, SQR uses the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.■ For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.■ For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

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 the SQR_DB_DATE_FORMAT setting
- One of the database-dependent formats (see Table 2-45, “Default Formats by Database,” on page 2-270)
- The database-independent format
'SYYYYYMMDD [HH24 [MI [SS [NNNNNN]]]].'

Examples

```
extract $state from $record 45 2
extract $foo from "Hyperion Rocks" 0 4 ! $foo='Hyperion'

code from &phone 0 3
extract $zip_four from &zip 5 4
extract $rec from $tape_block #loc #rec_len
```

See Also

- The substr function described in Table 2-36 on page 2-220 under the LET command.
- The FIND command.

FIND

Function	Determines the location of a character sequence within a string.
Syntax	<pre>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</pre>
Arguments	<p><i>{obj_txt_lit _var _col} {date_var _col}</i></p> <p>Specifies a text variable, column, or literal that is to be sought in <i>src_txt_var _col</i>.</p> <p><i>{src_txt_var _col} {date_var _col}</i></p> <p>Specifies a text variable or column to be searched.</p> <p><i>start_int_lit _var</i></p> <p>Specifies starting location of the search.</p> <p><i>dst_location_int_var</i></p> <p>Specifies the returned starting location of the left-most character of the matching text in <i>{src_txt_var _col} {date_var _col}</i>.</p>
Description	<p>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 <i>dst_location_int_var</i>.</p> <p>You must specify an offset from which to begin the search and supply a numeric variable for the return of the location.</p>

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 SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

Examples

```
find 'aw.2' in &code5 0 #loc
find ',' in &name 0 #comma_loc
if #comma_loc = -1
...comma not found...
```

See Also

- The instr function described in Table 2-36 on page 2-220 under the LET command.
- The EXTRACT command.

GET

Function Retrieves data from an array and places it into a date, string, or numeric variable.

Syntax

```
GET dst_any_var...FROM src_array_name(element)  
[field[(occurs)]]...
```

Arguments

dst_any_var
A date, string, or numeric variable (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, SQR converts the date to a string in the format specified by the `SQR_DB_DATE_FORMAT` setting. If this has not been set, SQR uses the first database-dependent format listed in Table 2-45 on page 2-270.

If the destination is a date variable, the string extracted from the source must be in the format specified by the `SQR_DB_DATE_FORMAT` setting, or one of the database-dependent formats (seeTable 2-45 on page 2-270), or the database-independent format '`'YYYYMMDD [HH24 [MI [SS [NNNNNN]]]]'`'.

src_array_name(element)

If the array's field names are listed, SQR 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

- The LET command for information on assigning the value of an expression.
- The PUT command for information on moving data into an array.

GET-COLOR

Function	Retrieves the current colors.
Syntax	<pre>GET-COLOR [PRINT-TEXT-FOREGROUND=({color_name_var} {rgb})] [PRINT-TEXT-BACKGROUND=({color_name_var} {rgb})] [PRINT-PAGE-BACKGROUND=({color_name_lit _var _col} {rgb})]</pre>
Arguments	<p>PRINT-TEXT-FOREGROUND Defines the color in which the text prints.</p> <p>PRINT-TEXT-BACKGROUND Defines the color used for the background behind the text.</p> <p>PRINT-PAGE-BACKGROUND Defines the color used for the page background.</p> <p>{color_name_var} A <i>color_name</i> is composed of the alphanumeric characters (A-Z, 0-9), the underscore (_) character, and the dash (-) character. The name must start with an alpha (A-Z) character and 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.</p> <p>{rgb} <i>red_lit _var _col, green_lit _var _col, blue_lit _var _col</i> where each component is a value in the range of 000 to 255. In the BEGIN-SETUP section, only literal values are allowed.</p>
	<p>The default colors implicitly installed with SQR include:</p> <pre>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)</pre>

```
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

The GET-COLOR command is allowed wherever the PRINT command is allowed. If the requested color settings does not map to a defined 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 specified area.

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

The DECLARE-COLOR-MAP, ALTER-COLOR-MAP, and SET-COLOR commands.

GOTO

Function	Skips to the specified label.
Syntax	<code>GOTO label</code>
Arguments	<code>label</code> Specifies a label within the same section or paragraph.
Description	Labels must end with a colon (:) and can appear anywhere within the same section or paragraph as the GOTO command.
Examples	<pre>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</pre>

GRAPHIC BOX

Function	Draws a box.
Syntax	<pre>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]]</pre>
Arguments	<p><i>width</i> and <i>depth</i> The <i>width</i> is the horizontal size in character columns; <i>depth</i> 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 <i>width</i> and <i>depth</i>. You can specify relative placement with (+), (-), or numeric variables, as with regular print positions.</p> <p><i>rule_width</i> 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.</p> <p><i>shading</i> 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 <i>rule-width</i> of zero, if a border is not desired.</p>
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, SQR changes the current print location to the starting location of the graphic. (This is different than the way the PRINT command works.)
Examples	<pre>graphic (1,1,66) box 58 20! Draw box around page graphic (30,25,10) box 10! Draw a 10-characters-wide-by-10- characters-long box graphic (1,1,66) box 5 0 8! Draw 5 line shaded box (without ! border) graphic (50,8,30) box 1! Draw box around 1 line</pre>
See Also	The ALTER-PRINTER and DECLARE-PRINTER commands for information on setting and changing the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

GRAPHIC HORZ-LINE

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	<i>rule_width</i> The default rule width is 2 decipoins.
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, SQR 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
See Also	The ALTER-PRINTER and DECLARE-PRINTER commands for information on setting and changing the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

GRAPHIC VERT-LINE

Function	Draws a vertical line.
Syntax	GRAPHIC ({line_int_lit _var}, {column_int_lit _var}, {length_int_lit _var}) VERT-LINE [<i>rule_width_int_lit _var</i>]
Arguments	<i>rule_width</i> 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, SQR 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
See Also	The ALTER-PRINTER and DECLARE-PRINTER commands for information on setting and changing the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

#IF

Function	Indicates that the commands following are to be compiled when the expression is TRUE. (#IF is a compiler directive.)
Syntax	<pre>#IF {txt_lit num_lit} comparison_operator {txt_lit num_lit}</pre>
Arguments	<p><i>txt_lit num_lit</i> Any text or numeric literal.</p> <p><i>comparison_operator</i> Any of the comparison operators as follows:</p> <ul style="list-style-type: none">= Equal!= Not Equal<> Not Equal< Less than> Greater than<= Less than or equal>= Greater than or equal
Description	<p>SQR has five compiler directives that allow different pieces of SQR code to be compiled, depending on the existence or value of substitution variables (not program variables, such as, string, numeric, or date).</p> <p>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 SQR program on or off from the command line, depending on the -DEBUGXXX flag.</p> <p>You can nest #IF, #IFDEF, or #IFNDEF directives to a maximum of 10 levels.</p> <p>The #IF, #IFDEF, or #IFNDEF directives cannot be broken across program lines.</p>

Table 2-26 lists the compiler directives.

Table 2-26 SQR Compiler Directives

Directive	Example	Description
#IF	#IF {option}='A'	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.
#ELSE	#ELSE	Compiles the commands following the #ELSE directive when the #IF expression is FALSE.
#ENDIF	#ENDIF	Ends the #IF directive. #ENDIF can also be typed #END-IF (with a hyphen).
#IFDEF	#IFDEF option	Compiles the commands following the #IFDEF directive if the substitution variable option is defined.
#IFNDEF	#IFNDEF option	Compiles the command following the #IFNDEF directive if the substitution variable option is not defined.

Examples

```
begin-setup
  ask type 'Use Male, Female or Both (M,F,B)'
end-setup
begin-procedure Main
#if {type} = 'M'
  ...code for M here
#else
#if {type} = 'F'
  ...code for F here
#else
#if {type} = 'B'
  ...code for B here
#else
  show 'M, F or B not selected. Report not created.'
  stop
#endif! for B
#endif! for F
#endif! for M

#endifdef debug
  show 'DEBUG: Cust_num = ' &cust_num edit 099999
#endiff

#ifndefdef debugB ! DebugB turned on with -DEBUGB on
  do test_procedure! SQR command line.
#endiff
```

See Also

The #DEBUG command for more information on the -DEBUG command-line flag.

IF

Function	Executes commands depending on the value of a condition.
Syntax	<pre>IF <i>logical_expression</i></pre> <p>IF commands have the following structure:</p> <pre>IF <i>logical_expression</i> <i>sqr_commands...</i> [ELSE <i>sqr_commands...</i>] END-IF</pre>
Arguments	<i>logical_expression</i> Any valid logical expression. See the LET command for a description of logical expressions.
Operators	See “Bit-Wise Operators” on page 2-196 for information on the bit-wise operators supported by the IF command.
Description	The expression is evaluated as a logical TRUE or FALSE. A value or expression that evaluates to nonzero is TRUE. Each IF command 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 SQR DATE variables, SQR uses the format specified by the `SQR_DB_DATE_FORMAT` setting, one of the database-dependent formats (see Table 2-45, “Default Formats by Database,” on page 2-270), or the database-independent format
`'SYYYYYMMDD [HH24 [MI [SS [NNNNNN]]]]'`.
- For DATE columns, SQR uses the format specified by the `SQR_DB_DATE_ONLY_FORMAT` setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the `SQR_DB_TIME_ONLY_FORMAT` setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

Examples

```
if &price > &old_price and instr(&code, 'M', 1) > 0
    add 1 to #price_count
    if #price_count > 50
        show 'More than 50 prices found.' noline
        input $x 'Continue? (Y/N)'
        if upper($x) = 'N'
            stop
        end-if
    end-if
else
    add 1 to #old_price_count
end-if
if #rows! Will be TRUE if #rows is non-zero.
    do print-it
end-if

if $date1 > 'Apr 21 1996 23:59'
    do past_due
end-if
```

See Also

- The LET command for a description of logical expressions.
- The EVALUATE command.

#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 SQR command line. (#IFDEF is a compiler directive.)
Syntax	<code>#IFDEF <i>substitution_variable</i></code>
Arguments	<i>substitution_variable</i> Is the variable to be used as the substitution variable.
See Also	The #IF command 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 SQR command line. (#IFNDEF is a compiler directive.)
Syntax	<code>#IFNDEF <i>substitution_variable</i></code>
Arguments	<i>substitution_variable</i> Is the variable to be used as the substitution variable.
See Also	The #IF command for a description of each compiler directive.

#INCLUDE

Function	Includes an external source file into the SQR report specification.
Syntax	<code>#INCLUDE filename_lit</code>
Arguments	<i>filename_lit</i> A filename that is valid for the platform on which this application is to be 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	<pre>#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.</pre>

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*
Specifies a text, numeric, or date variable for the input data.

MAXLEN
Specifies the maximum length for the data.

prompt
Specifies the prompt (literal not variable) displayed to the user.

TYPE
Specifies the datatype required for the input.

STATUS
Specifies a numeric variable for a return status code.

NOPROMPT
Prevents the prompt from being displayed before the INPUT command 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
Specifies the format for entering a date. Table 2-41, “Date Edit Format Characters,” on page 2-262 lists date edit format codes.

Description

Use MAXLEN to prevent the user from entering data that is too long. If an INSERT or UPDATE command references a variable whose length is greater than that defined in the database, the SQL is rejected and SQR 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, SQR 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. The INPUT command 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. The types CHAR, TEXT, and DATE are invalid types. The datatypes supported are described in Table 2-27.

Table 2-27 Datatypes Supported by the INPUT Command

Datatype	Description
CHAR, TEXT	Any character. This is the default datatype.
NUMBER	A floating point number in the format [+ -]9999.999[E[+ -]99]
INTEGER	An integer in the format [+ -]99999
DATE	A date in one of the following formats: MM/DD/YYYY [BC AD] [HH:MI[:SS[.NNNNNN]]] [AM PM]] MM-DD-YYYY [BC AD] [HH:MI[:SS[.NNNNNN]]] [AM PM]] MM.DD.YYYY [BC AD] [HH:MI[:SS[.NNNNNN]]] [AM PM]] YYYYMMDD[HH24[MI[SS[NNNNNN]]]]

Specifying STATUS causes the INPUT command 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 2-28 lists the values of the INPUT command's status argument

Table 2-28 Values of the STATUS Argument of the INPUT Command

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

By using NOPROMPT and STATUS with the SHOW command, 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 setting for the current locale. If FORMAT has not been set, use a database-independent format for the data as listed in Table 2-27 on page 2-189.

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 the BATCH-MODE option:

```
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

- The ALTER-LOCALE command.
- The INPUT-DATE-EDIT-MASK setting in Chapter 5, “SQR.INI.”

LAST-PAGE

Function Places the last page number on each page, as in “page *n* of *m*”.

Syntax LAST-PAGE *position* [*pre_txt_lit*[*post_txt_lit*]]

Arguments *position*
Specifies the position for printing the last page number.

pre_txt_lit
Specifies a text string to be printed before the last page number.

post_txt_lit
Specifies a text string to be printed after the last page number.

Description The text strings specified in *pre_txt_lit* and *post_txt_lit* are printed immediately before and after the number.

Using LAST-PAGE causes SQR and SQRT to delay printing until the last page has been processed so that the number of the last page is known.

Examples

```
begin-footing 1
    page-number(1,37)  'Page ' Will appear as
    last-page ()  ' of ' '..' ! "Page 12 of 25."
end-footing
```

See Also The PAGE-NUMBER, BEGIN-HEADING, and BEGIN-FOOTING commands.

LET

Function Assigns the value of an expression to a string, numeric, date, or list (DDO only) variable.

Syntax `LET dst_var=expression`

Arguments `dst_var`
A string, numeric, date, or list (DDO only) variable or array field to which the result of the expression is assigned.
`expression`
The 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. SQR supports a standardized set of mathematical operators and logical comparison operators working within a carefully defined set of precedence rules. SQR 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 SQR expression provides the user with a very powerful tool that can be tailored to suit any information processing need. (Note that all string indices are one-based, not zero-based.)

Examples The following examples show some complex expressions:

```
let #j = ((#a + #b) * #c) ^ 2
if #j > 2 and sqrt(#j) < 20 or #i + 2 > 17.4
while upper(substr(&descrip,1,#j+2)) != 'XXXX'
and not isnull(&price)
let #len = length(&fname || &initial || &lname) + 2
let $s = edit(&price * &rate, '99999.99')
let summary.total(#j) = summary.total(#j) + (&price * &rate)
if summary.total(#j) > 1000000
let store.total (#store_id, #dept)
    = store.total (#store_id, #dept) + #total
let #diff = datediff(datenow(), strtodate('1995', 'YYYY'), 'day')
```

```
let $newdate = dateadd(datenow(), 'month', 50)
let $date1 = datetostr(strtodate(&sale_date), 'Day Month DD,
YYYY')
```

SQR 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 SQR expression. Operands do not have to be the same type. You can combine string, numeric and array field operands to form a valid expression. SQR performs a sequence of automatic operand conversions as it evaluates expressions that contain dissimilar operand types. As the expression is evaluated, operands of lower precision are converted to match the operand of higher precision. Consider the following example:

```
let #answer = #float * #decimal / #integer
```

Since the *multiply* and *divide* operators are equal in precedence, the expression is evaluated as $(\#float * \#decimal) / \#integer$. Working from the inside out, the `#float` variable is converted to a decimal type where a multiply is performed yielding the simplified expression, $(\#decimal) / \#integer$. SQR now converts the `#integer` operand to a decimal type before performing the final divide. When finished with the expression evaluation, SQR 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. SQR considers integer variables as the lowest in the precision hierarchy, followed by float and then decimal.

Here are a few simple expression examples:

```
let #discount = round (&price * #rate / 100, 2)
let $name = $first_name || ' ' || $last_name
let customer.total (#customer_id) =
    customer.total (#customer_id) + #invoice_total
if not range(upper($code), 'A', 'G')
    ...processing when out of range...
let store.total (#store_id, #qtr) =
    store.total (#store_id, #qtr) + #invoice_total
let $date1 = strtodate ('Apr 10 1996', 'MON DD YYYY')
```

The following sections list operators and functions supported in expressions.

Operators

Table 2-29 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 2-29 Operators

Operator	Explanation
	Concatenate two strings or dates
+, -	Sign prefix (positive or negative)
^	Exponent
*, /, %	Multiply, divide, remainder: a % b = mod(a,b) for integers
+, -	Plus, minus
Note: SQR distinguishes between a sign prefix and arithmetic operation by the context of the expression.	
>, <, >=, <=, <>, !=, =	Comparison operators: greater than, less than, greater or equal to, less than or equal to, not equal (!= or <>), equal

Table 2-29 Operators (Continued)

Operator	Explanation
not	Logical NOT
and	Logical AND
or, xor	Logical OR, XOR (exclusive OR)

Bit-Wise Operators

Bit-Wise operators allow you to utilize bitmasks within your program.

Bit-Wise operators act just like their logical counterparts (AND, OR, XOR) except that instead of returning a TRUE or FALSE value, they return the actual result.

To facilitate the use of these operators, the LET command recognizes the hexadecimal notation of OX? for expressing a numerical constant. The ? character is from 1 to 8 hexadecimal characters (0-F).

Table 2-30 Bit-Wise Operators

Operator	Explanation
BitAND	Acts just like Logical AND except returns the actual result instead of TRUE or FALSE
BitOR	Acts just like Logical OR except returns the actual result instead of TRUE or FALSE
BitXOR	Acts just like Logical XOR except returns the actual result instead of TRUE or FALSE



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 OX?, which is the hexadecimal representation of the argument.

To prevent any loss of precision, declare the arguments to the Bit-Wise operators and the HEX function as an INTEGER.

Sample program:

```
!
! Validation test for Bit-Wise LET operators
!
Begin-Setup
    Declare-Variable
        Integer #Mask
    End-Declare
End-Setup

Begin-Program
    Let #Mask = 0x1000
    If Not (#Mask BitAND 0x1000)
        Let $Mask = Hex(#Mask)
        Show 'Impossible ' $Mask ' BitAND 0x1000 failed'
    End-If
    If (#Mask BitOR 0x1000) <> 0x1000
        Let $Mask = Hex(#Mask)
        Show 'Impossible ' $Mask ' BitOR 0x1000 failed'
    End-If
    If (#Mask BitXOR 0x1000)
        Let $Mask = Hex(#Mask)
        Show 'Impossible ' $Mask ' BitXOR 0x1000 failed'
    End-If
    Let #Mask = 0
    If (#Mask BitXOR 0x1000) <> 0x1000
        Let $Mask = Hex(#Mask)
        Show 'Impossible ' $Mask ' BitXOR 0x1000 failed'
    End-If
    Let $Mask = Hex(0xffffffff)
    If $Mask <> '0xffffffff'
        Show 'Impossible ' $Mask ' <> 0xffffffff'
    End-If
    Let $Mask = Hex(0x12345678)
    If $Mask <> '0x12345678'
        Show 'Impossible ' $Mask ' <> 0x12345678'
    End-If
    Let $Mask = Hex(0xabcddef12)
    If $Mask <> '0xabcddef12'
        Show 'Impossible ' $Mask ' <> 0xabcddef12'
    End-If
End-Program
```

Functions

SQR 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 2-31 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, SQR converts the date to a string according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

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 the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats listed in Table 2-45, “Default Formats by Database,” on page 2-270, or the database-independent format 'SYYYYMMDD [HH24 [MI [SS [NNNNNNN]]]]'

Numeric Functions

Table 2-31 lists the numeric functions.

Table 2-31 Numeric Functions

Function	Description
abs	Returns the absolute value of <i>num_value</i> . This function returns a value of the same type as <i>num_value</i> . Syntax: <i>dst_var</i> = abs(<i>num_value</i>) ■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. ■ <i>dst_var</i> = decimal, float, or integer variable. Example: let#dabsvar = abs(#dvar)
acos	Returns the arccosine of <i>num_value</i> in the range of 0 to π radians. The value of <i>num_value</i> must be between -1 and 1. This function returns a float value. Syntax: <i>dst_var</i> = acos(<i>num_value</i>) ■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #facosvar = acos(#fvar)
asin	Returns the arcsine of <i>num_value</i> in the range of $-\pi/2$ to $\pi/2$ radians. The value of <i>num_value</i> must be between -1 and 1. This function returns a float value. Syntax: <i>dst_var</i> = asin(<i>num_value</i>) ■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fasinvar = asin(#fvar)

Table 2-31 Numeric Functions (*Continued*)

Function	Description
atan	Returns the arctangent of <i>num_value</i> in the range of $-\pi/2$ to $\pi/2$ radians. The value of <i>num_value</i> must be between -1 and 1. This function returns a float value. Syntax: <i>dst_var</i> = atan(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fatanvar = atan(#fvar)
ceil	Returns a value representing the smallest integer that is greater than or equal to <i>num_value</i> . This function returns a value of the same type as <i>num_value</i> . Syntax: <i>dst_var</i> = ceil(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fceilvar = ceil(#fvar)
cos	Returns the cosine of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = cos(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fcosvar = cos(#fvar)
cosh	Returns the hyperbolic cosine of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = cosh(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fcoshvar = cosh(#fvar)

Table 2-31 Numeric Functions (Continued)

Function	Description
deg	Returns a value expressed in degrees of <i>num_value</i> which is expressed in radians. This function returns a float value. Syntax: <i>dst_var</i> = deg(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fdegvar = deg(#fvar)
e10	Returns the value of 10 raised to <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = e10(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fe10var = e10(#fvar)
exp	Returns the value of e raised to <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = exp(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fexpvar = exp(#fvar)
floor	Returns a value representing the largest integer that is less than or equal to <i>num_value</i> . This function returns a value of the same type as <i>num_value</i> . Syntax: <i>dst_var</i> = floor(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #ffloorvar = floor(#fvar)

Table 2-31 Numeric Functions (*Continued*)

Function	Description
hex	Returns a string, in the form of 0x?, which is the hexadecimal representation of the argument Syntax: <i>dst_var</i> = hex(<i>num_value</i>) ■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to an integer. ■ <i>dst_var</i> = string variable. Example: let \$hexvar = hex(#fvar)
log	Returns the natural logarithm of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = log(<i>num_value</i>) ■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #flogvar = log(#fvar)
log10	Returns the base-10 logarithm of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = log10(<i>num_value</i>) ■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #flog10var = log10(#fvar)
mod	Returns the fractional remainder, <i>f</i> , of <i>x_value</i> / <i>y_value</i> such that <i>x_value</i> = <i>i</i> * <i>y_value</i> + <i>f</i> , where <i>i</i> is an integer, <i>f</i> has the same sign as <i>x_value</i> , and the absolute value of <i>f</i> is less than the absolute value of <i>y_value</i> . The arguments are promoted to the type of the greatest precision and the function returns a value of that type. Syntax: <i>dst_var</i> = mod(<i>x_value</i> , <i>y_value</i>) ■ <i>x_value</i> = decimal, float, or integer literal, column, variable, or expression. ■ <i>y_value</i> = decimal, float, or integer literal, column, variable, or expression. ■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fmodvar = mod(#fxvar, #fyvar)

Table 2-31 Numeric Functions (*Continued*)

Function	Description
power	Returns the value of <i>x_value</i> raised to the power of <i>y_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = power(<i>x_value</i> , <i>y_value</i>) ■ <i>x_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>y_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fpowervar = power(#fxvar, #fyvar)
rad	Returns a value expressed in radians of <i>num_value</i> which is expressed in degrees. This function returns a float value. Syntax: <i>dst_var</i> = rad(<i>num_value</i>) ■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>place_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fradvar = rad(#fvar)
round	Returns a value that is <i>num_value</i> rounded to <i>place_value</i> digits after the decimal separator. This function returns a value of the same type as <i>num_value</i> . Syntax: <i>dst_var</i> = round(<i>num_value</i> , <i>place_value</i>) ■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. ■ <i>place_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #frndvar = round(#fvar, #fplace)

Table 2-31 Numeric Functions (*Continued*)

Function	Description
sign	Returns a -1, 0, or +1 depending on the sign of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = sign(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fsignvar = sign(#fvar)
sin	Returns the sine of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = sin(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fsinvar = sin(#fvar)
sinh	Returns the hyperbolic sine of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = sinh(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fsinhvar = sinh(#fvar)
sqrt	Returns the square root of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = sqrt(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #fsqrtvar = sqrt(#fvar)

Table 2-31 Numeric Functions (Continued)

Function	Description
tan	Returns the tangent of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = tan(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #ftanvar = tan(#fvar)
tanh	Returns the hyperbolic tangent of <i>num_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = tanh(<i>num_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. Example: let #ftanhvar = tanh(#fvar)
trunc	Returns a value that is <i>num_value</i> truncated to <i>place_value</i> digits after the decimal separator. This function returns a value of the same type as <i>num_value</i> . Syntax: <i>dst_var</i> = trunc(<i>num_value</i> , <i>place_value</i>) <ul style="list-style-type: none">■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.■ <i>place_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>dst_var</i> = decimal, float, or integer variable. 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, SQR 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 2-36 to prevent division by zero or other invalid function or operator argument values.

File-Related Functions

Table 2-32 lists the file-related functions. These functions return zero (0) if successful; otherwise, they return the system error code.

Table 2-32 File-Related Functions

Function	Description
<code>delete</code>	<p>Deletes the file <i>filename</i>. The function returns either a zero (0) to indicate success or the value returned from the operating system to indicate an error.</p> <p>Syntax: <code>stat_var = delete(filename)</code></p> <ul style="list-style-type: none">■ <i>filename</i> = text literal, column, variable, or expression.■ <i>stat_var</i> = decimal, float, or integer variable. <p>Example: <code>let #fstatus = delete(\$filename)</code></p>
<code>exists</code>	<p>Determines if the file, <i>filename</i>, exists. The function returns either a zero (0) to indicate success or the value returned from the operating system to indicate an error.</p> <p>Syntax: <code>stat_var = exists(filename)</code></p> <ul style="list-style-type: none">■ <i>filename</i> = text literal, column, variable, or expression.■ <i>stat_var</i> = decimal, float, or integer variable. <p>Example: <code>let #fstatus = exists(\$filename)</code></p>
<code>rename</code>	<p>Renames <i>old_filename</i> to <i>new_filename</i>. The function returns either a zero (0) to indicate success or the value returned from the operating system to indicate an error.</p> <p>Syntax: <code>stat_var = rename(old_filename, new_filename)</code></p> <ul style="list-style-type: none">■ <i>old_filename</i> = text literal, column, variable, or expression.■ <i>new_filename</i> = text literal, column, variable, or expression.■ <i>stat_var</i> = decimal, float, or integer variable. <p>Example: <code>let #fstatus = rename(\$old_filename, \$new_filename)</code></p>

String Functions

Table 2-33 lists the string functions.

Table 2-33 String Functions

Function	Description
ascii	Returns the ASCII value for the first character in <i>str_value</i> . This function returns a float value. Syntax: <i>ascii_var</i> = ascii(<i>str_value</i>) ■ <i>str_value</i> = date or text literal, column, variable, or expression ■ <i>ascii_var</i> = 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: <i>ascii_var</i> = asciic(<i>str_value</i>) ■ <i>str_value</i> = date or text literal, column, variable, or expression ■ <i>ascii_var</i> = decimal, float, or integer variable Example: let #fascii = asciic(\$filename)
chr	Returns a string that is composed of a character with the ASCII value of <i>num_value</i> . Syntax: <i>dst_var</i> = chr(<i>num_value</i>) ■ <i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float. ■ <i>dst_var</i> = text variable Example: let \$svar = chr(#num)
edit	Formats <i>source_value</i> according to <i>edit_mask</i> and returns a string containing the result. Syntax: <i>dst_var</i> = edit(<i>source_value</i> , <i>edit_mask</i>) ■ <i>source_value</i> = Any literal, column, variable, or expression ■ <i>edit_mask</i> = text literal, column, variable, or expression ■ <i>dst_var</i> = text variable Example: let \$phone = edit(&phone, '(xxx) xxx-xxxx') let \$price = edit(#price, '999.99') let \$today = edit(\$date, 'DD/MM/YYYY')

Table 2-33 String Functions (*Continued*)

Function	Description
instr	Returns the numeric position of <i>sub_value</i> in <i>source_value</i> or zero (0) if not found. The search begins at offset <i>offset_value</i> . This function returns a float value. Syntax: <i>dst_var</i> = instr(<i>source_value</i> , <i>sub_value</i> , <i>offset_value</i>) ■ <i>source_value</i> = date or text literal, column, variable, or expression ■ <i>sub_value</i> = text literal, column, variable, or expression ■ <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. ■ <i>dst_var</i> = decimal, float, or integer variable Example: let #offset = instr(&description, 'auto', 10)
instrb	Performs the same functionality as the instr function except that the starting point and returned value are expressed in bytes rather than in characters. Syntax: <i>dst_var</i> = instrb(<i>source_value</i> , <i>sub_value</i> , <i>offset_value</i>) ■ <i>source_value</i> = date or text literal, column, variable, or expression ■ <i>sub_value</i> = text literal, column, variable, or expression ■ <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. ■ <i>dst_var</i> = decimal, float, or integer variable Example: let #offset = instrb(&description, 'auto', 10) Note: instrb does <i>not</i> allow you to specify the target encoding. If you are using Unicode internally, you can specify the target encoding with lengthp, lenghtt, substrp, substrt, or transform.
isblank	Returns a value of one (1) if <i>source_val</i> is an empty string, null string, or composed entirely of whitespace characters; otherwise, returns a value of zero (0). Syntax: <i>dst_var</i> = isblank(<i>source_value</i>) ■ <i>source_value</i> = date or text literal, column, variable, or expression (character data type columns only, no numeric data type columns) ■ <i>dst_var</i> = decimal, float, or integer variable Example: let #blank = isblank(&description) Note: isblank can only be used for character data type columns.

Table 2-33 String Functions (*Continued*)

Function	Description
length	<p>Returns the number of characters in <i>source_value</i>.</p> <p>Syntax: <i>dst_var</i> = length(<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression■ <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #length = length(&description)</p>
lengthb	<p>Has the same functionality as the length function except that the return value is expressed in bytes, rather than in characters.</p> <p>Syntax: <i>dst_var</i> = lengthb(<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression■ <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #length = lengthb(&description)</p> <p>Note: lengthb does <i>not</i> allow you to specify the target encoding. If you are using Unicode internally, you can specify the target encoding with lengthp, lengtht, substrp, substrt, or transform.</p>
lower	<p>Converts the contents of <i>source_value</i> to lowercase and returns the result.</p> <p>Syntax: <i>dst_var</i> = lower(<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression■ <i>dst_var</i> = text variable <p>Example: let \$lower = lower(&description)</p>
lpad	<p>Pads the <i>source_value</i> on the left to a length of <i>length_value</i> using <i>pad_value</i> and returns the result.</p> <p>Syntax: <i>dst_var</i> = lpad(<i>source_value</i>, <i>length_value</i>, <i>pad_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression■ <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer■ <i>pad_value</i> = text literal, column, variable, or expression■ <i>dst_var</i> = text variable <p>Example: let \$lpad = lpad(\$notice, 25, ':')</p>

Table 2-33 String Functions (*Continued*)

Function	Description
ltrim	Trims characters in <i>source_value</i> from the left until a character is not in <i>set_value</i> and returns the result. Syntax: <i>dst_var</i> = ltrim(<i>source_value</i> , <i>set_value</i>) ■ <i>source_value</i> = date or text literal, column, variable, or expression ■ <i>set_value</i> = text literal, column, variable, or expression ■ <i>dst_var</i> = text variable Example: let \$ltrim = ltrim(&description, '!')
replace	Inspects the contents of <i>source_value</i> and replaces all occurrences of <i>from_string</i> with <i>to_string</i> and returns the modified string. Syntax: <i>dst_var</i> = replace(<i>source_value</i> , <i>from_string</i> , <i>to_string</i>) ■ <i>source_value</i> = date or text literal, column, variable, or expression ■ <i>from_string</i> = text literal, column, variable, or expression ■ <i>to_string</i> = text literal, column, variable, or expression ■ <i>dst_var</i> = text variable Example: let \$replaced = replace(\$paragraph, 'good', 'excellent')
rpad	Pads the <i>source_value</i> on the right to a length of <i>length_value</i> using <i>pad_value</i> and returns the result. Syntax: <i>dst_var</i> = rpad(<i>source_value</i> , <i>length_value</i> , <i>pad_value</i>) ■ <i>source_value</i> = date or text literal, column, variable, or expression ■ <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. ■ <i>pad_value</i> = text literal, column, variable, or expression ■ <i>dst_var</i> = text variable Example: let \$rpad = rpad(\$notice, 25, '!')
rtrim	Trims characters in <i>source_value</i> from the right until a character is not in <i>set_value</i> and returns the result. Syntax: <i>dst_var</i> = rtrim(<i>source_value</i> , <i>set_value</i>) ■ <i>source_value</i> = date, or text literal, column, variable, or expression ■ <i>set_value</i> = text literal, column, variable, or expression ■ <i>dst_var</i> = text variable Example: let \$rtrim = rtrim(&description, '!')

Table 2-33 String Functions (*Continued*)

Function	Description
substr	<p>Extracts the specified portion <i>source_value</i>. The extraction begins at <i>offset_value</i> (origin is 1) for a length of <i>length_value</i> characters.</p> <p>Syntax: <i>dst_var</i> = substr(<i>source_value</i>, <i>offset_value</i>, <i>length_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression.■ <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.■ <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.■ <i>dst_var</i> = text variable. <p>Example: let \$piece = substr(&record, 10, #len)</p>
substrb	<p>Has the same functionality as the substr function except that the starting point and length are expressed in bytes, rather than in characters.</p> <p>Syntax: <i>dst_var</i> = substrb(<i>source_value</i>, <i>offset_value</i>, <i>length_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression.■ <i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.■ <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.■ <i>dst_var</i> = text variable. <p>Example: let \$piece = substrb(&record, 10, #len)</p> <p>Note: substrb does <i>not</i> allow you to specify the target encoding. If you are using Unicode internally, you can specify the target encoding with lengthp, lenghtb, substrp, substrt, or transform.</p>
to_char	<p>Converts <i>source_value</i> to a string, using maximum precision.</p> <p>Syntax: <i>dst_var</i> = to_char(<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = decimal, float, or integer literal, column, variable, or expression■ <i>dst_var</i> = text variable <p>Example: let \$string = to_char(#number)</p>

Table 2-33 String Functions (*Continued*)

Function	Description
to_multi_byte	<p>Converts the specified string as follows: any occurrence of a single-byte character that also has a multi-byte representation (numerals, punctuation, roman characters, and katakana) is converted.</p> <p>Syntax: <i>dst_var</i> = to_multi_byte (<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression■ <i>dst_var</i> = text variable <p>Example: let \$multi = to_multi_byte (&text)</p>
to_number	<p>Converts <i>source_value</i> to a number. This function returns a float value.</p> <p>Syntax: <i>dst_var</i> = to_number(<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = decimal, float, or integer literal, column, variable, or expression■ <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #value = to_number(\$number)</p>
to_single_byte	<p>Converts the specified string as follows: any occurrence of a multi-byte character that also has a single-byte representation (numerals, punctuation, roman characters, and katakana) is converted.</p> <p>This function also converts a sequence of kana characters followed by certain grammatical marks into a single-byte character that combines the two elements. For all other encodings, the string is not modified.</p> <p>Note: If you are running SQR without the use of Unicode (UseUnicodeInternal=FALSE in the SQR.INI file), this conversion only occurs when the database encoding (ENCODING-DATABASE setting in the INI file) is set to SJIS, EBCDIK290, and EBCDIK1027.</p> <p>Syntax: <i>dst_var</i> = to_single_byte (<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression■ <i>dst_var</i> = text variable <p>Example: let \$single = to_single_byte (&text)</p>

Table 2-33 String Functions (*Continued*)

Function	Description
translate	<p>Inspects the contents of <i>source_value</i> and converts characters that match those in <i>from_set</i> to the corresponding character in <i>to_set</i> and returns the translated string.</p> <p>If <i>to_set</i> does not contain a matching translation character in the corresponding <i>from_set</i>, then the original is left unchanged with regard to that character. If the translation string in <i>to_set</i> is empty, then all characters specified in the <i>from_set</i> string are removed.</p> <p>Syntax: <i>dst_var</i> = translate(<i>source_value</i>, <i>from_set</i>, <i>to_set</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression■ <i>from_set</i> = text literal, column, variable, or expression■ <i>to_set</i> = text literal, column, variable, or expression■ <i>dst_var</i> = text variable <p>Example: let \$translated = translate(edit(&price, '999,999.99'), '.', ',')</p>
upper	<p>Converts the contents of <i>source_value</i> to uppercase and returns the result.</p> <p>Syntax: <i>dst_var</i> = upper(<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression■ <i>dst_var</i> = text variable <p>Example: let \$upper = upper(&description)</p>

Date Functions

Table 2-34 lists the date functions.

Table 2-34 Date Functions

Function	Description
dateadd	<p>Returns a date after adding (or subtracting) the specified units to the <i>date_value</i>.</p> <p>Syntax: <i>date_var</i> = dateadd(<i>date_value</i>, <i>units_value</i>, <i>quantity_value</i>)</p> <ul style="list-style-type: none">■ <i>date_value</i> = date variable or expression■ <i>units_value</i> = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second'■ <i>quantity_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>date_var</i> = date variable <p>Example: let \$date = dateadd(\$startdate, 'day', 7.5)</p>
datediff	<p>Returns the difference between the specified dates expressed in <i>units_value</i>. The function returns a float value. The result can be negative if the first date is earlier than the second date.</p> <p>Syntax: <i>dst_var</i> = datediff(<i>date1_value</i>, <i>date2_value</i>, <i>units_value</i>)</p> <ul style="list-style-type: none">■ <i>date1_value</i> = date variable or expression■ <i>date2_value</i> = date variable or expression■ <i>units_value</i> = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second'■ <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #diff = datediff(\$date1, \$date2, 'hour')</p>
datenow	<p>Returns the current local date and time from the client machine.</p> <p>Syntax: <i>dst_var</i> = datenow()</p> <ul style="list-style-type: none">■ <i>dst_var</i> = date variable <p>Example: let \$date = datenow()</p>

Table 2-34 Date Functions (*Continued*)

Function	Description
datetofstr	<p>Converts the date <i>date_value</i> to a string in the format <i>format_mask</i>.</p> <p>Syntax: <i>dst_var</i> = datetofstr(<i>date_value</i>, [<i>format_mask</i>])</p> <ul style="list-style-type: none">■ <i>date_value</i> = date variable or expression■ <i>format_mask</i> = text literal, column, variable, or expression. The keyword DATE can be used to specify the DATE-EDIT-MASK setting from the current locale. If this argument is not specified, then the format specified by the SQR_DB_DATE_FORMAT setting is used. If this has not been set, then the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270 is used.■ <i>dst_var</i> = text variable <p>Example:</p> <pre>let \$formdate = datetofstr(\$date, 'Day Mon DD, YYYY') let \$localedate = datetofstr(\$date, DATE)</pre>
strtodate	<p>Converts the string <i>source_value</i> in the format <i>format_mask</i> to a date type.</p> <p>Syntax: <i>dst_var</i> = strtodate(<i>source_value</i> [, <i>format_mask</i>])</p> <ul style="list-style-type: none">■ <i>source_value</i> = text literal, column, variable, or expression■ <i>format_mask</i> = text literal, column, variable, or expression that describes the exact format of the <i>source_value</i>. The keyword DATE can be used to specify the DATE-EDIT-MASK setting from the current locale. If this argument is not specified, then <i>source_value</i> must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats (see Table 2-45, “Default Formats by Database,” on page 2-270), or the database-independent format 'YYYYMMDD[HH24[MI]SS[NNNNNN]]'. Valid format codes are specified in Table 2-41 on page 2-262. See the command, “PRINT” on page 2-255 regarding the default date-time components as a result of converting an incomplete date.■ <i>dst_var</i> = date variable <p>Example:</p> <pre>let \$date = strtodate(\$str_date, 'Mon DD, YYYY') let \$date = strtodate(\$str_date, DATE)</pre>

Unicode Functions

Table 2-35 lists the unicode functions.

Table 2-35 Unicode Functions

Function	Description
lengthp	(Only allowed when converting to Unicode internally) Returns the length of the string in print position. Half-width characters take one print position, full-width characters take two, and combining characters take zero. Syntax: <i>dst_var</i> = <i>lengthp(source_value)</i> ■ <i>source_value</i> = date or text literal, column, variable, or expression ■ <i>dst_var</i> = decimal, float, or integer variable Example: let #printLen = lengthp(\$string))
lengtht	(Only allowed when converting to Unicode internally) Returns the length of a given string in bytes when converted (transformed) to a specified encoding. The encoding names are the same as those allowed in the OPEN command or within the INI file. String and column variables can be used in place of the literal encoding name. Syntax: <i>dst_var</i> = <i>lengtht(source_value, encoding_value)</i> ■ <i>source_value</i> = date or text literal, column, variable, or expression ■ <i>encoding_value</i> = text literal, column, variable, or expression ■ <i>dst_var</i> = decimal, float, or integer variable Example: let #sjisLen = lengtht(\$string, 'shift-jis')

Table 2-35 Unicode Functions (*Continued*)

Function	Description
substrp	<p>(Only allowed when converting to Unicode internally) 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, SQR “rounds up” to the next character. When #printLen ends in a partial character, SQR “rounds down” to the previous character.</p> <p>Syntax: <code>dst_var = substrb(source_value, offset_value, length_value)</code></p> <ul style="list-style-type: none">■ <code>source_value</code> = date or text literal, column, variable, or expression.■ <code>offset_value</code> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.■ <code>length_value</code> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.■ <code>dst_var</code> = text variable. <p>Example: <code>let \$sub = substrp(&string, #printPos, #printlen)</code></p>
substrt	<p>(Only allowed when converting to Unicode internally) Returns a Unicode string equivalent to a byte level substring of a given string after converting (transforming) the given string to a given encoding. If the substring of the converted string yields a partial character, that character will be truncated.</p> <p>Syntax: <code>dst_var = substrb(source_value, offset_value, length_value, encoding_value))</code></p> <ul style="list-style-type: none">■ <code>source_value</code> = date or text literal, column, variable, or expression.■ <code>offset_value</code> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.■ <code>length_value</code> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.■ <code>encoding_value</code> = text literal, column, variable, or expression■ <code>dst_var</code> = text variable. <p>Example: <code>let \$sjisPrep = SUBSTRT (\$string, 1, 10, 'Shift-JIS')</code></p>

Table 2-35 Unicode Functions (*Continued*)

Function	Description														
transform	<p>(Only allowed when converting to Unicode internally) Returns a Unicode string which is specified transform of a given string.</p> <p>Syntax: <code>dst_var = transform (source_value, transform_value)</code></p> <ul style="list-style-type: none">■ <code>source_value</code> = date or text literal, column, variable or expression■ <code>transform_value</code> = text literal, column, variable, or expression■ <code>dst_var</code> - text variable. <p>Example: <code>let \$hiragana = transform (&string, 'ToHiragana')</code></p> <p>SQR supports the following transforms:</p> <p>(***Source: Rosette API Reference)</p> <ul style="list-style-type: none">■ ToLowercase – Transforms all uppercase Latin letters to lowercase (this includes both "half-width" and "full-width" Latin characters).■ ToUppercase – Transforms all lowercase Latin letters to uppercase (this includes both "half-width" and "full-width" Latin characters).■ ToFullwidth – Transforms all half-width characters that also have a full-width representation to their full-width form. Characters with full-width representations are: Roman alphabet characters (A-z), digits (0-9), Japanese <i>katakana</i> characters, and the most commonly used punctuation characters (including Space).■ 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 <i>katakana</i> characters, and the most commonly used punctuation characters (including Space).■ ToHiragana – Transforms all full-width <i>katakana</i> characters to <i>hiragana</i>. To convert half-width <i>katakana</i> characters to <i>hiragana</i>, you must first convert the characters to full-width using the FullWidth transform.■ ToParagraphSeparator – Standardizes the line/paragraph separators in the text according to the following standards: <table><thead><tr><th>StandardCode Point</th><th>Line/Paragraph Separator</th></tr></thead><tbody><tr><td>Windows0x0DOA</td><td>0x0DOA</td></tr><tr><td>Macintosh0x0D</td><td>ToCR</td></tr><tr><td>UNIX0x0A</td><td>ToLF</td></tr><tr><td>UnicodeU+2028</td><td>ToLineSeparator</td></tr><tr><td>UnicodeU+2029</td><td>ToParagraphSeparator</td></tr><tr><td>EBCDIC 0x15</td><td>ToEBCDICNewLine</td></tr></tbody></table>	StandardCode Point	Line/Paragraph Separator	Windows0x0DOA	0x0DOA	Macintosh0x0D	ToCR	UNIX0x0A	ToLF	UnicodeU+2028	ToLineSeparator	UnicodeU+2029	ToParagraphSeparator	EBCDIC 0x15	ToEBCDICNewLine
StandardCode Point	Line/Paragraph Separator														
Windows0x0DOA	0x0DOA														
Macintosh0x0D	ToCR														
UNIX0x0A	ToLF														
UnicodeU+2028	ToLineSeparator														
UnicodeU+2029	ToParagraphSeparator														
EBCDIC 0x15	ToEBCDICNewLine														

Table 2-35 Unicode Functions (*Continued*)

Function	Description
	<ul style="list-style-type: none"> ■ HankakuKatakanaToZenkaku – Converts half-width (<i>hankaku</i>) Japanese katakana characters to the full-width (<i>zenkaku</i>) form. This conversion is almost identical to <code>ToFullwidth</code>, except that it automatically composes and combines katakana "accent" marks (<i>dakuten</i> and <i>handakuten</i>) appropriately, whereas <code>ToFullwidth</code> does not provide any special treatment for these marks. ■ ZenkakuKatakanaToHankaku – Converts full-width (<i>zenkaku</i>) Japanese katakana characters to the half-width (<i>hankaku</i>) form. This conversion is almost identical to <code>ToHalfwidth</code>, except that it automatically decomposes and separates katakana "accent" marks (<i>dakuten</i> and <i>handakuten</i>) appropriately, whereas <code>ToHalfwidth</code> does not provide any special treatment for these marks.
<code>unicode</code>	<p>(Only allowed when converting to Unicode internally) Returns a Unicode string from the string of hexadecimal values provided. The syntax of the literal for <code>UNICODE</code> is</p> <pre>' [whitespace U+ \u]XXXX...'</pre> <p>where X is a valid hexadecimal digit: 0-9, a-f, or A-F. The hexadecimal value will always be in big-endian form.</p> <p>Syntax: <code>dst_var = unicode(source_value)</code></p> <ul style="list-style-type: none"> ■ <code>source_value</code> = text literal, column, variable or expression ■ <code>dst_var</code> = text variable <p>Example: <code>let \$uniStr = unicode ('U+5E73 U+2294')</code></p>

Miscellaneous Functions

Table 2-36 lists miscellaneous functions. These functions return a string value unless otherwise indicated.

Table 2-36 Miscellaneous Functions

Function	Explanation
array	<p>Returns a pointer to the starting address of the specified array field. The value returned from this function can only be used by a user-defined function. See the routine printarray in the file UFUNC.C for complete instructions on how to use this function.</p> <p>Syntax: <i>array_var</i> = array(<i>array_name</i>, <i>field_name</i>)</p> <ul style="list-style-type: none">■ <i>array_name</i> = text literal, column, variable, or expression■ <i>field_name</i> = text literal, column, variable, or expression■ <i>array_var</i> = text variable <p>Example: let #fstatus = printarray(array('products', 'name'), 10, 2, 'c')</p>
command_line	<p>Returns the command line arguments passed to the SQR (or SQRT) module.</p> <p>Syntax: <i>dst_var</i> = command_line()</p> <ul style="list-style-type: none">■ <i>dst_var</i> = text variable <p>Example: let \$cmdline = command_line()</p>
cond	<p>Returns <i>y_value</i> if the <i>x_value</i> is nonzero (0) otherwise returns <i>z_value</i>. If <i>y_value</i> is numeric, then <i>z_value</i> must also be numeric; otherwise, date and textual arguments are compatible. If either the <i>y_value</i> or <i>z_value</i> is a date variable, column, or expression, a date is returned. The return value of the function depends on which value is returned.</p> <p>Syntax: <i>dst_var</i> = cond(<i>x_value</i>, <i>y_value</i>, <i>z_value</i>)</p> <ul style="list-style-type: none">■ <i>x_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.■ <i>y_value</i> = Any literal, column, variable, or expression■ <i>z_value</i> = Any literal, column, variable, or expression■ <i>dst_var</i> = Any variable <p>Example: let #avg = #total / cond(&rate != 0, &rate, 1)</p>

Table 2-36 Miscellaneous Functions (*Continued*)

Function	Explanation
getenv	<p>Returns the value of the specified environment variable. If the environment variable does not exist, an empty string is returned.</p> <p>Syntax: <i>dst_var</i> = getenv(<i>env_value</i>)</p> <ul style="list-style-type: none">■ <i>env_value</i> = text literal, column, variable, or expression■ <i>dst_var</i> = text variable <p>Example: let \$myuser = getenv('USER')</p>
isnull	<p>Returns a value of one (1) if <i>source_val</i> is null; otherwise, returns a value of zero (0).</p> <p>Syntax: <i>dst_var</i> = isnull(<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = date or text literal, column, variable, or expression■ <i>dst_var</i> = decimal, float, or integer variable <p>Example: let #null = isnull(\$date)</p>

Table 2-36 Miscellaneous Functions (*Continued*)

Function	Explanation
nvl	<p>Returns <i>y_value</i> if the <i>x_value</i> is null; otherwise, returns <i>x_value</i>. If <i>x_value</i> is numeric, <i>y_value</i> must also be numeric; otherwise, date and textual arguments are compatible. In any case, the <i>x_value</i> determines the type of expression returned. The return value of the function depends on which value is returned.</p> <p>Syntax: <i>dst_var</i> = nvl(<i>x_value</i>, <i>y_value</i>)</p> <ul style="list-style-type: none">■ <i>x_value</i> = Any literal, column, variable, or expression■ <i>y_value</i> = Any literal, column, variable, or expression■ <i>dst_var</i> = Any variable <p>Example: let \$city = nvl(&city, '-- not city --')</p> <p>If <i>x_value</i> is a date and <i>y_value</i> is textual, then <i>y_value</i> is validated according to the following rules:</p> <ul style="list-style-type: none">■ For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats (see Table 2-45, “Default Formats by Database,” on page 2-270), or the database-independent format 'SYYYMDD[HH24[MI]SS[NNNNNN]]]]'.■ For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting, or the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.■ For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

Table 2-36 Miscellaneous Functions (*Continued*)

Function	Explanation
range	<p>Returns a value of one (1) if <i>x_value</i> is between <i>y_value</i> and <i>z_value</i>; otherwise, returns a value of zero (0). If the first argument is text or numeric, the other arguments must be of the same type. If the first argument is a date, the remaining arguments can be dates and/or text. It is also possible to perform a date comparison on a mix of date and text arguments, for example, where <i>x_value</i> is a date and <i>y_value</i> and <i>z_value</i> are text arguments. In a comparison of this sort, <i>y_value</i> must represent a date that is earlier than that of <i>z_value</i>.</p> <p>Syntax: <i>dst_var</i> = range(<i>x_value</i>, <i>y_value</i>, <i>z_value</i>)</p> <ul style="list-style-type: none">■ <i>x_value</i> = Any literal, column, variable, or expression■ <i>y_value</i> = Any literal, column, variable, or expression■ <i>z_value</i> = Any literal, column, variable, or expression■ <i>dst_var</i> = decimal, float, or integer variable <p>Example:</p> <pre>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)</pre> <p>If <i>x_value</i> is a date and <i>y_value</i> and/or <i>z_value</i> is textual, then <i>y_value</i> and/or <i>z_value</i> is validated according to the following rules:</p> <ul style="list-style-type: none">■ For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats (see Table 2-45, “Default Formats by Database,” on page 2-270), or the database-independent format 'YYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.■ For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting, or the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.■ For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.
roman	<p>Returns a string that is the character representation of <i>source_value</i> expressed in lower case roman numerals.</p> <p>Syntax: <i>dst_var</i> = roman(<i>source_value</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = text literal, column, variable, or expression.■ <i>dst_var</i> = text variable <p>Example: let \$roman = roman(#page-count)</p>

Table 2-36 Miscellaneous Functions (*Continued*)

Function	Explanation
wrapdepth	<p>Returns the number of print lines required by <i>source_value</i>. See the WRAP argument of the PRINT command for detailed descriptions of the parameters to this function. This function returns a float value.</p> <p>Syntax: <i>dst_var</i> = wrapdepth(<i>source_value</i>, <i>wrap_width</i>, <i>line_height</i>, <i>on</i>, <i>strip</i>)</p> <ul style="list-style-type: none">■ <i>source_value</i> = text literal, column, variable, or expression■ <i>wrap_width</i> = decimal, float, or integer literal, column, variable, or expression■ <i>line_height</i> = decimal, float, or integer literal, column, variable, or expression■ <i>on</i> = text literal, column, variable, or expression.■ <i>strip</i> = text literal, column, variable, or expression■ <i>dst_var</i> = decimal, float, or integer variable <p>Example: <i>let #depth</i> = wrapdepth(&<i>description</i>,40,1,'<13>', '')</p>

Writing Custom Functions

In addition to using SQR's built-in functions, you can write your own functions in C, using the supplied source file UFUNC.C.

You can pass any number of arguments to your function and values can be returned by the function or passed back in variables.

After editing and recompiling UFUNC.C, you must relink SQR.

The following is a step-by-step example of how to add a user-defined function to SQR so that it can be invoked using the LET, IF, or WHILE command.

The example adds the C function random, which returns a random number. The function accepts a parameter that is used as the seed to start a new sequence of numbers. If the seed is zero, then the same sequence is used.

When adding functions to UFUNC.C, keep in mind the following:

- String functions require the following arguments:
 - (int) Number of arguments.
 - (char *) or (double *) Array of argument pointers, to either char[] or double.
 - (char *) Address for result string. If unchanged, function returns a NULL string.
 - (int) Maximum length of result string, in bytes.
- Numeric functions require the following arguments:
 - (int) Number of arguments.
 - (char *) or (double *) Array of argument pointers, to either char[] or double.
 - (double *) Address for result numeric value. If unchanged, function returns zero.

To add the random function to SQR, add the following modifications to the UFUNC.C file provided with SQR:

- Add the prototype for the random function:

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

- Add the function name to the declaration list. The name of the function called from sqr is random. The return type is *n* for numeric. The number of arguments passed is "1," and the argument type is *n* for numeric. The function name in UFUNC.C is random. The characters "PVR" must be entered before the function name.

Name	Return_type	Number of Arguments	Arg_Types	Function
"max",	'n',	0,	"n",	PVR max,
"max",	'n',	0,	"n",	PVR max,
"split",	'n',	0,	"C",	PVR split,
"printarray",	'n',	4,	"cnnc",	PVR printarray,
"random",	'n',	1,	"n",	PVR random,
/* Last entry must be NULL -- do not change */				
"" , '\0' , 0 , "", 0				
};				

- At the end of the of UFUNC.C file, add the following random routine. The routine name must be lowercase; however, in your SQR program it can be referenced in either upper or lower case.

```
static void random CC_ARGL((argc, argv, result))
CC_ARG(int, argc) /* The number arguments passed */
CC_ARG(double *, argv[]) /* The argument list */
CC_LARG(double *, result) /* Where to store result */
{
    if (*argv[0] != 0)
        srand(*argv[0]);
    *result = rand();
    return;
}
```

After these modifications, recompile UFUNC.C and relink SQR. See the programmer's reference manual for details about your particular machine.

The following is an example of a simple SQR program that uses the random function:

```
begin-program
    do get-random-number
    do process-calculations
end-program

begin-procedure
    let #seed = 44
    let #ran = random(#seed)
end-procedure

begin-procedure process-calculations
    .
    .
    .
```

LOAD-LOOKUP

Function Loads an internal table with columns from the database. Allows for quick search using the LOOKUP command.

Syntax In the SETUP section:

```
LOAD-LOOKUP
NAME=lookup_table_name
TABLE=database_table_name
KEY=key_column_name
RETURN_VALUE=return_column_name
[ROWS=initial_row_estimate_int_lit]
[EXTENT=size_to_grow_by_int_lit]
[WHERE=where_clause_txt_lit]
[SORT=sort_mode]
[QUIET]
[SCHEMA=schema_txt_lit]
[
    PROCEDURE=proc_txt_lit
        [PARAMETERS=({{arg1 [IN|INOUT] } |NULL} [, argi
            [IN|INOUT] |NULL] ... )]
        (or)
    COMMAND=command_txt_lit
        (or)
    GETDATA=getdata_txt_lit
]
[ {FROM-ROWSETS=( {m|m-n|m-|-n [, . . . ] } |{ALL}) } |
    {FROM-PARAMETER=parameter_txt_lit} ]
```

In the body of the report:

```
LOAD-LOOKUP
NAME=lookup_table_name
TABLE=database_table_name
KEY=key_column_name
RETURN_VALUE=return_column_name
[ROWS=initial_row_estimate_lit|_var|_col]
[EXTENT=size_to_grow_by_lit|_var|_col]
[WHERE=where_clause_txt_lit|_var|_col]
[SORT=sort_mode]
```

```

[QUIET]
[SCHEMA={txt_lit|_var}]
[
    PROCEDURE={txt_lit|_var}
        [PARAMETERS=({{arg1 [IN|INOUT]}|NULL} [,argi
            [IN|INOUT]|NULL] ... )]
    (or)
    COMMAND={txt_lit|_var}
    (or)
    GETDATA={txt_lit|_var}
]
[ {FROM-ROWSETS=( {m|m-n|m-|-n} [,...] } | {ALL} ) ] |
{FROM-PARAMETER={txt_lit|_var}}]

```



Note The following LOAD-LOOKUP elements are specific to SQR for DDO:

- SCHEMA
- PROCEDURE
- COMMAND
- GETDATA
- FROM ROWSETS
- FROM PARAMETER

The following LOAD-LOOKUP elements are *not* supported (processed but not used) in SQR for DDO:

- TABLE
- WHERE
- SORT-DC
- SORT-DI

Arguments

NAME

The name of the lookup table. The array name is referenced in the LOOKUP command.

TABLE

The name of the table in the database, where the KEY and RETURN_VALUE columns or expressions are stored (not supported for DDO).

KEY

The name of the column that is 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, SQR 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

The name of the column (expression) that is 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

The initial size of the lookup table. This argument is optional, and if not specified, a value of 100 is used.

EXTENT

The amount to increase the array when it becomes full. This argument is optional, and if not specified, a value of 25% of the ROWS value is used.

WHERE

A 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

The sorting method to be used. The following values are permitted:

- DC – Database sorts data, case-sensitive sort (not supported for DDO)
- DI – Database sorts data, case-insensitive sort (not supported for DDO)
- SC – SQR sorts data, case-sensitive sort
- SI – SQR 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 – The name of the datasource-stored procedure to be executed. 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 – A text string you pass to the datasource without modification by SQR. This string can include embedded SQR 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. If you specify more than one row set, 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 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 to be retrieved.

 **Note**

This is a similar concept to the PARAMETERS = statement in DECLARE- and ALTER-CONNECTION, with the minor difference being 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 the LOAD-LOOKUP command in conjunction 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 the LOOKUP command, SQR 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 to be 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, SQR lets the database sort the data. This works fine if the database and SQR 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 SQR is running on is faster than the machine the database is running on, letting SQR 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 the LOOKUP command to retrieve the full text of a state name from the abbreviation.

The following example uses the LOOKUP command to validate data entered by a user using an INPUT command:

```
get_state:
input $state 'Enter state abbreviation'
uppercase $state
lookup states $state $name
if $name = '' ! Lookup didn't find a match
    show 'No such state.'
    goto get_state
end-if
```

Surround any command argument with embedded spaces by single quotes, as shown in the following example:

```
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 a LOAD-LOOKUP command 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 mutiple 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

The LOOKUP command.

LOOKUP

Function	Searches a lookup table (an array) for a key value and returns the corresponding text string.
Syntax	<code>LOOKUP <i>lookup_table_name</i> {<i>key_any_lit _var _col</i>} {<i>ret_txt_var _date _var</i>}</code>
Arguments	<p><i>lookup_table_name</i> Specifies the lookup table. This table must be previously loaded with a LOAD-LOOKUP command.</p> <p><i>key_any_lit _var _col</i> The key used for the lookup.</p> <p><i>ret_txt_var _date _var</i> A string variable into which the corresponding value is returned.</p>
Description	Speeds up processing for long reports. For example, if you want to print the entire state name rather than the abbreviation, you could use LOAD-LOOKUP and then LOOKUP to do this.
Examples	The following example works in conjunction with the example for the LOAD-LOOKUP command: <code>lookup states &state_abbr \$state_name</code> This example searches the <i>states</i> lookup table for a matching <i>&state_abbr</i> value; if found, it returns the corresponding state name in <i>\$state_name</i> . If not found, a null is placed in <i>\$state_name</i> .
See Also	The LOAD-LOOKUP command.

LOWERCASE

Function Converts a text variable to lowercase.

Syntax LOWERCASE *txt_var*

Arguments *txt_var*
Specifies a text variable to be converted 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 2-36, “Miscellaneous Functions,” on page 2-220.

MBTOSBS

Function Converts a double-byte string to its single-byte equivalent.

Syntax MBTOSBS {*txt_var*}

Arguments *txt_var*
Specifies the string to be converted.

Description This command 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 the LET command.

MOVE

Function Moves one field to another field and optionally edits the field.

Syntax

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

Arguments

src_any_lit|_var|_col
Specifies any source column, variable, or literal.



Note A date can be stored in a date variable or column, or a string literal, column, or variable. When using a date *format_mask* or the keyword DATE with the MOVE command, the source, if a string literal, column, or variable, must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats as listed in Table 2-45, “Default Formats by Database,” on page 2-270, or the database-independent format 'YYYYMMDD[HH24[MI]SS[NNNNNN]]'.

When numerical precision is important, use the LET command. When no edit mask is specified, the MOVE command uses the 6 digit precision default mask.

dst_any_var
Specifies a destination variable.

format_mask
Specifies an optional format mask. For additional information regarding edit masks, see the command, “PRINT” on page 2-255.

NUMBER
Indicates that *src_any_lit|_var|_col* is to be formatted using the NUMBER-EDIT-MASK from the current locale. This option is not legal with date variables. (See the command, “ALTER-LOCALE” on page 2-9.)

MONEY
Indicates that *src_any_lit|_var|_col* is to be formatted using the MONEY-EDIT-MASK from the current locale. This option is not legal with date variables. (See the command, “ALTER-LOCALE” on page 2-9.)

DATE

Indicates that *src_any_lit|_var|_col* is to be formatted using the DATE-EDIT-MASK from the current locale. This option is not legal with numeric variables. (See the command, “ALTER-LOCALE” on page 2-9.)

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 SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

Finally, as this example shows, the edit mask can be contained in a string variable.

Examples

The following example illustrates the various features of the MOVE command:

The following code:

```
!
! Convert a string in place
!
move '123456789' to $ssn
move $ssn to $ssn xxx-xx-xxxx
show '$SSN = ' $ssn
```

Produces the following output:

```
$SSN = 123-45-6789
```

The following code:

```
!
! Convert a number to a string using an edit mask
!
move 1234567.89 to #value
move #value to $value 999,999,999.99
show '$Value = ' $value
```

Produces the following output:

```
$Value = 1,234,567.89
```

The following code:

```
!
! Convert a number to a string using a variable edit mask
!
move 123 to #counter
move '099999' to $mask
move #counter to $counter :$mask
show '$Counter = ' $counter
```

Produces the following output:

```
$Counter = 000123
```

The following code:

```
!
! Convert a number to a string using the default edit mask
!
! SQR, 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

- The LET command for information on copying, editing, or converting fields.
- The EDIT parameter of the PRINT command for a description of the edit masks.
- The ALTER-LOCALE command for a description of the arguments NUMBER-EDIT-MASK, MONEY-EDIT-MASK, and DATE-EDIT-MASK.
- The PRINT command regarding 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	<code>MULTIPLY {src_num_lit _var _col} TIMES dst_num_var [ROUND=nn]</code>
Arguments	<p><i>src_num_lit _var _col</i> Specifies a numeric source column, variable, or literal.</p> <p><i>dst_num_var</i> Specifies a destination numeric variable.</p> <p>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.</p>
Description	<p><code>MULTIPLY</code> multiplies the first field by the second and places the result into the second field.</p> <p>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.</p>
Examples	<pre>multiply &quantity times #cost multiply 1.5 times #result</pre>
See Also	<ul style="list-style-type: none">■ The <code>ADD</code> command for more information.■ The <code>LET</code> command for a discussion of complex arithmetic expressions.

NEW-PAGE

Function	Writes the current page and begins a new one.
Syntax	NEW-PAGE [<i>erase_from_line_num_lit _var _col</i>]
Arguments	<i>erase_from_line_num_lit _var _col</i> Specifies a numeric column, variable, or literal for line printers.
Description	For line printers, this command 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

```
! Write current page, then erase it
! beginning at line 5.
new-page 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*
Specifies a new file name.

Description This command is normally only used with single reports. When used with multiple report declarations, this command affects the *current* report only. The internal page counter is reset to 1 when NEW-REPORT is executed.



Note

SQR 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 SQR report by issuing the NEW-REPORT command before printing. You might even prompt for the filename to use, as shown in the following example:

```
begin-report  
    input $file 'Enter report filename'  
    new-report $file  
    ...
```

After execution of this command, the reserved variable *\$sqr-report* is updated to reflect the new report name.

See Also

- The DECLARE-REPORT and USE-REPORT commands.
- The -F command-line flag.

NEXT-COLUMN

Function Sets the current position on the page to the next column defined with the COLUMNS command.

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
Specifies 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  
from phonelist! within two columns.  
order by name  
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

The COLUMNS and USE-COLUMN commands.

NEXT-LISTING

Function	Ends the current set of detail lines and begins another.
Syntax	<pre>NEXT-LISTING [NO-ADVANCE] [SKIPLINES={num_lit _var _col}] [NEED={num_lit _var _col}]</pre>
Arguments	<p>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.</p> <p>SKIPLINES Causes the specified number of lines to be skipped before setting up the new offset.</p> <p>NEED Specifies the 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.</p>
Description	<p>NEXT-LISTING is used in tabular reports. This command causes a new vertical offset in the page to begin.</p> <p>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 SQR reserved variable #current-line still reflects the actual line number within the page body.</p> <p>The value of SKIPLINES must be a nonnegative integer. If it is less than 0, then 0 is assumed.</p> <p>The value of NEED must be an integer greater than 0. If it is less than or equal to 0, then 1 is assumed.</p>

Examples

```
begin-select
  cust_num  (1,1)edit 099999
  city(,+3)
  name(2,10,30)
  address(,+2)
    next-listing skiplines=1 need=2
  from customers order by cust_num
end-select
```

! Each detail group prints
! starting on line 1 since
! NEXT-LISTING keeps
! moving line 1 down the
! page. NEED=2 keeps 2
! line detail groups from
! breaking across
! pages.



Note The NEXT-LISTING command 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]}  
[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

Specifies 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

Specifies a 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, SQR procures all data sequentially. SQR 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. SQR 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

This file type assumes 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

This file type specifies that all records contained within the file are the same length with no line terminators. When writing records, SQR 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

This file type specifies 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 the OPEN succeeds and to -1 if it fails. Without the STATUS argument, a failure on OPEN causes SQR 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 SQR. When no encoding is specified, SQR uses the file input or output encoding specified in the INI file unless the file is UCS-2 encoded and auto-detection of UCS-2 files is enabled. Encoding is only allowed when converting to Unicode internally.

Description

After a file is opened, it remains open until explicitly closed by the CLOSE command. A maximum of 256 files can be opened at one time.

Examples

```
open 'stocks.dat' as 1 for-reading record=100
open 'log.dat'      as 5 for-writing record=70
open $filename     as #j for-append   record=80:fixed
open $filename     as 2 for-reading record=80:fixed_nolf

open $filename     as 6 for-reading record=132:vary
status=#filestat
if #filestat != 0
    ... error processing ...
end-if
```

See Also

The READ, WRITE, and CLOSE commands for more information about using files.

PAGE-NUMBER

Function	Places the current page number on the page.
Syntax	PAGE-NUMBER <i>position</i> [<i>pre_txt_lit</i> [<i>post_txt_lit</i>]]
Argument	<i>position</i> Specifies the position of the page number.
	<i>pre_txt_lit</i> Specifies a text string to be printed before the page number.
	<i>post_txt_lit</i> Specifies a text string to be printed after the page number.
Description	The text specified in <i>pre_txt_lit</i> and <i>post_txt_lit</i> are printed immediately before and after the number.
Examples	<pre>begin-footing 1 page-number(1,37) 'Page '! Will appear as last-page () ' of ' '.'! "Page 12 of 25." end-footing</pre>
See Also	The LAST-PAGE command.

POSITION

Function	Sets the current position on a page.
Syntax	<pre>POSITION position [@document_marker [COLUMNS{num_lit _var _col} [num_lit _var _col]...]]</pre>
Arguments	<p><code>@document_marker</code></p> <p>References 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.</p> <p><code>COLUMNS</code></p> <p>Defines columns beginning at the location of the document marker. The columns defined are relative to the position of the document marker.</p> <p>When COLUMNS is used, the entire command cannot be broken across more than one program line.</p>
Examples	<pre>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 ()</pre>
See Also	<ul style="list-style-type: none">■ The COLUMNS command for more information.■ The examples with the description of a DOCUMENT paragraph in Chapter 11, “Creating Form Letters” in the <i>Hyperion SQR Getting Started Guide</i>.

PRINT

Function Puts 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
Specifies the data to print.

 **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 the PRINT command, a date in a string literal, column, or variable must be in an acceptable format. See the description for “EDIT” on page 2-260 for further details.

SQR for DDO does not support printing of List variables.

position

Specifies the position where the data is to be printed. (For additional information on position, see Chapter 15, “Changing Fonts” in *Hyperion SQR Getting Started Guide*.)

format_command[format_cmd_params]

Specifies optional formatting commands and parameters.

Description The PRINT command has the following format commands:

BACKGROUND	MATCH
BOLD	MONEY
BOX	NOP
CENTER	NUMBER
CODE-PRINTER	ON-BREAK
DATE	POINT-SIZE
DELAY	SHADE
EDIT	UNDERLINE
FILL	URL
FONT	URL-TARGET
FOREGROUND	WRAP

Some of these format commands can be used in combination with others and some are mutually exclusive. Table 2-37 shows which can be used together. A “•” indicates that they can be used together.

Table 2-37 Valid Print Format Command Combinations

	B O L D	B O X	CENTER	CODE- PRINTER	D E L A Y	EDIT NUMBER MONEY DATE	F I L L	F O N T	F /B	MATCH	N O P	ON-BREAK	POINT SIZE	SHADE	UNDER- LINE	URL URL- Target	W R A P
BOLD		•	•		•	•	•	•	•	•	•	•		•	•	•	•
BOX	•		•		•	•	•	•	•	•	•	•		•		•	•
CENTER	•	•				•	•	•	•	•	•	•		•	•	•	
CODE- PRINTER											•						
DELAY	•	•				•	•	•	•					•	•	•	
EDIT NUMBER MONEY DATE	•	•	•		•			•			•	•		•	•	•	
FILL	•	•	•		•			•		•	•	•		•	•	•	•
FONT	•	•	•				•	•	•	•	•	•	•	•	•	•	•
F/B	•		•		•	•	•		•	•	•	•		•		•	•
MATCH	•	•	•					•		•	•			•	•	•	
NOP	•	•	•		•	•	•	•	•	•	•	•		•	•	•	•
ON-BREAK	•	•	•			•	•	•	•	•	•			•	•	•	
POINT- SIZE	•	•	•			•	•	•	•	•	•	•		•	•	•	•
SHADE	•	•	•		•	•	•	•	•	•	•	•			•	•	•
UNDER- LINE	•		•		•	•	•	•	•	•	•	•		•		•	•
URL URL- TARGET	•	•	•		•	•	•	•	•	•	•	•		•	•		•
WRAP	•	•					•	•		•	•			•	•	•	



Note In the above table, F/B stands for FOREGROUND/BACKGROUND.

The following sections 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 2-21, “DECLARE-PRINTER Command Arguments,” on page 2-124 for information on which fonts you can bold.

For line printers, when the BEFORE-BOLD and AFTER-BOLD arguments on the DECLARE-PRINTER command are used, the specified strings are added before and after the data to be bolded. If BEFORE-BOLD and AFTER-BOLD are not specified, then BOLD has no effect.

For example:

```
print &name (+1, 20) bold  
print 'Your account is in arrears' (1,1) bold
```

BOX

Draws a one-line deep graphical box around the printed data. This option 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 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.

CENTER

Centers the field on a line. The position qualifier for column is ignored. For example:

```
print 'Quarterly Sales' (1) center
```

CODE-PRINTER

Adds nondisplay characters to the program for the purpose of 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 to be 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 the PRINT command, 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

Indicates that the field is to be formatted using the DATE-EDIT-MASK from the current locale. (See “ALTER-LOCALE” on page 2-9.) If this entry is not set, then the date is printed according to the rules in Table 2-38.

Table 2-38 Date Formats if Column Type Not Set

Column Type	Default Mask	If not set
DATETIME	SQR_DB_DATE_FORMAT	See Table 2-45, “Default Formats by Database,” on page 2-270 for the format.
DATE	SQR_DB_DATE_ONLY_FORMAT	See Table 2-45, “Default Formats by Database,” on page 2-270 for the format.
TIME	SQR_DB_TIME_ONLY_FORMAT	See Table 2-47, “TIME Column Formats,” on page 2-270 for the format.

You cannot use DATE with numeric columns or variables.

DELAY

Directs SQR to delay the actual printing of the data until a SET-DELAY-PRINT command is issued against the variable used in PRINT. For example:

```
PRINT $Last_User (1,10) Delay  
.  
.  
.  
SET-DELAY-PRINT $Last_User with &Username
```

EDIT

Causes each field to be edited before it is printed. The three types of edits are:

- Text edit (see Table 2-39 on page 260)
- Numeric edit (see Table 2-40 on page 261)
- Date edit (see Table 2-41 on page 262)

Syntax

EDIT *edit_format*

Text Edit Format Characters

Table 2-39 lists the text edit format characters.

Table 2-39 Text Edit Format Characters

Character	Description
X	Use character in field.
B	Insert blank.
~ (tilde)	Skip character in field.
R[n]	Reverse sequence of string, for languages such as Hebrew. The optional number indicates right justification within length indicated.

Any other character (for example, punctuation) in a text edit mask is treated as a constant and is included in the edited field.

The characters 8, 9, 0, V, and \$ are illegal in a text edit mask because they are used to indicate that the mask is for a numeric edit.

Numeric Edit Format Characters

Table 2-40 lists the numeric edit format characters.

Table 2-40 Numeric Edit Format Characters

Character	Description
8	Digit, zero fill to the right of the decimal point, trim leading blanks (left justify the number).
9	Digit, zero fill to the right of the decimal point, space fill to the left.
0	Digit, zero fill to the left.
\$	Dollar sign, optionally floats to the right.
B	Treated as a "9", but if a value is zero, the field is converted to blanks.
C	Entered at the end of the mask, causes the comma and period characters to be transposed when the edit occurs. This is to support monetary values where periods delimit thousands and commas delimit decimals. (Example: 1.234,56).
E	Scientific format. The number of 9s after the decimal point determines the number of significant digits displayed. The "E" can be upper or lower case; the display follows the case of the mask.
V	Implied decimal point.
MI	Entered at the end of the mask, causes a minus to display at the right of the number.
PR	Entered at the end of the mask, causes angle brackets (<>) to display around the number if the number is negative.
PS	Entered at the end of the mask, causes parentheses to display around the number if the number is negative.
PF	Entered at the end of the mask, causes floating parentheses to display around the number if the number is negative.
NA	Entered at the end of the mask, causes "N/A" to display if the numeric column variable is null. The case of N/A follows that of the mask.
NU	Entered at the end of the mask, causes blanks to display if the numeric column variable is null.
.	Decimal point.
,	Comma.



Note Characters other than those listed in Table 2-40 are illegal for numeric edit masks and cause errors during processing.

Date Edit Format Characters

Table 2-41 lists the date edit format characters.

Table 2-41 Date Edit Format Characters

Character	Description
YYY YY Y	Last 3, 2, or 1 digit(s) of year. On input, for calculating the 4-digit year, the current century and/or decade are used. For example, a '9' using the 'Y' mask would result in 1999 as the year if the current year is in the 1990s.
YYYY SYYYY	4 digit year, "S" prefixes BC dates with "-".
RR	Last 2 digits of year; for years in other centuries. See Table 2-42 on page 2-263.
CC or SCC	Century; "S" prefixes BC dates with "-".
BC AD	BC/AD indicator.
Q	Quarter of year (1,2,3,4; JAN-MAR=1).
RM	Roman numeral month (I-XII; JAN=I).
WW	Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year.
W	Week of the month (1-5) where week 1 starts on the first day of the month and ends on the seventh.
DDD	Day of year (1-366).
DD	Day of month (1 - 31).
D	Day of week (1-7). Sunday is first day of week.
DAY	Name of day.
DY	Abbreviated name of day.
ER	Japanese Imperial Era. Returns the name of the Japanese Imperial Era in the appropriate kanji ('Heisei' is the current era).

Table 2-41 Date Edit Format Characters (*Continued*)

Character	Description
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).
SSSS	Seconds past midnight (0-86399).
N NN NNN NNNN NNNNN NNNNNN	Fractions of a second. Precise to microseconds; however, for most hardware and databases, this much accuracy will not be attainable.
MONTH	Name of month.
MON	Abbreviated name of month.
MM	Month (01-12; JAN=01).
MI	Minute (0-59).
SS	Second (0-59).
	Used to concatenate different masks.

Table 2-42 Date Edit Format Code-RR

Last 2 digits of current year	2-digit year is 00 - 49	2-digit year is 50 - 99
00 - 49	The return date is in the current century.	The return date is in the century before the current one.
50 - 99	The return date is in the century after the current one.	The return date is in the current century.

Edit Masks

As you work with edit masks, keep in mind the following:

- When using text, date, and numeric scientific edit masks with PRINT, the specified width value of PRINT sets the length allocated for the data displayed. For all other numeric edit masks, the edit mask sets the allocated length.
- All masks can be used by the `strtodate` function except for CC, SCC, Q, W, and WW.
- A backslash forces the next character into the output from the mask. For example, a mask of “*The cu\rrent \mo\nth is Month*” results in the output string of “*The current month is January*”. Without the backslashes the output string would be “*The cu95e7t january is January*”.
- You can use a vertical bar as a delimiter between format codes; however, in most cases the bar is not necessary. For example, the mask 'YYYY | MM | DD' is the same as 'YYYYMMDD'.
- Any other character (for example, punctuation) in a date edit mask is treated as a constant and is included in the edited field. If the edit mask contains spaces, you must enclose it in single quotes (').
- The masks MON, MONTH, DAY, DY, AM, PM, BC, AD, and RM are case-sensitive and follow the case of the mask entered. For example, if the month is January, the mask Mon yields “Jan” and MON yields “JAN”. All other masks are case-insensitive and can be entered in either uppercase or lowercase.
- National Language Support is provided for the following masks: MON, MONTH, DAY, DY, AM, PM, BC, and AD. See “ALTER-LOCALE” on page 2-9 or in Chapter 6, “SQR 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 2-43.

Table 2-43 Sample Date Edit Masks

Edit Mask	Result
dd/mm/yy	14/03/96
DD-MON-YYYY	14-MAR-1996
'Month dd, YYYY'	March 14, 1996
MONTH-YYYY	MARCH-1996

Table 2-43 Sample Date Edit Masks (*Continued*)

Edit Mask	Result
HH:MI	09:35
'HH:MI PM'	09:35 AM
YYYYMMDD	19960314
MM.DD.YYYY	03.14.1996
Mon	Mar
DD D DDD	143073

- In addition to the EDIT argument, you can use edit masks with the MOVE, CONCAT, DISPLAY, and SHOW commands, and with the edit function of the LET command. You edit the field using the supplied mask before storing or displaying it.
- When a date with missing date and/or time components displays or prints, the defaults are as follows:
 - The default year is the current year.
 - The default month is the current month.
 - The default day is one.
 - The default time is zero (00:00:00.000000).

For example, assuming today is September 7, 1996, the following assignment would produce an equivalent date-time of September 1, 1996 13:21:00.000000:

```
let $date1 = strtodate('13:21', 'HH:MI')
```

- You can dynamically change edit masks by storing them in a string variable and referencing the variable name preceded by a colon (:).

For example:

```
move '$999,999.99' to $mask
print #total (5,10) edit :$mask
show #total edit :$mask
```

- When a date stored in a string literal, column, or variable prints with an edit mask, it must be in one of the following formats:
 - The format specified by the SQR_DB_DATE_FORMAT environment variable, or the corresponding setting in the SQR.INI file.

- One of the database-dependent formats, as listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- The database-independent format,
'SYYYYYMMDD [HH24 [MI [SS [NNNNNN]]]]'.
- When a date column or variable prints without an edit mask, the date prints in the format specified by the environment variable SQR_DB_DATE_FORMAT or the corresponding setting in the SQR.INI file. If this has not been set, then the date prints in the primary database format (the first entry) listed in Table 2-45.

This applies to DISPLAY, MOVE, and SHOW commands as well as PRINT.

Sample Edit Masks

Table 2-44 shows sample edit masks and resulting fields.

Table 2-44 Sample Edit Masks

Mask	Value	Display
999.99	34.568	34.57
9,999,999V9999	123,456.7890	123,4567890
8,888,888.888	123,456.789	123,456.789
9,999	1234	1,234
9,999	123	123
09999	1234	01234
9999	-123	-123
9999	-1234	****
9999	12345	****
9999mi	-123	123-
9999pr	-123	< 123>
999999ps	-123	(123)
999999pf	-123	(123)
9999na	(null)	n/a

Table 2-44 Sample Edit Masks (*Continued*)

Mask	Value	Display
9999nu	(null)	(blank)
\$\$9,999.99c	1234.56	\$1.234,56
\$\$9,999.99	1234.56	\$1,234.56
\$\$9,999.99	12.34	\$ 12.34
\$\$,\$\$9.99	12.34	\$12.34
9.999e	123456	1.235e+05
B9,999	0	(blank)
B9,999	12345	12,345
(xxx)bxXX-xxxx	2169910551	(216) 991-0551
xxx-xx-xxxx	123456789	123-45-6789
~~xx~xx	ABCDEFGHIJ	CDFG
r10	ABCDEFG	GFEDCBA

Uses of Edit Masks

The following example shows some uses of edit masks:

```
print #total (7,55,0) edit $999,999.99 ! $ 12,345.67
print #total (7,55,0) edit $$9,999.99 ! $12,345.67
print #total (7,55,0) edit 999,999.99pr ! < 12,345.67>(if
! neg)
print #comm (7,55,0) edit b99,999.99 ! Blank if zero
print &cnum (16,1,0) edit 099999 ! 001234
print #cat (5,10,0) edit 9.999E ! 1.235E+04
print #phone (16,60,0) edit (xxx)bxXX-xxxxx ! (216) 397-0551
print #total (7,55,0) edit f££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.

Print Statement	Display	Current Position
Print 'ABCDEFGHIJ' (1,1, 5) Edit xxxxxxxx	ABCDE	(1,1, 6)
Print 'ABCDEFGHIJ' (1,1, 5) Edit xxxx	ABCDE	(1,1, 6)
Print 'ABCDE' (1,1, 10) Edit xxxxxxxx	ABCDE	(1,1, 11)
Print 'ABCDE' (1,1, 10) Edit xxxx	ABCDE	(1,1, 11)

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

Print Statement	Display	Current Position
Print \$current-date (1,1, 5) Edit DD/MM/YYYY	16/05	(1,1, 6)
Print \$current-date (1,1, 5) Edit DD/MM	16/05	(1,1, 6)
Print \$current-date (1,1, 10) Edit DD/MM/YYYY	16/05/2003	(1,1, 11)
Print \$current-date (1,1, 10) Edit DD/MM	16/05	(1,1, 11)

- Numeric Scientific Edit Masks – Print width sets the length allocated.

Print Statement	Display	Current Position
Print 1234567890 (1,1, 5) Edit 9e999999	1.23	(1,1, 6)
Print 1234567890 (1,1, 10) Edit 9e999999	1.234568e 1.234568e	(1,1, 11)
Print 1234567890 (1,1, 20) Edit 9e999999	1.234568e+009 1.234568e+009	(1,1, 21)

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

Print Statement	Display	Current Position
Print 1234567890 (1,1, 5) Edit 9999999999	12345	(1,1, 11)
Print 1234567890 (1,1, 5) Edit 99999	*****	(1,1, 6)
Print 12345 (1,1, 10) Edit 9999999999	_____12345	(1,1, 11)
Print 12345 (1,1, 10) Edit 99999	12345	(1,1, 6)
Print 1234567890 (1,1, 5) Edit 8888888888	12345	(1,1, 11)
Print 1234567890 (1,1, 5) Edit 88888	12345	(1,1, 6)
Print 12345 (1,1, 10) Edit 8888888888	12345	(1,1, 6)
Print 12345 (1,1, 10) Edit 88888	12345	(1,1, 6)

Default Formats

Review the following tables for information on default formats by database, date column formats, and time column formats.

Table 2-45 Default Formats by Database

Database	Default Formats
DB2	YYYY-MM-DD-HH:MI:SS.NNNNNN YYYY-MM-DD
Informix	YYYY-MM-DD HH:MI:SS.NNN MM/DD/YYYY MM-DD-YYYY MM.DD.YYYY
ODBC	'MON DD YYYY HH:MM:SS'
Oracle	DD-MON-YY
Sybase	MON DD YYYY HH:MM:SS MON DD YYYY [HH:MI[:SS[.NNNN]]][PM] MON DD YYYY [HH:MI[:SS[.NNNN]]][PM] YYYYMMDD [HH:MI[:SS[.NNNN]]]PM YYYYMMDD [HH:MI[:SS[.NNNN]]]PM

Table 2-46 DATE Column Formats

Database	DATE Column Formats
DB2	YYYY-MM-DD
Informix	MM/DD/YYYY
ODBC	DD-MON-YYYY

Table 2-47 TIME Column Formats

Database	TIME Column Formats
DB2	HH24.MI.SS
ODBC	HH24:MI:SS

FILL

Fills the page with the specified character or string as indicated by the print position and length.

The following example prints a line of stars and then a line of dashes followed by stars:

```
print '*' (1,1,79) fill! Fill line with '*'s  
print '-*' (+1,20,40) fill! Fill with '-*' characters.
```



Note When using the Text, Numeric, and Date edit masks with PRINT, the specified width value of the PRINT determines the length allocated for the displayed data. For all other "Numeric" edit masks, the edit mask sets the allocated length.

FONT

Causes the string to printed in the specified font. For example:

```
print 'Hello world' (3,3) font=5
```

FOREGROUND/BACKGROUND

When you specify a color on the PRINT command, its scope is that of the PRINT command. If you do not define the specified color name, then the setting for “default” will be 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 Refer to the example in the “ALTER-COLOR-MAP” command section to better understand the FOREGROUND and BACKGROUND commands.

MATCH

Compares a field to a list of key values and if a match is found, prints the corresponding string at the specified line and column.

If the *match_text* contains white space, it must be enclosed in single quotes (').

Any number of match text(s) can be tested, but each must have its own line, column, and *print_text*.

If a match is not found, the unmatched field is printed at the position specified in the parentheses.

Line and column positions for each matched string are treated as fixed or relative positions depending on the type of positioning used in the position qualifier for the PRINT command.

Syntax

```
MATCH match_text {line_num_lit|_var|_col}  
{column_num_lit|_var|_col} print_text ...
```

For example:

```
print &type_buyer (20,12) match  
A 20 12 Casual  
B 20 22 Impulsive  
C 21 12 Informed  
D 21 22 Choosey
```

To use relative line and fixed column positioning, for example, you could use the following example:

```
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

Indicates that the column or variable is to be formatted using the MONEY-EDIT-MASK from the current locale. (See the command, “ALTER-LOCALE” on page 2-9.) This can only be used with a numeric column or variable.

NOP

Suppresses the print command, causing “no operation” to be executed. This argument is useful for temporarily preventing a field from printing.

For example:

```
print  &ssn  (1,1)  nop    ! Hide the social security number.
```

NUMBER

Indicates that the column or variable is to be formatted using the NUMBER-EDIT-MASK from the current locale. (See the ALTER-LOCALE command.) This argument can only be used with a numeric column or variable.

ON-BREAK

Causes the specified action in a tabular report when the value of a field changes (a break occurs). The default action prints the field only when its value changes (PRINT=CHANGE).

Syntax

```
ON-BREAK [PRINT={ALWAYS|CHANGE|CHANGE/TOP-PAGE|NEVER} ]  
[SKIPLINES={num_lit|_var|_col}]  
[PROCEDURE=procedure_name[(arg1[,argi]...)]]  
[AFTER=procedure_name[(arg1[,argi]...)]]  
[BEFORE=procedure_name[(arg1[,argi]...)]]  
[SAVE=txt_var]  
[LEVEL=nn]  
[SET=nn]
```

ON-BREAK has the following qualifiers:

- PRINT – Specifies when the break field is printed.
 - 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 be invoked 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 skip-lines=2
county (,+2,14) on-break level=2 after=county-tot skip-lines=1
city   (,+2,14) on-break level=3 after=city-tot
...
end-select
```

The 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 would also be 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

Cause the string to be printed 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 the 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 be 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. (SQR 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 the following ULR in your report:

<http://www.somewebhost.com/~myusername/index.htm>

When you click on the "My web page" your browser is directed to the page.

URL-TARGET

The target within the specified URL. (SQR does not validate the target.) For example:

```
Print $URL (40,10)
      Point-Size=#Font
      Font=8
      URL=$URL
      URL-TARGET=$Target
      Background = (255, 0, 0)
      Foreground = ('yellow')
      Underline
```

WRAP

Wraps text at word spaces. Additional text is moved to a new line.

Syntax

```
WRAP {line_length_lit|_var|_col}
      {max_lines_lit|_var|_col} [KEEP-TOP]
      [STRIP=strip_chars] [ON=break_chars] [R]
      [LINE-HEIGHT={line_height_lit|_var|_col}]
```

line_length_lit|_var|_col

Specifies 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 the DECLARE-LAYOUT command to make the page width one character wider than the right edge of the wrapped text to avoid generating an implicit line feed.

For example:

```
print &comment (48,20,0) wrap 50 3 ! Paragraph is 50
                                         ! characters wide,
                                         ! with a maximum
                                         ! depth of 3 lines.

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 line position is 1 for each of the three wrapped fields: *note1*, *note2*, and *note3*. The current print position after a wrap occurs at the bottom right edge of the wrapped paragraph. To continue printing on the same line, you must use a fixed line number for the next field.

max_lines_lit|_var|_col

Specifies the maximum paragraph depth in lines. Usually, the line length and maximum lines are indicated with numeric literals. However, WRAP can also reference numeric variables or columns. This is useful when you want to change the width or depth of a wrapped paragraph during report processing. The numeric variable can optionally be preceded by a colon (:).

For example:

```
print $comments (1,30) wrap #wrap_width 6
print $message (5,45) wrap #msg_wid #msg_lines
```

KEEP-TOP retains the current line position except if a page break occurs, in which case, line 1 is used as the current line position. The default action is to set the next print position at the bottom of the wrapped data.

In the following example, the column *&resolution* prints on the same line as the first line of the column *&instructions*:

```
print &phone (+1,10) edit '(xxx) xxx-xxxx'
print &instructions (+1,10,30) wrap 6 10 keep-top
print &resolution (0,+3,25)
```

The STRIP and ON arguments affect which characters are to be converted before wrapping, and which characters force a wrap to occur.

- Characters in the STRIP string argument are converted to spaces before the wrap occurs.
- Characters in the ON string argument cause a wrap at each ON character found. The ON character is not printed.

Both arguments accept regular characters and nondisplay characters whose ASCII values are surrounded by angled brackets, <nn>.

For example, to print a long data type that contains embedded carriage returns, the setup would be:

```
print &long_field (5,20) wrap 42 30 on=<13>
```

The paragraph wraps at each carriage return, rather than at the usual word boundaries. If the ON character is not found within the width specified for the paragraph, the wrap occurs at a word space.

The following example converts the STRIP characters to spaces before wrapping on either a line feed <10> or a space (the default):

```
print &description (20,10) wrap 50 22 strip=/^@<13> on=<10>
```

WRAP can also be used to print reversed characters, for support of languages such as Hebrew. An R after the length and *max_lines* arguments causes the field to be reversed before the wrap takes place. In addition, the entire paragraph is right-justified within the length indicated.

```
! Reverse wrap, in 30 character field.  
print &comment (2,35) wrap 30 5 r  
print $notes (1,50) wrap 50 7 r
```

LINE-HEIGHT specifies the number of lines to skip between each line of the wrapped data. By default a value of 1 (single space) is assumed.

The following example prints the *comment* column with one blank line between each printed line for a maximum of four printed lines:

```
print &comment (1,1) wrap 40 4 line-height = 2! Double space  
text
```

See Also

- The LET command for information on copying, editing, or converting fields.
- The ALTER-LOCALE command for a description of the arguments NUMBER-EDIT-MASK, MONEY-EDIT-MASK, and DATE-EDIT-MASK.
- The DISPLAY and SHOW commands.

PRINT-BAR-CODE

Function	Prints bar codes.
Syntax	<pre>PRINT-BAR-CODE <i>position</i> {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={26}]</pre>
Arguments	<p><i>position</i></p> <p>Specifies the 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 bar code. (This is different than the way the PRINT command works.)</p> <p>TYPE</p> <p>Specifies the type of bar code to be printed. Types are shown in Table 2-48, “Bar Code Types,” on page 2-282.</p> <p>HEIGHT</p> <p>Specifies the 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.</p> <p>TEXT</p> <p>Specifies the text to be encoded and printed. The number and type of text characters permitted or required depends on the bar code type. See Table 2-48, “Bar Code Types,” on page 2-282 for specifications.</p> <p>CAPTION</p> <p>Specifies optional text to be printed under the bar code in the current font. SQR attempts to center the caption under the bar code; however, for proportional fonts this may vary slightly.</p> <p>CAPTION is not valid for Zip+4 Postnet. If specified, it is ignored.</p>

CHECKSUM

Specifies an optional check sum to be computed and printed within the bar code. Valid values are YES and NO, where NO is the default.



Note Some bar code types ignore the CHECKSUM qualier. See Table 2-48 for those bar code types for which CHECKSUM is relevant.

Description

PRINT-BAR-CODE prints industry standard bar codes. SQR supports the bar code types listed in Table 2-48.

Table 2-48 Bar Code Types

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

- * Text Type characters:
 - 9- Numbers (0-9)
 - X- Upper Case Letters (A-Z)
 - x- Lower Case Letters (a-z)
 - p- Punctuation
 - c- Control Characters

 **Note**

SQR 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 the PRINT-BAR-CODE command 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
```



0 12345 67890

This example shows how to use the PRINT-BAR-CODE command to create a ZIP+4 Postnet code.

```
begin-program
    print 'John Q. Public'      (3,1)
    print '1234 Main Street'   (4,1)
    print 'AnyTown, USA 12345-6789' (5,1)
    print-bar-code             (7,1)
        type=10
        height=0.2
        text='12345678934'
end-program
```

John Q. Public
1234 Main Street
AnyTown, USA 12345-6789



PRINT-CHART

Function Prints a chart. Only PostScript printers or HP printers that support HPGL (generally, this is HPLaserJet 3 and higher) render chart output.

Syntax

```
PRINT-CHART [chart_name]position  
[TYPE={chart_type_txt_lit|_var|_col}]  
[CHART-SIZE=(chart_width_num_lit|_var|_col,  
            chart_depth_num_lit|_var|_col)]  
[TITLE={title_txt_lit|_var|_col}]  
[SUB-TITLE={subtitle_txt_lit|_var|_col}]  
[FILL={fill_txt_lit|_var|_col}]  
[3D-EFFECTS={3d_effects_txt_lit|_var|_col}]  
[BORDER={border_txt_lit|_var|_col}]  
[COLOR-PALETTE=color_palette_lit|_var|_col]  
[POINT-MARKERS={point_markers_txt_lit|_var|_col}]  
[ATTRIBUTES={selector_lit|_var|_col}  
           LIST:{selector_list_name_lit|_var|_col|  
                  (selector_lit|_var|_col,...)}, {decl_key_lit|_var|  
                  _col, {decl_value_lit|_var|_col|  
                          LIST:{decl_val_list_name_lit|_var|_col|  
                                 (decl_val_lit|_var|_col,...)}|  
                          PALETTE:{color_palette_lit|_var|_col}}}, ...}]  
[DATA-ARRAY=array_name]  
[DATA-ARRAY-ROW-COUNT={x_num_lit|_var|_col}]  
[DATA-ARRAY-COLUMN-COUNT={x_num_lit|_var|_col}]  
[DATA-ARRAY-COLUMN-LABELS={NONE|array_name|({txt_lit|  
          _var|_col},...)}]  
[DATA-LABELS={data_labels_txt_lit|_var|_col}]  
[FOOTER-TEXT=NONE|text_lit|_var|_lit]  
[SUB-FOOTER-TEXT=NONE|text_lit|_var|_col]  
[ITEM-COLOR=(item_color_keyword|_lit|_var|_col,  
             {color_txt_lit_var|_col}|(r,g,b))]  
[ITEM-SIZE=(item_size_keyword_lit|_var|_col,  
            item_size_num_lit|_var|_col)]  
[LEGEND={legend_txt_lit|_var|_col}]  
[LEGEND-TITLE={legend_title_txt_lit|_var|_col}]  
[LEGEND-PLACEMENT={legend_placement_txt_lit|_var|_col}]  
[LEGEND-PRESENTATION={legend_presentation_txt_lit|  
                      _var|_col}]  
[PIE-SEGMENT-QUANTITY-DISPLAY={pie_segment_quantity_  
                                display_txt_lit|_var|_col}]
```

```
[PIE-SEGMENT-PERCENT_DISPLAY={pie_segment_percent_display_txt_lit|_var|_col}]
[PIE-SEGMENT-EXPLODE={pie_segment_explode_txt_lit|_var|_col}]
[X-AXIS-GRID={x_axis_grid_txt_lit|_var|_col}]
[X-AXIS-LABEL={x_axis_label_txt_lit|_var|_col}]
[X-AXIS-MIN-VALUE={x_axis_min_value_num_lit|_var|_col}]
[X-AXIS-MAX-VALUE={x_axis_max_value_num_lit|_var|_col}]
[X-AXIS-MAJOR_INCREMENT={x_axis_major_increment_num_lit|_var|_col}]
[X-AXIS-MINOR_INCREMENT={x_axis_minor_increment_num_lit|_var|_col}]
[X-AXIS-MAJOR-TICK_MARKS={x_axis_major_tick_marks_txt_lit|_var|_col}]
[X-AXIS-MINOR-TICK-MARKS={x_axis_minor_tick_marks_txt_lit|_var|_col}]
[X-AXIS-TICK-MARK-PLACEMENT={x_axis_tick_mark_placement_txt_lit|_var|_col}]
[X-AXIS-ROTATE={x_num_lit|_var|_col}]
[X-AXIS-SCALE={x_axis_scale_txt_lit|_var|_col}]
[Y-AXIS-GRID={y_axis_grid_txt_lit|_var|_col}]
[Y-AXIS-LABEL={y_axis_label_txt_lit|_var|_col}]
[Y-AXIS-MASK={mask_txt_lit|_var|_col}]
[Y-AXIS-MIN-VALUE={y_axis_min_value_num_lit|_var|_col}]
[Y-AXIS-MAX-VALUE={y_axis_max_value_num_lit|_var|_col}]
[Y-AXIS-MAJOR_INCREMENT={y_axis_major_increment_num_lit|_var|_col}]
[Y-AXIS-MINOR_INCREMENT={y_axis_minor_increment_num_lit|_var|_col}]
[Y-AXIS-MAJOR-TICK-MARKS={y_axis_major_tick_marks_txt_lit|_var|_col}]
[Y-AXIS-MINOR-TICK-MARKS={y_axis_minor_tick_marks_txt_lit|_var|_col}]
[Y-AXIS-TICK-MARK-PLACEMENT={y_axis_tick_mark_placement_txt_lit|_var|_col}]
[Y-AXIS-SCALE={y_axis_scale_txt_lit|_var|_col}]
[Y2-AXIS-LABEL={y2_axis_label_txt_lit|_var|_col}]
[Y2-AXIS-MASK={mask_txt_lit|_var|_col}]
[Y2-AXIS-MIN-VALUE={y2_axis_min_value_num_lit|_var|_col}]
[Y2-AXIS-MAX-VALUE={y2_axis_max_value_num_lit|_num_lit|_var|_col}]
[Y2-AXIS-MAJOR_INCREMENT={y2_axis_major_increment_num_lit|_var|_col}]
```

```
[Y2-AXIS-MINOR-INCREMENT={y2_axis_minor_increment  
_num_lit|_var|_col}]  
[Y2-AXIS-MAJOR-TICK-MARKS={y2_axis_major_tick_marks  
_txt_lit|_var|_col}]  
[Y2-AXIS-MINOR-TICK-MARKS={y2_axis_minor_tick_marks  
_txt_lit|_var|_col}]  
[Y2-AXIS-SCALE={y2_axis_scale_txt_lit|_var|_col}]  
[Y2-COLOR-PALETTE=color_palette_lit|_var|_col]  
[Y2-DATA-ARRAY=array_name]  
[Y2-DATA-ARRAY-ROW-COUNT={x_num_lit|_var|_col}]  
[Y2-DATA-ARRAY-COLUMN-COUNT={x_num_lit|_var|_col}]  
[Y2-DATA-ARRAY-COLUMN-LABELS={NONE|array_name|  
({txt_lit|_var|_col},...)}]  
[Y2-TYPE={chart_type_txt_lit|_var|_col}]
```

 **Note**

If you do not define CHART-SIZE with this command, you must define it with DECLARE-CHART.

Arguments

chart_name

Specifies the name of the chart from the DECLARE-CHART command. This name is not necessary if you specify the CHART-SIZE and all other pertinent attributes in the PRINT-CHART command.

position

(row, column) Specifies the 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 chart area. (This is different than the way the PRINT command works.)

 **Note**

For definitions of the other arguments in the PRINT-CHART command, see “DECLARE-CHART” on page 2-83. For information on NewGraphics, see NewGraphics under “[Default-Settings] Section” on page 5-4.

Description

The PRINT-CHART command directs SQR to output a chart according to the named chart, if any, and the overridden attributes, if any.

As you use the PRINT-CHART command, keep in mind the following points:

- All the arguments defined for DECLARE-CHART are valid for PRINT-CHART. (See Table 2-11 on page 2-86 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.
- The PRINT-CHART command directs SQR 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 the PRINT-CHART command has been executed.

For example, if you execute a PRINT-CHART with TITLE=\$ttl and \$ttl='Encouraging Results', and then change the value of \$ttl to 'Discouraging Results' immediately afterward, then the chart is printed with first value, 'Encouraging Results'.

- PRINT-CHART expects the DATA-ARRAY to be organized in a particular way. See Table 2-49 on page 2-289 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 SQR 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 mentioned earlier, each chart type meets a specific organizational requirement. Table 2-49 and Table 2-50 describe these requirements.

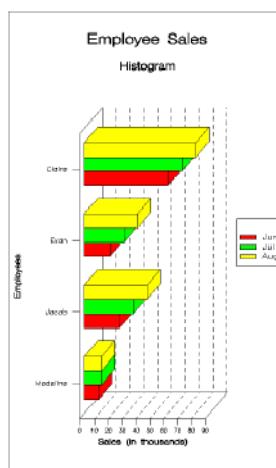
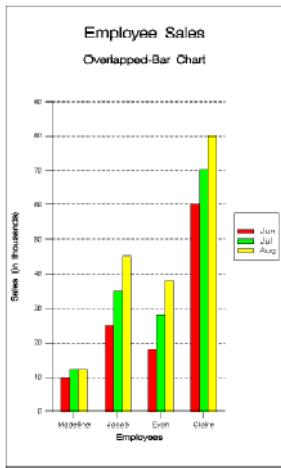
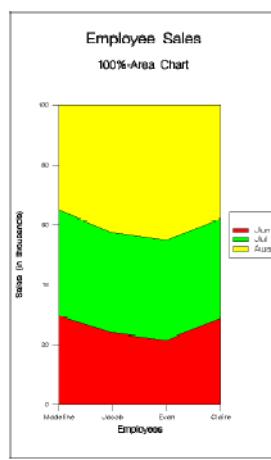
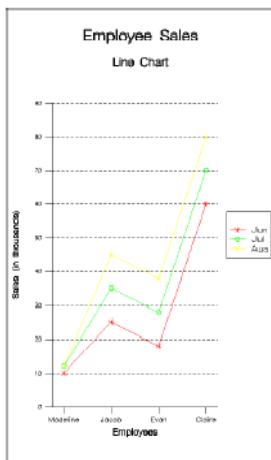
Table 2-49 Chart Array Field Types (fewer than four fields)

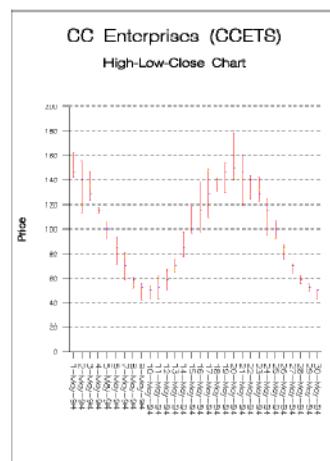
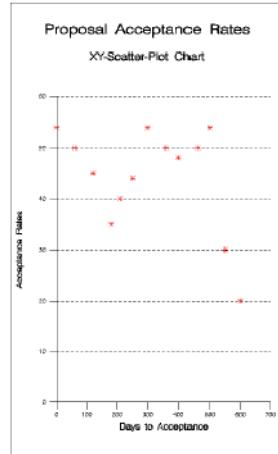
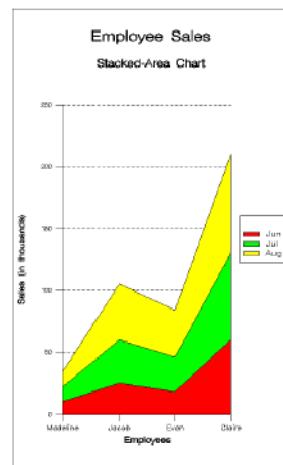
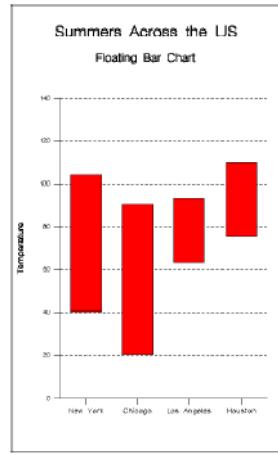
Chart Type	Field 0	Field 1	Field 2	Field 3
BUBBLE	Type=num X-Axis values	Type=num Y-Axis values	Type=num radius of bubble at (x,y)	
PIE	Type=char Pie segment labels, the names associated with each segment	Type=num The value associated with each pie segment	(Optional) Type=char Pie segment explode flag setting, 'Y' or 'N'	
LINE	Type=char	Type=num	(Optional)	(Optional)
BAR				
STACKED-BAR	X-Axis values	Series 1	Type=num Series 2	Type=num
100%-BAR				
OVERLAPPED-BAR		Y-Axis values	Y-Axis values	Series 3...
HISTOGRAM				Y-Axis values
AREA				
STACKED-AREA				
100%-AREA				
XY-SCATTER-PLOT	Type=num Series 1 X-Axis values	Type=num Series 1 Y-Axis values	(Optional) Type=num Series 2 X-Axis values Y-Axis values	(Optional) Type=num Series 2 ... Y-Axis values
FLOATING-BAR	Type=char X-Axis values	Type=num Series 1 Y-Axis offset	Type=num Series 1 Y-Axis duration	(Optional) Type=Num Series 2 ... Y-Axis offset

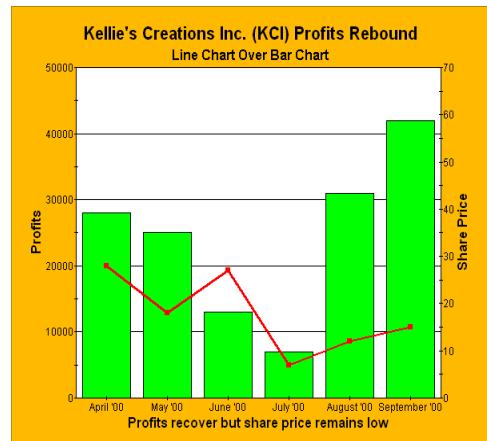
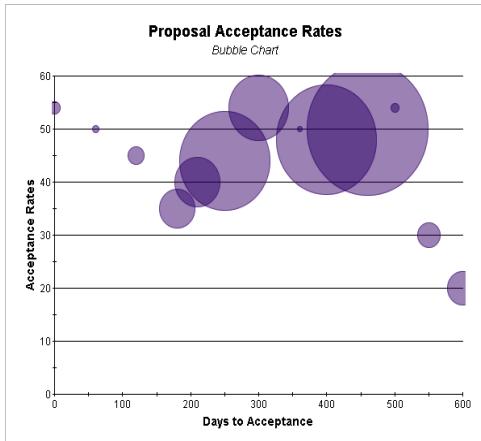
Table 2-50 Chart Array Field Types for HIGH-LOW-CLOSE

Chart Type	Field 0	Field 1	Field 2	Field 3	Field 4
HIGH-LOW-CLOSE	Type=char X-Axis values	Type=num High value	Type=num Low value	Type=num Closing value	(Optional) Type=num Opening value

Examples





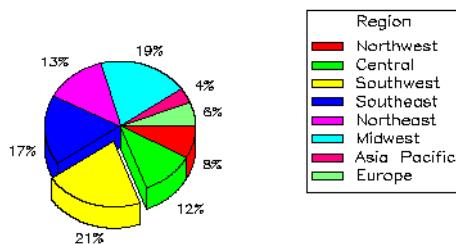


See “Creating Bubble Charts” on page 14-24 in *Hyperion SQR Getting Started Guide* for information on creating a Bubble Chart.

See “Use the ATTRIBUTES Argument in the DECLARE-CHART and PRINT-CHART Commands” on page 14-15 in *Hyperion SQR Getting Started Guide* for information on combination charts created using the Y2-Axis syntax in the DECLARE-CHART and PRINT-CHART commands.

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.  
.br/>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

The DECLARE-CHART command.

PRINT-DIRECT

Function Writes directly to the print output file without using the SQR page buffer.

Syntax

```
PRINT-DIRECT  
[NOLF]  
[PRINTER={LINEPRINTER | POSTSCRIPT | HPLASERJET | HTML  
| LP | PS | HP | HT} ]  
{txt_lit|_var|_col} ...
```

Arguments

NOLF

Specifies that no carriage return and line feed is to be printed. By default, printed text is followed by a carriage return and line feed character.

PRINTER

Specifies the type of printer to which this text applies.

txt_lit|_var|_col

The text to be printed.

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 in conjunction with PRINT commands, be aware that the SQR 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

Specifies the name of an image specified by a DECLARE-IMAGE.

position

(row, column) Specifies the 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 the PRINT command works.)

TYPE

Specifies the image type. Types can be EPS-FILE, HPGL-FILE, GIF-FILE, JPEG-FILE, BMP-FILE., or PNG-FILE.

IMAGE-SIZE

Specifies the width and height of the image in SQR coordinates.

SOURCE

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

FOR-PRINTER

Specifies a 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 the DECLARE-IMAGE command.)

Description

The PRINT-IMAGE command can be placed in any section of a report with the exception of 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 2-17, “Valid Images Types,” on page 2-113 illustrates the valid relationships between image type and report output type.

Examples

For PostScript:

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

For Windows:

```
print-image company-logo (+21, 25)
  type=bmp-file
  source='m:\logos\gustavs.bmp'
  image-size=(75,50)
```



 **Note**

For an example of the FOR-PRINTER argument used with DECLARE-IMAGE and PRINT-IMAGE, see the example under DECLARE-IMAGE on page 2-114.

See Also

- The DECLARE-IMAGE command.
- “Adding Graphics” on page 13-4 in the *Hyperion SQR Getting Started Guide*.

PUT

Function	Moves data into an array.
Syntax	<pre>PUT {src_any_lit _var _col}... INTO dst_array_name(element) [field[(occurs)]]...</pre>
Arguments	<p><i>src_any_lit _var _col</i></p> <p>The source variable or literal to be moved 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 <i>char</i>, <i>text</i>, or <i>date</i> fields. Date variables can be put into <i>date</i>, <i>char</i>, or <i>text</i> fields.</p> <p>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:</p> <ul style="list-style-type: none">■ For DATETIME columns and SQR DATE variables, SQR uses the format specified by the <code>SQR_DB_DATE_FORMAT</code> setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.■ For DATE columns, SQR uses the format specified by the <code>SQR_DB_DATE_ONLY_FORMAT</code> setting. If is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.■ For TIME columns, SQR uses the format specified by the <code>SQR_DB_TIME_ONLY_FORMAT</code> setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270. <p>When a string variable, column, or literal is moved to a date array field, the string must be in the format specified by the <code>SQR_DB_DATE_FORMAT</code> setting, one of the database-dependent formats as listed in Table 2-45, “Default Formats by Database,” on page 2-270, or the database-independent format '<code>'YYYYMMDD [HH24 [MI [SS [NNNNNN]]]]'</code>'.</p>

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 the CREATE-ARRAY command; data is copied into occurrence zero of each field of the element specified in the array.

field[(occurs)]

Array element and field occurrence numbers can be numeric literals (123) or numeric variables (#j).

If no occurrence is specified, occurrence zero is used.

Description

Columns retrieved from the database and SQR variables or literals can be moved into an array. The array must have been created previously using the CREATE-ARRAY command.

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	<pre>READ {filenum_lit _var _col} INTO {any_var:length_int_lit _var _col}... [STATUS=status_num_var]</pre>
Arguments	<p><i>filenum_lit _var _col</i> Specifies the number assigned in the OPEN command to the file to be read.</p> <p><i>any_var:length_int_lit _var _col</i> Specifies one or more variables into which data from the record read are to be put. <i>length_int_lit _var _col</i> specifies the length of each field of data.</p> <p>STATUS Specifies an optional variable into which a read status is returned.</p>
Description	<p>Text and binary data is parsed according to the following criteria:</p> <ul style="list-style-type: none">■ 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 the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats as listed in Table 2-45 on page 2-270 or the database-independent format 'YYYYMMDD [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, SQR 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 `strtotime` function.

```
.
.
.
declare-variable
  date $date1 $date2
  text $text
end-declare
.
.
.
read 4 into $date1:18 $text1:18
let $date2 = strtotime($text1,'SYYYYMMDDHHMISSNNN')
      or
let $date2 = strtotime($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

The OPEN, CLOSE, and WRITE commands for more information on files.

ROLLBACK

Function Causes a database rollback to the last commit.

Syntax ROLLBACK

Description An automatic rollback is performed whenever SQR aborts due to program errors. ROLLBACK is useful in testing or in certain error conditions.

ROLLBACK is an SQR command and should not be used inside an SQL paragraph.



Note The ROLLBACK command can be used with DB2, ODBC, DDO, Teradata, and Oracle. For Sybase, use BEGIN TRANSACTION and ROLLBACK TRANSACTION within SQL paragraphs as in the following example. See the COMMIT command for an example of ROLLBACK.

Examples

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

See Also The COMMIT command.

SBTOMBS

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

Syntax SBTOMBS {*txt_var*}

Arguments *txt_var*
Specifies the string to be converted.

Description This command 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. This command 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 the LET command.

SECURITY

Function	Marks sections of a report for security purposes.
Syntax	<pre>SECURITY [SET=(<i>sid</i> [,<i>sid</i>]...)] [APPEND=(<i>sid</i> [,<i>sid</i>]...)] [REMOVE=(<i>sid</i> [,<i>sid</i>]...)] [MODE=<i>mode</i>]</pre>
Arguments	SET Sets the 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. <i>sid</i> Can be 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 Used to turn on (reactivate) or turn off (suspend) the security feature for the current report. This argument is optional and can only be used once. <i>mode</i> Can be any string literal, column, or variable. The value is <i>not</i> case sensitive and can be either ON or OFF.

Description

The SECURITY command can be repeated as many times as desired for the current report. After the SECURITY command 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

The SECURITY command is useful only when used in conjunction with the Hyperion Performance Suite. The Security IDs refer to Hyperion Performance Suite 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 the SECURITY command wherever you use the PRINT command.

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} ) ]
```

Arguments

PRINT-TEXT-FOREGROUND
Defines the color in which the text is printed.

PRINT-TEXT-BACKGROUND
Defines the color to print as a background behind the text.

PRINT-PAGE-BACKGROUND
Defines the color to print as a background on the page.

{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 SQR 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

The SET-COLOR command is allowed wherever the PRINT command is allowed. If the specified color name is not defined, SQR uses the settings for the color name 'default.' Use the color name 'none' to turn off color for the specified area.

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

The DECLARE-COLOR-MAP, ALTER-COLOR-MAP, and GET-COLOR commands.

SET-DELAY-PRINT

Function	Sets the values of a DELAY variable.
Syntax	SET-DELAY-PRINT <i>delay_var</i> WITH { <i>src_lit _var _col</i> }
Arguments	<i>delay_var</i> Specifies the delay variable to be affected. { <i>src_lit _var _col</i> } Specifies the source variable.
Description	Replaces each reference of <i>delay_var</i> 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 PRINT command with the DELAY parameter.

SHOW

Function Displays one or more variables or literals on the screen. In addition, 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
Specifies the position on the screen 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
Specifies the information to be displayed.

EDIT
Shows variables under an edit mask. If the mask contains spaces, enclose it in single quotes. For additional information regarding edit masks, see the PRINT command.

NUMBER
Indicates that *any_lit|_var|_col* is to be formatted using the NUMBER-EDIT-MASK from the current locale. (See the ALTER-LOCALE command.) This option is not legal for date variables.

MONEY
Indicates that *any_lit|_var|_col* is to be formatted using the MONEY-EDIT-MASK from the current locale. (See the ALTER-LOCALE command.) This option is not legal for date variables.

DATE

Indicates that *any_lit|var|col* is to be formatted using the DATE-EDIT-MASK from the current locale. (See the ALTER-LOCALE command.) This option is not legal for numeric variables. If DATE-EDIT-MASK has not been specified, then the date is displayed using the default format for that database (see Table 2-45 on page 2-270).

BOLD, BLINK, UNDERLINE, and REVERSE

Changes the display of characters on terminals that support those characteristics. 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/or 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 a SHOW command without any cursor position when you want to scroll. Also, you cannot mix SHOW and DISPLAY commands while referencing relative cursor positions.

The SHOW command does not advance to the next line if a cursor location (...), CLEAR-SCREEN, CLEAR-LINE, or BEEP is used. (A SHOW command without any of these arguments automatically advances the line.) To add a line advance, add (+1,1) to the end of the line or use an extra empty SHOW command.

Only ANSI terminals are supported for cursor control, screen blanking, line blanking, and display characteristics.

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

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

When displaying a date in a string literal, column, or variable using EDIT or DATE, the string must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats as listed in Table 2-45 on page 2-270, or the database-independent format 'SYYYYYMMDD [HH24 [MI [SS [NNNNNN]]]].

Examples

The following program segments illustrate the various features of the SHOW command:

The following code:

```
!
! Show a string using an edit mask
!
let $ssn = '123456789'
show $ssn edit xxx-xx-xxxx
```

Produces the following output:

123-45-6789

The following code:

```
!
! Show a number using an edit mask
!
show 1234567.89 edit 999,999,999.99
```

Produces the following output:

1,234,567.89

The following code:

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

Produces the following output:

```
123.780000
```

The following code:

```
!
! Show a number using the locale default numeric edit mask
!
alter-locale number-edit-mask = '99,999,999.99'
show 123456.78 number
```

Produces the following output:

```
123,456.78
```

The following code:

```
!
! Show a number using the locale default money edit mask
!
alter-locale money-edit-mask = '$$, $$., $$8.99'
show 123456.78 money
```

Produces the following output:

```
$123,456.78
```

The following code:

```
!
! Show a date column using the locale default date edit mask
!
begin-select
dcol
  from tables
end-select
alter-locale date-edit-mask = 'DD-Mon-YYYY'
show &dcol date
```

Produces the following output:

```
01-Jan-1999
```

The following code:

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

Produces the following output:

```
Hello World
```

The following code:

```
!
! Show two values on the same line with editing of the values
!
let #taxes = 123456.78
show 'You owe ' #taxes money ' in back taxes.'
```

Produces the following output:

```
You owe $123,456.78 in back taxes.
```

The following program illustrates the usage of additional options of the SHOW command. Only terminals that support the ANSI escape characters can use the cursor control, screen blanking, line blanking and display attributes.

```
begin-program
!
! Produces a menu for the user to select from
!
show clear-screen
    (3,30) bold 'Accounting Reports for XYZ Company' normal
    (+2,10) '1. Monthly Details of Accounts'
    (+1,10) '2. Monthly Summary'
    (+1,10) '3. Quarterly Details of Accounts'
    (+1,10) '4. Quarterly Summary'
!
! Show a line of text and numerics combined
!
show (+2,1)
    'The price is ' #price edit 999.99
        '      Total = ' #total edit 99999.99
!
! Put an error message on a particular line
!
show (24,1) clear-line 'Error in SQL. Please try again.' beep
end-program
```

See Also

- The `LET` command for information on copying, editing, or converting fields.
- The `EDIT` parameter of the `PRINT` command for a description of the edit masks.
- The `ALTER-LOCALE` command for a description of the arguments `NUMBER-EDIT-MASK`, `MONEY-EDIT-MASK`, and `DATE-EDIT-MASK`.
- The `DISPLAY` command.

STOP

Function	Halts SQR.
Syntax	STOP [QUIET]
Arguments	QUIET Causes the report to complete with the “SQR: End Of Run” message, instead of aborting with an error message.
Description	The STOP command halts SQR 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	<pre>if #error-status = 1 rollback stop else commit stop quiet end-if</pre>

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
Specifies one or more fields to be concatenated, 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 SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

delim_txt_lit|_var|_col
Specifies one or more characters to be used as separator characters between the source fields.

dst_txt_var
Specifies the destination field for the concatenated result.

Description The destination string must not be included 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

- The UNSTRING command for additional information.
- The “| |” concatenation operator in Table 2-29 on page 2-195 under the LET command.

SUBTRACT

Function	Subtracts one value from another.
Syntax	<code>SUBTRACT {src_num_lit _var _col} FROM dst_num_var[ROUND=nn]</code>
Arguments	<p><i>src_num_lit _var _col</i> Is subtracted from the contents of <i>dst_num_var</i>.</p> <p><i>dst_num_var</i> Contains the result after execution.</p> <p>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.</p>
Description	<p>Subtracts the first value from the second and moves the result into the second field.</p> <p>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.</p>
Examples	<pre>subtract 1 from #total ! #total - 1 subtract &discount from #price ! #price - &discount</pre>
See Also	<ul style="list-style-type: none">■ The ADD command for more information.■ The LET command for information on complex arithmetic expressions.

TOC-ENTRY

Function Places an entry into the Table of Contents.

Syntax TOC-ENTRY

TEXT={*src_txt_lit|_var|_col*}
[LEVEL={*level_num_lit|_var|_col*}]

Arguments

TEXT
Specifies the text to be placed in the Table of Contents.

LEVEL
Specifies the 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 The DECLARE-TOC command.

UNSTRING

Function	Copies portions of a string into one or more text variables.
Syntax	<pre>UNSTRING {{src_txt_lit _var _col} {src_date_var _col}} BY {delim_txt_lit _var _col} INTO dst_txt_var...</pre>
Arguments	<p><i>{src_txt_lit _var _col} {src_date_var _col}</i></p> <p>Specifies the source field to be parsed.</p> <p><i>delim_txt_lit _var _col</i></p> <p>Specifies one or more characters to be used to delimit the fields within <i>{src_txt_lit _var _col} {src_date_var _col}</i></p> <p><i>dst_txt_var</i></p> <p>Specifies one or more destination fields to receive the results.</p>
Description	<p>Each substring is located using the specified delimiter. The source string must not be included in the list of destination strings.</p> <p>If more destination strings than substrings are found in the source strings, the extra destination strings are each set to an empty string.</p> <p>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.</p>

If the source is a date variable or column, it is converted to a string according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format as listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed Table 2-47, “TIME Column Formats,” on page 2-270.

Examples

```
unstring $show-info by ' - ' into $name $city $state $zip  
unstring $cust-data by ',' into $cust_num $entry-date $total
```

See Also

- The STRING and EXTRACT commands.
- The substr and instr functions in Table 2-36, “Miscellaneous Functions,” on page 2-220 under the LET command.

UPPERCASE

Function	Converts a string variable to uppercase.
Syntax	<code>UPPERCASE <i>txt_var</i></code>
Arguments	<i>txt_var</i> Specifies the field to be converted to uppercase.
Examples	<pre>input \$state 'Enter state abbreviation' uppercase \$state ! Force uppercase.</pre>
See Also	The <code>upper</code> function in Table 2-36, “Miscellaneous Functions,” on page 2-220 under the <code>LET</code> command.

USE

Function	Uses the named database, rather than the default database associated with your user name. (Sybase and ODBC only)
Syntax	<code>USE <i>database</i></code>
Arguments	<code><i>database</i></code> Specifies 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:
	<pre>from sqdb.sqr.customers</pre>
	You cannot issue the Sybase or ODBC USE command from within an SQL paragraph.
Examples	<pre>begin-setup use pubs end-setup</pre>
See Also	The -DB command-line flag described in “SQR Command-line Flags” on page 1-15.

USE-COLUMN

Function	Sets the current column.
Syntax	USE-COLUMN { <i>column_number_int_lit _var _col</i> }
Arguments	<i>column_number_int_lit _var _col</i> Specifies the number of the defined column (not the location on the page). For example, if five columns are defined, then the <i>column_number_int_lit _var _col</i> can be 1 to 5.
Description	The column must be previously defined with the COLUMNS command. 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	<pre>use-column 3 ! Print total in 3rd column. print #total () 999,999 use-column 0 ! End of column printing.</pre>

USE-PRINTER-TYPE

Function	Sets the printer type to be used for the current report.
Syntax	USE-PRINTER-TYPE <i>printer-type</i>
Arguments	<i>printer-type</i> Specifies the printer type to be used for the current report. See DECLARE-PRINTER for valid types.
Description	The USE-PRINTER-TYPE command sets or alters the printer type used for the current report. The USE-PRINTER-TYPE command must appear before the first output is written to that report. If output has already been written to the report file, the USE-PRINTER-TYPE command is ignored.
Examples	<pre>use-report customer_orders use-printer-type PostScript print (1, 1) 'Customer Name: ' print () \$customer_name</pre>
See Also	The DECLARE-PRINTER, DECLARE-REPORT, and USE-REPORT commands.

USE-PROCEDURE

Function	Changes the procedure usage.
Syntax	<pre>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]...)]]</pre>
Arguments	<p>FOR-REPORTS Specifies the reports that are to use these 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.</p> <p>BEFORE-REPORT Specifies a procedure to execute at the time of execution of the first command, which causes output to be generated. You can use the command, for example, to create a report heading.</p> <p>AFTER-REPORT Specifies a 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.</p> <p>BEFORE-PAGE Specifies a 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.</p> <p>AFTER-PAGE Specifies a procedure to execute just before each page is written to the file. This argument can be used, for example, to display page totals.</p> <p>You can also specify arguments to be passed to the procedure. Arguments can be any variable, column, or literal.</p>

Description

The USE-PROCEDURE must be issued in the PROGRAM or PROCEDURE sections of an SQR 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 the BEFORE-REPORT only before the first output is written to that report, since that causes the BEFORE-REPORT procedure to be executed.

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
before-page=page_setup   ! report
after-page=page_totals
use-procedure           ! The after-report
for-reports=(summary)   ! procedure will be
after-report=none        ! disabled for the
                         ! summary report
```

See Also

The DECLARE-PROCEDURE command.

USE-REPORT

Function	For programs with multiple reports, allows the user to switch between reports.
Syntax	USE-REPORT { <i>report_name_lit _var _col</i> }
Arguments	<i>report_name_lit _var _col</i> Specifies the report to become the “ <i>current</i> ” report. All subsequent PRINT and PRINT-DIRECT statements are written to this report until the next USE-REPORT is encountered.
Description	The USE-REPORT command specifies 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	<pre>use-report customer_orders use-printer-type PostScript print (1, 1) 'Customer Name: ' print () \$customer_name</pre>
See Also	The DECLARE-REPORT, DECLARE-LAYOUT, DECLARE-PRINTER, and USE-PRINTER-TYPE commands.

WHILE

Function Begins a WHILE ... END-WHILE loop.

Syntax `WHILE logical_expression`

The general format of a WHILE command is as follows:

```
WHILE logical_expression  
SQR_commands...  
[BREAK]  
[CONTINUE]  
SQR_commands...  
END-WHILE
```

Arguments `logical_expression`

A valid logical expression. See the LET command for a description of logical expressions.

Operators See “Bit-Wise Operators” on page 2-196 for information on the bit-wise operators supported by the WHILE command.

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; SQR continues with the command immediately following END-WHILE.

CONTINUE ends the current iteration of a loop. Program control passes from the CONTINUE parameter to the end of the WHILE loop body.

WHILE commands can be nested to any level and can include or be included within IF and EVALUATE commands.

Examples

The following example shows an IF nested within a WHILE:

```
while #count < 50
    do get_statistics
    if #stat_count = 100
        break      ! Exit WHILE loop.
    end-if
    add 1 to #count
end-while
```

You can use single numeric variables in your expression to make your program more readable, for example when using flags.

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

The following example sets up an infinite loop:

```
while 1
    ...processing...
    if ...
        break      ! Exit loop
    end-if
end-while
```

You can use any complex expression in the WHILE command as shown in the following example:

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

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

```
while #count < 50
    if #count = 10
        continue
    end-if
    do get-statistics(#count)
    add 1 to #count
end-while
```

See Also

The LET command for a description of expressions.

WRITE

Function Writes a record to a file from data stored in variables, columns, or literals.

Syntax

```
WRITE {filenum_lit|_var|_col} FROM  
{{txt_lit|_var|_col}|{date_var|_col}|num_col}  
[:len_int_lit|_var|_col}]|{num_lit|_var:len_int_lit|  
_var|_col}}...  
[STATUS=status_num_var]
```

Arguments

filenum_lit|_var|_col

Specifies the number assigned in the OPEN command to the file to be written.

```
{txt_lit|_var|_col}|{date_var|_col}|num_col}  
[:len_int_lit|_var|_col}]|{num_lit|_var:len_int_lit|_var|  
_col}
```

Specifies one or more variables to be written. *len_int_lit|_var|_col* specifies the length of each field of data.

STATUS

Specifies 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 the WRITE command. 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 SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this is not set, SQR uses the first database-dependent format listed in Table 2-45, “Default Formats by Database,” on page 2-270.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-46, “DATE Column Formats,” on page 2-270.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this is not set, SQR uses the format listed in Table 2-47, “TIME Column Formats,” on page 2-270.

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 the WRITE command.

If STATUS is specified, SQR returns 0 if the write is successful; otherwise, it returns the value of *errno*, which is system-dependent.

Examples

```
write 5 from $name:20 $city:15 $state:2
write 17 from $company ' - ' $city ' - ' $state ' ' $zip
write #j2 from #rate:2 #amount:4 #quantity:1
move #total to $tot 99999.99      ! Convert floating point to
                                ! string.
write 1 from $tot
let $date1 = datenow()           ! Put the current date and time
                                ! into DATE variable
write 3 from $date1:20
```

See Also

The OPEN, CLOSE, and READ commands.

3

HTML Procedures

This chapter describes the procedures that enable SQR to generate HTML output. You can publish the output on an Internet, Intranet, or Extranet Web site.

An SQR program without HTML procedures has limited HTML capabilities. Adding HTML procedures to the SQR program enhances the appearance of the HTML output.

For information on using these procedures, see Chapter 31, “Working with HTML” in the *Hyperion SQR Getting Started Guide*.

In This Chapter	HTML General Purpose Procedures	3-2
	HTML Heading Procedures	3-5
	HTML Highlighting Procedures	3-7
	HTML Hypertext Link Procedures	3-10
	HTML List Procedures	3-11
	HTML Table Procedures	3-15

HTML General Purpose Procedures

Table 3-1 lists the HTML general purpose procedures.

Table 3-1 HTML General Purpose Procedures

Procedure	Description
html_br	<p>Produces the specified number of line breaks in a paragraph using the HTML
 tag. This causes the paragraph to continue onto the next line.</p> <p>Syntax: html_br(<i>number count, string attributes</i>)</p> <ul style="list-style-type: none">■ <i>count</i> = The number of
 tags.■ <i>attributes</i> = The HTML attributes inside the HTML
 tag. <p>Example: Producing a line break:</p> <pre>print 'Here is some text' () do html_br(3, '') print 'Here is some three lines down' ()</pre>
html_center	<p>Marks the start of text centered in the HTML document using the HTML <CENTER> tag. (You can also use the SQR PRINT command and specify CENTER in the code.)</p> <p>Syntax: html_center(<i>string attributes</i>)</p> <ul style="list-style-type: none">■ <i>attributes</i> = The HTML attributes inside the <CENTER> tag. <p>Example: Centering text using the <CENTER> tag:</p> <pre>do html_center('') print 'Here is some text' () do html_center_end</pre>
html_center_end	<p>Marks the end of text previously specified to be centered.</p> <p>Syntax: html_center_end</p>
html_hr	<p>Produces a horizontal divider between sections of text using the HTML <HR> tag.</p> <p>Syntax: html_hr(<i>string attributes</i>)</p> <ul style="list-style-type: none">■ <i>attributes</i> = The HTML attributes inside the <HR> tag. <p>Example: Producing a horizontal divider:</p> <pre>print 'Here is some text' () do html_hr('') print 'And some more text' ()</pre>

Table 3-1 HTML General Purpose Procedures (*Continued*)

Procedure	Description
html_img	<p>Inserts an image using the HTML tag. (You can also use the PRINT-IMAGE command; however, the HTML tag allows you to specify the full set of available HTML attributes.)</p> <p>Syntax: html_img(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the tag.</p> <p>Some common attributes:</p> <ul style="list-style-type: none">src - URL of the image to be inserted. (Example: src=/images/abc.gif)height - Height of the image in pixels. (Example: height=200)width - Width of the image in pixels. (Example: width=400) <p>Example: Producing an image:</p> <pre>do html_img('src="/images/stop.gif'')</pre>
html_nobr	<p>Marks the start of text that cannot be wrapped with the HTML <NOBR> tag.</p> <p>Syntax: html_nobr</p> <p>Example: Preventing line wrapping:</p> <pre>do html_nobr('') print 'Here's long text that should not wrap'() do html_nobr_end</pre>
html_nobr_end	<p>Marks the end of text that cannot be wrapped with the HTML </NOBR> tag.</p> <p>Syntax: html_nobr_end</p>
html_on	<p>Turns on the HTML procedures. Call this procedure at the start of an SQR program; otherwise, the HTML procedures do not turn on. After the HTML procedures turn on, the appearance of the Web page must be formatted using the various HTML procedures.</p> <p>Syntax: html_on</p> <p>Example: Turning on the HTML procedures</p> <pre>do html_on</pre>

Table 3-1 HTML General Purpose Procedures (*Continued*)

Procedure	Description
html_p	<p>Marks the start of a new paragraph using the HTML <P> tag.</p> <p>Syntax: html_p(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <P> tag.</p> <p>Some common attributes: align = left right center - Specifies the alignment of the paragraph.</p> <p>Example: Displaying one paragraph with right-aligned text, then another paragraph with normal text:</p> <pre>do html_p('ALIGN=RIGHT') print 'Right aligned text' (1,1) do html_p_end print 'Normal text' (+1,1)</pre>
html_p_end	<p>Marks the end of a paragraph using the HTML </P> tag. The end of a paragraph is typically implied and not needed; however, it is a good idea to specify it for completeness.</p> <p>Syntax: html_p_end</p>
html_set_body_attributes	<p>Specifies the attributes inside the HTML <BODY> tag. This must be called at the start of the SQR program.</p> <p>Syntax: html_set_body_attributes(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <BODY> tag.</p> <p>Some common attributes: background - Specifies the image to display on the background of the Web page. (Example: background=/images/logo.gif) bgcolor=#rrggbb - Specifies the background color of the Web page. (Example: bgcolor=#80FFF)</p> <p>Example: Using the BACKGROUND attribute to display a marble.gif image in the background of the Web page:</p> <pre>do html_set_body_attributes('BACKGROUND="/images/x.gif"')</pre>
html_set_head_tags	<p>Specifies the tags between the <HEAD> and </HEAD> HTML tags. By default these are empty. One common tag to set is <TITLE>, which specifies the title to display for the Web page. This must be called at the start of the SQR program.</p> <p>Syntax: html_set_head_tags(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes between the <HEAD> and </HEAD> tags.</p> <p>Example: Displaying the title <i>My Report</i> for the Web page:</p> <pre>do html_set_head_tags('<TITLE>My Report</TITLE>')</pre>

HTML Heading Procedures

Table 3-2 lists the HTML heading procedures.

Table 3-2 HTML Heading Procedures

Procedure	Description
html_h1	<p>Marks the start of heading level one text using the HTML <H1> tag. Text under this heading displays more prominently than that of heading level two.</p> <p>Syntax: html_h1(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <H1> tag.</p> <p>Example: Displaying text as a heading level one:</p> <pre>do html_h1('') print 'This is a heading' () do html_h1_end</pre>
html_h1_end	<p>Marks the end of heading level one text using the HTML </H1> tag.</p> <p>Syntax: html_h1_end</p>
html_h2	<p>Marks the start of heading level two text using the HTML <H2> tag. Text under this heading displays more prominently than that of heading level 3.</p> <p>Syntax: html_h2(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <H2> tag.</p> <p>Example: Displaying text as a heading level two:</p> <pre>do html_h2('') print 'This is a heading' () do html_h2_end</pre>
html_h2_end	<p>Marks the end of heading level two text using the HTML </H2> tag.</p> <p>Syntax: html_h2_end</p>
html_h3	<p>Marks the start of heading level three text using the HTML <H3> tag. Text under this heading displays less prominently than that of heading level two and more prominently than that of heading level four.</p> <p>Note: This heading level is the default.</p> <p>Syntax: html_h3(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <H3> tag.</p>

Table 3-2 HTML Heading Procedures (*Continued*)

Procedure	Description
html_h3_end	Marks the end of heading level three text using the HTML </H3> tag. Syntax: html_h3_end
html_h4	Marks the start of heading level four text using the HTML <H4> tag. Text under this heading displays less prominently than that of heading level three and more prominently than that of heading level five. Syntax: html_h4(<i>string attributes</i>) ■ <i>attributes</i> = The HTML attributes inside the HTML <H4> tag.
html_h4_end	Marks the end of heading level four text using the HTML </H4> tag. Syntax: html_h4_end
html_h5	Marks the start of heading level five text using the HTML <H5> tag. Text under this heading displays less prominently than that of heading level four and more prominently than that of heading level six. Syntax: html_h5(<i>string attributes</i>) ■ <i>attributes</i> = The HTML attributes inside the <H5> tag.
html_h5_end	Marks the end of heading level five text using the HTML </H5> tag. Syntax: html_h5_end
html_h6	Marks the start of heading level six text using the HTML <H6> tag. Text under this heading displays less prominently than that of heading level five and more prominently than that of heading level seven. Syntax: html_h6(<i>string attributes</i>) ■ <i>attributes</i> = The HTML attributes inside the <H6> tag.
html_h6_end	Marks the end of heading level six text using the HTML </H6> tag. Syntax: html_h6_end

HTML Highlighting Procedures

Table 3-3 lists the HTML highlighting procedures.

Table 3-3 HTML Highlighting Procedures

Procedure	Description
html_blink	<p>Marks the start of blinking style text using the HTML <BLINK> tag.</p> <p>Syntax: html_blink(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <BLINK> tag.</p> <p>Example: Displaying text with the address style:</p> <pre>do html_blink('') print 'This is blinking' () do html_blink_end</pre>
html_blink_end	<p>Marks the end of blinking style using the HTML </BLINK> tag.</p> <p>Syntax: html_blink_end</p>
html_cite	<p>Marks the start of citation style text using the HTML <CITE> tag.</p> <p>Syntax: html_cite(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <CITE> tag.</p> <p>Example: Displaying text with the citation style:</p> <pre>do html_cite('') print 'This is a citation' () do html_cite_end</pre>
html_cite_end	<p>Marks the end of citation style text using the HTML </CITE> tag.</p> <p>Syntax: html_cite_end</p>

Table 3-3 HTML Highlighting Procedures (*Continued*)

Procedure	Description
html_code	<p>Marks the start of code style text using the HTML <CODE> tag.</p> <p>Syntax: html_code(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <CODE> tag.</p> <p>Example: Displaying text with the code style:</p> <pre>do html_code('') print 'Here is the code' () do html_code_end</pre>
html_code_end	<p>Marks the end of code style text using the HTML </CODE> tag.</p> <p>Syntax: html_code_end</p>
html_kbd	<p>Marks the start of keyboard input style text using the HTML <KBD> tag.</p> <p>Syntax: html_kbd(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <KBD> tag.</p> <p>Example: Displaying text with the keyboard style:</p> <pre>do html_kbd('') print 'Here is keyboard' () do html_kbd_end</pre>
html_kbd_end	<p>Marks the end of keyboard style text using the HTML </KBD> tag.</p> <p>Syntax: html_kbd_end</p>
html_samp	<p>Marks the start of sample style text using the HTML <SAMP> tag.</p> <p>Syntax: html_samp(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <SAMP> tag.</p> <p>Example: Displaying text with the sample style:</p> <pre>do html_samp('') print 'Here is sample' () do html_samp_end</pre>
html_samp_end	<p>Marks the end of sample style text using the HTML </SAMP> tag.</p> <p>Syntax: html_samp_end</p>

Table 3-3 HTML Highlighting Procedures (*Continued*)

Procedure	Description
html_strike	<p>Marks the start of strike-through style text using the HTML <STRIKE> tag.</p> <p>Syntax: html_strike(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <STRIKE> tag.</p> <p>Example: Displaying text with the strike-through style:</p> <pre>do html_strike('') print 'Here is strike-through' () do html_strike_end</pre>
html_strike_end	<p>Marks the end of strike-through style text using the HTML </STRIKE> tag.</p> <p>Syntax: html_strike_end</p>
html_sub	<p>Marks the start of subscript style text using the HTML <SUB> tag.</p> <p>Syntax: html_sub(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <SUB> tag.</p> <p>Example: Displaying text with the subscript style:</p> <pre>print 'Here is' () do html_sub('') print 'subscript text' () do html_sub_end</pre>
html_sub_end	<p>Marks the end of subscript style text using the HTML </SUB> tag.</p> <p>Syntax: html_sub_end</p>
html_sup	<p>Marks the start of superscript style text using the HTML <SUP> tag.</p> <p>Syntax: html_sup(<i>string attributes</i>)</p> <p>■ <i>attributes</i> = The HTML attributes inside the <SUP> tag.</p> <p>Example: Displaying text with the superscript style:</p> <pre>print 'Here is' () do html_sup('') print 'superscript text' () do html_sup_end</pre>
html_sup_end	<p>Marks the end of superscript style text using the HTML </SUP> tag.</p> <p>Syntax: html_sup_end</p>

HTML Hypertext Link Procedures

Table 3-4 lists the HTML hypertext link procedures.

Table 3-4 HTML Hypertext Link Procedures

Procedure	Description
html_a	<p>Marks the start of a hypertext link using the HTML <A> tag. When you click an area with a hypertext link, the Web browser switches to the specified HTML document.</p> <p>Syntax: html_a(<i>string attributes</i>)</p> <ul style="list-style-type: none">■ <i>attributes</i> = The HTML attributes inside the HTML <A> tag. At a minimum, you should define the HREF attribute, which specifies the URL of an HTML document. <p>Some common attributes:</p> <ul style="list-style-type: none">href – Where the hypertext link points. (Example: href=home.html)name – An anchor to which a hypertext link can point. (Example: name=marker1) <p>Example: Creating an anchor with two hypertext links. The anchor is positioned at the top of the document. The first hypertext link points to the HTML document <i>otherdoc.html</i>. The second hypertext link points to the anchor named <i>TOP</i>.</p> <pre>do html_a('NAME=TOP') do html_a_end print 'At the top of document' () do html_br(20, '') do html_a('HREF=otherdoc.html') print 'Goto other document' () do html_a_end do html_p('') do html_a('HREF=#TOP') print 'Goto top of document' () do html_a_end</pre>
html_a_end	<p>Marks the end of a hypertext link using the HTML tag.</p> <p>Syntax: html_a_end</p>

HTML List Procedures

Table 3-5 lists the HTML list procedures.

Table 3-5 HTML List Procedures

Procedure	Description
<code>html_dd</code>	Marks the start of a definition in a definition list using the HTML <DD> tag. Syntax: <code>html_dd(string attributes)</code> ■ <i>attributes</i> = The HTML attributes inside the <DD> tag.
<code>html_dd_end</code>	Marks the end of a definition in a definition list using the HTML </DD> tag. The end of a definition in a definition list is typically implied and not needed; however, it is good idea to specify it for completeness. Syntax: <code>html_dd_end</code>
<code>html_dir</code>	Marks the start of a directory list using the HTML <DIR> tag. Syntax: <code>html_dir(string attributes)</code> ■ <i>attributes</i> = The HTML attributes inside the <DIR> tag. Example: Displaying a directory list with three items: <pre>do html_dir('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_dir_end</pre>
<code>html_dir_end</code>	Marks the end of a directory using the HTML </DIR> tag. Syntax: <code>html_dir_end</code>

Table 3-5 **HTML List Procedures (Continued)**

Procedure	Description
html_dl	<p>Marks the start of a definition list using the HTML <DL> tag. A definition list displays a list of terms and definitions. The term displays above and to the left of the definition. Use the <i>html_dt</i> procedure to display a term. Use the <i>html_dd</i> procedure to display a defintion.</p> <p>Syntax: <code>html_dl(string attributes)</code></p> <p>■ <i>attributes</i> = The HTML attributes inside the <DL> tag.</p> <p>Example: Displaying a defintion list with two terms and definitions:</p> <pre>do html_dl('') do html_dt('') print 'A Daisy' () do html_dd('') print 'A sweet and innocent flower.' () do html_dt('') print 'A Rose' () do html_dd('') print 'A very passionate flower.' () do html_dl_end</pre>
html_dl_end	<p>Marks the end of a definition list using the HTML </DL> tag.</p> <p>Syntax: <code>html_dl_end</code></p>
html_dt	<p>Marks the start of a term in a definition list using the HTML <DT> tag.</p> <p>Syntax: <code>html_dt(string attributes)</code></p> <p>■ <i>attributes</i> = The HTML attributes inside the<DT> tag.</p>
html_dt_end	<p>Marks the end of a term in a definition list using the HTML </DT> tag.</p> <p>Syntax: <code>html_dt_end</code></p>
html_li	<p>Marks the start of a list item using the HTML tag.</p> <p>Syntax: <code>html_li(string attributes)</code></p> <p>■ <i>attributes</i> = The HTML attributes inside the tag.</p>
html_li_end	<p>Marks the end of a list item using the HTML tag. The end of a list item is typically implied and not needed; however, it is a good idea to specify it for completeness.</p> <p>Syntax: <code>html_li_end</code></p>

Table 3-5 HTML List Procedures (*Continued*)

Procedure	Description
html_menu	<p>Marks the start of a menu using the HTML <MENU> tag. Use the <i>html_li</i> procedure to identify each item in the list.</p> <p>Syntax: <code>html_menu(string attributes)</code></p> <p>■ <i>attributes</i> = The HTML attributes inside the <MENU> tag.</p> <p>Example: Displaying a menu with three items:</p> <pre>do html_menu('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_menu_end</pre>
html_menu_end	<p>Marks the end of a menu using the HTML </MENU> tag.</p> <p>Syntax: <code>html_menu_end</code></p>
html.ol	<p>Marks the start of an ordered list using the HTML tag. Each item in the list typically displays indented to the right with a number to the left. Use the <i>html_li</i> procedure to identify each item in the list.</p> <p>Syntax: <code>html.ol(string attributes)</code></p> <p>■ <i>attributes</i> = The HTML attributes inside the tag.</p> <p>Example: Displaying an ordered list with three items:</p> <pre>do html.ol('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html.ol_end</pre>

Table 3-5 HTML List Procedures (*Continued*)

Procedure	Description
<code>html.ol_end</code>	Marks the end of an ordered list using the HTML tag. Syntax: <code>html.ol_end</code>
<code>html.ul</code>	Marks the start of an unordered list using the HTML tag. Each item in the list typically displays indented to the right with a bullet to the left. Use the <code>html.li</code> procedure to identify each item in the list. Syntax: <code>html.ul(string attributes)</code> ■ <i>attributes</i> = The HTML attributes inside the tag. Example: Displaying an unordered list with three items: <code>do html.ul('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_ul_end</code>
<code>html.ul_end</code>	Marks the end of an unordered list using the HTML tag. Syntax: <code>html.ul_end</code>

HTML Table Procedures

Table 3-6 lists the HTML table procedures.

Table 3-6 HTML Table Procedures

Procedure	Description
<code>html_caption</code>	<p>Marks the start of a table caption using the HTML <CAPTION> tag.</p> <p>Syntax: <code>html_caption(string attributes)</code></p> <ul style="list-style-type: none">■ <i>attributes</i> = The HTML attributes inside the <CAPTION> tag.
<code>html_caption_end</code>	<p>Marks the end of a table caption using the HTML </CAPTION> tag. The end of a table caption is typically implied and not needed; however, it is good idea to specify it for completeness.</p> <p>Syntax: <code>html_caption_end</code></p>
<code>html_table</code>	<p>Marks the start of a table using the HTML <TABLE> tag.</p> <p>Syntax: <code>html_table(string attributes)</code></p> <ul style="list-style-type: none">■ <i>attributes</i> = The HTML attributes inside the <TABLE> tag. <p>Some common attributes:</p> <ul style="list-style-type: none">border – Specifies that a border displays around each cell of the table.width – Specifies the width of the entire table in pixels.cols – Specifies the number of columns in the table. (Example: COLS=4) <p>Example: Displaying database records in a tabular format. The <code>html_caption_end</code>, <code>html_tr_end</code>, <code>html_td_end</code>, and <code>html_th_end</code> procedures are used for completeness; however, they are typically implied and not needed.</p>

Table 3-6 **HTML Table Procedures (Continued)**

Procedure	Description
html_table (<i>Continued</i>)	<pre>! start the table & display the column headings do html_table('border') do html_caption('') print 'Customer Records' (1,1) do html_caption_end do html_tr('') do html_th('') print 'Cust No' (+1,1) do html_th_end do html_th('') print 'Name" (,10) do html_th_end do html_tr_end ! display each record begin-select do html_tr('') do html_td('') cust_num (1,1,6) edit 099999 do html_td_end do html_td('') name (1,10,25) do html_td_end do html_tr_end next-listing skiplines=1 need=1 from customers end-select ! end the table do html_table_end</pre>
html_table_end	Marks the end of a table using the HTML </TABLE> tag.
	Syntax: html_table_end
html_td	Marks the start of a new column in a table row using the HTML <TD> tag. This specifies that the text that follows displays within the column.
	Syntax: html_td(<i>string attributes</i>)
	■ <i>attributes</i> = The HTML attributes inside the <TD> tag.
html_td_end	Marks the end of a column in a table using the HTML </TD> tag. The end of a column is typically implied and not needed; however, it is a good idea to specify it for completeness.
	Syntax: html_td_end

Table 3-6 HTML Table Procedures (*Continued*)

Procedure	Description
html_th	Marks the start of a new column header in a table row using the HTML <TH> tag. This specifies that the text that follows displays as the header of the column. Syntax: <code>html_th(string attributes)</code> ■ <i>attributes</i> = The HTML attributes inside the <TH> tag.
html_th_end	Marks the end of a column header in a table using the HTML </TH> tag. The end of a column header is typically implied and not needed; however, it is a good idea to specify it for completeness. Syntax: <code>html_th_end</code>
html_tr	Marks the start of a new row in a table using the HTML <TR> tag. Syntax: <code>html_tr(string attributes)</code> ■ <i>attributes</i> = The HTML attributes inside the <TR> tag.
html_tr_end	Marks the end of a row in a table using the HTML </TR> tag. The end of a row in a table is typically implied and not needed; however, it is a good idea to specify it for completeness. Syntax: <code>html_tr_end</code>

4

Encoding in SQR

This chapter describes the SQR encoding methods and lists the encodings used by SQR to support many languages, including Japanese, Chinese, and Korean.

In This Chapter	Encoding Methods.....	4-2
	Encoding Keys in the SQR.INI File.....	4-2
	Encodings Supported without Using Unicode Internally.....	4-6
	Encodings Supported in SQR.....	4-8

Encoding Methods

You can setup SQR to use one the following encoding methods.

- Read character streams into the system by "widening" them into 16-bit character strings. (This is the default method.)
- Use Unicode internally to normalize the data.

The default method is to read character streams into the system. To override the default encoding method and use Unicode internally, set the *UseUnicodeInternal* key in the SQR.INI file to TRUE.

`UseUnicodeInternal=TRUE`

When `UseUnicodeInternal=TRUE`, any combination of encodings is valid in a single SQR run, including ASCII and EBCDIC.

Encoding Keys in the SQR.INI File

You can define encoding keys in the following sections of the SQR.INI file:

- [Default-Settings]
- [Environment]

See the following sections for more information on the encoding keys in each of these SQR.INI sections.

Encoding Keys in the [Default-Settings] Section

Table 4-1 lists the encoding keys in the [Default-Settings] section of the SQR.INI. Detailed descriptions of each key follow the table.

Table 4-1 Encoding Keys in the [Default-Settings] Section

Encoding Key	Description
<code>UseUnicodeInternal</code>	Set to TRUE to use Unicode internally.
<code>AutoDetectUnicodeFiles</code>	Set to TRUE to automatically detect UCS-2 encoded files.
<code>Substitution-Character</code>	Allows a substitution character to be defined on a character set by character set basis.

UseUnicodeInternal Key

Use this key to force the use of Unicode internally. When UseUnicodeInternal=TRUE, any combination of encodings is valid in a single SQR 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 0xFFFFE. On big-endian architectures, the BOM is recorded as 0xFEFF.

If auto-detection of *UCS-2* encoded files is enabled, SQR checks whether the first two bytes of each file that it opens equal either 0xFEFF or 0xFFFFE. If so, the file reads as a *UCS-2* encoded file.

If an ENCODING directive is specified on an OPEN statement, SQR 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, SQR writes a BOM to the file as the file's first two bytes.

Substitution-Character Key

This SQR.INI 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 4-2 describes the encoding keys in the [Environment] section of the SQR.INI file. 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 4-2 Encoding Keys in the [Environment] Section

Encoding Key	Description
Encoding	Default encoding.
Encoding-Console	Encoding used for console input and output.
Encoding-Database	Encoding used for interfacing with the database.
Encoding-File-Input	Default encoding used for “OPEN” for-reading files and argument files.
Encoding-File-Output	Default encoding used for “OPEN” for-writing files.
Encoding-Report-Output	Default encoding used for report output, such as, SPF, LIS, HTM, etc.
Encoding-SQR-Source	Encoding used for SQR source and include files.

The following is an example of encoding settings in the [Environment] section:

```
[Environment:Common]
Encoding=ISO-8859-1
Encoding-File-Output=Greek
Encoding-File-Input=Shift-Jis
Encoding-Report-Output=UCS-2
Encoding-database=utf-8
Encoding-console=ascii
Encoding-SQR-Source=ucs-2
```

If these keys are not specified, encodings default to ASCII. The Encoding setting, which specifies the default encoding, can be overridden by the other encoding settings.

As with other [Environment] section settings, SQR first checks the [Environment] section of its database type and then checks the [Common Environment] section. For example, an ODBC version of SQR 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 SQR program, use the following reserved variables.

- \$SQR-ENCODING-REPORT-OUTPUT
 {SQR-ENCODING-REPORT-OUTPUT}
- \$SQR-ENCODING-FILE-INPUT
 {SQR-ENCODING-FILE-INPUT}
- \$SQR-ENCODING-FILE-OUTPUT
 {SQR-ENCODING-FILE-OUTPUT}
- \$SQR-ENCODING-CONSOLE
 {SQR-ENCODING-CONSOLE}
- \$SQR-ENCODING-SOURCE
 {SQR-ENCODING-SOURCE}
- \$SQR-ENCODING-DATABASE
 {SQR-ENCODING-DATABASE}

Encodings Supported without Using Unicode Internally

When you do not use Unicode internally (`UseUnicodeInternal=FALSE`), SQR supports the following encodings:

- ASCII
- EBCDIC
- Shift-JIS
- EUC-J
- EBCDIK290
- EBCDIK1027
- UTF-8
- UCS-2

**Note**

You can only specify UCS-2 as the value for the Encoding-Database key. If you specify UCS-2 as the value for any other encoding keys, SQR changes the encoding to UTF-8.

When you do not use Unicode internally, SQR 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 4-3 identifies a valid set of encoding settings for an SQR 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 4-3 Compatible Encodings without Unicode

Encoding-File-Input	Encoding-Database	Encoding-File-Output	Encoding-Report-Output
ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	Shift-JIS
ASCII	ASCII	ASCII	JEUC
ASCII	ASCII	ASCII	UTF-8/UCS-2
ASCII	Shift-JIS	ASCII	ASCII
ASCII	Shift-JIS	ASCII	Shift-JIS
ASCII	JEUC	ASCII	ASCII
ASCII	JEUC	ASCII	JEUC
ASCII	UTF-8/UCS-2	ASCII	ASCII
ASCII	UTF-8/UCS-2	ASCII	UTF-8/UCS-2
ASCII	Shift-JIS	Shift-JIS	Shift-JIS
ASCII	JEUC	JEUC	JEUC
ASCII	UTF-8/UCS-2	UTF-8/UCS-2	UTF-8/UCS-2
EBCDIC	EBCDIC	EBCDIC	EBCDIC
EBCDIC	EBCDIC	EBCDIC	EBCDIK290 or 1027
EBCDIC	EBCDIK290 or 1027	EBCDIC	EBCDIC
EBCDIC	EBCDIK290 or 1027	EBCDIC	EBCDIK290 or 1027
EBCDIC	EBCDIK290 or 1027	EBCDIK290 or 1027	EBCDIK290 or 1027
EBCDIK290 or 1027	EBCDIK290 or 1027	EBCDIC	EBCDIK290 or 1027
EBCDIK290 or 1027	EBCDIK290 or 1027	EBCDIK290 or 1027	EBCDIK290 or 1027
Shift-JIS	Shift-JIS	ASCII	Shift-JIS
Shift-JIS	Shift-JIS	Shift-JIS	Shift-JIS
JEUC	JEUC	ASCII	JEUC

Table 4-3 Compatible Encodings without Unicode (Continued)

Encoding-File-Input	Encoding-Database	Encoding-File-Output	Encoding-Report-Output
JEUC	JEUC	JEUC	JEUC
UTF-8/UCS-2	UTF-8/UCS-2	ASCII	UTF-8/UCS-2
UTF-8/UCS-2	UTF-8/UCS-2	UTF-8/UCS-2	UTF-8/UCS-2

**Note**

SQR 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 SQR

Table 4-4 is the complete list of encodings supported in SQR.

Table 4-4 SQR-supported Encodings

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable SQR Names
CP10004	Arabic	Macintosh Arabic	Microsoft & IBM	CP10004
CP1256	Arabic		Microsoft & IBM	CP1256
CP20420	Arabic	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20420
CP28596	Arabic	Arabic Alphabet (ISO)	Microsoft & IBM	CP28596
CP720	Arabic	Transparent ASMO	Microsoft & IBM	CP720
CP864	Arabic		Microsoft & IBM	CP864
ISO 8859-6	Arabic	ISOLatinArabic	International or National Standard	Arabic
CP708	Arabic	ASMO708	Microsoft & IBM	CP708
CP1257	Baltic		Microsoft & IBM	CP1257

Table 4-4 SQR-supported Encodings (*Continued*)

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable SQR Names
CP28594	Baltic	Baltic Alphabet (ISO)	Microsoft & IBM	CP28594
CP775	Baltic		Microsoft & IBM	CP775
ISO 8859-4	Baltic	Latin4	International or National Standard	Latin4, ISO-8859-4
ISO 8859-13	Baltic	Latin7	International or National Standard	Latin7, ISO-8859-13
ISO 8859-14	Celtic	Latin8	International or National Standard	ISO-8859-14
ISO 2022-CN	Chinese		International or National Standard	ISO-2022-CN
GB18030	Chinese		International or National Standard	GB18030
HKSCS	Chinese	Big5-HKSCS	International or National Standard	Big5-HKSCS
CP936	Chinese, Simplified	GBK	Microsoft & IBM	CP936
GB2312	Chinese, Simplified	EUC-CN, EUC-SC	International or National Standard	GB2312, EUC-CN
HZ-GB-2312	Chinese, Simplified	HZ-GB-2312	International or National Standard	HZ
Big5	Chinese, Traditional		International or National Standard	Big5
BIG5+	Chinese, Traditional		International or National Standard	BIG5+
CNS-11643-1986	Chinese, Traditional	EUC-TW	International or National Standard	CNS-11643-1986
CNS-11643-1992	Chinese, Traditional	EUC-TW	International or National Standard	CNS-11643-1992
EUC-TW	Chinese, Traditional	CNS-11643-1986, CNS-11643-1992	UNIX	CNS-11643-1992

Table 4-4 SQR-supported Encodings (*Continued*)

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable SQR Names
GB12345	Chinese, Traditional		International or National Standard	GB12345
CP10002	Chinese, Traditional	Macintosh Traditional Chinese	Microsoft & IBM	CP10002
CP950	Chinese, Traditional		Microsoft & IBM	CP950
CP10007	Cyrillic	Macintosh Cyrillic	Microsoft & IBM	CP10007
CP1251	Cyrillic	MS Windows Cyrillic (Slavic)	Microsoft & IBM	CP1251
CP20866	Cyrillic	Cyrillic Alphabet, KOI8-R	Microsoft & IBM	CP20866
CP20880	Cyrillic	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20880
CP21025	Cyrillic	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP21025
CP21866	Cyrillic	Ukrainian KOI8-RU	Microsoft & IBM	CP21866
CP28595	Cyrillic	Cyrillic Alphabet (ISO)	Microsoft & IBM	CP28595
CP855	Cyrillic	IBM Cyrillic	Microsoft & IBM	CP855
CP866	Cyrillic	MS DOS Russian	Microsoft & IBM	CP866
ISO 8859-5	Cyrillic	ISOLatinCyrillic	International or National Standard	ISOLatinCyrillic
CP10006	Greek	Macintosh Greek 1	Microsoft & IBM	CP10006
CP1253	Greek		Microsoft & IBM	CP1253
CP20423	Greek	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20423
CP28597	Greek	Greek Alphabet (ISO)	Microsoft & IBM	CP28597
CP737	Greek		Microsoft & IBM	CP737
CP869	Greek	IBM Modern Greek	Microsoft & IBM	CP869

Table 4-4 SQR-supported Encodings (*Continued*)

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable SQR Names
ISO 8859-7	Greek	ISO LatinGreek	International or National Standard	Greek
CP10010	Gurmukhi	Macintosh Gurmukhi	Microsoft & IBM	CP10010
CP10005	Hebrew	Macintosh Hebrew	Microsoft & IBM	CP10005
CP1255	Hebrew		Microsoft & IBM	CP1255
CP28598	Hebrew	Hebrew Alphabet (ISO)	Microsoft & IBM	CP28598
CP38598	Hebrew	ASCII + Hebrew and private use characters	Microsoft & IBM	CP38598
CP862	Hebrew		Microsoft & IBM	CP862
ISO 8859-8	Hebrew	ISO LatinHebrew	International or National Standard	Hebrew
CP10079	Icelandic	Macintosh Icelandic	Microsoft & IBM	CP10079
CP861	Icelandic	MS DOS Icelandic	Microsoft & IBM	CP861
CCSID 1027	Japanese	EBCDIK	Microsoft & IBM	CCSID-1027, EBCDIK1027
CCSID 290	Japanese	EBCDIK	Microsoft & IBM	CCSID-290, EBCDIK290
CCSID 942	Japanese		Microsoft & IBM	CCSID-942
CP10001	Japanese	Macintosh Japanese	Microsoft & IBM	CP10001
CP20290	Japanese	(full/half width Latin & halfwidth katakana)	Microsoft & IBM	CP20290
CP21027	Japanese	(halfwidth Latin, half-width katakana&private use)	Microsoft & IBM	CP21027
CP932	Japanese		Microsoft & IBM	CP932
EUC-JP	Japanese		UNIX	EUC-J, JEUC
EUC-JP-	Japanese		UNIX	EUC-JP-
JISROMAN				JISROMAN

Table 4-4 SQR-supported Encodings (*Continued*)

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable SQR Names
ISO-2022-JP	Japanese		International or National Standard	ISO-2022-JP
JIS_X_0201	Japanese	HalfWidthKatakana	International or National Standard	JIS_X_0201, IBM897
JIS_X_0208	Japanese		International or National Standard	JIS_X_0208
Shift-JIS	Japanese	MS_Kanji	Microsoft & IBM	Shift-JIS, SJIS
CP10003	Korean	Macintosh Korean	Microsoft & IBM	CP10003
CP1361	Korean	Korean Johab (based on KSC 5861-1992)	Microsoft & IBM	CP1361
CP949	Korean		Microsoft & IBM	CP949
EUC-KR	Korean	KS_C_5861-1992	UNIX	EUC-KR, EUC-K
ISO-2022-KR	Korean	KS_C_5601-1987	International or National Standard	ISO-2022-KR
Johab	Korean		International or National Standard	Johab
CP10000	Latin	Macintosh Roman	Microsoft & IBM	CP10000
CP10029	Latin	Macintosh Latin2	Microsoft & IBM	CP10029
CP10082	Latin	(with mathematical symbols)	Microsoft & IBM	CP10082
CCSID1047	Latin	EBCDIC (for IBM Open Systems platform)	Microsoft & IBM	CCSID1047
CP20261	Latin	(with private use characters)	Microsoft & IBM	CP20261
CP20269	Latin		Microsoft & IBM	CP20269
CP20273	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20273
CP20277	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20277

Table 4-4 SQR-supported Encodings (*Continued*)

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable SQR Names
CP20278	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20278
CP20280	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20280
CP20284	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20284
CP20285	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20285
CP20297	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20297
CP20833	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20833
CP20871	Latin	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20871
CP28591	Latin	ASCII + Latin accented vowels	Microsoft & IBM	CP28591
CP28593	Latin	Latin 3 Alphabet (ISO)	Microsoft & IBM	CP28593
CP850	Latin	MS DOS Multilingual, MS-DOS Latin1	Microsoft & IBM	CP850
CP870	Latin	(with fullwidth punctuation)	Microsoft & IBM	CP870
HP-ROMAN8	Latin	csHPRoman8, r8, roman8	HP	HP-ROMAN8
ISO 8859-1	Latin	Latin1	International or National Standard	Latin1, ISO-8859-1
ISO 8859-15	Latin	Latin1 + Euro symbol & accented characters	International or National Standard	ISO-8859-15
ISO 8859-2	Latin	Latin2	International or National Standard	Latin2, ISO-8859-2
UTF8-EBCDIC	Latin		Unicode	UTF8-EBCDIC

Table 4-4 SQR-supported Encodings (*Continued*)

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable SQR Names
CP863	Latin, Canadian French	MS DOS Canadian French	Microsoft & IBM	CP863
CP28592	Latin, Central European	Central European Alphabet (ISO)	Microsoft & IBM	CP28592
CP1250	Latin, Eastern Europe		Microsoft & IBM	CP1250
CP20905	Latin, Esperanto	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20905
CP860	Latin, Portuguese	MS DOS Portugese	Microsoft & IBM	CP860
ISO 8859-3	Latin, Southeast European	Latin3	International or National Standard	Latin3, ISO-8859-3
ASCII	Latin, US English	US-ASCII, CP367	International or National Standard	ASCII, ANSI
CP037	Latin, US English	EBCDIC	Microsoft & IBM	CP037
CP1026	Latin, US English	EBCDIC	Microsoft & IBM	CP1026
CP1252	Latin, US English	MS Windows Latin1 (ANSI)	Microsoft & IBM	CP1252
CP20105	Latin, US English	US ASCII	Microsoft & IBM	CP20105
CP437	Latin, US English	MS-DOS Latin US	Microsoft & IBM	CP437
CP500	Latin, US English	EBCDIC	Microsoft & IBM	CP500
CP875	Latin, US English	EBCDIC	Microsoft & IBM	CP875
CP10017	Malayalam	Macintosh Malayalam	Microsoft & IBM	CP10017
CP865	Nordic	MS DOS Nordic	Microsoft & IBM	CP865
ISO 8859-10	Nordic	Latin6	International or National Standard	Latin6, ISO-8859-10
CP852	Slavic	MS DOS Slavic	Microsoft & IBM	CP852
Adobe-Symbol-Encoding	Symbol	(used in PS printers)	Adobe	Adobe-Symbol-Encoding

Table 4-4 SQR-supported Encodings (*Continued*)

Encoding	Character Set	Also Known As	Vendor / Standard Body	Acceptable SQR Names
CP10008	Symbol	Macintosh RSymbol (Right-left symbol)	Microsoft & IBM	CP10008
CP20838	Thai	(with fullwidth Latin & punctuation)	Microsoft & IBM	CP20838
CP874	Thai	IBMThai	Microsoft & IBM	CP874
ISO 8859-11 (draft)	Thai	ISOLatinThai	International or National Standard	Thai
CP10081	Turkish	Macintosh Turkish	Microsoft & IBM	CP10081
CP1254	Turkish		Microsoft & IBM	CP1254
CP28599	Turkish	Turkish (ISO)	Microsoft & IBM	CP28599
CP857	Turkish	IBM Turkish	Microsoft & IBM	CP857
ISO 8859-9	Turkish	Latin5	International or National Standard	Latin5, ISO-8859-9
BMP	Unicode		Unicode	BMP
Java	Unicode	(way of representing Unicode chars in ASCII)	Sun	Java
UCS2	Unicode	ISO-10646-UCS2, UTF16	Unicode	UCS2
Unicode Big-endian	Unicode		Unicode	big-endian
Unicode Little-endian	Unicode		Unicode	little-endian
UTF7	Unicode		Unicode	UTF7
UTF8	Unicode		Unicode	UTF8
UTF8-EBCDIC	Unicode		Unicode	UTF8-EBCDIC
CP1258	Vietnamese		Microsoft & IBM	CP1258



SQR.INI

The SQR.INI file is the initialization file for SQR. This file contains settings and parameters that SQR uses during the compile and execution phases.

In This Chapter	
Installation of SQR.INI	5-2
[Default-Settings] Section	5-4
[Environment: environment] Section	5-9
[SQR Extension] Section	5-11
[Locale:local-name] Section	5-12
[Fonts] Section	5-14
[PDF Fonts] Section	5-16
[PDF Settings] Section	5-18
[HTML:Images] Section	5-20
[Enhanced-HTML] Section	5-21
[Color Map] Section	5-22
[MAP-ODBC-DB] Section	5-23
[MAP-DDO-DB] Section	5-23
[SQR Remote] Section	5-24

Installation of SQR.INI

The installation process installs a default initialization file called SQR.INI. On Windows platforms, this file is placed in the main Windows directory. (On the Windows XP platform, the default directory name is "WINDOWS." On Windows 2000, the default directory name is "WINNT.") On all other platforms, this file is placed in the same directory as the executable images (where SQRDIR points).

For the Windows Platforms Only

SQR looks for the initialization file in the following locations:

1. The file name specified by the `-ZIF{file}` command-line flag.
2. The directory where the executable image resides.
3. The Windows system directory.

Since the required environment variable `SQRDIR` is defined in the initialization file, SQR produces an error message if it cannot find the file.

For All Other Platforms

SQR looks for the initialization file in the following order:

1. The file name specified by the `-ZIF{file}` command-line flag.
2. The current working directory.
3. The directory specified with the `SQRDIR` environment variable.

Since the required environment variable `SQRDIR` is defined at the operating system level, the initialization file does not need to be available.

You can make changes or additions to the SQR.INI file if desired.

The format of the file is as follows:

```
; Comments are lines which start with a semicolon. The semicolon  
; must be the first character of the line and therefore cannot be  
; part of another line.  
;  
; Leading and trailing space characters are ignored. To preserve  
; the space characters you must surround the value with either  
; single ('') or double ("") quote characters. SQR will remove  
; them when the entry is processed.  
;  
[Section_Name]  
Entry = Value  
.  
. .  
  
[Another_Section_Name]  
Entry = Value  
. .
```

[Default-Settings] Section

The [Default-Settings] section defines the various SQR default actions.

Table 5-1 Entries in the [Default-Settings] Section

Entry	Value	Description
AllowDateAsChar	TRUE FALSE	<p>The default setting is FALSE. By default, SQR 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 this value is set to TRUE, SQR allows character to equal either character or date columns.</p> <p>When a date column is “type cast” to be a character, SQR creates the string according to the following rules:</p> <ul style="list-style-type: none">■ For DATETIME columns, SQR uses the first database-dependent format as listed in Table 2-45 on page 270.■ For DATE columns, SQR uses the format listed in Table 2-46 on page 270.■ For TIME columns, SQR uses the format listed in Table 2-47 on page 270. <p>In the following example, AllowDateAsChar=True. This allows \$Col1 to be either date or text.</p> <pre>Begin-Select [\$Col1] &col1=Text [\$Col2] &col2=Date [\$Col3] &col3=Number from MyTable End-Select</pre>
AutoDetectUnicodeFiles	TRUE FALSE	<p>The default is FALSE.</p> <p>When this value is set to TRUE, SQR auto-detects UCS-2 encoded files.</p> <p>See “AutoDetectUnicodeFiles Key” on page 4-3 for more information.</p>
DEFAULT-NUMERIC	INTEGER FLOAT DECIMAL[(p)] V30	Specifies the default numeric type for variables. The -DNT command line flag and the DECLARE-VARIABLE command override this setting. See “DECLARE-VARIABLE” on page 2-136 for complete details on the meaning of the values.

Table 5-1 Entries in the [Default-Settings] Section (*Continued*)

Entry	Value	Description
LOCALE	Name of a locale defined in the SQR.INI file or the name SYSTEM.	Specifies the initial locale that SQR loads when the program starts to execute. The value of SYSTEM is used to reference the default locale. See “ALTER-LOCALE” on page 2-9 for a complete description.
NewGraphics	TRUE FALSE	<p>The default value is FALSE.</p> <p>When set to FALSE, SQR uses Graftsman chart package. When set to TRUE, SQR uses Jchart chart package.</p> <p>You must set to NewGraphics equal to TRUE to use the following features:</p> <ul style="list-style-type: none">■ COLOR-PALETTE (See “Specifying Chart Data Series Colors” on page 14-12 in the <i>Hyperion SQR Getting Started Guide</i>.)■ ITEM-COLOR (See “Specifying Chart Item Colors” on page 14-13 in the <i>Hyperion SQR Getting Started Guide</i>.)■ ITEM-SIZE (See “DECLARE-CHART” on page 2-83 and “PRINT-CHART” on page 2-284.)■ Y-AXIS-MASK and Y2-AXIS-MASK (See “DECLARE-CHART” on page 2-83 and “PRINT-CHART” on page 2-284.)■ Y2 Syntax (See “DECLARE-CHART” on page 2-83 and “PRINT-CHART” on page 2-284.)■ Combination Charts (See “Creating Combination Charts” on page 14-16 in the <i>Hyperion SQR Getting Started Guide</i>.)■ Bubble Charts (See “Creating Bubble Charts” on page 14-24 in the <i>Hyperion SQR Getting Started Guide</i>.)
OracleWeakCursor	STOP WARN SKIP	<p>The default value is WARN.</p> <p>Describes what to do when a 'Weak' reference cursor is processed.</p> <p>When set to STOP, an error or warning message displays and the SQR procedure terminates.</p> <p>When set to WARN, an error or warning message displays and the SQR procedure continues.</p> <p>When set to SKIP, an error or warning message does <i>not</i> display, and the SQR procedure continues.</p>

Table 5-1 Entries in the [Default-Settings] Section (Continued)

Entry	Value	Description
OUTPUT-FILE-MODE	LONG SHORT	<p>Specifies the filename convention used for HTML output. SHORT specifies DOS style (8.3) and LONG specifies UNIX style (non 8.3). The default is LONG. (Ignored on 16-bit platforms) The DECLARE-TOC command and the -Burst command-line flag force Output-File-Mode = LONG.</p> <p>The following represent the file formats for UNIX, DOS, and Windows.</p> <p>SQR and SQRT: {Program} is the name of the SQR/SQT file without the extension</p> <p>For Output-File-Mode = SHORT, SQR-generated filenames are limited to a DOS 8.3 format</p> <ul style="list-style-type: none">■ Output file = {Program}.LIS for first, and {Program}.Lnn for multi-reports■ SPF file = {Program}.SPF for first, and {Program}.Snn for multi-reports■ PDF file = {Program}.PDF for first; and {Program}.Pnn for multi-reports■ HTM file = {Program}.HTM for “frame, and {Program}.Hbb for report bodies■ GIF file={Program}.Gxx for all reports <p>bb ranges from 00 to 99 and represents the report number.</p> <p>nn ranges from 01 to 99 and represents the report number.</p> <p>xx ranges from 00 to ZZ and represents the graphic number.</p> <p>For Output-File-Mode = LONG, SQR-generated filenames are <i>not</i> constrained to a DOS 8.3 format. {Output}={Program} of first report and {Program}_nn for multi-reports.</p> <ul style="list-style-type: none">■ Output file = {Output}.LIS■ SPF file = {Output}.SPF■ PDF file = {Output}.PDF■ GIF file = {Output}_zz.SPF■ HTM files = {Output}.HTM, {Output}_bb.HTM, {Output}_frm.HTM, {Output}_toc.HTM, {Output}_nav.htm <p>bb ranges from 01 to ZHJOZI and represents the bursted page group number in radix 36.</p> <p>nn ranges from 01 to 99 and represents the report number.</p> <p>zz ranges from 01 to ZHJOZI and represents the graphic number in radix 36.</p>

Table 5-1 Entries in the [Default-Settings] Section (*Continued*)

Entry	Value	Description
		SQRP: {Filename} is the name of the SPF file without the extension
		For Output-File-Mode = SHORT , SQR-generated filenames are limited to a DOS 8.3 format. <ul style="list-style-type: none">■ Output file = {Filename}.LIS■ GIF file = {Filename}.Gxx■ PDF file = {Filename}.PDF■ HTM file = .HTM and {Filename}.H00 xx ranges from 00 to ZZ and represents the graphic number.
		For Output-File-Mode = LONG , SQR-generated filenames are <i>not</i> limited to a DOS 8.3 format. <ul style="list-style-type: none">■ Output file = {Filename}.LIS■ PDF file = {Filename}.PDF■ GIF file = {Filename}_zz.SPF■ HTM files = {Filename}.HTM, {Filename}_bb.HTM, {Filename}_frm.HTM, {Filename}_toc.HTM, {Filename}_nav.htm bb ranges from 01 to ZHJOZI and represents the bursted page group number in radix 36.
		zz ranges from 01 to ZHJOZI and represents the graphic number in radix 36.
OutputFormFeedWithDashD	TRUE FALSE	The default value is FALSE. When set to TRUE, the -Dnn command line flag outputs the Form-Feed character that denotes a page break.
OutputTwoDigitYearWarningMsg	TRUE FALSE	The default value is TRUE. When set to TRUE, SQR generates a warning message (sent to the warning file) when a YY or RR date edit mask is encountered during a program run. This affects only SQR code that is processed.

Table 5-1 Entries in the [Default-Settings] Section (*Continued*)

Entry	Value	Description
PrinterHT	Standard Enhanced	The default value is Enhanced. Controls the HTML output produced by the –PRINTER: HT command-line flag. When set to Standard, –PRINTER: HT produces version 2.0 HTML files with report content inside of <PRE></PRE> tags. When set to Enhanced, –PRINTER: HT is mapped to –PRINTER: EH and produces reports in which content is fully formatted with version 1.1 XHTML tags.
SUBSTITUTION-CHARACTER=XX EncodingName1 [,xx EncodingName2...]	XX is the hexadecimal representation of the substitution variable	Allows a substitution character to be defined on a character set by character set basis. See “Substitution-Character Key” on page 4-3 for more information.
UseUnicodeInternal	TRUE FALSE	The default value is FALSE. By default, SQR reads character streams into the system by “widening” them into 16-bit character strings. When set to TRUE, SQR uses Unicode internally to normalize the data.
UseY2kCenturyAlgorithm	TRUE FALSE	The default value is FALSE. When set to TRUE, SQR treats the YY date edit mask as though it is an RR edit mask. See the RR edit mask.



Note Use the setting V30 to handle numbers in the same manner as in prior releases (before V4.0). Specifically, all numeric variables and literals are declared as FLOAT, including integer literals.

[Environment: environment] Section

The [Environment:{ Common | DB2 | Informix | ODBC | Oracle | Sybase | DDO}] section defines environment variables to be used by SQR. 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 the [Environment:Common] section.

The following environment variables can be set:

- **SQRDIR** – Specifies the default directory for all invocations of SQR.
- **SQRFLAGS** – Specifies the default command-line flags for all invocations of SQR.
- **DSQUERY** (Sybase only) – Identifies the default Sybase server to use.

On Windows systems, SQRDIR is required and is automatically defined in the appropriate database-specific environment section during the SQR installation. The other environment variables are optional.

Using the Java Virtual Machine

When you use SQR to produce reports with HTML files or charts, SQR must make several calls to Java routines. The number of HTML reports and charts you want to produce can affect the runtime of your SQR program. To reduce the total runtime of an SQR program, you can embed the Java Virtual Machine (JVM) directly into SQR. To do this, use the following environment variable:

Table 5-2 Java Virtual Machine Environment Variable

Entry	Value	Description
SQR_USEJVM	TRUE FALSE	<p>The default value is TRUE.</p> <p>When set to TRUE, SQR uses the JVM for Enhanced HTML and JClass charting applications (NewGraphics). The JVM is embedded directly into SQR.</p> <p>When set to FALSE, SQR invokes separate subprocesses for Enhanced HTML and JClass charting applications (NewGraphics).</p> <p>Note: Callable SQR uses the JVM only when this entry is set to TRUE.</p>

DDO Variables in the [Environment] Section

Table 5-3 describes the DDO variables in the [Environment] section.

Table 5-3 DDO Variables in the [Environment] Section

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

Each of these entries is automatically entered upon product installation (Windows only). You can specify additional classpath entries using up to nine SQR_DDO_JRE_CLASSn variables, where n is a number from 1-9. These additional variables are available to augment the normal 512-character line limit for entries in the SQR.INI file.

Encoding Keys in the [Environment] Section

For detailed information about the encoding keys that you can set in the [Environment] section of the SQR.INI file, see “Encoding Keys in the [Environment] Section” on page 4-4.

[SQR Extension] Section

On Windows systems only, the [SQR Extension] section defines DLLs containing new user functions (ufunc) and user calls (ucall). Ufunc and ucall reside inside SQREXT.DLL and/or other DLLs.

When SQRW.DLL and SQRWT.DLL are loaded, they look for SQREXT.DLL in the same directory, and for any DLLs specified in the [SQR Extension] section in SQR.INI such as:

```
[SQR Extension]
c:\sqrexts\sqrexxt1.dll=
c:\sqrexts\sqrexxt2.dll=
c:\sqrexts\sqrexxt3.dll=
```

Any new extension DLLs containing new user functions must be listed in the [SQR Extension] section in SQR.INI. For more information, see Chapter 25, “Interoperability” in *Hyperion SQR Getting Started Guide*.

For Windows/Oracle, SQR uses dynamic binding of Oracle routines. When SQR tries to access an Oracle database, it searches for the Oracle DLL as follows:

- The file described by the value of ORACLE_DLL entry in the [Environment:Oracle] section of the SQR.INI file.
- OCIW32.DLL (Oracle supplied)

[Locale:*locale-name*] Section

The [Locale:*locale-name*] section specifies 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 the SQR.INI file. 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 the ALTER-LOCALE command. The entries for a locale section are described in Table 5-4.



Note

The SYSTEM locale is provided for your reference, but is commented out. The settings for the SYSTEM locale, if set, are ignored. Use the ALTER-LOCALE command to change the SYSTEM locale settings at run time.

Table 5-4 Entries in the [Locale:*locale-name*] Section

Entry	Description
NUMBER-EDIT-MASK	The default numeric edit mask format when the keyword NUMBER accompanies the DISPLAY, MOVE, PRINT, or SHOW command.
MONEY-EDIT-MASK	The default numeric edit mask format when the keyword MONEY accompanies the DISPLAY, MOVE, PRINT, or SHOW command.
DATE-EDIT-MASK	The default date edit mask format when the keyword DATE accompanies the DISPLAY, MOVE, PRINT, or SHOW command, or the LET datetostr() or strtodate() functions.
INPUT-DATE-EDIT-MASK	The 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 listed in Table 2-32 on page 2-206.
MONEY-SIGN	Specifies the character(s) to replace the '\$' edit character.
MONEY-SIGN-LOCATION	Specifies the MONEY-SIGN character(s) location. Valid values are LEFT and RIGHT.
THOUSAND-SEPARATOR	Specifies the character to replace the ',' edit character.
DECIMAL-SEPARATOR	Specifies the character to replace the '.' edit character.
DATE-SEPARATOR	Specifies the character to replace the '/' character.
TIME-SEPARATOR	Specifies the character to replace the ':' character.
EDIT-OPTION-NA	Specifies the character(s) to replace the 'na' option.

Table 5-4 Entries in the [Locale:locale-name] Section (*Continued*)

Entry	Description
EDIT-OPTION-AM	Specifies the character(s) to replace 'AM'.
EDIT-OPTION-PM	Specifies the character(s) to replace 'PM'.
EDIT-OPTION-AD	Specifies the character(s) to replace 'AD'.
EDIT-OPTION-BC	Specifies the character(s) to replace 'BC'.
DAY-OF-WEEK-CASE	Specifies how the case for the DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT entries are affected when used with the format codes 'DAY' or 'DY'. Valid values are UPPER, LOWER, EDIT, and NO-CHANGE. UPPER and LOWER forces the output to either all uppercase or lowercase, ignoring the case of the format code in the edit mask. Use EDIT to follow the case as specified with the format code in the edit mask. Use NO-CHANGE to ignore the case of the format code and output the day of week as explicitly listed in the DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT entries.
DAY-OF-WEEK-FULL	Specifies the full names for the days of the week. SQR considers the first day of the week to be Sunday. All seven days must be specified.
DAY-OF-WEEK-SHORT	Specifies the abbreviated names for the days of the week. SQR considers the first day of the week to be Sunday. All seven abbreviations must be specified.
MONTHS-CASE	Specifies how the case for the MONTHS-FULL or MONTHS-SHORT entries is affected when used with the format codes 'MONTH' or 'MON'. Valid values are UPPER, LOWER, EDIT, and NO-CHANGE. UPPER and LOWER force the output to either all uppercase or lowercase, ignoring the case of the format code in the edit mask. Use EDIT to follow the case as specified with the format code in the edit mask. Use NO-CHANGE to ignore the case of the format code and output the month as explicitly listed in the MONTHS-FULL or MONTHS-SHORT entries.
MONTHS-FULL	Specifies the full names for the months of the year. SQR considers the first month of the year to be January. All 12 months must be specified.
MONTHS-SHORT	Specifies the abbreviated names for the months of the year. SQR considers the first month of the year to be January. All 12 abbreviations must be specified.

Table 5-5 Date Column Formats

Database	DATE Column Formats
DB2	YYYY-MM-DD
INFORMIX	MM/DD/YYYY
ODBC	DD-MON-YYYY

Table 5-6 Time Column Formats

Database	TIME Column Formats
DB2	HH24.MI.SS
ODBC	HH24:MI:SS

[Fonts] Section

The [Fonts] section lists the fonts available to SQR when printing on Windows printer devices (using the -PRINTER:WP command-line flag). This section does not apply to PostScript or HP LaserJet printer types. See the DECLARE-PRINTER command for a listing of available fonts for these alternate printer types.

Adding [Fonts] Entries

Within the [Fonts] section, there are a number of predefined font entries. You can add entries by using the font numbers 900 through 999. Each entry consists of a font name, a font style (fixed or proportional), and a bold indicator, all of which are associated with a font number.

For example:

```
4=Arial,proportional  
or  
300=Courier New,fixed,bold
```

**Note**

Proportional is assumed if the second parameter starts with a "P". Also, bold is assumed if a third parameter is supplied.

Using the font number commands such as ALTER-PRINTER and DECLARE-PRINTER, you can reference a particular font style.

Specifying Character Sets in Windows

If you use Windows, you can use the CharacterSet entry 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 SQR 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

The [PDF Fonts] section allows you to specify whether to embed fonts into your PDF document. In addition, it lists the available fonts for SQR when printing using the -PRINTER : PD command-line flag. Fonts specified are case sensitive.

Embedding Fonts

Use the following entry to specify whether to embed fonts; and if you do embed fonts, what fonts to embed.

Table 5-7 [PDF Fonts] Section Entry

Entry	Value	Description
Embed	All {font numbers}	<p>Specifies whether to embed fonts; and if so, what fonts to embed. Enter one of the following values:</p> <ul style="list-style-type: none">■ ALL For example: Embed=All■ A list of fonts numbers to embed. (Font numbers must be separated by spaces.) For example: Embed=1 3 5 8 64 <p>If you do not specify a value, SQR does not embed any fonts in the PDF file. This is the default action.</p>

Available Fonts

Each entry in the [PDF Fonts} section is defined to be:

`font_number=Roman_Typeface[, CJK_Typeface, Character_Map]`

The following describes each part of the entry:

`font_number`

Font number used within the SQR 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 5-8describes the legal Chinese, Japanese, and Korean Typeface / Character Map combinations.

Table 5-8 Legal Typeface - Character Map Combinations

Local	CJK Typeface	Character Map	Encoding Supported
Simplified Chinese	STSong-Light	GB-EUC-H	GB2312
	STSongStd-Light-Acro	GBpc-EUC-H	GB2312
		GBK-EUC-H	CP936
		GBKp-EUC-H	CP936
		GBK2K-EUC-H	GB18030
		UniGB-UCS2-H	UCS-2
Traditional Chinese	MHei-Medium	B5pc-H	Big5
	MSung-Light	HKscs-B5-H	Big5-HKSCS
	MSung-Std-Light-Acro	Big5	Big5
		ETen-B5-H	EUC-TW
		ETenms-B5-H	UCS-2
		CNS-EUC-H	
		UniCNS-UCS2-H	
Korean	HYGoThic-Medium	KSC-EUC-H	EUC-KR
	HYSMyeongJo-Medium	KSCcms-UHC-H	JOHAB
	HYSMyeongJoStd-Medium-Acro	KSCcms-UHC-HW-H	JOHAB
		KSCpc-EUC-H	EUC-KR
		UniKS-UCS2-H	UCS-2
Japanese	HeiseiKakuGo-W5	83pv-RKSJ-H	Shift-JIS
	HeiseiMin-W3	90ms-RKSJ-H	Shift-JIS
	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

The [PDF Settings] section lists the available PDF settings for SQR when producing PDF output.

Table 5-9 Entries in the [PDF Settings] Section

Entry	Value	Description
Bookmarks	True False	When set to True, a PDF bookmark is created for each Table of Contents entry. When set to False, bookmarks are disabled. The default is False.
Compatibility	4 - 6	The minimum version of Acrobat required to read the PDF document. The default is 5.
CompressionText	0 - 9	The amount of compression applied to text in the PDF document. Values range from 0 (no compression) to 9 (maximum compression). The default is 6, which is the best value for compression versus speed.
CompressionGraphics	0 - 9	The amount of compression applied to graphics in the PDF document. Values range from 0 (no compression) to 9 (maximum compression). The default is 6, which is the best value for compression versus speed.
EmbedAction	Stop Warn Skip	The action to take when an error occurs while trying to embed a font: <ul style="list-style-type: none">■ Stop – An error message is issued, and program execution stops.■ Warn – A warning message is issued, and the font is used without being embedded.■ Skip – No message is issued and the font is used without being embedded. The default is Stop.

Table 5-9 Entries in the [PDF Settings] Section (*Continued*)

Entry	Value	Description
EncodingNotInFontAction	Stop Warn Skip	The action to take when a font does not support an encoding. <ul style="list-style-type: none">■ Stop – An error message is issued and program execution stops.■ Warn – A warning message is issued and CP1252 is used.■ Skip – No message is issued and CP1252 is used. The default is Warn.
Subsetting	True False	When set to True subsetting is enabled for all fonts where subsetting is possible. When set to False, subsetting is disabled. The default is True.
SubsetLimit	1-100	If a document uses more than the specified percentage of glyphs in a font, then subsetting is disabled for that particular font, and the complete font is embedded. The default is 100.
SubsetMinSize	Numbers > zero	If the original font file is smaller than the specified size (in KB), then font subsetting is disabled for that particular font. The default is 100.
UnsupportedEncodingAction	Stop Warn Skip	The action to take when an unsupported ENCODING-REPORT-OUTPUT value is set. (The ENCODING-REPORT-OUTPUT value is the value set as the encoding for non-CJK fonts in the [Environment] section.) <ul style="list-style-type: none">■ Stop – An error message is issued and program execution stops.■ Warn – A warning message is issued and CP1252 is used.■ Skip – No message is issued and CP1252 is used. The default is Skip.

[HTML:Images] Section

The [HTML:Images] section defines the parameters that SQR uses when generating HTML report output files.

Table 5-10 Entries in the [HTML:Images] Section

Entry	Value	Default Value	Description
FIRST-PAGE	HEIGHT, WIDTH, NAME	60,60,firstpg.gif	Specifies the NAME of the graphic image file that accesses the first page of the report. The HEIGHT and WIDTH are values specified in pixels.
PREV-PAGE	HEIGHT, WIDTH, NAME	60,60,prevpg.gif	Specifies the NAME of the graphic image file that accesses the previous page of the report. The HEIGHT and WIDTH are values specified in pixels.
NEXT-PAGE	HEIGHT, WIDTH, NAME	60,60,nextpg.gif	Specifies the NAME of the graphic image file that accesses the next page of the report. The HEIGHT and WIDTH are values specified in pixels.
LAST-PAGE	HEIGHT, WIDTH, NAME	60,60,lastpg.gif	Specifies the NAME of the graphic image file that accesses the last page of the report. The HEIGHT and WIDTH are values specified in pixels.
WALLPAPER	NAME		Specifies the NAME of the graphic image file used as the background image for the report.



Note SQR 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

The [Enhanced-HTML] section defines various default actions that SQR takes when generating HTML output with the -EH command-line flag.

Table 5-11 Entries in the [Enhanced-HTML] Section

Entry	Value	Description
Browser	BASIC IE NETSCAPE ALL	<p>Specifies the browser and generates the appropriate HTML.</p> <ul style="list-style-type: none">■ BASIC – SQR generates HTML suitable for all browsers.■ IE – SQR generates HTML designed for Internet Explorer.■ NETSCAPE – SQR generates HTML designed for Netscape.■ ALL – If necessary, SQR generates Basic, IE, and Netscape HTML files. <i>Report_frm.htm</i> contains Javascript to “sense” the browser on the user’s machine and displays the appropriate version. (In this case, the user’s machine is the machine of the person reading the report, not the person writing it.) <p>The default action is to generate BASIC HTML files, suitable for all browsers</p>
DefaultTemplate	Path to default template.	Specifies the location of the default template for SQR HTML output. SQR provides a <code>defaultTemplate.xml</code> file that can be customized and used as the default for all SQR HTML output. Refer to Chapter 33, “Customizing the HTML Navigation Bar Template” in the <i>Hyperion SQR Getting Started Guide</i> for more details.
IncludePDFInZip	TRUE FALSE	<p>Defines whether to include the PDF file in the ZIP file SQR creates when -PRINTER:EP is combined with -EH_ZIP.</p> <ul style="list-style-type: none">■ TRUE – Includes the PDF file in the ZIP file.■ FALSE – Does <i>not</i> include the PDF file in the ZIP file. <p>The default value is FALSE.</p>
Language	English French German Portuguese Spanish	<p>Sets the language used for the HTML navigation bar.</p> <p>The default setting is English.</p>
FullHTML	3.2	<p>Specifies the level of HTML that the browser supports so appropriate Enhanced HTML code is generated.</p> <p>There is no default value. If you do not specify a value, XHTML 1.1 is produced. If you specify 3.2, HTML version 3.2 is produced.</p> <p>Note: For information on deprecated values for the FullHTML keyword see “Deprecated SQR.INI Entries” on page 8-2.</p>

[Color Map] Section

The [Color Map] section defines the default colors that you can use in your SQR reports. Enter the default colors in the format of:

```
[Color Map]
color_name=( {rgb} )
color_name=( {rgb} )
.
.
.
color_name=( {rgb} )
```

The default colors implicitly installed are:

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

[MAP-ODBC-DB] Section

The [MAP-ODBC-DB] section maps unknown ODBC driver names (using the SQR reserved variable `$sqr-connected-db-name`) to a valid `$sqr-connected-db` setting.

For example:

```
[MAP-ODBC-DB]  
MyDriver=SYBASE
```



Note See “`$sqr-connected-db {sqr-connected-db}`” and “`$sqr-connected-db-name {sqr-connected-db-name}`” on page 1-11 for additional information on the SQR reserved variables referenced from the [MAP-ODBC-DB] section.

[MAP-DDO-DB] Section

The [MAP-DDO-DB] section maps unknown DDO driver names (using the SQR reserved variable `$sqr-connected-db-name`) to a valid `$sqr-connected-db` class.

For example:

```
[MAP-DDO-DB]  
NewOracleDriver=Oracle
```

Table 5-12 Entry in the [MAP-DDO-DB] Section

Entry	Valid Values	Description
<code>driver_name</code>	CSV DB2 ESSBASE INFORMIX MSOLAP ORACLE SAP SQLSERVER SYBASE XML	The database class to which the unknown driver name is mapped.

**Note**

See “\$sqr-connected-db {sqr-connected-db}” and “\$sqr-connected-db-name {sqr-connected-db-name}” on page 1-11 for additional information on the SQR reserved variables referenced from the [MAP-DDO-DB] section.

[SQR Remote] Section

The [SQR Remote] section defines default settings for running an SQR program remotely.

Table 5-13 Entries in the [SQR Remote] Section

Entry	Description
HostName	The default host name.
UserName	The default username used to log into the host.
TimeOut	The timeout duration in seconds. The default value is 30.
Passive	Defines whether the FTP transfer is in (#0) passive or (0) non-passive mode. The default value is 0.

6

SQR Samples

SQR samples is a library of SQR programs you can use to customize and experiment with. These programs are stored in the SQR directory called SAMPLE on your installation media. Modify these programs any way you like to create customized SQR reports.

Table 6-1 lists the sample SQR programs and provides a brief description of each. Each program consists of a report specification and a sample of the output.

Table 6-1 SQR Sample Reports SQR Sample Reports

Name	Description
_____ .DAT	Data files used by the LOADALL.SQR programs
_____ .MEM	SQR startup files to run tiny, medium, and big SQR programs
APPEND.SQR	Demonstrates the append and fixed-nolf commands
APTDIARY.SQR	Demonstrates columns, text wrapping
AREA100.SQR	Demonstrates a 100% area chart
BAR100.SQR	Demonstrates a 100% bar chart
BARCODE.SQR	Demonstrates printing a bar code
CALENDAR.SQR	Demonstrates nondatabase formatting
COMP _FOR.SQR	Prints a graph of the forecasted and actual sales for a given employee
COMP _F_G.SQR	Prints a graph of the forecasted and actual sales for month or quarter
COMP _PLN.SQR	Prints a graph of the planned and actual sales for a given employee
COMP _P_G.SQR	Prints a graph of the planned and actual sales for month or quarter
COVLET02.SQR	Uses SQR to input data from user, enter data in the database, and write a form letter using a DOCUMENT paragraph

Table 6-1 SQR Sample Reports (*Continued*)SQR Sample Reports (*Continued*)

Name	Description
CRUPSL.SQR	(Oracle) Creates stored functions and procedures for Oracle Version 7
CUST.SQR	Prints a list of all of the customers bursted by page
CUSTLBL.SQR	Demonstrates printing mailing labels within columns
CUSTOMER.SQR	Demonstrates multiple detail lines, NEXT-LISTING command
CUSTOMR2.SQR	Demonstrates the use of the ON-BREAK argument to the PRINT command
CUSTOMR3.SQR	Demonstrates the use of the INPUT command to change report output
CUSTOMR4.SQR	Demonstrates the use of arrays
CUSTOMR5.SQR	Demonstrates dynamic queries to allow user to qualify a report as it runs
CUST_SUM.SQR	Prints and charts on a bar chart information about each customer in the customer table
CUSTTAPE.SQR	Demonstrates the flat file output for magnetic tape or other post-processing
DATAA.DAT	Needed for <code>append .sqr</code>
DATAB.DAT	Needed for <code>append .sqr</code>
DROPALL.SQR	Drops all the SQR sample tables created by the LOADALL program
DROPPROC.SQR	(Sybase) Deletes leftover temporary stored procedures belonging to the user
DYNAMCOL.SQR	Demonstrates use of dynamic columns, dynamic tables and variables passed to ON-ERROR procedure
EMPSQR	Prints a list of all of the employees bursted by page
EMP_COMM.SQR	Calculates each employee's commission based on sales
EMP_P_Q.SQR	List all employee quotas for a given month or quarter
ENVELOPE.SQR	Demonstrates use of printing envelope with proper bar code
EXPORT.SQR	Creates two SQR reports: one to export a database table, the second to import that table. Data from the table is stored in an external operating system file in compressed format, with trailing blanks removed.
FLATFILE.SQR	Creates an SQR report to extract a database table and place it into a flat file
FLOATBAR.SQR	Demonstrates a floating bar chart

Table 6-1 SQR Sample Reports (*Continued*)SQR Sample Reports (*Continued*)

Name	Description
FOR_CUST.SQR	Sales forecast for given customer grouped by month or quarter
FOR_EMPSQR	Sales forecast for given employee grouped by month or quarter
FOR_PROD.SQR	Sales forecast for given product grouped by month or quarter
FOR_REG.SQR	Sales forecast for given region grouped by month or quarter
FOR_SUM.SQR	Creates a table of projected product sales with links to more information
FORMLETR.SQR	Demonstrates form letters using a DOCUMENT paragraph
HILO.SQR	Demonstrates a high-low-close chart
HISTGRAM.SQR	Demonstrates a histogram chart
INQUIRY.SQR	Creates an SQR program to display rows at your terminal selected from a database table you specify. The resulting SQR program prompts you to qualify rows to be selected, display those rows, then repeat.
INVOICE.SQR	Demonstrates multiple reports, printing invoices, and printing envelopes
LOADALL.SQR	Creates and loads sample tables used in the above SQR programs
Note: The sample tables are in ASCII format. As a result, you must specify a valid ASCII-derived encoding value in your SQR.INI file. For more information on encoding values, see “Encoding Keys in the [Environment] Section” on page 4-4.	
MAKEDATA.SQR	Creates a data file with fixed length and NOLF attributes
MAKEREPT.SQR	Helps you create SQR reports more quickly
MITI1.EPS	Needed for <code>sqrllogo.sqr</code>
MULTIPLE.SQR	Demonstrates creating multiple reports
NESTREPT.SQR	Demonstrates nesting of procedures
ORDERS.SQR	Lists all the orders and the orderlines associated with them
ORD_MONG.SQR	List all orders for a given month and group them by employee number
ORD_M_Q.SQR	List all orders for a given month or quarter
ORD_PROD.SQR	List all orders for a given product
ORD_REGG.SQR	Creates a report of all orders from a given region grouped by month or grouped by quarter

Table 6-1 SQR Sample Reports (*Continued*)
SQR Sample Reports (*Continued*)

Name	Description
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
PHONELIST.SQR	Demonstrates printing within columns, page headings, and page footings
PLN_EMPSQR	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
STCKAREA.SQR	Demonstrates a stacked area chart
SQR3DBAR.SQR	Demonstrates a 3D bar chart
SQRLASER.SQR	Demonstrates graphic and file I/O commands
SQRLINE.SQR	Demonstrates a line chart
SQRLOGO.SQR	Demonstrates printing images
SQRPIE.SQR	Demonstrates a pie chart
TABREPSQR	Creates a tabular SQR report for a table you choose
UPDATE.SQR	Generates an SQR program that allows you to query and update database tables. The created program uses the SHOW command to simulate a menu interface.
UPDSAL.SQR	A sample report that demonstrates use of stored functions and procedures in Oracle



SQR Messages

This chapter lists all the messages produced by SQR. Table 7-1 lists unnumbered messages. Table 7-2 lists numbered messages.

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.

In This Chapter	Unnumbered Messages	7-2
	Numbered Messages	7-5

Unnumbered Messages

Table 7-1 Unnumbered Messages

Error Message	Suggestion/Interpretation
Out of memory.	This occurs when a call to the C routine 'malloc()' fails. <ul style="list-style-type: none">■ PC – Use the <i>-Mfile</i> to reduce some of the different memory requirements. Remove unneeded TSRs■ UNIX – Increase the size of the system swap file.■ VAX – Increase the amount of memory allowed for that user.
No cursors defined.	From the <i>-S</i> command line flag. The SQR program did not contain any commands that required a database cursor.
Not processed due to report errors.	From the <i>-S</i> command line flag. SQR cannot provide information about the cursor due to errors in the program.
Enter `01`02	Type the value to be assigned to the specified variable. `01 = First character of the variable name `02 = Rest of the variable name
NOPROMPT used - Enter value below	(Windows) This message appears when an INPUT command is defined with the NOPROMPT argument.
Enter `01	Type the value to be assigned to the specified substitution variable. `01 = Name of the substitution variable
Enter this run's parameters:	Enter the values for the parameters defined in the program.
Error on line `01: `02	SQR detected an error while processing the report file. Correct the error and rerun. `01 = Source line number `02 = Source line
Error in include file ``01" on line `02: `03	SQR detected an error while processing the report file. Correct the error and rerun. `01 = Name of the include file `02 = Source line number `03 = Source line
Warning on line `01: `02	SQR detected a non-fatal error while processing the report file. `01 = Source line number `02 = Source line

Table 7-1 Unnumbered Messages (*Continued*)

Error Message	Suggestion/Interpretation
Warning in include file ``01" on line `02: `03	SQR detected a nonfatal error while processing the report file. `01 = Name of the include file `02 = Source line number `03 = Source line
Type RETURN for more, C to continue w/o display, X to exit run:	Informational message that is used in conjunction with the -D command line flag.
Loading Oracle DLL Failed!!!	(Oracle) Title for the dialog box that informs the user that SQR could not load the Oracle DLL.
Errors were found in the program file.	Correct the errors and rerun.
Errors were found during the program run.	Correct the errors and rerun.
`01: End of Run.	Informational message. `01 = Image name (for example, SQR)
Enter report name:	Enter the name of the report (.SQR or .SQT) to run.
Enter database name:	Enter the name of the database.
Enter Username:	Enter the user name to log onto the database.
Enter Password:	Enter the password. For security reasons, the password is not echoed.
Customer ID:	Text message
Press Enter to close...	Text message
Enter Subsystem Name:	Enter the subsystem name.
`01: Program Aborting	Informational message. `01 = Image name (for example, SQR)
*** Internal Coding Error ***	Informational message.
SQL DataServer Message	(Windows) A title for the error message dialog box.
Operating-System error	(Windows) A title for the error message dialog box.
DB-Library error	(Windows) A title for the error message dialog box.
`01 is running.	(Windows) This is the body of the -C cancel dialog box. The user can click the Cancel button to abort the program run.
Click the Cancel button to interrupt it.	
Table of Contents	Text for HTML driver

Table 7-1 Unnumbered Messages (*Continued*)

Error Message	Suggestion/Interpretation
Previous	Text for HTML driver
Next	Text for HTML driver
First Page	Text for HTML driver
Last Page	Text for HTML driver
PAGE	Text for HTML driver

Numbered Messages

Table 7-2 Numbered Messages

Error Number	Error Message	Suggestion/Interpretation
000001	Error while opening the message file: `01'(`02):`03	Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = Name of the error message file `02 = System error code `03 = System error message
000002	Error while reading the message file. (`01):`02	Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = Name of the error message file `02 = System error code `03 = System error message
000003	Error while closing the message file. (`01):`02	Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = Name of the error message file `02 = System error code `03 = System error message
000004	Error while seeking the message file. (`01):`02	Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = Name of the error message file `02 = System error code `03 = System error message
000005	Corrupt message file: Invalid header information.	Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support.
000006	Corrupt message file: Invalid count (Got `01, Should be `02).	The header contains an invalid entry count. Make sure SQRDIR points to the correct directory. Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = The value read from the header `02 = What the value should be
000007	Cannot handle message file version `01.	This release of SQR does not support the header version. Make sure SQRDIR points to the correct directory. Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support.. `01 = Unsupported version read from the header

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
000010	Invalid SEMCode encountered: `01.	An invalid code was passed to the error message handler. Try reloading the files from the release media. If the error persists, contact technical support. `01 = Invalid code
000011	Unknown conversion type (`01) for code `02.	Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = Invalid type `02 = Internal error code
000012	Message `01 must be either Preload or BuiltIn.	The type error code is not correct. Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = Error code
000013	Cannot point to message `01.	The error handler cannot position to the desired error code. Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = Error code
000014	The required environment variable `01 has not been defined.	Define the named environment variable and restart SQR. `01 = Environment variable name
000015	The Meta ESC characters do not match (Got ``01', Should be ``02').	The meta escape character defined in the header does not match what the error message handler expects. Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = What was found in the header `02 = What was expected to be found
000016	`01() called to process (`02) and the message file is not open.	The specified error routine was called but the error message file was not open. Try reloading the files from the release media. If the error persists, contact technical support. `01 = Name of the routine `02 = Error code
000017	Message `01 must be ReportParameters or CopyrightNotice.	Try reloading the <code>sqrerr.dat</code> file from the release media. If the error persists, contact technical support. `01 = Error code
000018	Allocation header does not point to a valid heap.	(Windows) This is the result of a memory overwrite. Record the steps leading up to the error and contact technical support.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
000019	Allocation header has an invalid size.	(Windows) This is the result of a memory overwrite. Record the steps leading up to the error and contact technical support.
000020	GLOBAL header has an invalid size.	(Windows) This is the result of a memory overwrite. Record the steps leading up to the error, and contact technical support.
000021	Cannot free GLOBAL allocation.	(Windows) This is the result of a memory overwrite. Record the steps leading up to the error and contact technical support.
000028	Cannot access the initialization file: `01 (^02): `03	The initialization file specified by the -ZIF command line flag cannot be accessed. `01 = Name of the file `02 = System error code `03 = System error message
000029	Unknown encoding name: `01	The encoding name specified by the -ZEN command line flag is not valid `01 = Encoding name
000202	DPUT: Bad field number.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000203	DARRAY: Unknown command number.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000204	`01: Cannot find `02 command.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine `02 = Name of the command
000205	DDO: DO arguments do not match procedure's.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000206	SDO: Bad params for DO command.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000207	SDO: Bad params for BEGIN-PROCEDURE command.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
000208	SGOTO: Bad command numbers.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000209	SGOTO: Bad goto function parameters.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000210	SGOTO: Could not find beginning of section or paragraph.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000211	SGOTO: Bad label: from parameters.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000212	COMPAR: Unknown relational (numeric) operator.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000213	COMPAR: Unknown relational (string) operator.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000214	DONBRK: Unknown case for putin.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000215	`01: Bad length case for numeric `02.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine `02 = name of the variable
000216	GARRAY: Unknown command number.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000217	GCMDS: No Gfunc found.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000218	GDOC: Unknown document type.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
000219	GLET: Bad operator.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000220	GLET: Stack incorrect for expression - arg `01.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Number of the argument
000221	GLET: Unknown operator type.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000222	GLET: Unknown operator in expression.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000223	GPARS: Column not SCOL, TCOL, or NCOL type.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000224	GPARS: Bad parameter format: `01 =`02=	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string `02 = Bad format field found
000225	GPARS: No end of required word in parfmt: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string
000226	GPARS: Bad parfmt entry: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string
000227	GPARS: Bad parameter string.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000228	GPARS: Repeat count bad: `01	This is an 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

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
000229	GPARS: Only a,b,8,9 allowed for repeats: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string
000230	GPARS: Missing required x: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string
000231	GPARS: Bad type in 'ckvrpr()'.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000232	GPROC: No Gfunc found.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000233	GRDWRT: Unknown command number.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000234	GSHOW: Unknown SHOW option.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000235	PGMPARS: 'addvar()' passed maxlen but not column.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000238	PGMPARS: ``01' passed invalid parameter number: `02.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Routine name `02 = Invalid parameter number
000239	PGMPARS: 'fxclrf()' encountered bad column reference type: `01.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal variable type code
000240	PLCMNT: 'getplc()' passed invalid element number: `01.	This is an 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

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
000241	RDPGM: Command array size exceeded (change COMDMAX to at least `01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Maximum internal command number supported.
000242	RDPGM: Bad match adding internal variable: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal variable name
000243	RDPGM: No cmdget function found for BEGIN_S.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000244	Function `01 not included in run-time package.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the SQR routine
000245	SETSQL: Could not find variable ``01', in Run Time.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable name
000248	SIFWHL: Command number incorrect.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000249	SPINIT: Bad parameters.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000251	DBFFIX: DBDATLEN returned out of range status.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000252	DPRPST: Error converting Sybase type for EXECUTE.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000254	SETSQL: Could not find variable entry in list.	(Oracle) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000255	DBDESC: SQLD not = number of select columns.	(DB2 and Informix) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
000256	DBFETCH: Unknown variable dbtype encountered: `01 (`02)	(DB2 and Informix) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable name `02 = Unknown database type
000257	WRITE_SPF: Unknown code encountered: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Unknown SPF code
000258	`01: Cannot find LOAD-LOOKUP table: `02	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine `02 = Name of the table
000259	PGMPARS: ``01' called with wrong variable ``02'	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine `02 = Name of the variable
000260	SQTMGT: Could not find 'vars' entry with 'nvars' index of ``01'.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Index into nvars table
000261	MODIFYVAR: Attempt to change variable which is not xVAR(`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = name of the variable
000262	MODIFYVAR: Incompatible variable types(`01) and(`02).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable type (from) `02 = Variable type (to)
001201	Cannot open the argument file: ``01'.(`02);`03	Depends on the system error message. `01 = Name of the file `02 = System error code `03 = System error message

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
001202	Cannot close the argument file. (`01): `02	Depends on the system error message. `01 = System error code `02 = System error message
001203	Cannot open the -Mfile: '01'.(`02): `03	Depends on the system error message. `01 = Name of the file `02 = System error code `03 = System error message
001204	Minimum value for ``01' in the -Mfile is `02.	Correct the -Mfile entry. `01 = Keyword in question `02 = Minimum value allowed
001205	Maximum value for ``01' in the -Mfile is `02.	Correct the -Mfile entry. `01 = Keyword in question `02 = Maximum value allowed
001206	Invalid -Mfile entry: ``01'.	Correct the -Mfile entry. `01 = The line from the -Mfile
001207	Cannot close the -Mfile. (`01): `02	Depends on the system error message. `01 = System error code `02 = System error message
001209	The minimum value for ``01' (`02) is `03.	Value out of range. `01 = Entry name `02 = Specified value `03 = Minimum value
001210	The maximum value for ``01' (`02) is `03.	Value out of range. `01 = Entry name `02 = Specified value `03 = Maximum value
001211	The value for ``01' (`02) is not an integer number.	Value must be a integer value. `01 = Entry name `02 = Specified value
001300	Bind list does not match query (do not use '@__p' string).	SQR reserves the variable names that start with "@__p" for internal use. Edit the source code and use different variable names.
001301	Forward references not permitted in select list bind variables.	Within the body of BEGIN-SQL paragraphs, forward references to &column names are not permitted. Move the BEGIN-SQL paragraph after the &column definition.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
001303	Error in SQL (perhaps missing &name after expression):	The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement.
001304	Check SELECT columns, expressions and 'where' clause for syntax.	The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement.
001305	CMPSQL: Unknown data type in database: `01.	Contact technical support with the version of the database you are connected to. `01 = Datatype in question
001306	Bind value too large (IMAGE, TEXT not allowed).	IMAGE and TEXT data types cannot be used as bind variables. Modify your SQL statement to use other columns to perform the same selection logic.
001307	CMPSQL: DBDEFN failed.	(ODBC, Oracle, Informix) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
001308	`01: Could not bind column `02.	(ODBC, Oracle, Informix) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the SQR routine `02 = Name of the column
001309	The type for '&`01'(`02) does not match the type from the database (`03).	Correct the source code. `01 = Name of the column/expression pseudonym `02 = User specified type `03 = Database type
001400	Only numerics allowed for arithmetic.	Only #numeric variables, &columns, and literals are permitted in the arithmetic commands. Correct the source code.
001401	Optional qualifier is ROUND=n (0-`01).	Correct the syntax. `01 = Maximum value for ROUND=
001402	Optional qualifiers for DIVIDE are ON-ERROR={HIGH ZERO} and ROUND=n.	Correct the syntax.
001403	Attempting division by zero.	Use the ON-ERROR = HIGH ZERO option to prevent this error from halting the program.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
001404	Bad number of digits to ROUND or TRUNC (0-15).	Correct the syntax.
001405	WARNING: The ROUND or TRUNC qualifier is greater than the number's precision.	Correct the syntax.
001500	Array element out of range (`01) for array `02' on line `03.	Correct the source logic. `01 = Element number passed `02 = Name of the array `03 = Program line number
001501	Field element out of range (`01) for array `02', field `03', on line `04.	Correct the source logic. `01 = Element number passed `02 = Name of the array `03 = Name of the field `04 = Program line number
001502	WARNING: Attempting division by zero on line `01. Array field `02' unchanged. Run continuing...	The ARRAY-DIVIDE command has attempted division by zero. The division has been ignored; the result field is unchanged. Add logic to account for this possibility. `01 = Program line number `02 = Name of field
001601	'FILL' not appropriate for numeric data.	The FILL argument to the PRINT command may be used only for text fields. Move the #numeric variable to a \$string variable, and then print the string variable.
001700	Report ``01': Columns must be between 1 and the page width (`02).	The specified value is wider than the width of the page. Correct the source line. `01 = Name of the current report `02 = Page width
001702	Report ``01': GOTO-TOP=`02 must be between 0 and the page depth (`03).	The value specified on the GOTO-TOP argument of the NEXT-COLUMN command was either less than 1 or greater than the page depth. Correct the source line. `01 = Name of the current report `02 = Goto-Top value `03 = Page width
001703	Report ``01': ERASE-PAGE=`02 must be between 0 and the page depth (`03).	The line number specified on the ERASE-PAGE argument of the NEXT-COLUMN command is greater than the page depth. Correct the source line. `01 = Name of the current report `02 = Erase-Page value `03 = Page width

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
001704	Report ``01': The NEXT-COLUMN command is not legal in the `02 section with the qualifier AT-END=NEWPAGE.	Correct the source line. `01 = Name of the current report `02 = Name of the section
001705	Report ``01': Column number `02 is not defined.	The column number specified with the USE-COLUMN command is greater than the highest column defined in the COLUMNS command. Correct the source line. `01 = Name of the current report `02 = Column number
001800	Format for CONNECT: username/password [ON-ERROR=procedure[(arg1[,arg1]...)]]	Correct the syntax.
001801	Cannot use CONNECT while SQL statements are active.	Correct the program logic to ensure that all BEGIN-SELECT paragraphs have completed before executing the CONNECT command.
001802	Logoff failed prior to CONNECT.	The database server returned an error while trying to log off from the database. SQR aborts the program run since it cannot continue.
001803	CONNECT failed. Perhaps username/password incorrect.	The specified connectivity information is incorrect or there might have been a network failure. Use the ON-ERROR flag to trap any errors during the program run; otherwise SQR aborts the program run.
001804	Sybase extensions SET and SETUSER not permitted in SQR.	Remove SET and SETUSER from the source.
001805	USE allowed once in SETUP section only, not in BEGIN-SQL. Elsewhere, specify db.[user].table...	Correct the source.
001807	The requested database connection (`01) is already active.	The -Cnn value specified is being used by another BEGIN-SELECT paragraph that is currently selecting data. Use another connection number. `01 = Connection number
001808	Cannot find inactive database cursor. Program too large.	Too many BEGIN-SELECT and BEGIN-SQL paragraphs are active at the same time. Reduce the complexity of the program.
001809	Database commit failed.	(DB2, ODBC, Oracle) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
001810	Database rollback failed.	(DB2, ODBC, Oracle) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.
001811	Cannot open database cursor.	(ODBC, Oracle) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.
001901	Variable for date-time must begin with '&'.	Correct the syntax.
001913	Format code must be SYYYY when specifying signed year.	Correct the edit mask.
001914	Bad input data (`01) for edit mask: `02'.	Correct the input. `01 = Data being converted `02 = Edit mask
001915	Year cannot be zero.	Correct the date.
001916	Year must be between -4713 and 9999 inclusive.	Correct the date.
001917	Ambiguous date-time.	Correct the date.
001918	`01' is not a valid date part.	Correct the date part. `01 = Date part.
001919	Invalid day of week.	Correct the date.
001920	Format code cannot appear in date input format: `01'.	Correct the edit mask. `01 = Improper format characters.
001921	Bad date mask starting at: `01'.	Correct the edit mask. `01 = Improper format characters.
001922	Seconds past midnight must be between 0 and 86399.	Correct the date.
001923	Seconds must be between 0 and 59.	Correct the date.
001924	Minutes must be between 0 and 59.	Correct the date.
001925	Month must be between 1 and 12.	Correct the date.
001926	Day must be between 1 and `01.	Correct the date.
001927	Hour must be between 1 and 12.	Correct the date.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
001928	Hour must be between 0 to 23.	Correct the date.
001929	HH24 precludes the use of meridian indicator.	Correct the edit mask.
001930	HH12 requires meridian indicator.	Correct the edit mask.
001931	Day of year must be between 1 and 365 (366 for leap year).	Correct the date.
001932	Date string too long.	Correct the date.
001933	The month (` 01) is not valid for the current locale or database.	Correct the date. ` 01 = Name of the month.
001934	The format mask must be a literal when the date-time is not loaded into a variable.	Correct the format mask. The format mask must be a literal when the date-time is not loaded into a variable.
001935	Date-time format too long.	Correct the format mask.
001936	Bad date-time format.	Correct the format mask.
001937	Bad SQL for default date-time. (Table DUAL required for syntax.)	(Oracle) Possibly the format mask needs to be corrected; otherwise, there is a problem with the database server.
001937	Bad SQL for default date-time. (Table DUAL required for syntax.)	(DB2) Possibly the format mask needs to be corrected; otherwise, there is a problem with the database server.
001938	Cannot recompile sql.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.
001939	Problem executing cursor.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.
001940	Error fetching row.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.
001941	Cannot redefine variable addresses.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
001942	The date ``01' is not in the format: YYYYMMDD[HH24[MI[SS[NNNNNN]]]].	When specifying an SQR date, at a minimum, the date must be specified; the time is optional. `01 = The invalid date.
001943	The date ``01' is not in one of the accepted formats listed below: MM/DD/YYYY [BC AD] [HH:MI[:SS[.NNNNNNN]]] [AM PM]] MM-DD-YYYY [BC AD] [HH:MI[:SS[.NNNNNNN]]] [AM PM]] MM.DD.YYYY [BC AD] [HH:MI[:SS[.NNNNNNN]]] [AM PM]] YYYYMMDD[HH24[MI[SS[NNNNNN]]]]	The date specified with the INPUT command was not in one the default formats. Please re-enter the date in a valid format. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: DD-MON-YY YYYYMMDD[HH24[MI[SS[NNNNNN]]]]	(Oracle) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: Mon DD YYYY [HH:MI[:SS[.NNN]]][AM PM]] Mon DD YYYY [HH:MI[:SS[:NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[.NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[:NNN]]][AM PM]] YYYYMMDD[HH24[MI[SS[NNNNNN]]]]	(Sybase) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: Mon DD YYYY [HH:MI[:SS[.NNN]]][AM PM]] Mon DD YYYY [HH:MI[:SS[:NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[.NNN]]][AM PM]] YYYYMMDD [HH:MI[:SS[:NNN]]][AM PM]] YYYYMMDD[HH24[MI[SS[NNNNNN]]]]	(ODBC) The date was not in one of the expected formats for this database. `01 = The invalid date.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: Mon DD YYYY [HH:MI[:SS[.NNN]][AM PM]] Mon DD YYYY [HH:MI[:SS[:NNN]][AM PM]] YYYYMMDD [HH:MI[:SS[.NNN]][AM PM]] YYYYMMDD [HH:MI[:SS[:NNN]][AM PM]] YYYYMMDD[HH24[MI[SS[NNNNNN]]]]]	(DDO) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: YYYY-MM-DD HH:MI:SS.NNN YYYYMMDD[HH24[MI[SS[NNNNNN]]]]	(Informix) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: YYYY-MM-DD[-HH.MI.SS[.NNNNNN]] MM/DD/YYYY DD.MM.YYYY YYYYMMDD[HH24[MI[SS[NNNNNN]]]]	(DB2) The date was not in one of the expected formats for this database. `01 = The invalid date.
001944	The date ``01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: YYYY-MM-DD YYYYMMDD[HH24[MI[SS[NNNNNN]]]]	(Teradata) The date was not in one of the expected formats for this database. `01 = The invalid date
001945	SQR does not support dates before ``01'.	SQR does not support dates before the one specified in the message. `01 = Smallest date
001946	The date variables are incompatible with each other.	The SQR function references two date variables which cannot be logically be used together (for example, DateDiff of 'date-only' and 'time-only' dates).
002000	Procedure name used more than once: ``01'.	Give the procedure a unique name. `01 = Procedure name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
002001	Could not find procedure: `01'.	Check for a misspelled procedure name. `01 = Procedure name
002002	DO arguments do not match procedure's.	The argument lists for the DO and BEGIN-PROCEDURE commands must match in both type and count. Correct the source line.
002003	DO argument must be \$string or #number to accept returned value.	Correct the syntax.
002100	Edit string too long.	The edit mask must be less than 255 characters. Reduce the length of the edit mask.
002101	Bad numeric 'edit' format: `01	The numeric edit mask contains an invalid character. See the PRINT command for the valid numeric edit mask characters. `01 = Invalid character
002103	DOLLAR-SYMBOL must be a single alphanumeric character or its decimal value enclosed in brackets: <nnn>.	Correct the syntax.
002104	DOLLAR-SYMBOL cannot be any of the following characters: `01	Correct the syntax. `01 = List of invalid characters
002106	MONEY-SYMBOL must be a single alphanumeric character or its decimal value enclosed in brackets: <nnn>.	Correct the syntax.
002107	MONEY-SYMBOL cannot be any of the following characters: `01	Correct the syntax.
002200	ENCODE string too large; maximum is `01.	Break up the ENCODE command. `01 = Maximum length of an ENCODE string supported by this version of SQR
002300	EXIT-SELECT failed.	The database command to cancel the query returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
002301	EXIT-SELECT valid only within SELECT paragraph.	Remove the EXIT-SELECT command.
002400	Duplicate label's - do not know which one to GOTO.	Labels must be unique within the section or paragraph where they are defined. Give each label a unique name.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
002401	(Labels must be in same section or paragraph as GOTO.) Cannot find a matching label for GOTO command.	Check the source code.
002500	Error getting INPUT.	The C routine "fgets()" returned an error and SQR aborts the program run.
002501	Unknown INPUT datatype: type={char number integer date}	Correct the syntax.
002502	INPUT STATUS= must reference #variable.	Correct the syntax.
002503	Unknown qualifier for INPUT.	Correct the syntax.
002506	Too long. Maximum `01 characters.	The response to the INPUT statement was too long. Re-enter the data. `01 = Maximum characters allowed
002507	Incorrect. Format for floating point number: [+ -]99.99[E99]	Invalid number was entered for an INPUT request. Re-enter the data.
002508	Incorrect. Format for integer: [+ -]999999	Invalid integer was entered for an INPUT request. Re-enter the data.
002510	A format mask can only be specified when TYPE=DATE is used.	Correct the syntax.
002511	The format mask cannot be stored in a date variable.	Correct the syntax.
002512	The input variable type does not match the TYPE qualifier.	Correct the syntax.
002513	Number too large for INTEGER. Valid range is -2147483648 to 2147483647.	The number was too large to be stored as an integer. Values are from -2147483648 to 2147483647. Re-enter the data.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
002514	Enter a date in one of the following formats: MM/DD/YYYY [HH:MI[:SS[.NNNNNN]]] [AM PM] MM-DD-YYYY [HH:MI[:SS[.NNNNNN]]] [AM PM] MM.DD.YYYY [HH:MI[:SS[.NNNNNN]]] [AM PM] YYYYMMDD[HH24[MI[SS[NNNNNN]]]]	The date cannot be blank. Enter a date in one of the specified formats.
002515	`01 required user interaction but user interaction was disabled by the -XI command line flag.	The specified command required user interaction, but user interaction was disabled by the -XI command line flag. `01 = Name of the command
002600	LOAD-LOOKUP table ``01' has not been defined.	Add a LOAD-LOOKUP command. `01 = Load lookup table name
002601	Missing value for `01= in LOAD-LOOKUP.	Correct the syntax. `01 = Name of missing required parameter
002602	Bad value for `01= in LOAD-LOOKUP.	Correct the syntax. `01 = Name of the parameter
002603	LOAD-LOOKUP `01= cannot reference a variable in the Setup section.	Either move the LOAD-LOOKUP command from the Setup section or remove the variable reference. `01 = Name of the parameter
002604	LOAD-LOOKUP names must be unique.	Give each LOAD-LOOKUP array a unique name.
002605	Cannot compile SQL for LOAD-LOOKUP table ``01'.	The database server returned an error while trying to compile the SQL statement needed to process the LOAD-LOOKUP command. Check the column and table names. Also check the WHERE= clause for errors. `01 = Load lookup table name
002606	Could not set up cursor for LOAD-LOOKUP table ``01'.	The database server returned an error while trying to compile the SQL statement needed to set up the LOAD-LOOKUP command. Check the column and table names. Also check the WHERE= clause for errors. `01 = Load lookup table name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
002607	Problem executing the cursor for LOAD-LOOKUP table `01'.	The database server returned an error while trying to execute the SQL statement needed to process the LOAD-LOOKUP command. `01 = Load lookup table name
002609	Integers only allowed in numeric lookup keys.	Correct the source line.
002610	Numeric lookup keys must be <= `01 digits.	Correct the source line. `01 = maximum length supported
002611	Bad return fetching row from database in LOAD-LOOKUP table `01'.	The database server returned an error while fetching the data. `01 = Load lookup table name
002613	Loading `01' lookup table ...	This message can be inhibited by using the QUIET argument on the LOAD-LOOKUP command. `01 = Name of the load lookup table `02 = Number of rows loaded
002615	Warning: `01 duplicate keys found in `02' lookup table.	This message can be inhibited by using the QUIET argument on the LOAD-LOOKUP command. `01 = Number of duplicate keys `02 = Name of the load lookup table
002616	LOAD-LOOKUP `01= must reference a numeric variable or literal.	Correct the source line. `01 = Name of the parameter
002617	LOAD-LOOKUP `01= must reference a string variable or literal.	Correct the source line. `01 = Name of the parameter
002618	LOAD-LOOKUP `01= variable `02' has not been defined.	Correct the source line. `01 = Name of the parameter `02 = Name of the undefined variable
002619	LOAD-LOOKUP cannot support `01 rows; maximum is `02.	Reduce the ROWS= value. `01 = ROWS= value `02 = Maximum value allowed
002620	`01 command not allowed with -XL option in effect.	Either use the #IF command to conditionally compile the program when -XL is being used or do not execute this SQR report with the -XL option. `01 = SQR command
002700	Line to stop erasing for 'NEW-PAGE' is larger than the page depth.	Correct the source line.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
002800	'ON-BREAK' not appropriate for numeric data.	The ON-BREAK argument to the PRINT command may be used only for text fields. Move the #numeric variable to a \$string variable, and then print the \$string variable.
002801	SET= and LEVEL= must be >= zero when indicated.	Correct the source line.
002802	Cannot use old style PROCEDURE= with BEFORE= or AFTER=.	Correct the syntax.
002804	SET= must be same for all ON-BREAKs in Select.	All the ON-BREAKS in a query must belong to the same SET. Use SET= to differentiate between ON-BREAKs in different queries. Correct the source line.
002805	ON-BREAK with BEFORE or AFTER must be inside Select.	Correct the source line.
002806	SAVE= must be a \$string variable.	Correct the syntax.
002900	Record :types are FIXED, VARY, or FIXED_NOLF (default is VARY).	Correct the syntax.
002901	STATUS variable for `01 must be #Numeric.	Correct the syntax. `01 = SQR command affected
002902	OPEN missing required qualifiers: RECORD={rec_len} FOR-READING FOR-WRITING FOR-APPEND	Correct the syntax.
002903	Too many external files opened; maximum is `01.	Reduce the number of open external files needed by the program. `01 = Maximum number of open external files supported by this version of SQR
002904	File number already opened.	Check your program logic.
002905	Cannot open file ``01' AS `02.(`03):`04	SQR aborts. `01 = Filename `02 = File number `03 = System error code `04 = System error message
002906	Cannot close file `01(`02):`03	SQR aborts. `01 = File number `02 = System error code `03 = System error message

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
002907	Problem closing user file(s) at the end of run.	This message may indicate system problems.
002908	Warning: Cannot CLOSE file `01 -- file not opened.	While not an error, this message indicates a problem with your SQR code. `01 = File number.
002909	Missing or invalid character set encoding: ``01'.	Specify a valid encoding: UCS-2, ASCII, EBCDIK1027, etc.
002910	This has been declared a UCS-2 file; however, the Byte-Order-Mark is missing or invalid.	A UCS-2 file must contain a Byte-Order-Mark.
002911	Cannot use `01 without setting UseUnicodeInternal to TRUE in the SQR.INI file.	The UseUnicodeInternal setting in the DEFAULT-SETTINGS section of the SQR.INI must be set to TRUE in order to use this functionality.
002912	The encoding `01 cannot be supported without the use of unicode. Set the UseUnicodeInternal setting in SQR.INI file to TRUE.	The UseUnicodeInternal setting in the DEFAULT-SETTINGS section of the SQR.INI file must be set to TRUE in order to use this encoding.
002913	Invalid Unicode character representation used in the UNICODE Let function.	The string of hex characters representing a set of Unicode characters must be in the format '[whitespace U+ u]XXXX' for the UNICODE Let function.
002914	The encoding `01 is not compatible with `02 unless unicode is used internally. Set the UseUnicodeInternal setting in the INI file to TRUE and rerun.	Certain encodings are not compatible unless unicode is used. The UseUnicodeInternal setting in the DEFAULT-SETTINGS section of the SQR.INI file must be set to TRUE in order to use these encodings in the same run.
002915	The file encoding `01 is not compatible with `02 unless unicode is used internally. Set the UseUnicodeInternal setting in the INI file to TRUE and rerun.	Certain encodings are not compatible unless unicode is used. The UseUnicodeInternal setting in the DEFAULT-SETTINGS section of the SQR.INI file must be set to TRUE in order to use these encodings in the same run.
002916	The SQT encoding `01 is not compatible with `02. Set the UseUnicodeInternal setting in the INI file to TRUE and rerun.	Certain encodings are not compatible unless unicode is used. The SQT was not generated using unicode; therefore, the encodings specified are incompatible.
		Reset the encoding in conflict or regenerate the SQT with the UseUnicodeInternal setting in the DEFAULT-SETTINGS section of the SQR.INI file to TRUE in order to use these encodings in the same run.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
002917	The SQT file was created using an SQR executable that does not support unicode. Please run the SQT using the appropriate executable.	(EBCDIC) The SQT file was created using SQR that does not support unicode. Therefore, it must be run with a non-unicode SQR or SQRT executable.
002918	An operation was detected that requires unicode; however, this executable cannot perform unicode processing. Relink or use the appropriate executable.	Unicode processing is required; however, the executable has not been linked with the Rosette Unicode library. Either relink the executable with the Rosette library or, for EBCDIC platforms, use the alternate executable that supports unicode processing.
002919	The field length `01 is invalid for I/O with UCS-2 data.	By definition, UCS-2 requires two bytes per character; however, the length specifier was odd. Make the length a multiple of two.
002920	:length must be a numeric literal, variable, or column.	Correct the syntax.
002921	Bad :length (`01) for OPEN command.	The length must be greater than zero. `01 = Length value
003000	PAGE-NUMBER strings too long.	The pre-and post-PAGE-NUMBER strings must be less than 74 characters. Correct the source line.
003100	Cannot find document marker referenced in POSITION command.	Defines the specified @ marker in a BEGIN-DOCUMENT paragraph. Check for a misspelled @ marker name.
003101	Only 'COLUMNS nn...' allowed after document marker in POSITION command.	Correct the syntax.
003200	Specified file number not opened for reading.	Files must be opened for reading in order to use the READ command with them. Correct the program logic.
003201	Line `01: Error reading the file. (`02):`03	`01 = Program line number `02 = System error code `03 = System error message
003202	Specified file number not opened for writing.	Files must be opened for writing in order to use the WRITE command with them. Correct the program logic.
003203	Line `01: Error writing the file. (`02):`03	`01 = Program line number `02 = System error code `03 = System error message
003204	Length of variables exceeds record length.	The total of the lengths indicated in the command must be less than the RECORD= argument used on the OPEN command. Check for a typographical error or recalculate the RECORD= value.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003205	Numeric binary transfer allowed with FIXED or FIXED_NOLF records only.	By default, all files are opened in VARY (variable length) mode, thus prohibiting the transfer of numeric binary data. Add the:FIXED or FIXED_NOLF option to the RECORD= argument on the appropriate OPEN command.
003206	Command not complete.	Correct the syntax.
003207	File number must be a numeric literal, variable, or column.	Correct the syntax.
003208	Missing required :length in READ command.	Correct the syntax.
003209	Bad :length (`01) for READ or WRITE command.	The length must not be less than zero. `01 = Length value
003210	\$String or #numeric variables required for READ.	Correct the syntax.
003211	#Numeric variables and literals must have :length of 1, 2 or 4 bytes.	Correct the syntax.
003212	#Numeric variables and literals on CDC may only have :length of 1 or 3 bytes.	Correct the syntax.
003213	:Length not allowed for \$date variables, length of 18 is assumed.	Correct the syntax.
003214	:length must be a numeric literal, variable, or column.	Correct the syntax.
003300	Unknown qualifier for STOP.	Correct the syntax.
003301	Program stopped by user request.	Informational message.
003400	Wrap not appropriate for numeric data.	The WRAP argument to the PRINT command may be used only for text fields. Move the #numeric variable to a \$string variable first, and then print the \$string variable.
003401	Max `01 chars/line for reverse WRAP.	Reduce the number of characters specified. `01 = Maximum number of characters supported by this version of SQR.
003402	Max `01 chars/line for WRAP with ON= or STRIP=	Reduce the number of characters specified. `01 = Maximum number of characters supported by this version of SQR

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003403	Bad <number> in WRAP qualifier.	The number inside the angled brackets must be a valid ASCII number (1 - 255). Correct the source line.
003404	Missing '>' in WRAP qualifier.	A leading "<" in the ON= or STRIP= qualifier indicates that a numeric value is following, which must be ended by a closing ">". Correct the source line.
003405	The value for ``01' (`02) must be `03 0.	The value specified for the specified qualifier is invalid. Correct the program logic. `01 = Qualifier name `02 = Value encountered `03 = Relation to zero (<,<=,=,>=,>)
003500	PUT, GET or ARRAY-xxxx command incomplete. Required word missing.	Correct the syntax.
003501	Did not find end of literal.	The ending quote character (') was not found at the end of the literal. Add the ending quote character.
003502	Literal too long.	Literal strings can be up to 256 characters long. Break up the literal into smaller pieces and combine using the LET command.
003503	Unknown variable type.	Variable names must begin with \$, #, or &. Correct the source line.
003504	Cannot find 'array_name (#element)'.	The element number was not specified. Correct the source line.
003505	'(#Element)' variable not found for array.	Each GET or PUT command must indicate the element or row number to access in the array. Correct the source line.
003506	Array specified not defined with CREATE-ARRAY.	Use the CREATE-ARRAY command to define each array before referencing that array in other commands. Check for a misspelled array name.
003507	Bad element reference for array (#variable 123).	The element number is larger than the number of rows defined in the CREATE-ARRAY command. Check program logic to make sure that the element number was not inadvertently changed.
003508	Did not find ending ')' for field.	The "occurs" number for an array field is missing a right parenthesis. Correct the source line.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003509	Field not defined in array: `01	Check for a misspelled field name against the CREATE-ARRAY command. `01 = Undefined field name
003510	More variables than fields specified in array command.	The ARRAY command must not have more variables listed to the left of the array name than there are matching fields defined for the array. Check against the CREATE-ARRAY command.
003511	More variables in command than fields in array.	The ARRAY command must not have more variables listed to the left of the array name than there are matching fields defined for the array. Check against the CREATE-ARRAY command.
003512	Only numeric variables and fields allowed with array arithmetic commands.	The ARRAY-ADD, ARRAY-SUBTRACT, ARRAY-MULTIPLY, and ARRAY-DIVIDE commands may have only numeric variables or literals as the source fields. Move the string data into a #numeric variable and then reference the #numeric variable.
003513	GET can only be used with \$string or #numeric variables.	You can move array fields only into \$string variables or #numeric variables. Correct the source line.
003514	PUT and GET variables must match array field types.	When moving data into or out of arrays, the source or destination variables must match the array fields in type. CHAR fields can be stored into/from strings, NUMBER fields into/from numeric variables. Check the CREATE-ARRAY command.
003515	More fields than variables found in array command.	The ARRAY command must not have more variables listed to the left of the array name than there are matching fields defined for the array. Check against the CREATE-ARRAY command.
003516	Too many arrays defined; maximum is `01.	Reduce the number of arrays needed by the program. `01 = Maximum number of arrays supported by this version of SQR
003517	Missing '=specifier' in qualifier: `01	Correct the syntax. 01 = Name of missing required parameter
003518	Duplicate array name: `01	Change the name of the array. `01 = Array name in question
003519	Too many fields defined; maximum is `01.	Reduce the number of fields. `01 = Maximum number of fields allowed per array

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003520	Missing ':type' in CREATE-ARRAY FIELD= `01	Correct the syntax. `01 = The name of the field
003521	Duplicate FIELD name: `01	Change the name of one of the fields. `01 = The name of the field
003522	Optional :nn for FIELD must be between 1 and 64K.	Correct the source line.
003523	CREATE-ARRAY FIELDS :type must be one of the following: `01	Correct the syntax.
003525	Missing NAME= in CREATE-ARRAY.	Correct the syntax.
003526	Missing or incorrect SIZE= in CREATE-ARRAY.	Correct the syntax.
003527	Missing FIELD= statements in CREATE-ARRAY.	Correct the syntax.
003528	Array dimensioned too large for PC in CREATE-ARRAY.	On PC-based systems, the maximum allocation that can be made is 65520 characters. The array as specified would exceed this limit. Reduce the number of entries.
003529	Missing or invalid initialization value for field `01.	Correct the syntax. 01 = Name of the field
003530	Invalid EXTENT= value in CREATE-ARRAY.	Correct the syntax.
003600	Missing 'ask' variable name.	Correct the syntax.
003603	WARNING: Substitution variables do not vary when saved with run-time.	Informational message.
003605	No substitution variable entered.	The C routine "fgets()" returned an error and SQR aborts the program run.
003700	Did not find end of paragraph: `01	Missing the END-paragraph command to match the specified paragraph. Correct the source file. `01 = BEGIN-paragraph in question
003701	Invalid command.	Check for a misspelled command.
003702	Command not allowed in this section: `01	Correct the syntax. `01 = Offending command name
003703	Paragraph not allowed inside procedure.	The BEGIN-paragraph command is not allowed here. Check your SQR code for a misplaced paragraph.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003704	Missing procedure name.	Correct the syntax.
003705	Extra argument found.	Correct the syntax.
003706	Missing Comma.	Correct the syntax.
003707	Bad Argument List.	The DO or BEGIN-PROCEDURE command has an error in its argument list, possibly extra characters after the final right parentheses. Correct the source line.
003708	Empty Argument.	The DO or BEGIN-PROCEDURE command has an error in its argument list, possibly two commas in a row inside the parentheses. Correct the source line.
003709	Only \$string and #number variables allowed for BEGIN-PROCEDURE parameters.	Correct the syntax.
003710	Unknown argument type.	An argument in a DO or BEGIN-PROCEDURE command is incorrect. Check for a misspelled variable type.
003711	Indicate :\$string or :#number returned values in BEGIN-PROCEDURE only.	Correct the syntax.
003712	Missing).	Correct the syntax.
003713	`01 paragraph not allowed with -XL option in effect.	Either use the #IF command to conditionally compile the program when -XL is being used or do not execute this SQR report with the -XL option. `01 = Name of the BEGIN-paragraph
003714	Bad database connection number.	The -Cnn value must be a non-zero value. Correct the source line.
003715	Did not find end of paragraph: `01 (No 'from...' clause found.)	Correct the source code. `01 = BEGIN-command in question
003716	Error in SQL statement.	The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement.
003717	Extra characters after expression continuation.	Remove the extra characters after the dash.
003718	Did not find end of expression.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003719	Columns names and expressions must be unique or be given unique pseudonyms (&name).	Columns retrieved from the database are assigned names by prepending an "&" to the beginning of the name. You are trying to select the same &column name more than once. Change the assigned &column name by using an alias after the name.
003720	Bad number specified for 'LOOPS=' on 'BEGIN-SELECT; Maximum is 32767'.	If your program logic requires that you stop processing after more than 32767 rows have been retrieved, you could count the rows manually and use the EXIT-SELECT command to break out of the SELECT loop.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [-Bnn] [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(DB2) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Informix) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]] [-DB=database]	(ODBC) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]] [BEFORE=procedure[(arg1[,argi]...)]] [AFTER=procedure[(arg1[,argi]...)]]	(DDO) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [-Bnn] [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Oracle) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [-XP] [LOOPS=nnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Sybase) Correct the syntax.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003721	Bad param found on 'BEGIN-SELECT' line; Format is: -SELECT [DISTINCT] [-Cnn] [-Bnn] [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Teradata) Correct the syntax.
003722	Could not set up cursor.	An error occurred while trying to compile the SQL statement. Look closely at any \$string variable references. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003723	Problem executing cursor.	An error occurred while trying to execute the SQL statement. Look closely at any \$string variable references. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003724	Could not exit query loop.	The database command to cancel the query returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
003725	Bad return fetching row from database.	The database returned an error status for the last row that was fetched, commonly due to the buffer not being large enough. If selecting expressions, make sure that the length of the first expression will be adequate for all rows selected.
003726	Literal in SQL expression missing closing quote.	Literals must be surrounded by single quotes ('). To embed a quote within a literal use two single quotes in sequence (""). Correct the source line.
003727	SQL expression not ended, perhaps parentheses not balanced.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003728	SQL expression not ended, perhaps missing &name.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003729	SQL expression is missing &name or has unbalanced parentheses.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(DB2) Correct the syntax.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Informix) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [-NR] [ON-ERROR=procedure(arg1[,argi]...)] [-DB=]	(ODBC) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Oracle) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]] [-Connection=]	(DDO) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [-XP] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Sybase) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Teradata) Correct the syntax.
003731	Did not find 'END-SQL' after 'BEGIN-SQL'.	Correct the source file.
003732	ON-ERROR= for 'BEGIN-SQL' in SETUP section must be STOP, WARN or SKIP.	Correct the syntax.
003733	Could not create procedure for SQL.	(Sybase) SQR could not create a stored procedure for the SQL statement. The most likely cause for failure is that the user name you are using to run the report under does not have the proper privileges. Either grant the user CREATE PROCEDURE privilege or use the -XP command line option to inhibit SQR from creating temporary stored procedures for SQL statements.
003734	Could not compile SQL.	Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003735	Could not execute SQL.	An error occurred while trying to compile the SQL statement. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003736	Please use BEGIN-SELECT - END-SELECT section for SELECT statements.	(Informix, ODBC, Oracle, DDO) Correct the source code.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003737	Bad fetch buffer count.	(Oracle, Sybase) The -B flag specifies an illegal value. Correct the source code.
003738	Report interrupted by request.	Informational message.
003741	Dynamic column must be \$string variable.	Correct the syntax.
003742	Dynamic column missing ``01'.	Correct the syntax. `01 = Missing character
003743	Dynamic columns must have a &pseudonym.	Correct the syntax.
003744	&pseudonym =type must be 'char', 'number', or 'date'.	Correct the syntax.
003745	Only a variable name may be between the ``01' and ``02' characters.	Correct the syntax. `01 = Leading character `02 = Trailing character
003746	When dynamic columns are used all non-dynamic columns and expressions must be defined with &name=type.	Add &name=type to all expressions and non-dynamic columns.
003747	When the table name is dynamic each column and expression must be defined with &name=type.	Add &name=type to all expressions and non-dynamic columns.
003748	When selecting multiple rowsets, they must have the same columns (order, type, width).	Non-contiguous rowsets selected in ROWSETS=().
003749	The highest numbered rowset named in ROWSET=() must be less than 10.	Rowset upper bound exceeded in ROWSETS=().
003750	Correct syntax is ROWSETS=(-n,m,n-m,i-) or (all).	Correct the syntax in ROWSETS=().
003751	Cannot select more than 25 Rowsets in a single BEGIN-SELECT.	Correct the syntax in ROWSETS=().
003752	One of the selected fields ('`01') was not found in the specified row set(s). The available fields are: ``02'.	Field not found in BEGIN-SELECT.
003753	The datasource returned no rowsets for the query.	Rowsets not available in BEGIN-SELECT.
003754	One of the specified rowsets was not found.	Specific Rowset not available in BEGIN-SELECT.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003755	Rowsets must be listed in ascending order, and may not include duplicates.	Bad rowset numbering sequence in BEGIN-SELECT.
003756	Column variable data types must be specified, either using 'TYPE =dtype' or '&pseudonym =dtype', where dtype must be 'char', 'number', or 'date'.	Correct the syntax.
003757	One of the variables ('`01') referenced in the ROWSETS=() portion of the BEGIN-SELECT block was not found.	Correct the syntax.
003800	Too many document paragraphs; maximum is `01.	There are too many BEGIN-DOCUMENT paragraphs. Reduce the number of DOCUMENT paragraphs needed by the program. `01 = Maximum number supported by this version of SQR
003801	Too many document markers; maximum is `01.	There are too many BEGIN-DOCUMENT paragraphs. Reduce the number of DOCUMENT paragraphs needed by the program. `01 = Maximum number supported by this version of SQR
003802	Duplicate document marker.	Give the document marker a unique name.
003803	Did not find 'END-DOCUMENT' after 'BEGIN-DOCUMENT'.	The BEGIN-DOCUMENT paragraph must end with END-DOCUMENT. Correct the source code.
003900	EXECUTE command is incomplete.	Correct the syntax.
003901	Bad -Cnn connection number for EXECUTE.	The -Cnn value must be a nonzero value. Correct the source line.
003902	@#Return_status must be #numeric (missing #).	(Sybase) Correct the source line.
003902	@#Return_status must be #numeric (missing #).	(ODBC) Correct the source line.
003902	@(# or \$)Return_status must be #numeric or \$variable.	(Oracle) Correct the source line.
003902	@#Return_status must be #numeric (missing #).	(DB2) Correct the source line.
003903	Missing '=' after `01.	Correct the source line. `01 = The parameter in question

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003904	Unknown variable type.	Variable names must begin with \$, #, or &. Correct the source line.
003905	OUT[PUT] variables for EXECUTE may only be \$variable or #variable.	Correct the syntax.
003906	The only EXECUTE option is WITH RECOMPILE.	Correct the syntax.
003907	You must EXECUTE ... INTO &columns.	Correct the syntax.
003908	Unknown datatype for EXECUTE...INTO &columns.	Check for a misspelled data type. If the data type is correct, then contact customer technical support so SQR can be updated.
003909	EXECUTE...INTO &columns must be unique.	The &column name assigned to the column must be unique throughout the report. Give the column a unique name.
003910	Missing (length) for datatype in EXECUTE.	Correct the source line.
003911	Datatype should not have (length) in EXECUTE.	Correct the source line.
003912	DO= in EXECUTE requires INTO... variables.	Correct the syntax.
003913	Could not EXECUTE stored procedure.	(Sybase) Record the database error message displayed with this message. If needed, contact your system administrator.
003913	Could not EXECUTE stored procedure.	(ODBC) Record the database error message displayed with this message. If needed, contact your system administrator.
003913	Could not EXECUTE stored procedure or function.	(Oracle) Record the database error message displayed with this message. If needed, contact your system administrator.
003913	Could not EXECUTE stored procedure or function.	(DB2) Record the database error message displayed with this message. If needed, contact your system administrator.
003914	Bad return fetching row from database.	Record the database error message displayed with this message. If needed, contact your system administrator.
003915	Could not set up EXECUTE cursor.	The database server returned an error while trying to compile the SQL statement needed to set up the EXECUTE command.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
003918	Missing Stored Procedure or Function - ``01'.	(Oracle) A describe of the store procedure or function could not be performed. `01 = Procedure or function name
003918	Missing Stored Procedure - ``01'.	(DB2) A describe of the store procedure or function could not be performed. `01 = Procedure or function name
003919	The EXECUTE command is not supported when using the Oracle 7 API.	(Oracle) The EXECUTE command is not supported when using the Oracle 7 API.
003920	The EXECUTE command had an erroneous return status data type of ``02' declared. The Stored Function expects a return status of ``01'.	(Oracle) The wrong data type has been specified for the return status of a Stored Function. `01 = Expected Data Type `02 = Return Data Type
003921	The EXECUTE command detected an erroneous data type of ``02' declared for the IN/OUT parameter - name/position: `03/`04. The Stored Function or Procedure expects a parameter data type of ``01'.	(Oracle) The wrong data type has been specified for the return variable of a Stored Procedure. `01 = Expected Data Type `02 = Return Data Type `03 = Name of Parameter `04 = Position of Parameter
003921	The EXECUTE command detected an erroneous data type of ``02' declared for the IN/OUT parameter - name/position: `03/`04. The Stored Procedure expects a parameter data type of ``01'.	(DB2) The wrong data type has been specified for the return variable of a Stored Procedure. `01 = Expected Data Type `02 = Return Data Type `03 = Name of Parameter `04 = Position of Parameter
003922	The maximum number of allowable IN/OUT parameters was reached. No more than ``01' can be processed by the EXECUTE command.	(DB2) The maximum number of allowable IN/OUT parameters was reached. `01 = Maximum Number of Parameters
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

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
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
004000	Result #variable or \$variable or '=' missing in expression.	The LET command is not properly formatted. Correct the source line.
004001	Expression too complex.	The expression is either too long or is too deeply nested. Break the expression into smaller expressions.
004002	Parentheses unbalanced in expression.	A left or right parenthesis is missing. Correct the source line.
004003	Too many variables; maximum is `01'.	Break the expression into smaller expressions. `01 = Maximum number supported by this version of SQR
004004	Empty expression.	The expression is invalid. Correct the source line.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004005	Extra comma in expression.	An argument is missing after a comma in the expression. Correct the source line.
004006	Unknown operator ``01'. Do you mean `02 ?	The concatenation operator is . Correct the source line.
004007	Too many &column forward references in expression; maximum is `01.	The expression contains too many forward references. Break the expression into smaller expressions. `01 = Maximum number supported by this version of SQR
004008	Unknown function or variable in expression: `01	The specified function is not an SQR built-in function nor does it exist in the user-modifiable file UFUNC.C. Check for a misspelled function name. `01 = Function name
004009	Function ``01' missing parentheses.	All functions in an expression must be followed by their arguments enclosed in parentheses. Correct the source line.
004010	Empty parentheses or expression.	A pair of parentheses were found with nothing inside them. Remove the () in question from the source line.
004011	User function ``01' has incorrect number of arguments.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004012	Function ``01' has incorrect number of arguments.	Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name
004013	Missing operator in expression.	Correct the source line.
004014	Operator ``01' missing argument.	Correct the syntax of the function. Functions are described under the LET command. `01 = Operator
004015	Function ``01' missing argument.	Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name
004016	Function or operator ``01' missing arguments.	Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004017	User function ``01' requires character argument.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004018	User function ``01' requires numeric argument.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004019	User function ``01' requires \$string variable.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004020	User function ``01' requires #numeric variable.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004021	User function ``01' has incorrect argument type list. Must be of: c,n,C,N	The UFUNC.C file has a bad definition for the specified function. Correct the UFUNC.C program file; recompile UFUNC.C; and recreate the SQR executable. `01 = User function name
004022	User function ``01' missing arguments.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004023	User function ``01' has incorrect return type. Must be c or n.	The UFUNC.C file has a bad definition for the specified function. Correct the UFUNC.C program file; recompile UFUNC.C; and recreate the SQR executable. `01 = User function name
004024	'isnull' requires a &column, \$string, or \$date argument.	#numeric variables cannot be NULL. Correct the source line.
004025	'nvl' requires a &column, \$string, or \$date as its first argument.	#numeric variables cannot be NULL. Correct the source line.
004026	Function or operator ``01' requires character argument.	Correct the source line. `01 = Function or operator
004027	Function or operator ``01' requires numeric argument.	Correct the source line. `01 = Function or operator

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004028	IF or WHILE expression must return logical result.	The expression used must evaluate a statement that will be TRUE or FALSE. Correct the source line.
004029	Attempting division by zero in expression.	The expression tried to divide a number by zero. Use the COND() function to check if the divisor is zero; then divide by something else (for example, 1).
004030	Attempting division by zero with '%'.	An attempt was made to divide a number using the "%" operator. Use the COND() function to check if the divisor is zero; then divide by something else (for example, 1).
004031	The number used with '%' (`01) is out of range.	The "%" operator works with integers only. Correct the program logic. `01 = Maximum value allowed
004032	User function has unknown return type -- expecting n or c -- need to recompile Run-Time file?	SQR detected an error while processing a user defined function. If you are running an .sqt file, it probably needs to be recompiled because the user function has changed its definition. If you are running an .sqr file, then you need to correct the UFUNC.C program file; recompile UFUNC.C, and recreate the SQR executable.
004033	In user function use C type with allocated string to change \$variable.	SQR detected an error while processing a user defined function. Correct the UFUNC.C program file recompile UFUNC.C and recreate the SQR executable.
004034	Could not find array `^01' in ARRAY function.	Check for a misspelled array name. `01 = Array name
004035	Could not find array field ``01' in ARRAY function.	Check for a misspelled array field name. `01 = Array field name
004036	Math error in expression (usually over- or under-flow).	Most of the SQR mathematical built-in functions have a corresponding C library routine. One returned an error. Break the expression into discrete expressions in order to identify the function that caused the error.
004037	Error executing expression.	Record the steps leading up to the error and contact technical support.
004038	Out of space while processing expression; Use -Mfile to increase EXPRESSIONSPACE.	The expression requires more temporary string storage than is currently allocated. Use the -Mfile flag on the command line to specify a file that contains an entry that increases by a greater value than is currently defined.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004039	'`01' assumed to be a variable name, not an expression.	Warning message. `01 = Expression in question
004040	The array '`01' has not been defined.	Define the array using the CREATE-ARRAY command. `01 = Array name
004041	The field '`01' is not valid for array '`02'.	Correct the source code. `01 = Field name `02 = Array name
004042	The array reference '`01' has an incorrect number of parameters specified.	Correct the source code. `01 = Array name
004043	The array reference '`01' requires numeric parameters for the element and occurs arguments.	Correct the source code. `01 = Array name.
004045	Function or operator '`01' requires date argument.	Correct the source code. `01 = Array name
004046	Incompatible types between expression and variable.	Correct the source code.
004047	The field '`01' is must be 'char' or 'float'.	Correct the source code. `01 = Field name
004048	Function or operator '`01' must be a string or date argument.	Correct the source line. `01 = Function or operator
004049	Unknown transform value '`01' in TRANSFORMATION function.	Check for a misspelled transform value. `01 = Transform value
004100	Use 'print' command to format data outside SELECT query.	You must precede PRINT command arguments (WRAP, ON-BREAK.) with an explicit PRINT command when outside of a BEGIN-SELECT paragraph. Correct the source line.
004101	Cannot find required parameter.	Correct the syntax.
004102	Bad number found.	A command expecting a numeric literal or :#numeric variable reference found an illegal number definition or a reference to a string variable or column. Correct the source line.
004103	Cannot find required numeric parameter.	Correct the syntax.
004104	Cannot find placement parameters.	The position qualifier "(Row,Col,Len)" was not found. Check for a missing parentheses.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004105	Placement parameter incorrect.	The "Row", "Column" or "Length" fields are invalid or ill-formed. Correct the source line.
004106	Invalid second function on line.	An SQR command used as a qualifier for a primary command (for example, PRINT) is incorrect. Correct the source line.
004107	Second function must be FORMAT type.	The PRINT command may have format command qualifiers such as WRAP, CENTER, or FILL. Other qualifier commands are not permitted.
004108	Missing operator =, <, >, ...	Correct the source line.
004109	Invalid operator.	Correct the source line.
004110	Missing variable.	Correct the syntax.
004111	Please give this expression a &pseudonym.	Expressions selected in a BEGIN-SELECT paragraph should be given an &Name or be followed by a print position "(Row,Col,Len)". Correct the source line.
004112	Wrong variable type.	Correct the syntax.
004113	Command incomplete, expected ``01'.	Correct the syntax. `01 = What was expected
004114	Expecting ``01', found ``02'.	Correct the syntax. `01 = What was expected `02 = What was encountered
004115	Unknown command or extra parameters found (missing quotes?).	Correct the syntax.
004116	Duplicate references to parameter ``01'.	Correct the syntax. `01 = Duplicated parameter
004117	Unexpected equal sign found with ``01'.	Correct the syntax. `01 = Parameter name
004118	Qualifier ``01' cannot be used with the following qualifiers:	Correct the syntax. `01 = Qualifier name
004119	Expecting numeric column, found string column.	Correct the syntax.
004120	Date variables (`01) cannot be used with this command.	Correct the syntax. `01 = Parameter name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004200	Page width and depth must be > 0 and < 32767.	The values specified with the PAGE-SIZE command are out of bounds. Specify legal values.
004201	Page buffer must be < 65536 on PC SQR.	The maximum allocation on a PC is 65536. The Page-Depth * Page-Width cannot exceed this value. Reduce the Page-Depth or Page-Width.
004202	Cannot generate line printer output for this report because position qualifier(s) may be out of range. If you are running this report from Hyperion SQR Developer, omit the -PRINTER command line option. Otherwise specify PRINTER:{HPEH,HT,PS,WP} for graphical printer output.	The report output cannot be generated for a Line Printer. If your report was designed for a graphics printer, specify -PRINTER:{HPEH,HT,PS,WP} for graphical printer output.
004300	Missing end of placement (...) in SHOW.	The placement parameter is ill-formed. Correct the source line.
004301	Bad (...) location in SHOW.	Screen positions must be valid numbers. Correct the source line.
004302	Missing literal or variable name to EDIT in SHOW.	The literal or variable name must immediately precede the EDIT, NUMBER, MONEY, or DATE keywords.
004303	Missing edit mask in SHOW.	The word EDIT must be followed by a valid edit mask. Correct the source line.
004304	Only string variable allowed for dynamic edit mask.	Dynamic edit masks may only be stored in \$Variables. Correct the line.
004305	Unknown option for SHOW.	Correct the syntax.
004406	Number `01 not allowed.	Use a different value. `01 = Internal number
004407	Referenced variables not defined:	References were made to column variables (&var) that were not defined in the program. The list of variable names follows this message.
004501	Use '+' and negate variable for reverse relative placement.	The use of "-#Variable" is not legal here. Negate the #Variable value and use "+#Variable".
004503	Fixed line placement #variable must be > 0. Use relative positioning, (+#line,10,0).	Correct the source line as indicated.
004504	Fixed column placement #variable must be > 0. Use relative positioning, (5,+#col,0).	Correct the source line as indicated.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004505	Length placement #variable must be >= 0.	The length field cannot be a negative value. Correct the source line.
004600	CODE not appropriate for numeric data.	The CODE qualifier to the PRINT command may only be used for text fields. Move the “#Variable” to a “\$Variable” first and then print the “\$Variable”.
004601	Unknown option for GRAPHIC command: BOX, HORZ-LINE, VERT-LINE or FONT	Correct the syntax.
004602	GRAPHIC BOX out of bounds. Row: `01, Column: `02, Width: `03, Depth: `04	SQR aborts the program run. `01 = Row `02 = Column `03 = Width `04 = Depth
004603	GRAPHIC VERT-LINE out of bounds. Row: `01, Column: `02, Length: `03	SQR aborts the program run. `01 = Row `02 = Column `03 = Length
004604	GRAPHIC HORZ-LINE out of bounds. Row: `01, Column: `02, Length: `03	SQR aborts the program run. `01 = Row `02 = Column `03 = Length
004700	Cannot open the program file: `01`(`02): `03	Depends on the system error message. `01 = Name of the program file `02 = System error code `03 = System error message
004701	Cannot logon to the database.	The connectivity information is either incorrect or the database server is unavailable. Check the connectivity information and the server availability.
004702	Line found outside paragraph.	All commands must be within BEGIN-... END-statements. Correct the source code.
004703	Cannot close the program file. (`01): `02	Depends on the system error message. `01 = System error code `02 = System error message
004704	#ENDIF not found for #IF.	Missing an #ENDIF to complete conditional compilation. Correct the source code.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004705	Program line too long; maximum is `01.	Break the program line into smaller lines. `01 = Maximum line length supported by this version of SQR
004706	Substitution variable {`01} would cause this line to exceed the maximum line length of `02 characters.	The substitution variable value would cause this line to exceed the maximum line size. Break the program line into smaller lines. `01 = Name of the substitution variable `02 = Maximum line length supported by this version of SQR
004707	No value found for substitution variable: {`01}	An empty value was found for the substitution variable. Check for a misspelled name. `01 = Name of the substitution variable
004708	#ELSE without preceding #IF.	Missing an #IF or #IFDEF or #IFNDEF to begin conditional compilation. Correct the source code.
004709	#ENDIF without preceding #IF.	Missing an #IF or #IFDEF or #IFNDEF to begin conditional compilation. Correct the source code.
004710	#IF's nested too deeply; maximum is `01.	Reduce the number of nested #IF directives. `01 = The maximum depth supported by this version of SQR
004711	#INCLUDE files nested too deeply; maximum is `01.	Reduce the number of nested #INCLUDE directives. `01 = The maximum depth supported by this version of SQR
004712	Include file name too long; Modify -I flag.	The combined -I directory name with the #INCLUDE file name exceeds the maximum length permitted for a complete pathname. Check the spelling of both the -I command flag and the #INCLUDE filename.
004713	Cannot open the #INCLUDE file: ``01'(`02): `03	`01 = Include file name `02 = System error code `03 = System error message
004714	Cannot close the #INCLUDE file: ``01'(`02): `03	`01 = Include file name `02 = System error code `03 = System error message
004716	'BEGIN-PROGRAM' command not found in program.	This section is required for all reports. Correct the source code.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004717	Cannot open the report output file: `^01'(`02):`03	`01 = Output file name `02 = System error code `03 = System error message
004719	Cannot logoff the database.	The database server returned an error while trying to log off from the database. SQR aborts the program run.
004720	Cannot open the run-time file: `^01'.(`02):`03	SQR aborts the program run. `01 = Run-Time file name `02 = System error code `03 = System error message
004721	Cannot close the run-time file. (`01):`02	SQR aborts the program run. `01 = System error code `02 = System error message
004722	Error reading the run-time file. (`01):`02	SQR aborts the program run. `01 = System error code `02 = System error message
004723	Run time file must be recreated for this version of SQR.	The run-time file was created by a earlier version of SQR and is incompatible with the current version. Recreate the .sqt (run-time) file.
004724	The -XL option cannot be specified with this run-time file because access to the database is required.	Do not use the -XL option.
004725	Cannot open cursor.	The database server returned an error indicating that a new database cursor or logon could not be completed. See the error message from the database server.
004726	Cannot create procedure for SQL statement.	(Sybase) SQR could not create a stored procedure for the SQL statement. The most likely cause for failure is that the user name you are running the report under does not have the proper privileges. Either grant the user CREATE PROCEDURE privilege or use the -XP command line option to inhibit SQR from creating temporary stored procedures for SQL statements.
004727	Error writing the run-time file. (`01):`02	`01 = System error code `02 = System error message
004729	Cannot find inactive database cursor. Program too large.	(DB2, Oracle) The program has too many concurrent database cursors. Reduce the complexity of the program.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004730	Run-time saved in file: `01	Informational message. `01 = Name of the .sqt file created
004734	Cannot select user.	SQR could not select the default user name from the database. Contact your system administrator.
004735	Unknown variable type encountered in run-time file: `01	SQR aborts loading the run-time file. `01 = Variable type
004736	Unexpected End-Of-File while processing the run-time file.	SQR aborts loading the run-time file.
004737	Cannot load the run-time file because it was built for the `01database and `02 is built for the `03 database.	SQR aborts loading the run-time file. `01 = Database name from run-time file `02 = SQR image name `03 = Database that SQR is built for
004738	'END-REPORT' not paired with 'BEGIN-REPORT'.	Correct the source code.
004739	'END-PROGRAM' not paired with 'BEGIN-PROGRAM'.	Correct the source code.
004743	#INCLUDE filename must be enclosed in quotation marks.	Correct the syntax.
004744	#INCLUDE command format is: #Include 'filename'.	Correct the syntax.
004745	Array field (`01.`02) specification exceeds the PC 64K limit.	Reduce the size of the field requirements. `01 = Array name `02 = Field name
004746	Layout ``01' specifications exceeds the PC 64K limit.	The layout is too large for the PC version of SQR to handle. `01 = Layout name
004747	The SQT file is corrupted and cannot be processed.	SQR aborts loading the run-time file.
004748	The user function ``01' needs to be defined as entry `02 in the user function table. It requires a definition of: Return Type = ``03' Arg Count = `04 Arg Types = ``05"	The SQT file requires that the specified user function be defined. `01 = User function name `02 = Entry in the user function table `03 = Return type `04 = Argument count `05 = Argument types

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
004749	An attempt was made to move `01 characters into ``02'. The maximum allowed is `03 characters.	An attempt was made to move too much data into an SQR string variable. `01 = Number of characters to be moved `02 = Variable name `03 = Maximum characters allowed
004750	SQR has reached the architectural limit for `01'(`02).	While attempting to increase an internal table SQR reached its architectural limit for that table. Processing will stop as SQR cannot continue. `01 = Internal table classification `02 = Architectural limit
004802	PRINTER TYPE must be HTML, HPLASERJET, POSTSCRIPT, or LINEPRINTER.	Correct the syntax.
004805	Both BEFORE-BOLD and AFTER-BOLD must be specified.	Correct the syntax.
004807	Unknown DECLARE qualifier.	Correct the syntax.
004901	Date variables (`01) cannot be used in BEGIN-SQL or BEGIN-SELECT paragraphs.	Correct the source code. `01 = Variable name
005000	Report ``01' heading section size exceeds the page depth.	Reduce the size of the heading or increase the page depth.
005001	Report ``01' footing location must be less than the page depth.	Reduce the size of the footing or increase the page depth.
005002	Check 'BEGIN-HEADING' commands: Discovered 2nd page-initialization while heading in progress.	The BEGIN-HEADING procedure either caused an overflow of the current page or it issued a command that caused a page eject to occur. Check any procedure invoked by the BEGIN-HEADING section to ensure that the commands do not overflow the page or cause a page eject.
005003	Check 'BEGIN-FOOTING' commands; perhaps number of footing lines is too small. Discovered 2nd page-write while footing in progress.	The BEGIN-FOOTING procedure either caused an overflow of the current page or it issued a command that caused a page eject to occur. Check any procedure invoked by the BEGIN-FOOTING section to ensure that the commands do not overflow the page or cause a page eject.
005004	Attempt to execute the `01 command while processing the `02 section.	Change the SQR program logic to prevent the command from executing while the specified section is active. `01 = Command name `02 = Section name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005005	Report ``01' already has been assigned a `02 section.	Correct the source code. `01 = Report name `02 = Duplicated section name
005006	You cannot define more than one default ``01' section.	Correct the source code. `01 = Duplicated section name
005007	Report ``01' has overlapping heading and footing sections.	Correct the source code. `01 = Report name
005008	TOC ``01' already has been assigned a `02 section.	Correct the source code. `01 = Table of Contents name `02 = Duplicated section name
005009	The name can only contain characters [0-9 A-Z _ -].	Correct the source code.
005010	The name cannot be the reserved names 'none' or 'default'.	Correct the source code.
005011	This name has already be used.	Correct the source code.
005012	The specified `01 (`02) does not exist.	Correct the source code. `01 = Heading or Footing `02 = Heading/Footing name
005013	FOR-REPORTS and FOR-TOC cannot be specified when NAME= is used.	Correct the source code.
005014	TOC (`01) has already been defined as the default.	Correct the source code. `01 = Default TOC name
005100	'IF', 'WHILE', 'EVALUATE' commands nested too deeply; maximum is `01.	Reduce the nested commands. `01 = Maximum depth allowed by this version of SQR
005101	'BREAK' found outside 'WHILE' or 'EVALUATE' statement.	The BREAK command is valid only in the context of a WHILE or EVALUATE statement. Correct the source code.
005103	END-WHILE found without matching 'WHILE'.	Correct the source code.
005104	'IF' or 'EVALUATE' command not completed before 'END-WHILE'.	Correct the syntax.
005105	'ELSE' found without matching 'IF'.	ELSE can be used only within the context of an IF command. Correct the source code.
005106	Single 'ELSE' found inside 'WHILE' or 'EVALUATE' statement.	ELSE can be used only within the context of an IF command. Correct the source code.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005107	Only one 'ELSE' allowed per 'IF'.	Rewrite the source code to use nested IF statements.
005108	Found 'END-IF' without matching 'IF'.	Each IF command must have a matching END-IF command. Correct the source code.
005109	'WHILE' or 'EVALUATE' command not completed before 'END-IF'.	You are missing a closing END-WHILE or END-EVALUATE command before END-IF. IF, WHILE, and EVALUATE statements can be nested, but they cannot cross each other's boundaries. Each inner statement must be complete before a closing statement is ended. Correct the source code.
005110	EVALUATE statements nested too deep; maximum is `01.	Reduce the number of nested statements. `01 = Maximum depth supported by this version of SQR
005111	'WHEN' found outside 'EVALUATE' clause.	WHEN may be used only in the context of an EVALUATE clause. Correct the source code.
005112	'IF' or 'WHILE' not completed before 'WHEN' statement.	Correct the syntax.
005114	Incorrect types for comparison. Both must be of the same type (string, numeric or date).	Correct the source line.
005115	'When-other' found outside 'Evaluate' statement.	WHEN can be used only in the context of an EVALUATE statement. Correct the source code.
005116	'IF' or 'WHILE' not ended before 'WHEN-OTHER' command.	Correct the syntax.
005117	Only one 'WHEN-OTHER' allowed per 'EVALUATE'.	Correct the syntax.
005118	Found 'END-EVALUATE' without matching 'EVALUATE'.	Each EVALUATE command must have a matching END-EVALUATE command. Correct the source code.
005119	'IF' or 'WHILE' command not completed before 'END-EVALUATE'.	Correct the syntax.
005120	'WHEN-OTHER' must be after all 'WHEN's.	Correct the syntax.
005121	No 'WHEN's found inside 'EVALUATE' statement.	Correct the syntax.
005122	'IF', 'EVALUATE' and 'WHILE' statements cannot cross sections or paragraphs.	These commands must be contained within a single section or paragraph. Correct the source code.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005123	'CONTINUE' found outside 'WHILE' statement.	The CONTINUE command is valid only in the context of a WHILE statement. Correct the source code.
005200	Did not find '>' after <....	A leading left angled bracket "<" indicates that you are beginning an ASCII value, which must be ended by a right angled bracket ">". Correct the source line.
005201	Bad ascii character in <...>.	Numbers in angled brackets <> must be between 1 and 255. Correct the source line.
005202	Bad ascii number in <...>.	Numbers in angled brackets <> must be between 1 and 255. Correct the source line.
005203	<...> string is too long; maximum is `01 characters.	Reduce the length of the string. If this is not possible, use a PRINT-DIRECT command in a BEGIN-REPORT or END-REPORT procedure. `01 = Maximum number of characters supported by this version of SQR
005300	Did not find '=' after qualifier: `01	Correct the syntax. `01 = Qualifier name
005301	Qualifier ``01' requires a numeric value.	Correct the syntax. `01 = Qualifier name
005302	Incorrect value for qualifier ``01'. Valid values are:	Correct the source line. `01 = Qualifier name
005303	Invalid qualifier ``01'. Valid qualifiers are:	Correct the source line. `01 = Qualifier name
005304	Qualifier ``01' requires a numeric literal, variable, or column.	Correct the source line. `01 = Qualifier name
005305	Qualifier ``01' references a numeric variable that has not been defined.	Correct the source line. `01 = Qualifier name
005306	Qualifier ``01' requires a string literal, variable, or column.	Correct the source line. `01 = Qualifier name
005307	List not terminated.	Correct the syntax.
005308	Missing comma in list.	Correct the syntax.
005309	Required argument ``01' was not specified.	Correct the source line. `01 = Qualifier name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005310	Qualifier ``01' has already been specified.	Correct the source line. `01 = Qualifier name
005311	Qualifier ``01' requires a string literal.	Correct the source line. `01 = Qualifier name
005312	Qualifier ``01' requires a list of values: (val [,val]...).	Correct the source line. `01 = Qualifier name
005313	Qualifier ``01' requires a integer value.	Correct the source line. `01 = Qualifier name
005314	Invalid character in variable name ``01'.	Correct the source line. `01 = Invalid character
005315	Qualifier ``01' references a string variable that has not been defined.	Correct the source line. `01 = Qualifier name
005316	Qualifier ``01' uses an invalid Unit-Of-Measure suffix. Valid suffixes are: dp pt mm cm in	Correct the source line. `01 = Qualifier name
005317	Qualifier ``01' can only reference string literals or variables.	Correct the source line. `01 = Qualifier name
005318	Qualifier ``01' can only reference string or numeric literals.	Correct the source line. `01 = Qualifier name
005400	Second page write attempted while writing current page. Check BEFORE-PAGE, AFTER-PAGE procedures.	Check any procedure invoked by the BEFORE-PAGE or AFTER-PAGE procedures to ensure that the commands do not overflow the page or cause a page eject.
005402	String cannot be placed on page: `01 -- placement specified is out of range. (`02,`03,`04)	Ensure the values are within the page limits. `01 = Text value `02 = Row `03 = Column `04 = Length
005403	Error writing the output file. (`01): `02	`01 = System error code `02 = System error message
005404	Cannot open the Postscript startup file: `01 (`02): `03	`01 = Name of the file `02 = System error code `03 = System error message
005405	SQR trial copy exiting after `01 pages.	`01 = Number of pages.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005406	Exiting after requested number of test pages (`01).	`01 = Number of pages
005408	Program stopped by user request.	Informational message.
005500	Cannot set parse_only option.	(Sybase) The DB-Library routine dbsetopt() returned an error. This should never happen. Contact technical support.
005501	Cannot reset parse_only option.	(Sybase) The DB-Library routine dbclropt() returned an error. This should never happen. Contact technical support.
005502	Cannot drop SQR generated stored procedure: `01.	(Sybase) See the database server error message that was also output. This should never happen. Contact technical support. `01 = Stored procedure name
005503	Cannot use `01 datatype as bind variable.	(Sybase) Use another database column. `01 = The database datatype.
005504	Unknown datatype for bind variable: `01 Cannot create stored procedure.	(Sybase) Please contact technical support. `01 = Unknown database datatype
005505	SQL too large to create stored procedure.	(Sybase) The size of the SQL text needed to create the stored procedure is too large for SQR to handle. Add the -XP option to the BEGIN-SQL or BEGIN-SELECT command.
005506	SQR's EXECUTE command not available for this version of Sybase.	(Sybase) Some early versions of Sybase SQL Server or Microsoft SQL Server do not support Remote Procedure Calls (RPCs). Update your database server.
005507	Could not add param to remote procedure call.	(Sybase) A DB-Library routine returned an unexpected error. See the error message from the database.
005508	The number of EXECUTE...INTO &columns does not match the procedure.	(Sybase) Check the definition for the stored procedure you are referencing.
005509	Incorrect number of INTO &columns defined in EXECUTE.	(Sybase) Check the definition for the stored procedure you are referencing.
005510	Error converting OUTPUT Sybase type for EXECUTE.	(Sybase) The DB-Library routine dbconvert() failed to convert the data from the stored procedure. Contact technical support.
005511	Number of OUTPUT parameters from EXECUTE is incorrect.	(Sybase) Check the definition for the stored procedure you are referencing.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005512	Missing default database name for USE.	(Sybase) Correct the syntax.
005512	Missing default database name for USE.	(ODBC) Could not connect to the specified datasource.
005513	You may only specify 'USE db' once, before any SQL statements are executed.	(Sybase) Only one USE command is allowed in a report. Place the SETUP section at the beginning of the SQR report.
005515	Undefined variable referenced in -DB flag: `01	(ODBC) Check for a misspelling. `01 = Variable name
005523	Database commit failed.	The database command to perform a commit returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
005524	Cannot close database cursor.	The database command to close the database cursor returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
005528	DB2 SQL `01 error `02 in cursor `03: INFORMIX SQL `01 error `02 (ISAM: `03) in cursor `04: `05	(DB2) `01 = Routine name `02 = Error code `03 = SQR cursor number (Informix) `01 = Routine name `02 = Error code `03 = ISAM code `04 = SQR cursor number `05 = Error message from database
	ODBC SQL `01 error `02 in cursor `03: `04	(ODBC) `01 = Routine name `02 = Error code `03 = SQR cursor number `04 = Error message from database
	ODBC SQL `01 error `02 in cursor `03: `04	(DDO) `01 = Routine name `02 = Error code `03 = SQR cursor number `04 = Error message from database

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
	ORACLE `01 error `02 in cursor `03: `04	(Oracle) `01 = Routine name `02 = Error code `03 = SQR cursor number `04 = Error message from database
	Sybase `01 error in cursor `02: `03	(Sybase) `01 = Routine name `02 = SQR cursor number `03 = Error message from database
	Teradata SQL `01 error `02 in cursor `03:	(Teradata) `01 = Routine name `02 = Error code `03 = SQR cursor number
005532	System 10 files are missing.	(Sybase) Contact your system administrator.
005533	Not a System 10 SQL Server.	(Sybase) The CT-Library version of SQR can only connect to a System 10 server. Use the DB-Library version of SQR to connect to a pre-System 10 server.
005534	SQL too long for PREPARE/DECLARE; maximum `01 characters.	(DB2) The SQL statement is too large. `01 = Maximum number of characters supported by this version of SQR
005534	SQL too long for PREPARE/DECLARE; maximum `01 characters.	(Teradata) The SQL statement is too large. `01 = Maximum number of characters supported by this version of SQR
005536	Unknown error message number: `01.	(DB2) `01 = Error message number
005537	Empty error message returned from system for error number: `01.	(DB2) `01 = Error message number
005538	Invalid SELECT statement; COMPUTE clauses are not supported.	(Sybase) The select statement contains a COMPUTE clause that is not supported.
005539	Could not connect to datasource specified in -db variable: `01'.	(ODBC) Could not connect to the specified datasource.
005540	Not connected to a database, database access is not allowed.	The SQR program is no longer connected to a database. Commands that access the database can no longer be used. This situation can occur if the CONNECT fails and the ON-ERROR option was used.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005543	Specify the Oracle DLL name in the SQR.INI file in [Environment:Oracle] section for ORACLE_DLL entry, such as ORACLE_DLL=orant71.dll	(Oracle) SQR was unable to load the Oracle DLL. By default, SQR looks first for "ociw32.dll" or the DLL specified by the ORACLE_DLL entry in the [Environment:Oracle] section of the SQR.INI file. If that DLL could not be loaded, then SQR attempts to load 'orant71.dll'.
005600	GETWRD: Word too long; maximum is `01.	Reduce the length of the "word". `01 = Maximum size of a "word" supported by this version of SQR
005700	Cannot call SQR recursively.	SQR cannot be called recursively. This error can only occur if a User Function from either UFUNC.C or UCALL.C calls the sqr() routine. Do not call sqr() from a UFUNC.C or UCALL.C routine.
005701	Too many SQR command line arguments; maximum is `01	To pass more than this number of arguments, use a @file argument file containing one argument per line. `01 = Maximum number supported by this version of SQR.
005702	Log file name specified is too long.	Reduce the length of the log file name.
005703	Error opening the SQR log file: ``01' (`02):`03	`01 = Name of the file `02 = System error code `03 = System error message
005704	Missing program name.	The name of the program file was not found on the command line. The program name must be the first parameter on the command line.
005705	Program file name specified is too long.	Reduce the length of the program file name.
005707	Error opening the -E error file: ``01' (`02):`03	`01 = Name of the file `02 = System error code `03 = System error message
005708	Cannot find `01 in SQRDIR, PATH or \SQR.	The specified file cannot be located in any of the directories pointed to by the mentioned environment variables or default directories. Make sure the "file" is present in one of the locations searched. `01 = File name
005709	`01 environment variable is not defined.	As of version 2.5, the environment variable SQRDIR must be defined. `01 = Name of the environment variable

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005710	`01 path too long.	The length of the directory path plus the length of the file name to be opened is too long for SQR to handle. Reduce the length of the directory path. `01 = Environment variable name
005711	Bad number in -T test flag.	The number specified must be > zero. Correct the value.
005712	-G option requires arguments.	(VAX) The command line option is ill-formed. Correct the syntax.
005713	Too many arguments to -G option; maximum is `01.	(VAX) The command line option is ill-formed. Correct the syntax. `01 = Maximum number of arguments supported by this version of SQR
005714	-G attribute too long; maximum is `01.	(VAX) The command line option is ill-formed. Correct the syntax. `01 = Maximum number of each attribute supported by this version of SQR
005716	Unknown flag on command line: `01	Correct the syntax. `01 = Unknown command line flag
005717	Cannot open channel to TT; status = `01	(VAX) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System status
005718	Cannot read from TT; status = `01	(VAX) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System status
005719	Cannot close channel to TT; status = `01	(VAX) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System status
005720	Error opening tty. (`01): `02	(DG, UNIX) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005721	Error with 'ioctl()'. (`01): `02	(DG, UNIX) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005722	Error reading tty. (`01): `02	(DG, UNIX) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005723	Error closing tty. (`01): `02	(DG, UNIX) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005724	Bad number in -B flag.	(Oracle, Sybase) The number specified must be > zero. Correct the value.
005734	No program name given.	The report name must be the first command line argument.
005737	Unknown printer type specified with -PRINTER: switch.	The printer type can be EH, HT, LP, HP, PS, or WP. WP is valid only with PC/Windows.
005738	Database name needs to be included with -DB switch.	(ODBC) Could not connect to the specified datasource.
005738	Database name needs to be included with -DB switch.	(Sybase) Supply the database name.
005739	Too many -F switches; maximum is `01.	Reduce the number of -F switches. `01 = Maximum number allowed
005742	Attempt to invoke viewer (using WinExec) failed; error code = `01.	(Windows) `01 = System error code
005743	Unknown numeric type specified with -DNT: switch.	Correct the command line.
005744	-DNT:Decimal precision (`01) is out of range (`02 - `03).	Correct the command line. `01 = Specified precision `02 = Minimum allowed `03 = Maximum allowed

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005745	The specified default numeric type `01 = `02' is invalid.	Correct the SQR.INI file entry. `01 = Entry `02 = Value
005746	The decimal precision `01 = `02' is out of range (`03 - `04).	Correct the SQR.INI file entry. `01 = Entry `02 = Value `03 = Minimum allowed `04 = Maximum allowed
005747	The following error(s) occurred while processing the [`01] section from the SQR.INI file.	See the error message(s) that follow. `01 = Name of the section
005750	The -Burst switch is not properly formatted.	The "Burst" command line flag is not properly formatted.
005751	The -Burst switch cannot be used with the -NOLIS switch.	The "Burst" command line flag cannot be specified when the -NOLIS command line flag is also specified.
005752	The -Burst switch requires either the -Printer:HT or -Printer:EH switch to be specified.	The "Burst" command line flag is applicable only when HTML code is produced. You must specify either the -PRINTER:HT or -PRINTER:EH switch.
005753	The -Burst:S and -Burst:T switches can only be used against an SPF file which was generated with SQR v4.1 and above.	The "Burst" command line flag can only be specified when processing a SPF file that was generated by SQR v4.1 and above. Older SPF files do not contain the proper information that permits bursting.
005754	The -Burst switch caused no output to be generated.	The "Burst" command line flag was specified with a set of page ranges that prevented any output to be created. Change the page ranges.
005755	The -Printer:HT switch does not support UTF-8 encoded data. Use the -Printer:EH switch instead.	Spf_ht.c can't handle UTF-8
005756	The -EH_FullHTML switch support the following values: 30, 32, and 40.	The 'EH_FullHTML' command line flag is not properly formatted.
005757	The -EH_Browser switch can be specified with one of the following values: Basic, Netscape, IE, or ALL.	The 'EH_Browser' command line flag is not properly formatted.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
005758	The -EH_Language switch can be specified with one of the following values: English, French, German, Portuguese, Spanish, SChinese, TChinese, or Japanese.	The 'EH_Language' command line flag is not properly formatted.
005781	An ASCII-based encoding (ASCII, CP1252, etc) must be specified in order to generate barcodes for HPLaserJet output.	An ASCII-based encoding must be specified for the ENCODING-REPORT-OUTPUT setting in the INI file when generating barcodes for HPLaserJet (-printer:hp)
005900	Bad number in `01	(Windows) Specify a valid number. `01 = Command line option
005901	Bad filename in `01	(Windows) Specify a valid file name. `01 = Command line option
005902	Bad directory in `01	(Windows) Specify a valid directory path. `01 = Command line option
005903	Cannot access the @ parameter file (`01): `02	(Windows) Depends on the system error message. `01 = System error code `02 = System error message
005904	The argument list is too long; maximum is `01.	(Windows) To pass more than this number of arguments, use a @file argument file containing one argument per line. `01 = Maximum number supported by this version of SQR.
005905	Cannot open the report file (`01): `02	(Windows) Depends on the system error message. `01 = System error code `02 = System error message
005906	Invalid filename entered.	(Windows) Re-enter with a valid file name.
006000	Error writing the printer file. (`01): `02	This is an error that 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

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006001	Error reading the printer file. (` 01): ` 02	This is an error that can occur during normal operations due to the system environment (for example, file locking, permissions). Record the steps leading up to the error and contact your system administrator. ` 01 = System error code ` 02 = System error message
006002	Cannot open the printer file: ` 01 (` 02): ` 03	This is an error that can occur during normal operations due to the system environment (for example, file locking, permissions). Record the steps leading up to the error and contact your system administrator. ` 01 = Name of the file ` 02 = System error code ` 03 = System error message
006003	Unexpected End-Of-File while processing the printer file.	Possibly the file got corrupted. Try to recreate the .spf file. If the error persists, contact technical support.
006004	Encountered unknown SPF code (` 01) while reading the printer file.	Possibly the file got corrupted. Try to recreate the .spf file. If the error persists, contact technical support. ` 01 = Unknown SPF code
006100	Duplicate chart (` 01).	Each chart must be given a unique name. ` 01 = Chart name
006101	Unknown chart (` 01).	Chart could not be found. ` 01 = Chart name
006102	Number of chart data-array columns specified (` 01) exceeds the number of array columns (` 02).	Correct the source code. ` 01 = Number of data-array columns ` 02 = Number of array columns
006103	Number of chart data-array rows specified (` 01) exceeds the number of array rows (` 02).	Correct the source code. ` 01 = Number of data-array rows ` 02 = Number of array rows
006104	Too many pie segments (` 01). Max is ` 02.	Correct the source code. ` 01 = Number of segments ` 02 = Maximum allowed segments
006105	Chart module is not initialized.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006106	XY charts may have only numeric columns.	Correct the syntax.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006107	The 3rd column in the data array must be a character column to specify USE-3RD-DATA-COLUMN.	Correct the syntax.
006108	Invalid chart size or placement.	Correct the source code.
006120	INTERNAL: Bad chart index from stack (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Chart index
006121	INTERNAL: Unknown SQRBGInterface message (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Message code
006122	INTERNAL: Unsupported Graftsman chart type (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Chart type
006123	INTERNAL: Unsupported pie-explode setting (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Setting value
006124	INTERNAL: Unsupported tick-mark placement (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Placement value
006125	Graftsman interface message (`01) not supported.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Message code
006126	Unrecognized return code (`01) from Graftsman command message (`02).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Return code `02 = Message code
006127	Cannot fit Chart/Image into the current page. Position: (`01, `02) Size: (`03, `04)	Correct the source code. SQR aborts the program run. `01 = Row `02 = Column `03 = Width `04 = Depth

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006128	Check coordinate values.	Correct the syntax.
006140	Duplicate image (`01).	Images must be given unique names. `01 = Image name
006141	Unknown image (`01).	Image name could not be found. `01 = Image name
006142	Cannot open image file (`01). (`02): `03	`01 = Name of the file `02 = System error code `03 = System error message
006143	Unknown or missing image type (`01).	Enter a valid image type. `01 = Image type
006144	Unknown or missing printer type (`01).	Enter a valid Printer type. `01 = Printer type
006145	Duplicate FOR-PRINTER entries for printer (`01).	Only a single FOR-PRINTER can be specified for a printer type . `01 = Printer type
006146	The image type (`01) is not supported by printer type (`02).	The image, based on its type is invalid for the printer specified. For example, an EPS image is only valid for Postscript printer. `01 = The image type `02 = The printer type
006147	Invalid number of items in FOR-PRINTER list.	Too few or too many items in the FOR-PRINTER list. Correct the syntax.
006150	INTERNAL: Bad image index from stack (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Image name
006200	This report has already been defined.	Each report must be given a unique name.
006201	This layout has already been defined.	Each layout must be given a unique name.
006202	This printer has already been defined.	Each printer must be given a unique name.
006203	The values for ``01' must be > 0.	Correct the syntax. `01 = Qualifier name
006204	Qualifiers ``01' and ``02' are mutually exclusive.	Correct the syntax. `01 = Qualifier name `02 = Qualifier name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006205	Qualifier ``01' is not applicable with a 'default' printer.	Correct the syntax. `01 = Qualifier name
006206	The list must contain report names or ALL.	Correct the syntax.
006207	'ALL' must be specified by itself.	Correct the syntax.
006208	No report name was specified.	Correct the syntax.
006209	No layout name was specified.	Correct the syntax.
006210	No printer name was specified.	Correct the syntax.
006211	The name cannot be 'ALL'.	Correct the syntax.
006212	The name can only contain characters [0-9 A-Z _ -].	Correct the syntax.
006213	Report ``01' is referenced by multiple ``02' printers.	Correct the syntax. `01 = Report name `02 = Printer type
006214	Qualifier ``01' is not allowed with a ``02' printer.	Correct the syntax. `01 = Qualifier name `02 = Printer type
006215	The value for ``01' must be ``02 0.	Correct the syntax. `01 = Qualifier name `02 = Relation to zero (<,<=,=,>=,>)
006216	Report ``01' does not exist.	Correct the syntax. `01 = Report name
006217	The report name can be a string literal, variable, or column.	Correct the syntax. `01 = Report name
006218	Referenced layouts not defined:	A list of undefined layouts follows this message.
006219	Referenced reports not defined:	A list of undefined reports follows this message.
006220	Referenced printers not defined:	A list of undefined printers follows this message.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006221	The following SQR commands (listed below) cannot be used when any of the following NEW SQR commands are also used in the same report:	Correct the syntax.
006224	No printer type was specified.	Correct the syntax.
006225	Incorrect value for printer type. Valid values are:	Correct the syntax. A list of valid printer types follows this message.
006226	Attempt to execute the `01 command while processing the `02 procedure.	SQR aborts the program run. `01 = SQR command `02 = Procedure name
006227	Incorrect value for 'paper-size'. Specify the actual dimensions or one of the following names:	Correct the syntax. A list of valid predefined paper-size names follows this message.
006228	Referenced TOC (Table Of Contents) not defined:	A list of undefined Table of Contents follows this message.
006229	This TOC (Table Of Contents) has already been defined.	Each Table of Contents must be given a unique name.
006230	The list must contain TOC (Table of Contents) names or ALL.	Correct the syntax.
006231	The TOC (Table Of Contents) entry cannot be positioned given the LEVEL (^01) and INDENTATION (^02) values.	The Table of Contents entry will not fit given the specified level and current indentation values. ^01 = Specified LEVEL= value ^02 = Current INDENTATION= value
006232	`01 command not allowed while generating the Table of Contents.	The specified command cannot be used while the Table of Contents is being generated. `01 = SQR command
006233	The TOC (Table of Contents) entry "A" cannot be processed because the existing entry "B" is positioned below it. A: Line = `01, Level = `02, Text = ``03' B: Line = `04, Level = `05, Text = ``06'	Correct the program logic to eliminate the conflict between the two TOC (Table of Contents) entries. `01 = A: Line number `02 = A: Level value `03 = A: Text value `04 = B: Line number `05 = B: Level value `06 = B: Text value

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006300	Unknown parameter (`01).	Correct the syntax. `01 = Parameter name
006301	Value not valid for parameter (`01).	Correct the syntax. `01 = Parameter name
006302	Invalid option (`02) for parameter (`01).	Correct the syntax. `01 = Parameter name `02 = Option
006303	Parameter (`01) is required, but has not been specified.	Correct the syntax. `01 = Parameter name
006304	Parameter (`01) already specified.	Correct the syntax. `01 = Parameter name
006305	Parameter (`01) does not support &columns.	Correct the syntax. `01 = Parameter name
006306	Parameter (`01) requires equal sign.	Correct the syntax. `01 = Parameter name
006307	Parameter (`01) has an unquoted string.	Correct the syntax. `01 = Parameter name
006308	Missing part of specification for parameter (`01).	Correct the syntax. `01 = Parameter name
006309	Parameter (`01) requires literal.	Correct the syntax. `01 = Parameter name
006310	Parameter (`01) requires valid numeric value.	Correct the syntax. `01 = Parameter name
006311	Parameter (`01) requires integer value.	Correct the syntax. `01 = Parameter name
006312	Parameter (`01) does not support type supplied.	Correct the syntax. `01 = Parameter name
006313	Parameter (`01) requires valid string. Perhaps quote or \$ is missing.	Correct the syntax. `01 = Parameter name
006314	Parameter (`01) does not accept 'NONE' in this context.	Correct the syntax. `01 = Parameter name
006315	Parameter (`01) requires proper object name.	Correct the syntax. `01 = Parameter name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006316	Parameter (`01) requires array name.	Correct the syntax. `01 = Parameter name
006317	Parameter (`01) does not accept 'AUTOSCALE' in this context.	Correct the syntax. `01 = Parameter name
006318	Parameter (`01) has improper value list.	Correct the syntax. `01 = Parameter name
006320	Parameter (`01) does not support relative values.	Correct the syntax. `01 = Parameter name
006350	Conversion [(`01) to (`02)] is not supported.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = From type `02 = To type
006352	INTERNAL: Unsupported option/request (`01) in (`02).	This is an 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
006354	INTERNAL: Unknown data type, (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Data type
006355	INTERNAL: Unable to retrieve parameter value, (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Parameter name
006356	INTERNAL: Data type (`02) not valid for parameter (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Parameter name `02 = Data type
006357	INTERNAL: Data location (`02) not valid for data type (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Data location `02 = Data type

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006358	INTERNAL: Cannot decode string (`01) to index.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = String to decode
006359	INTERNAL: Cannot set bit value (`02) for parameter (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Parameter name `02 = Value
006360	INTERNAL: Unknown program state (`01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = State
006400	Unsupported background color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006401	Unsupported border color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006402	Border width out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006403	X position out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006404	Y position out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006405	X size out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006406	Y size out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006407	Unsupported font.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006408	Unsupported font style.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006409	Unsupported font color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006410	Unsupported horizontal text justification value.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006411	Unsupported vertical text justification value.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006412	Unsupported font path.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006413	Unsupported font rotation.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006414	Font size out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006415	Text line id# out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006416	Unsupported chart type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006417	Unsupported chart sub-type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006418	Unsupported chart orientation (not H or V).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006419	Unsupported perspective (not 2D or 3D).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006420	Unsupported axis (not X or Y).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006421	Unsupported axis label data type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006422	Dataset id# out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006423	Unsupported dataset type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006424	Unsupported dataset color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006425	Unsupported dataset line style.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006426	Unsupported dataset fill pattern.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006427	Unsupported dataset marker.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006428	Chart type does not support Y-axis datasets.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006429	Pie-chart segment id# is out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006430	Unsupported pie-segment color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006431	Unsupported pie-segment border color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006432	Unsupported pie-segment pattern.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006433	Unsupported pie-segment explode setting.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006434	Command only valid for charts of type 'pie'.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006435	Pie-chart radius out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006436	Pie-chart starting angle out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006437	Unsupported pie-chart fill direction. Must be clockwise or counter-clockwise.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006438	Unsupported pie-segment label position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006439	Unsupported pie-segment quantity display position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006440	Unsupported pie-segment per-cent display position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006441	Unsupported legend style.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006442	Unsupported legend horizontal position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006443	Unsupported legend vertical position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006444	Text charts do not support legend.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006445	Number of datasets specified does not match data.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006446	Unsupported axis label position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006447	Unsupported axis type (not LINEAR or LOG).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006448	Pie and text charts do not support axis control.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006449	Unsupported axis min scaling.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006450	Unsupported axis max scaling.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006451	Unsupported axis max scaling.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006452	Beginning of tickmarks is after end.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006453	Unsupported tickmark type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006454	Unsupported grid type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006455	Unsupported grid color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006456	Grid line width out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006457	Unable to open grafcap file.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006458	Unsupported grafcap device.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006459	Error in grafcap entry specification.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006460	Unable to open chart output destination.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006461	Internal error during ggDraw.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006462	Improper parameters passed to gscale.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006463	The shared library specified in the grafcap file could not be found.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006464	A function called from the shared library specified in the grafcap file could not be found.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006500	The bar code could not be positioned on the page. Row: `01, Column: `02, Height: `03	Correct the source code. `01 = Row `02 = Column `03 = Height
006501	Unknown BCL error (`01) encountered.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = BCL error code

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006502	Invalid bar code type (`01): Valid values are from 1 to 15.	Correct the source code. `01 = Bar code type.
006503	The length of the bar code text '01' must be between 1 and 30 characters.	Correct the source code. `01 = Bar code text
006504	The length of the caption text '01' must be between 1 and 30 characters.	Correct the source code. `01 = Caption text
006505	Invalid printer type (`01): Valid values are from 0 to 13.	Correct the source code. `01 = Printer type
006506	Invalid offset: Valid values are from 0 to 250.	Correct the source code.
006507	Invalid height (`01): Valid values are from 0.1 to 2.0 inches.	Correct the source code. `01 = Height
006508	Invalid checksum: Valid values are from 0 to 2.	Correct the source code.
006509	Invalid pass: Valid values are from 1 to 6.	Correct the source code.
006510	The bar code text '01' is not valid for the type of bar code (`02) selected.	Correct the source code. `01 = Bar code text `02 = Bar code type
006511	Internal error: Could not generate the bar code.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006512	Internal error: Bar code buffer required too large (>32K).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006601	Cannot allocate the device context for the default printer.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006602	Failed to start printing the document.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006603	New-page (start) failed on page `01.	(Windows) This is an error that 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

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006604	New-page (end) failed on page `01.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator. `01 = Page number
006605	End document failed.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006606	Error reading font information from the [Fonts] section in SQR.INI. Using the default font.	(Windows) Correct the [Fonts] section in the SQR.INI file.
006607	Failed to create a brush for shading.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006608	Failed to select font `01.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Font name
006609	Failed to modify font `01.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Font name
006610	Failed to create a pen that was required to draw a box.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006611	Failed to create a pen that was required to draw a horizontal line.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006612	Failed to create a pen that was required to draw a vertical line.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006613	Failed to open the image bitmap file (`01).(`02):`03	(Windows) This is an error that can occur during normal operations due to the system environment (file locking, permissions). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006614	The file (`01) does not contain a valid bitmap.	(Windows) Specify a valid bitmap file. `01 = Name of the file
006615	Failed to create the palette for image (`01).	(Windows) This is an error that can occur due to lack of system resources or an invalid bitmap. Record the steps leading up to the error and contact your system administrator. `01 = Name of the file
006616	Failed to load RLE into memory for image (`01).	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Name of the file
006617	Failed to convert DIB to DDB for image (`01).	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Name of the file
006618	Failed to draw the bitmap image (`01).	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Name of the file
006619	Cannot access the default printer's driver.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006620	Cannot select the charting clip area onto the printers DC.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006621	Cannot select create a metafile required for business graphics.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006622	Cannot create a region required for business graphics.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006623	Cannot create a DC required for business graphics.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006624	Cannot create a bitmap required for business graphics.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006625	Business graphics failed while setting up the device (ggWinDevice).	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006626	Cannot draw business graphics.	(Windows) This is an error that 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 BINW directory. Record the steps leading up to the error and contact your system administrator.
006700	SQRDIR is not defined.	(Windows) The variable SQRDIR must be defined in the SQR.INI file.
006701	Could not allocate memory while attempting to register the .spf filename extension.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006702	Could not allocate memory for the page cache.	(Windows) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006704	Cannot open or read file (`01)`02): `03	(Windows) This is an error that can occur during normal operations due to the system environment (e.g. file locking, permissions, etc.). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message
006705	File (`01) is not in SPF packet format.	(Windows) The file was not produced by SQR or it has been corrupted. `01 = Name of the file
006706	Failed to identify the start of the report (`01).	(Windows) The file was not produced by SQR or it has been corrupted. `01 = Name of the file
006707	An invalid seek was made for page `01.	(Windows) This is an internal error which should not occur under normal operations. Please contact technical support. `01 = Page number

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006708	Too many errors were encountered while processing the file. Processing has been stopped.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006709	Failed to open the image bitmap file (`01). (`02); `03 This message is displayed only once per SPF file.	(Windows) This is an error that can occur during normal operations due to the system environment (e.g. file locking, permissions, etc.). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message
006800	`01: Detected internal program error.	This is an internal error that should never occur during normal operation. Record the steps leading up to the error and contact technical support. `01 = Name of the routine
006801	`01: Null Operand Passed as input.	This is an internal error that should never occur during normal operation. Record the steps leading up to the error and contact technical support. `01 = Name of the routine
006802	`01: Decimal Exponent Under/Overflow.	Exponent Under/Overflow: Exponent of decimal number has exceeded the valid boundaries established for the decimal type. Check the documentation for the current upper and lower bounds of a decimal object. `01 = Name of the routine
006803	`01: Decimal to Integer Conversion Under/Overflow.	Integer Under/Overflow: Cannot convert input decimal object into a valid integer number. Decimal object exceeds the established integer boundaries for this machine architecture. Check the magnitude and sign of the decimal object to ensure that it falls within the upper and lower bounds of an integer number. `01 = Name of the routine
006804	`01: Decimal to Float Conversion Under/Overflow.	Floating Point Under/Overflow: Cannot convert input decimal object into a valid floating point number. Decimal object exceeds the established floating point 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 a floating point number. `01 = Name of the routine

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
006805	`01: Decimal Precision Under/Overflow.	Decimal Precision Under/Overflow: Attempt made to initialize decimal object with an invalid precision. Check the input precision value against the documented upper and lower boundaries for a decimal object. `01 = Name of the routine
006806	`01: String to Decimal Object Conversion Error.	String To Decimal Conversion Error: Length of input string is greater than precision of underlying decimal object. Either increase the precision of the decimal object or reduce the size of the input mantissa to match the decimal object precision. `01 = Name of the routine
006807	`01: Truncation/Rounding Error - Outside Valid Range for Decimal Object.	Truncation/Rounding Error: Input truncation or round value is outside the valid range for this decimal object. Please ensure that the truncation/round value is greater than or equal to zero and less than the precision of the underlying decimal object. `01 = Name of the routine
006808	`01: Decimal Error: Cannot Divide by Zero.	Decimal Math Divide by Zero Error: Attempt made to divide a decimal object by zero. Please check divisor to ensure that it does not equal zero before attempting to divide. `01 = Name of the routine
006900	There is no default printer set up on your system. Use the Control Panel "Printers" applet to define it.	(Windows) SQR Print requires that a default printer be defined. Use the "Printers" applet in the Control Panel to define one.
007000	The locale ``01' is not defined in the SQR.INI file.	Check for a misspelled locale name and/or the SQR.INI file. `01 = Locale name
007001	At least one qualifier must be specified.	Correct the source code.
007002	The value for '01' must be a list of 02 string literals, variables or columns.	Correct the source code. `01 = Qualifier `02 = Number of entities in list
007003	The values for '01' and '02' cannot be the same.	Correct the source code. `01 = Qualifier `02 = Qualifier

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007004	The value for '01' (^ 02) must be a single character which is not in the list: "03".	Correct the source code. `01 = Qualifier `02 = Value `03 = List of invalid characters
007005	The value for `` 01' (^ 02) is invalid. Valid values are:	Correct the source code. `01 = Qualifier `02 = Value
007006	The last character of the `` 01' value (^ 02) cannot be a digit or the minus sign or the same as either of the separators.	Correct the source code. `01 = Qualifier `02 = Invalid character
007007	The first character of the `` 01' value (^ 02) cannot be a digit or the minus sign or the same as either of the separators.	Correct the source code. `01 = Qualifier `02 = Invalid character
007008	The following errors occurred while processing the (^ 01) locale from the SQR.INI file.	This message precedes error messages encountered while processing the SQR.INI file. `01 = Locale name
007009	The value for `` 01' cannot be 'DEFAULT' or 'SYSTEM'.	Correct the syntax. `01 = Qualifier
007010	The value for `` 01' (^ 02) is not properly formatted: Did not find the '>' for the '<nnn>' construct.	Correct the syntax. `01 = Qualifier `02 = Value
007011	The value for `` 01' (^ 02) is not properly formatted: The value of an '<nnn>' construct must be from 1 to 255.	Correct the syntax. `01 = Qualifier `02 = Value
007012	The default locale (^ 01) specified in the [^ 02] section of the SQR.INI file has not been defined.	Correct the syntax. `01 = Locale name `02 = Section name
007013	The value for `` 01' (^ 02) must be a list of `03 quoted string literals.	Correct the syntax. `01 = Qualifier `02 = Value `03 = Number of entities in list
007014	The entry (^ 01 = ^ 02) is not valid.	Correct the SQR.INI entry. `01 = Qualifier from the SQR.INI file `02 = Qualifier's value

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007100	The use of an edit mask or the keywords NUMBER, MONEY, or DATE is not legal when storing numeric variables.	Correct the source code.
007101	The last keyword is not ``01'.	Correct the source code. `01 = Keyword
007102	Incompatible source and destination variable types.	Correct the source code.
007103	The keyword (`01) is not compatible with the variable (`02).	Correct the source code. `01 = Keyword `02 = Variable name
007104	The use of an edit mask or the keyword DATE is not legal if both variables are date variables.	Correct the source code.
007200	The specified precision (`01) is out of range (`02 - `03).	Correct the source code. `01 = Specified precision `02 = Minimum precision `03 = Maximum precision
007201	The precision is specified by a value from `01 to `02 surrounded by parentheses.	Correct the source code. `01 = Minimum precision `02 = Maximum precision
007202	Variable (`01) is not a decimal variable and cannot have a precision associated with it.	Correct the source code. `01 = Variable name
007203	A string variable name is required here.	Correct the source code.
007204	A numeric variable name is required here.	Correct the source code.
007205	The variable (`01) has already been defined as ``02' and may not be redefined.	Correct the source code. `01 = Variable name `02 = Variable type
007206	The variable type has not been specified.	Correct the source code.
007207	This command is only allowed within local procedures.	Correct the source code.
007208	This command must be before all other commands in the procedure.	Correct the source code.
007209	Only string (\$) and numeric (#) variables may be declared.	Correct the source code.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007210	Invalid variable name specified.	Correct the source code.
007211	You cannot declare a global variable from within a procedure.	Correct the source code.
007400	The specified character is invalid in the current character set.	Correct the program logic.
007401	'`01' is not a valid value for the ENCODING environment variable.	The specified encoding scheme is not known by SQR. `01 = ENCODING environment variable setting.
007402	The Double-Byte LET function '`01' is not supported in this version of SQR.	The SQT file contains a reference to a LET function, which is not supported by this version of SQR. `01 = LET function name
007403	The Double-Byte SQR command '`01' is not supported in this version of SQR.	The SQT file contains a reference to an SQR command, which is not supported by this version of SQR. `01 = SQR command name
007404	Double-Byte .sqt files are not supported by this version of SQR.	The run-time file was created by the Double-Byte version of SQR and is incompatible with the current version.
007405	The barcode text '`01' cannot contain double-byte characters.	Correct the source code. `01 = Bar code text
007501	Using `01 edit mask from (`02) against (`03)	A date edit mask element was detected which could cause date data to be incorrectly interpreted. This warning message can be turned off by setting the "OutputTwoDigitYearWarningMsg" entry to the [Default-Settings] section of the SQR.INI file to FALSE. `01 = Edit mask element `02 = Edit mask being used `03 = Value being applied to the edit mask
007601	Cannot access the Java file (`01) (`02): `03	SQR cannot access the required file. `01 = Name of the file `02 = System error code `03 = System error message
007602	-EH_Scale: value (`01) is out of range (`02 - `03).	Correct the command line. `01 = Specified scale `02 = Minimum allowed `03 = Maximum allowed
007603	-Printer:EH functionality is not available on this platform.	Enhanced HTML functionality is not available on this platform.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007604	-Printer:PD functionality is not available on this platform.	PDF functionality is not available on this platform.
007605	Cannot support Unicode internally. Please reset the UseUnicodeInternal setting in the SQR.INI file to FALSE.	Cannot support Unicode internally. Please reset the UseUnicodeInternal setting in the SQR.INI file to FALSE.
007701	Did not find end of paragraph: `01 (No 'end-execute' clause found.)	Correct the source code. `01 = BEGIN-command in question.
007702	Invalid entry for keyword, ``01``02'	Correct the source code.
007703	May only specify either PROCEDURE=, or COMMAND=, or GETDATA=, exclusive.	Correct the source code.
007704	Must specify a SCHEMA.	Correct the source code.
007705	Must specify either a PROCEDURE, COMMAND, or GETDATA.	Correct the source code.
007706	CONNECTION ``01' not found. No such connection.	Correct the source code.
007707	The returned set of Procedure parameters (INOUT and OUT) (length = `01 items) did not include one or more of the specified items.	Stored procedure error.
007708	Encountered a parameter of type ``01'. Valid types are either IN, OUT, or INOUT. If no type is entered, the type defaults to IN.	Stored procedure error.
007709	The datasource failed to provide the expected return status value. Verify the query metadata.	Datasource error.
007710	The datasource failed to provide the expected number of elements in the return status list.	Datasource error.
007711	Failed to login to the requested datasource (Connection=``01', username=``02'). DETAILS: `03	Logon failed.
007712	The requested rowset (`01) was not available. Verify the query metadata.	Not enough rowsets.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007713	Missing or invalid Registry.properties file. Verify that the CLASSPATH includes SQDIR, that SQDIR contains the folder with the Registry.properties file, and that the Registry.properties file is valid.	Incorrect environment setup.
007714	The datasource ('`01') does not support the requested capability ('`02'). Check the capabilities list for the datasource, located in the Properties folder.	Invalid query for datasource.
007715	Failed to start the Java Virtual Machine (JVM). Possible causes are: missing or invalid jdk files, incorrect CLASSPATH, or insufficient resources.	Incorrect environment setup.
007716	The current rowset (`01) contained no rows. Check the return status and/or metadata for the requested service to determine the cause.	No data.
007717	The query failed. DETAILS: `01	Query failed.
007718	Failure setting property '`01'. DETAILS: `02	Property-set failed.
007719	The value for keyword '`01' exceeds the maximum length of '02 characters.	Keyword value too long.
007720	A fatal error occurred while fetching against the current rowset: DETAILS: `01	A failure occurred during row fetch.
007721	Parameter `01 (`02) was passed to the PROCEDURE as data type `03; expected (`04) type `05. Verify the query metadata.	A failure occurred during row fetch.
007722	Invalid query parameter: Reason: `01	Bad procedure parameter.
007723	Too many parameters (= `01) were supplied to the query. Verify the query metadata.	Bad procedure parameter.
007724	Parameter `01 (`02) was passed to the PROCEDURE as type `03; expected type `04. Verify the query metadata.	Bad procedure parameter.
007725	Parameter `01 ('`02', JDO-type `03), specified 'NULL', is a required-parameter. Specify a value or variable name.	Bad procedure parameter.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007726	The list-variable parameter to the query is too long. Maximum number of elements is 30.	List too long.
007727	Unable to retrieve metadata for Procedure= `01, Schema= `02. DETAILS: `03	Metadata check failed.
007728	Parameter list type mismatch (#`01, SQR type = `02). The datasource expected a parameter of type `03. Verify the query metadata.	Parameter list mismatch.
007729	List size mismatch detected while fetching data of type ROW, `01 items, into SQR list-variable, `02 items. Fetching will proceed to the smaller size.	List size mismatch.
007730	Incorrect syntax for BEGIN-SELECT ... FROM. Options are: FROM ROWSETS=... FROM PARAMETER= \$strvar strlit	Bad begin-select syntax.
007731	Attempted to pass as INOUT or OUT a parameter which was of type ROWSET (`01). Use of such parameters is supported as IN only, after which they may be used in a BEGIN-SELECT construct.	Bad parameter keyword.
007732	Attempt to use a scalar SQR variable ('01') to reference a ROWSET procedure parameter ('02'). Use either the keyword 'NULL', or an SQR LIST variable (%var). Verify the query metadata.	Bad proc parameter.
007733	The list of keywords entered to the PARAMETERS keyword must be terminated with a semicolon.	Bad proc parameter. Correct the source code.
007734	Datasource '01' not found. The Connection being used by this query specifies a datasource which is not listed in the DDO Registry ('02'). DETAILS: `03.	Bad proc parameter. Correct the source code.
007735	Missing one or more DDO {fname}.jar files. Verify the location of the original-installation files, and that they are accessible. Error code: `01. Classpath: `02.	Bad environment.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007736	Unable to open Connection ('01') to datasource ('02'). Possible causes: (a) the Declare- or Alter-connection specification is invalid, or (b) the datasource is no longer available. DETAILS: `03.	Bad environment.
007737	Unable to locate one or more entry points in an SQR {fname}.jar file. Verify that the original-installation files have not become corrupted.	Bad environment.
007738	At least one JNI method pointer was lost. This should never occur: record the steps leading up to this failure, and contact Technical Support. DETAILS: Schema='`01', Proc='`02'.	Bad environment.
007739	Unable to locate query object ``01' in the specified schema (`02). DETAILS: `03.	Bad environment.
007740	Invalid &pseudonym or 'TYPE=' data-type specified for a begin-select column-variable. Valid types are: CHAR, TEXT, DATE, NUMBER.	Correct the syntax.
007741	Illegal attempt to fetch a non-scalar field into a column variable. Correct the query.	Correct the syntax.
007742	The output parameter specified in 'Begin-Select ... From Parameter = `01' is not available. Available parameters: `02.	Bad command.
007743	The output parameter specified in 'Begin-Select ... From Parameter = `01' is not of type ROWSET. Verify the query metadata.	Bad command.
007744	Illegal attempt to assign an SQR variable ('01') of type '02' the value from a DDO object ('03') of type '04'. Verify the query metadata.	Bad var assignment.
007745	Illegal attempt to assign an SQR column variable ('01') of type '02' the value from a DDO object of type '03'. Verify the query metadata.	Bad var assignment.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007746	Failed to locate the requested Rowset (`01) while processing the query. The last available Rowset number is `02. Verify the query metadata.	Not enough RowSets.
007747	The query raised a DDO exception. DETAILS: `01.	Bad query.
007748	A BEGIN-SELECT paragraph was coded, but the query returned no Rows.	No data warning.
007749	Invalid syntax for PARAMETERS=(...) statement. Use: PARAMETERS=(%v \$v #v &v NULL SKIP numlit datelit textlit [IN OUT], ...) All parameters must be specified. Optional parameters which are to be ignored may be specified by the keyword 'NULL' or 'SKIP'. Correct the syntax.	Incorrect syntax.
007750	FATAL: Failure creating Java object.	General failure.
007751	Attempt to create a List variable of size greater than the maximum size of `01 items.	General failure.
007752	Parameter-list item ``01' is not a member of the parameter list for this Query. Verify the query metadata.	No such input/inout parameter.
007753	Attempt to access List-row (`01) beyond the List size (`02 rows).	Bad list assignment/setup.
007754	Attempt to assign/modify a List row is not compatible with the List definition.	Bad list assignment/setup.
007755	Attempt to assign a row to a non-existant List variable. Define the List first, using the syntax: let %!name[size] = list(NUMBER DATE TEXT #var \$var [, ...])	Bad list assignment/setup.
007756	Incorrect syntax for List-variable reference. Use: let [\$ #]var = %listname[nlit #var].colname	Bad list assignment/setup.
007757	Alter-connection statement missing 'DSN='...'. Improper alter-conn.	Improper alter-conn.
007758	List-definition size specifier must be literal.	Improper alter-conn.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007759	Attempt to access a non-existent List-column ('01').	No such list column name.
007760	Must specify one of the keywords, FROM-ROWSETS or FROM_PARAMETER.	Incorrect syntax for Load-lookup.
007761	Incorrect syntax to Load-lookup 'PARAMETERS=' keyword. Use: PARAMETERS=(slit nlit \$var #var %var &var, ...) No line wrapping is allowed for this usage.	Incorrect syntax for Load-lookup.
007762	Too many parameters (`02) entered to Load-Lookup command. Max parameters is `01.	Incorrect syntax for Load-lookup.
007763	Problem executing the cursor for LOAD-LOOKUP table ``01'. DETAILS: `02.	The database server returned an error while trying to execute the SQL statement needed to process the LOAD-LOOKUP command. `01 = Load lookup table name
007764	Bad return fetching row from database in LOAD-LOOKUP table ``01'. DETAILS: `02	The database server returned an error while fetching the data. `01 = Load lookup table name
007765	DC, DI sort options not supported with this SQR version. To sort, use SORT=SC or SORT=SI.	Database sort not supported for Load-Lookup with DDO.
007766	Must specify a query keyword; PROCEDURE=, COMMAND= or GETDATA=.	Incorrect syntax for Load-lookup. Specify a keyword representing the query.
007767	Unknown column variable type.	Unknown data type returned by the server.
007768	The property `01` was not found in the property sheet for the specified datasource (`02). Available property names are: `03. The datasource property sheet does not include the named property.	Verify the metadata and correct the syntax.
007769	The specified CONNECTION ('01') references a datasource whose property sheet does not show support for the Get-Data query method.The datasource property sheet does not show support for Get-Data.	Verify the metadata and property sheet and correct the syntax.
007770	Attempt to create a Selector (or MDSelector) object failed. This event should not occur. Contact your system administrator.	Failed to create the requested object. Contact your system administrator.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007771	Did not find value after ``01 ='	The code specified a Connection = keyword, but no matching literal. Correct the syntax.
007772	The value for the keyword Connection= must be a literal string.	The code specified a Connection = keyword, which must be a text literal. Correct the syntax.
007773	FATAL: Non-unique list variable allocation. Contact Technical Support.	This should never occur. Contact technical support.
007774	Invalid attempt to establish a second connection to datasource ``02', using Connection ``01'. The Connection ``01' is declared to allow only one active login (no-duplicate=TRUE).	Duplicate logins specified as not allowed. Correct the source code, declare a new Connection, or omit the use of no-duplicate in the subject Connection
007775	Bad value ('`01') for Alter-connection keyword (`02). Valid values are: `03.	Bad keyword value. Refer to language reference and correct the syntax.
007776	Missing column variable type. Each select variable must include 'type=[TEXT NUMBER DECIMAL DATE]'	Rowset Metadata could not be retrieved to support early-binding. Check the source code or the datasource. Possible causes: (a) if type=datatype not specified, and From Rowsets statement includes a variable, (b) if type=datatype not specified, and the datasource did not provide metadata, or (c) if type=datatype not specified, and one of the specified ResultSets was not found in the metadata.
007777	Can't select from an INPUT-only parameter ('`01').	The selected parameter is an Input-only. Correct the source.
007778	Datasource login not available.	Connection non-existent. Correct the source. Possible causes: No BEGIN-EXECUTE statement.
007779	Unable to verify ResultSet column types due to use of variable. Variables are not allowed either for CONNECTION, SCHEMA, PROCEDURE, From-Parameter, or the first element of the From-Rowsets. Use literals or define column types using 'type=<datatype>'.	Can't verify colvar types with variable entry; use literals or define column types. Correct the source.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007780	Unable to verify ResultSet column types. Must specify column types using 'type=CHAR NUMBER DATE' construct when selecting from datasources which do not supply metadata, or when using the COMMAND= and GETDATA= keywords.	Can't verify colvar types with for COMMAND, GETDATA or from [TABLES]. Use type=<datatype> on select variables. Correct the source.
007781	Unable to log onto datasource to obtain query metadata. Specify the Connection for this query using a complete Declare-Connection statement, or specify type=<datatype> for each column variable in the Begin-Select.	Must declare a complete connection if use early binding of select column variables.
007782	Column variables must specify datatype. Use the construct 'colname type=[CHAR NUMBER DATE]', or &colname=<type> for all relational queries.	Must supply type=datatype on column variables for non-procedure queries.
007783	Could not execute SQL. DETAILS: `01	An error occurred while trying to compile the SQL statement. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
007784	Bad CONNECTION specification (`01'). Possible causes: Syntax error, Dimension name not found, Dimension attributes not found, Dimension name not found in Begin-Select list. DETAILS: `02	The OLAP-related members of the named CONNECTION could not be processed, either due to syntax or no such name.
007785	The column specified (`01') is ambiguous. It appears more than once in the data.	A column specified in the query appears multiple times in the data. Change the query to avoid that column, or rename the column in the data. `01 = The column name
007786	Column (`01') not found.	A column specified in the query was not found in the data. `01 = The column name
007787	Unsupported datatype (`02') found in column (`01'). Only Text, Numeric, and Date are currently supported.	A column specified in the query contained values which are not currently supported. Change the query to use a different column. `01 = The column name `02 = The unsupported datatype

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
007788	Multiple occurrences of column ('`01') detected. Only one occurrence of a column is permitted in each row.	A column specified in the query was detected multiple times within one row of data. This is not currently supported. `01 = The column name
007789	The query is improperly specified. The result would involve a Cartesian Join which is not currently supported. Rewrite the query to involve only columns in the same branch of the tree.	The columns specified in the query are found in different branches of a 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.
007790	Must specify PROCEDURE= to use PARAMETERS= in BEGIN-EXECUTE.	Parameters may only be specified for PROCEDURE queries, not for COMMAND or GETDATA queries. Remove the PARAMETERS= line, or specify PROCEDURE= instead.
007791	Unable to locate one or more JAVA classes in the DDO JAR file.	Verify that the original installation files are not corrupted.
007792	Unable to locate one or more JAVA methods in the DDO JAR file.	Verify that the original installation files are not corrupted.
007793	An incompatible version of the DDO JAR file was found.	Verify that the original installation files are not corrupted.
008000	Delay not appropriate for database columns or literals.	The DELAY argument to the PRINT command can only be used with SQR #variables or \$variables.
008001	The width must also be specified when DELAY is used.	The DELAY argument to the PRINT command requires that the width argument be specified.
008002	The PRINT DELAY statement on line `01 has not been satisfied by a matching SET-DELAY-PRINT command.	An attempt was made to process an PRINT DELAY statement without an intervening SET-DELAY-PRINT command against the same SQR variable.
008003	The SET-PRINT-DELAY command cannot find a pending PRINT DELAYstatement.	An attempt was made to process an SET-DELAY-PRINT command against an SQR variable for which there was no pending PRINT DELAY statement.
008004	The PRINT DELAY statement did not have an SET-PRINT-DELAY command executed against it.	This PRINT DELAY statement did not have an SET-DELAY-PRINT command executed against it when SQR ended its run.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
008005	The variable (`01`02) was referenced by a PRINT DELAY statement but the SQR program does not contain a matching SET-PRINT-DELAY command.	The referenced variable was used with a PRINT DELAY statement but the SQR program did not contain a SET-PRINT-DELAY command for that variable.
008006	The variable (`01`02) was referenced by a SET-PRINT-DELAY command but the SQR program does not contain a matching PRINT DELAY statement.	The referenced variable was used with a SET-PRINT-DELAY command but the SQR program did not contain a PRINT DELAY statement for that variable.
008101	The specified MODE value ``01' is not legal. Legal values are 'ON' or 'OFF'.	The MODE= qualifier values are ON or OFF. `01 = Invalid value
008102	At least one qualifier must be specified.	Correct the source line.
008200	The specified color (`01') is not defined.	The specified color is not defined in the color map. `01 = Undefined color
008201	Qualifier ``01' has a malformed color reference.	The specified color reference is not properly formed. It can reference a single string literal, column, or variable (i.e. (\$name)) or it can reference three numeric literals, columns, or variables (i.e. (10,20,30)) which represent the Red, Green, and Blue components of the color.
008202	Qualifier ``01' must reference SQR variables only.	The specified color reference is not properly formed. It can reference a single string variable (i.e. (\$name)) or it can reference three numeric variables (i.e. (#R,#G,#B)) which represent the Red, Green, and Blue components of the color.
008203	Invalid RGB value (`01,`02,`03)	The RGB values are out of range. Each value can be from 0 to 255. `01 = Red value `02 = Green value `03 = Blue value
008204	At least one qualifier must be specified.	Correct the source line.
008205	The Declare-Color-Map entry is not properly defined.	The Declare-Color-Map is not properly defined: 1) The color name can only contain characters [0-9 A-Z _]. 2) The color name cannot be 'none' 3) The RGB values are not valid (each can be 0 to 255)
008206	Duplicate palette name: `01	Change the name of the palette. `01 = Palette name in question

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
008207	The name can only contain characters [0-9 A-Z _ -].	Correct the syntax.
008208	The palette cannot have gaps. All colors up to the highest one defined (`01) must be specified.	An SQR palette cannot have gaps. Correct the source code. `01 = Highest color defined for this palette
008209	The specified palette (`01) does not exist.	Change the name of the palette. `01 = Palette name in question
008300	For font (`01) the specified typeface (`02) is not legal.	Correct the name of the CJK typeface `01 = Font id `02 = Typeface name
008301	For font (`01) the specified character map (`02) is not legal.	Correct the name of the CJK character map `01 = Font id `02 = Character map name
008302	For font (`01) both a typeface and character map must be specified.	If a CJK typeface is specified with a font then a character map must also be specified. `01 = Font id
008304	The current report encoding (`01) requires that a typeface and a character map be specified with each font.	PDF support requires that a proper typeface and character map be associated with a font in order to generate a PDF file with the following output encodings: <ul style="list-style-type: none">■ Simplified Chinese: EUC-CN, GBK (CP936), UCS-2■ Traditional Chinese: EUC-TW, BIG5, USC-2■ Korean: EUC-KR, UHC (Johab), UCS-2■ Japanese: EUC-JP, Shift-JIS, ISO-2022-JP, UCS-2
008305	For font (`01) the encoding (`02) is incompatible with the report output encoding (`03).	The encoding for the current font is incompatible with the encoding used for the PDF file. `01 = Font id `02 = Font encoding `03 = Report output encoding
008400	The table name (`01) can only contain the following characters [0-9 A-Z _ -].	The specified table name contains invalid characters. `01 = Table name
008401	The table name (`01) is already being used.	The specified table name is already being used. `01 = Table name
008402	The table definition (`01) is already being used.	The specified table definition is already being used. `01 = Table definition name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
008403	The table definition (` 01) does not exist.	The specified table definition does not exist. ` 01 = Table definition name
008404	The table (` 01) does not exist.	The specified table does not exist. ` 01 = Table name
008405	No table definition name was specified.	Correct the source line.
008406	When ACTION=ERASE no other parameters are allowed.	Correct the source line.
008407	When ACTION=INFO only the ROW and COUNT parameters are allowed. At least one must be specified.	Correct the source line.
008408	When ACTION=GET only the following combination of parameters are allowed: <ul style="list-style-type: none">■ ARRAY with optional ROW and COUNT■ VALUES with optional ROW	Correct the source line.
008409	When ACTION=REPLACE only the following combination of parameters are allowed: <ul style="list-style-type: none">■ ARRAY with optional FIRST, ROW and COUNT■ BLANK with optional ROW and COUNT■ VALUES with optional ROW	Correct the source line.
008410	When ACTION=INSERT only the following combination of parameters are allowed: <ul style="list-style-type: none">■ ARRAY with optional MODE, FIRST, ROW and COUNT■ BLANK with optional MODE, ROW and COUNT■ VALUES with optional MODE and ROW	Correct the source line.
008411	When ACTION=DELETE only the ROW and COUNT parameters are allowed.	Correct the source line.
008412	When ACTION=APPEND only the following combination of parameters are allowed: <ul style="list-style-type: none">■ ARRAY with optional FIRST, and COUNT■ BLANK with optional COUNT VALUES	Correct the source line.
008413	Qualifier `` 01' requires a numeric variable.	Correct the source line.

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
008414	The value for ``01'(`02) must be `03`04 and `05`06.	Correct the source line. `01 = Qualifier name `02 = Value `03 = Minimum value relation `04 = Minimum value `05 = Maximum value relation `06 = Maximum value
008415	The first argument in ``01' must be the column number (>= 0 and <= `02).	Correct the source line. `01 = Qualifier name `02 = Maximum value
008416	The specified column number (`01) exceeds the number of columns for this table as specified by the COLUMN-COUNT qualifier.	Correct the source line. `01 = Column number
008417	The specified column number (`01) has already been defined by another `02 qualifier.	Correct the source line. `01 = Column number `02 = Qualifier name
008418	Incorrect value (`01) for qualifier ``02'. Valid values are:	Correct the source line. `01 = Value `02 = Qualifier name
008419	Unknown keyword (`01) for qualifier ``02'. Valid keywords are:	Correct the source line. `01 = Value `02 = Qualifier name
008420	Incorrect value (`01) for qualifier ``02'. It must be an integer value > 0.	Correct the source line. `01 = Value `02 = Qualifier name
008421	Incorrect value (`01) for qualifier ``02'. It must be a numeric value > 0.	Correct the source line. `01 = Value `02 = Qualifier name
008422	Incorrect value (`01) for qualifier ``02'. It must be a string literal.	Correct the source line. `01 = Value `02 = Qualifier name
008423	Incorrect value (`01) for qualifier ``02'. It must be either YES or NO.	Correct the source line. `01 = Value `02 = Qualifier name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
008424	Incorrect value (`01) for qualifier ``02'. It must be either YES, NO, or the number of lines.	Correct the source line. `01 = Value `02 = Qualifier name
008425	Qualifier ``01' has already been defined.	Correct the source line. `01 = Qualifier name `02 = List Keyword
008426	There are TABLE-FORMAT entries for column numbers which exceed the number of columns as defined by the COLUMN-COUNT qualifier.	Correct the source line.
008427	The value for SIZE= (`01) must be > 0.	Correct the source line.
008428	The value for EXTENT= (`01) must be > 0.	Correct the source line.
008429	Encountered an invalid VALUES= argument. Legal aruments types are \$Variable and #Variable.	Correct the source line.
008430	Encountered an invalid (`01) VALUES= argument. Legal argument types are \$Variable, #Variable, &Variable, 'Literal', Numerics, and the word NULL.	Correct the source line. `01 = List contents
008431	The type of VALUES= parameter (`01) is not compatible with the corresponding table column (`02) defined as ``03'.	The VALUES= parameter type is not compatible with the corresponding TABLE column type. `01 = VALUES parameter `02 = Table column number `03 = Table column type
008432	The number of parameters in the VALUES= list exceeds the number of columns (`01) for table (`02).	The number of parameters in the VALUES= list exceeds the number of columns defined in the table. `01 = Table column count `02 = Table name
008433	The value for MODE= (`01) must be either BEFORE or AFTER.	Correct the source line.
008434	Cannot perform GET because the specified table (`01) is empty.	The ACTION=GET option cannot be used against an empty table. `01 = Table name

Table 7-2 Numbered Messages (*Continued*)

Error Number	Error Message	Suggestion/Interpretation
008435	The number of columns (`01) defined for array (`02) does not match the number of columns (`03) defined for table (`04).	The number of columns defined for the specified array must be the same as the number of columns defined for the specified table. `01 = Array column count `02 = Array name `03 = Table column count `04 = Table name
008436	Column type mismatch in column (`01): Array (`02) is defined as (`03) and Table (`04) is defined as (`05)	The column types for the specified array must be compatible with the columns types for the specified table. `01 = Column number `02 = Array name `03 = Array column type `04 = Table name `05 = Table column type
008437	Cannot perform GET from table (`01) for `02 rows because the array (`03) can only support (`04) rows.	The ACTION=GET option cannot be used to increase the size of array. `01 = Table name `02 = Table rows `03 = Array name `04 = Array size
009999	The printer (`01) specified with the -Printer:WP command line flag is invalid.	(Windows) The specified printer is not valid.



Deprecated Information

This chapter discusses the current set of older (deprecated) SQR commands, command-line arguments, and a section in the SQR.INI file. We recommend that you phase this information out of your code as soon as it is feasible to do so. SQR may not support this information in future releases.

In This Chapter	Deprecated SQR Command-line Flags	8-2
	Deprecated SQR.INI Entries	8-2
	Deprecated Transforms	8-5
	Deprecated SQR Commands	8-6

Deprecated SQR Command-line Flags

Table 8-1 describes the SQR command-line flags that have been deprecated.

Table 8-1 Deprecated SQR Command-line Flags

Flag	Description
-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 8-2 for more information.)

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 8-2 lists the deprecated values for the FullHTML keyword.

Table 8-2 Deprecated Values for the FullHTML Keyword

Entry	Deprecated Values	Description
FullHTML	3.0 – Was used to generate HTML 3.0. TRUE – Was used to generate HTML 3.2. FALSE – Was used to generate HTML 3.0.	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 5-21 for more information on current values.

[Processing-Limits] Section

SQR has built-in default values as to how much memory to allocate to certain SQR internal structures. In versions of SQR 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, SQR automatically adjusts the internal structures until the architectural limit is reached.

The sizes and limitations of SQR's internal structures are defined in the [Processing-Limits] section of the SQR.INI file. 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 8-3.

Table 8-3 Entries for [Processing-Limits] Section

Entry	Old Default	New Default	Maximum Value	Description
BREAKS	100	1024	65535	Number of BREAK arguments allowed per EVALUATE command.
DYNAMICARGS	70	4096	32767	Maximum number of dynamic SQL arguments.
EXPRESSIONSPACE	8192	65535	65535	Maximum length, in bytes, of temporary string storage used during LET operations.
FORWARDREFS	200	1024	32767	Maximum number of column forward references.
ONBREAKS	30	1024	65535	Maximum number of ON-BREAK LEVEL=values per SET.
POSITIONS	1800	32767	65535	Maximum number of placement parameters, "(10,5,30)".
PROGLINEPARS	18000	32767	65535	Maximum number of arguments for all program lines.
PROGLINES	5000	16384	32767	Maximum number of program lines (SQR commands).
QUERIES	60	1024	32767	Maximum number of BEGIN-SQL and BEGIN-SELECT paragraphs.

Table 8-3 Entries for [Processing-Limits] Section (*Continued*)

Entry	Old Default	New Default	Maximum Value	Description
QUERYARGS	240	4096	65535	Maximum number of arguments (bind variables) for all BEGIN-SQL or BEGIN-SELECT paragraphs.
SQLSIZE	4000	16384	65535	Maximum length of an SQL statement in characters.
STRINGSPACE	15000	32767	65535	Maximum size of string space for program line arguments, in bytes.
SUBVARS	100	4096	32767	Maximum number of substitution variables.
VARIABLES	1500	16384	32767	Maximum number of variables (string, float, integer, decimal, date), literal values, and database columns.
WHENS	70	1024	65535	Maximum number of WHEN arguments allowed per EVALUATE command.



Note The entries in the [Processing-Limits] section are the same as those specified with the `-Mfile` command line flag. If the `-Mfile` command line flag is used, then the [Processing-Limits] section of the SQR.INI file is not processed. (See “`-Mfile`” on page 8-2 for more information.)

Values for PDFCompressionText and PDFCompressionGraphics in the [Default-Settings] Section

The PDFCompressionText and PDFCompressionGraphics settings in the [Default-Settings] section now appear in the [PDF Settings] section as CompressionText and CompressionGraphics. See “[PDF Settings] Section” on page 5-18 for more information.

Table 8-4 Deprecated Entries in the [Default-Settings] Section

Entry	Value	Description
PDFCompressionText	0 - 9	Each of these entries specifies the amount of compression to apply. The values range from 0 (no compression) to 9 (maximum compression).
PDFCompressionGraphics		The default value for both is 6, which is the best value for the compression versus speed.

Deprecated Transforms

Table 8-5 describes the transforms that have been deprecated.

Table 8-5 Deprecated Transforms

Transform	Description
ToCanonical	Transforms Unicode "compatibility characters" to their standard equivalents.
ToTraditionalChinese	Converts all Simplified Chinese characters to their Traditional Chinese equivalent.
ToSimplifiedChinese	Converts all Traditional Chinese characters to their Simplified Chinese equivalent.



Note

A transform is a function of the LET command. See “transform” on page 2-218 for information on the available transforms.

Deprecated SQR Commands

If you still have older SQR commands in your program code, refer to Table 8-6 as you replace them with their updated alternatives. Even though the commands are technically supported in this release, they do not interact well with the current SQR lexicon. Incorporating the deprecated commands into your SQR code can cause unpredictable results.

Table 8-6 **Deprecated SQR Commands**

Old Commands	Use Instead
BEGIN-REPORT (END-REPORT)	BEGIN-PROGRAM (END-PROGRAM)
DATE-TIME	datenow function
DECLARE PRINTER	DECLARE-PRINTER
DECLARE PROCEDURE	DECLARE-PROCEDURE
DOLLAR-SYMBOL	ALTER-LOCALE
GRAPHIC FONT	ALTER-PRINTER
MONEY-SYMBOL	ALTER-LOCALE
NO-FORMFEED	DECLARE-LAYOUT
PAGE-SIZE	DECLARE-LAYOUT
PRINTER-DEINIT	DECLARE-PRINTER
PRINTER-INIT	DECLARE-PRINTER
PRINT...CODE	PRINT...CODE-PRINTER



Note Two older commands, DECLARE PRINTER and DECLARE PROCEDURE, do not contain hyphens. The new commands, DECLARE-PRINTER and DECLARE-PROCEDURE, contain hyphens.

BEGIN-REPORT

**Note**

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer SQR functionality, use BEGIN-PROGRAM.

Function Begins a report.

Syntax BEGIN-REPORT

Description After processing the commands in the SETUP section, SQR 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 SQR 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 SQR 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

Specifies the position for printing the date.

date_format

A string literal containing the date format mask.

col_var

Causes the retrieved date-time to be placed 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 8-7 shows the default date-time formats for SQR-supported databases.

Table 8-7 Default Date-Time Formats

Database	Default Date-Time Format
DB2	YYYY-MM-DD-HH:MI YYYY-MM-DD HH:MI:SS.NNNNNN
Informix	YYYY-MM-DD HH:MI YYYY-MM-DD HH:MI:SS.NNN
Oracle	DD-Mon-YYYY HH:MI PM
Sybase	DD-MON-YYYY HH:MI

For some databases, there are two default formats. The first format prints the date-time, as in the following example:

```
date-time (+1,1)
```

The second format retrieves the date-time into a column variable, as follows:

```
date-time () '' &date1
```

Obviously, for those databases with only one default format, that format is always used in either of these cases.

For information on the valid edit mask format codes, see Table 2-36, “Miscellaneous Functions,” on page 2-220.

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
```

See Also

See the \$current-date reserved and datenow function described in Table 2-36, “Miscellaneous Functions,” on page 2-220 under the LET command. See also the ALTER-LOCALE command.

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 SQR 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 8-8 describes the arguments for the DECLARE PRINTER command.

Table 8-8 **DECLARE PRINTER** Command Arguments

Argument	Choice or Measure	Default Value	Description
TYPE	LINEPRINTER, POSTSCRIPT, HPLASERJET	LINEPRINTER	SQR creates output specific to each printer. LINEPRINTER files generally consist of ASCII characters and can be viewed by a text editor. POSTSCRIPT files consist of ASCII characters, but you need to know PostScript to understand what will be shown on the printer. HP Laserjet files are binary files and cannot be edited or viewed.
ORIENTATION	PORTRAIT, LANDSCAPE	PORTRAIT	Portrait pages are printed vertically. Landscape pages are printed horizontally. Printing in landscape on HP Laserjet printers requires landscape fonts.
LEFT-MARGIN	inches	0.5	This argument does not apply to LINEPRINTER printers. This is the amount of blank space to leave at the left side of the page.
TOP-MARGIN	inches	0.5	This argument does not apply to LINEPRINTER printers. This is the amount of blank space to leave at the top of the page.
LINE-SIZE	points	12	This argument does not apply to LINEPRINTER printers. This is the size of each SQR 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.
CHAR-SIZE	points	7.2	This argument does not apply to LINEPRINTER printers. This is the size of each SQR 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.
LINES-INCH	lines	6	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.
CHARS-INCH	characters	10	This argument does not apply to LINEPRINTER printers. This is an alternate way of indicating the width of each SQR character column, in characters per inch, rather than points for CHAR-SIZE.
POINT-SIZE	points	12	This argument does not apply to Lineprinter printers. This is the beginning size of the selected font.

Table 8-8 DECLARE PRINTER Command Arguments (*Continued*)

Argument	Choice or Measure	Default Value	Description
FONT-TYPE	PROPORTIONAL, FIXED	Depends on the font	This argument applies only to HP Laserjet printers and needs to be specified only for font types not defined in Table 2-22, "Fonts Available for HP LaserJet Printers in SQR," on page 2-126.
SYMBOL-SET	HP defined sets	OU	This argument applies only to HP Laserjet printers. The default value of "OU" is for the ASCII symbol set. For a complete list of the symbol sets, see the <i>HP Laserjet Technical Reference Manual</i> .
STARTUP-FILE	filename	POSTSCRI.STR	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.
FONT	font_number	3	This is the font number of the typeface to use. For HP Laserjet printers, this is the typeface value as defined by Hewlett-Packard. For a complete list of the typeface numbers, see the <i>HP Laserjet Technical Reference Manual</i> . For PostScript printers, SQR 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, 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 2-22, "Fonts Available for HP LaserJet Printers in SQR," on page 2-126 lists the fonts available in SQR internally. Table 2-23, "Fonts Available for PostScript Printers," on page 2-127 lists the fonts available in the SQR POSTSCRI.STR file.
BEFORE-BOLD	any string	(none)	The BEFORE-BOLD and AFTER-BOLD arguments are for Line-printer 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=<27>[r ! Turn on bold AFTER-BOLD=<27>[u ! Turn it off These arguments work in conjunction with the BOLD argument of the PRINT command.
AFTER-BOLD	any string	(none)	See BEFORE-BOLD.

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

For fonts not listed in Table 2-22, “Fonts Available for HP LaserJet Printers in SQR,” on page 2-126, you must indicate the font style using the FONT-TYPE argument, or the correct typeface cannot be selected by the printer.

Description

The DECLARE PRINTER command 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

SQR maps its line and column positions on the page by using a grid determined by the LINE-SIZE and CHAR-SIZE (or LINES-INCH and CHARS-INCH) arguments. Each printed piece of text is placed on the page using this grid. Because the characters in proportional fonts vary in width, it is possible that a word or string is wider than the horizontal space you have allotted, especially in words containing uppercase letters. To account for this behavior, you can either move the column position in the PRINT statement or indicate a larger CHAR-SIZE in the DECLARE PRINTER command.

DECLARE PROCEDURE



Note This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer SQR functionality, use DECLARE-PROCEDURE.

Function

Defines specific event procedures.

Syntax

```
DECLARE PROCEDURE  
[BEFORE-REPORT=procedure_name]  
[AFTER-REPORT=procedure_name]  
[BEFORE-PAGE=procedure_name]  
[AFTER-PAGE=procedure_name]
```

Arguments

BEFORE-REPORT

Specifies a procedure to execute at the time of the first PRINT command. It may be used, for example, to create a report heading.

AFTER-REPORT

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

BEFORE-PAGE

Specifies a procedure to execute at the beginning of every page, just before the first PRINT command for the page. It can be used, for example, to set up page totals.

AFTER-PAGE

Specifies a procedure to execute just before each page is written to the file. It can be used, for example, to display page totals.

Description

DECLARE PROCEDURE can be issued either in the SETUP section or in the body of the report. You can use the command as often as you like.

If you issue multiple DECLARE PROCEDURE commands, the last one takes precedence. In this way, you can turn procedures on and off while the report is executing. The referenced procedures do not take any arguments; however, the variables can be local by using the LOCAL argument. In addition, they can only PRINT into the body of the report, that is, they cannot PRINT into the header and/or footer areas.

Examples

```
declare procedure  
    before-page=page_setup  
    after-page=page_totals
```

DOLLAR-SYMBOL

**Note**

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer SQR functionality, use ALTER-LOCALE.

Function

Redefines the currency symbol within numeric edit masks.

Syntax

DOLLAR-SYMBOL *new_symbol*

Arguments

new_symbol

Specifies 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 SQR programs and the reports they produce.

This command is used only in the SETUP section.

**Note**

The MONEY-SYMBOL command has the same effect as these options of the ALTER-LOCALE command: MONEY-SIGN and MONEY-SIGN-LOCATION=LEFT.

Table 8-9 lists the characters that DOLLAR-SYMBOL cannot take.

Table 8-9 Characters Disallowed in the DOLLAR-SYMBOL Command

Type	Characters	
Numbers	0, 8, 9	
Alphabetical	b	B
	e	E
	n	N
	r	R
	v	V
Symbols	.	,
	-	+
	!	*
	-	`
	<	>
	()

Examples

The following example shows how to use the DOLLAR-SYMBOL command:

```
begin-setup
  dollar-symbol £! Define £ as the currency symbol
end-setup
begin-procedure
...
print #amount () edit fff,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
```

See Also

See the ALTER-LOCALE command for a description of other locale-specific parameters.

GRAPHIC FONT



Note This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer SQR 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 SQR does not know about. (See Table 2-23, “Fonts Available for PostScript Printers,” on page 2-127 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! Century Schoolbook, 8.5 points
graphic () font 6 12 0 10! Letter Gothic, 12 points,
               ! fixed, 10 characters per inch
graphic () font :#font_number :#point_size
```

See Also

See ALTER-PRINTER and DECLARE-PRINTER for information on setting and changing the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

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 SQR functionality, use the ALTER-LOCALE command.

Function

Redefines the currency symbol to be printed.

Syntax

MONEY-SYMBOL *new_symbol*

Arguments

new_symbol

Specifies a new, single character to replace the dollar sign (\$) or DOLLAR-SYMBOL character on the printed report.

Description

If you wish 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 SQR application programs and the reports they produce.

To indicate a non-edit character, surround its decimal value with angle brackets (<>). Refer to Table 8-9 on page 8-17 for characters that cannot be used with MONEY-SYMBOL.

**Note**

The MONEY-SYMBOL command has the same effect as these options of the ALTER-LOCALE command: MONEY-SIGN and MONEY-SIGN-LOCATION=LEFT.

Examples

The following example shows how to use the DOLLAR-SYMBOL and MONEY-SYMBOL commands:

```
begin-setup
  dollar-symbol f! Define f as the
    ! currency symbol
end-setup
begin-procedure! If #Amount=1234.56
...
money-symbol f
print #Amount () Edit fff,999.99! Prints as: f1,234.56
...
money-symbol $
print #Amount () Edit fff,999.99! Prints as: $1,234.56
...
money-symbol
print #Amount () Edit fff,999.99! Prints as: 1,234.56
...
end-procedure
```

See Also

See the DOLLAR-SYMBOL and ALTER-LOCALE commands.

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 SQR 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
```

See Also

See the FORMFEED qualifier in DECLARE-LAYOUT.

PAGE-SIZE

 **Note**

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer SQR 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, SQR 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 SQR trims trailing blanks on each line before writing the report file.

The size of the internal page buffer used to store a complete page in memory can be determined by multiplying the page depth by the width in the PAGE-SIZE command. For PCs, the page buffer is limited to 64K bytes. On other computers, the page buffer is limited only by the amount of memory available.

Examples

```
begin-setup
  page-size 57 132! 57 lines long by 132 columns wide
end-setup
```

See Also

See the PAPER-SIZE parameter of the DECLARE-LAYOUT command.

PRINT ...CODE

The PRINT command has the following format option:

CODE

CODE is a qualifier that may be discontinued in a future release. Use CODE-PRINTER instead.

If you use CODE, the sequence is assumed to be for the printer type specified in the DECLARE-REPORT or default printer, if none is specified.

PRINTER-DEINIT

**Note**

This command may be discontinued in a future release. We highly recommend that you no longer use this command. To take advantage of newer SQR functionality, use the RESET-STRING parameter of the DECLARE-PRINTER command.

Function

Sends control or other characters to the printer at the end of a report.

Syntax

PRINTER-DEINIT *initialization_string*

Description

Specify nondisplay characters by placing their decimal values inside angled brackets. For example, <27> is the ESC or escape character.

The PRINTER-DEINIT command 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
```

See Also

See the ENCODE command for another method of printing nondisplay characters. See the *chr* function in Table 2-36, “Miscellaneous Functions,” on page 2-220 under the LET command.

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 SQR 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.

The PRINTER-INIT command 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
```

See Also

See the ENCODE command for another method of printing non-display characters. See the *chr* function in Table 2-36, “Miscellaneous Functions,” on page 2-220 under the LET command.

Index

Symbols

#character, defined, 1-7
#current-column, 1-10
#current-line, 1-10
#DEFINE, 2-140, 2-185 – 2-186
#ELSE, 2-150, 2-182
#end-file, 1-10, 2-301
#END-IF, 2-155
#ENDIF, 2-155, 2-182
#IF, 2-181 – 2-182
#IFDEF, 2-185
#IFNDEF, 2-182, 2-186
#INCLUDE, 1-20, 2-187
#page-count, 1-10
#return-status, 1-10
#sql-count, 1-10
#sql-status, 1-11
#sqr-max-columns, 1-12
#sqr-max-lines, 1-12
#sqr-pid, 1-12
\$character, defined, 1-7
\$current-date, 1-10
\$sql-error, 1-11
\$sqr-database, 1-11
\$sqr-dbcs, 1-11
\$sqr-encoding, 1-11
\$sqr-encoding-console, 1-12
\$sqr-locale, 1-12
\$sqr-platform, 1-12

\$sqr-program, 1-12

\$sqr-report, 1-12

\$sqr-ver, 1-12

\$username, 1-12

&character, defined, 1-7

@character, defined, 1-7

A

-A, 1-15

abs, 2-199

acos, 2-199

ADD, 2-2

AFTER, 2-33, 2-35

AFTER qualifier, ON-BREAK argument, 2-274

AFTER-BOLD argument

DECLARE-PRINTER command, 2-124, 8-12

PRINT command, 2-257

AFTER-PAGE argument

DECLARE PROCEDURE command, 8-14

DECLARE-PROCEDURE command, 2-130

DECLARE-TOC command, 2-134 – 2-135

USE-PROCEDURE command, 2-327

AFTER-REPORT argument

DECLARE PROCEDURE command, 8-14

DECLARE-PROCEDURE command, 2-130

USE-PROCEDURE command, 2-327

AFTER-TOC argument

DECLARE-TOC command, 2-134

ALTER-COLOR-MAP, 2-3

ALTER-CONNECTION, 2-4, 2-176, 2-308
ALTER-LOCALE, 2-9
ALTER-PRINTER, 2-18, 2-123, 2-128
ALTER-REPORT, 2-20
arccosine, 2-199
arcsine, 2-199
arctangent, 2-200
array
 defined, 2-220
 elements, 2-77
arrays
 ARRAY-ADD command, 2-28
 ARRAY-DIVIDE, 2-28
 ARRAY-MULTIPLY, 2-28
 ARRAY-SUBTRACT command, 2-28
 maximums, 2-76
ascii, 2-207
asciic, 2-207
asin, 2-199
ASK, 2-31, 2-185 – 2-186
atan, 2-200
AT-END argument, NEXT-COLUMN command,
 2-246
ATTRIBUTES argument
 declaration keywords, 2-100
 definition of, 2-98
 selector/sub-selector keywords, 2-99
audience for this guide, 1-xiii

DECLARE PROCEDURE command, 8-14
DECLARE-PROCEDURE command, 2-130
DECLARE-TOC command, 2-134
USE-PROCEDURE command, 2-327
BEFORE-REPORT argument
 DECLARE PROCEDURE command, 8-14
 DECLARE-PROCEDURE command, 2-130
 USE-PROCEDURE command, 2-327
BEFORE-TOC argument, DECLARE-TOC
 command, 2-134
BEGIN-DOCUMENT, 2-32
BEGIN-EXECUTE, 2-33
BEGIN-HEADING
 See also Page Header
BEGIN-PROCEDURE, 2-43
BEGIN-PROGRAM, 2-46
BEGIN-REPORT, 8-7
BEGIN-SELECT, 2-33, 2-47
BEGIN-SETUP, 2-52
BEGIN-SQL, 2-54
bind variables, 2-48
BLINK argument, SHOW command, 2-311
-Bnn
 BEGIN-SELECT command, 2-47
 definition, 1-15
BOLD argument
 PRINT command, 2-257
 SHOW command, 2-311
BOTTOM-MARGIN argument
 DECLARE-LAYOUT command, 2-117
BOX and SHADE, 2-257
BOX argument
 GRAPHIC command, 2-178 – 2-179
 PRINT command, 2-257
BREAK, 2-58
BREAK argument, EVALUATE command, 2-159
break processing, field changes, 2-273
-BURST,{xx}, 1-16

B

bar codes, 2-281
BATCH-MODE, INPUT command, 2-188
BEEP argument, SHOW command, 2-311
BEFORE, 2-33, 2-35
BEFORE qualifier, ON-BREAK argument, 2-274
BEFORE-BOLD argument
 DECLARE-PRINTER command, 2-124, 8-12
 PRINT command, 2-257
BEFORE-PAGE argument

C

-C, 1-16
CALL, 2-59
callable SQR, 1-16, 1-23
cancel dialog box, 1-16
CAPTION argument, PRINT-BAR-CODE, 2-281
-CB, 1-16
ceil, 2-200
CHARS-INCH argument, DECLARE-PRINTER command, 8-11
CHAR-SIZE argument, DECLARE-PRINTER command, 8-11
charts, 2-284
CHAR-WIDTH argument, DECLARE-LAYOUT command, 2-117
CHECKSUM argument, PRINT-BAR-CODE, 2-282
chr, 2-207
CLEAR-ARRAY, 2-66, 2-75
CLEAR-LINE argument, SHOW command, 2-310
CLEAR-SCREEN argument, SHOW command, 2-310
CLOSE, 2-67
-Cnn argument
 BEGIN-SELECT command, 2-47
 BEGIN-SQL command, 2-54
COLOR argument
 ALTER-COLOR-MAP command, 2-3
 DECLARE-COLOR-MAP command, 2-107
 DECLARE-PRINTER command, 2-124
 GET-COLOR command, 2-175
 SET-COLOR command, 2-307
COLUMNS, 1-7, 2-68, 2-246, 2-325
COLUMNS argument, POSITION command, 2-254
COMMAND, 2-33 – 2-34, 2-228 – 2-229, 2-231
command line
 arguments, 1-13
 flags, 1-15
command_line, 2-220
compiler directives, 2-150, 2-155, 2-181, 2-185 – 2-186
CONCAT, 2-71

concatenation, 2-317
cond, 2-220
conditional processing, 2-159, 2-183, 2-330
CONNECT, 2-73
CONNECTION, 2-33 – 2-34
connection_name, 2-3, 2-5, 2-109, 2-175, 2-307
consulting services, 1-xvii
copyright banner, 1-20
cos, 2-200
cosh, 2-200
cosine, 2-200
CREATE-ARRAY, 2-75, 2-299
CREATE-COLOR-PALETTE, 2-78
CREATE-LIST, 2-80
csv file, 1-18
csv icon, 1-18
currency symbol, 8-16, 8-19
custom functions, 2-225

D

data elements, 1-7
data input, 2-188
data types supported
 by INPUT command, 2-189
database cursors, 2-74
database type, 1-11
DATE argument
 DECLARE-VARIABLE command, 2-136
 DISPLAY command, 2-142
 MOVE command, 2-239
 PRINT command, 2-259
 SHOW command, 2-311
date edit format characters, 2-263
date functions
 dateadd, 2-214
 datediff, 2-214
 datenow, 2-214
 datetotstr, 2-215
 strtotime, 2-215
 dateadd, 2-214

datediff, 2-214
DATE-EDIT-MASK argument, ALTER-LOCALE command, 2-10
datenow, 2-214
DATE-SEPARATOR argument, ALTER-LOCALE command, 2-11
DATE-TIME, 8-8
datetosr, 2-215
DAY-OF-WEEK-CASE argument, ALTER-LOCALE command, 2-11
DAY-OF-WEEK-FULL argument, ALTER-LOCALE command, 2-11, 5-13
DAY-OF-WEEK-SHORT argument, ALTER-LOCALE command, 2-12, 5-13
DB2, 8-9
-DBconnectionstring argument, BEGIN-SELECT, 2-48, 2-54
-DBdatabase, 1-16
DDO, 2-229 – 2-231, 2-255
ALTER-CONNECTION command, 2-5
BEGIN-EXECUTE command, 2-33
BEGIN-SELECT command, 2-49
BEGIN-SQL command, 2-55
COMMIT command, 2-69
CONNECT command, 2-73
DECLARE-CONNECTION command, 2-109
DECLARE-VARIABLE command, 2-138
LET command, 2-193
LOAD-LOOKUP command, 2-229
PRINT command, 2-255
ROLLBACK command, 2-303
variables in the Environment section, 5-10
-DEBUG, 1-16, 2-181, 2-185 – 2-186
DECIMAL argument, DECLARE-VARIABLE command, 2-136
DECIMAL-SEPARATOR argument, ALTER-LOCALE command, 2-11
DECLARE PRINTER, 8-10
DECLARE PROCEDURE, 8-14
DECLARE-CHART, 2-83, 2-287
DECLARE-COLOR-MAP, 2-107
DECLARE-CONNECTION, 2-34, 2-109
DECLARE-IMAGE, 2-112
DECLARE-LAYOUT, 2-115
 arguments with, 2-116
 PAPER-SIZE parameter, 8-22
DECLARE-PRINTER, 2-96, 2-123, 2-257
DECLARE-PROCEDURE, 2-130
DECLARE-REPORT, 2-132
DECLARE-TOC, 2-134
DECLARE-VARIABLE, 2-136
DEFAULT-NUMERIC argument
 DECLARE-VARIABLE command, 2-136
Default-Settings section, in SQR.INI, 5-4
deg, 2-201
delete, 2-206
destination field, 2-239
DISPLAY, 2-142
DISTINCT argument, BEGIN-SELECT command, 2-47
DIVIDE, 2-146
-Dnn, 1-16
-DNT, 1-17
DO, 2-43, 2-46, 2-148, 8-7
DO argument, EXECUTE command, 2-162
document marker, 2-32, 2-254
DOCUMENT paragraphs, 6-3
documents
 conventions used, 1-xvi
 feedback, 1-xviii
 ordering print documents, 1-xvi
 structure of, 1-xiv
documents, accessing
 Hyperion Download Center, 1-xv – 1-xvi
 Hyperion Solutions Web site, 1-xv
 Information Map, 1-xv
 online help, 1-xv
DOLLAR-SYMBOL, 8-16, 8-19
DOT-LEADER argument, DECLARE-TOC command, 2-134

DSN, 2-3, 2-5, 2-107, 2-109
DSQUERY, 5-9
dynamic query variables, 2-48

E

-E, 1-17
edit, 2-207
EDIT argument
 SHOW command, 2-310
edit masks
 case sensitivity, 2-264
 default formats, 2-270
 description of, 2-264
 samples, 2-266
 uses, 2-267
 with specified width values, 2-268
edit types, 2-260
EDIT-OPTION-AD argument, ALTER-LOCALE command, 2-11, 5-13
EDIT-OPTION-AM argument, ALTER-LOCALE command, 2-10 – 2-11
EDIT-OPTION-BC argument, ALTER-LOCALE command, 2-11, 5-13
EDIT-OPTION-NA argument, ALTER-LOCALE command, 2-11
EDIT-OPTION-PM argument, ALTER-LOCALE command, 2-11
education services, 1-xvii
-EH_APPLETS, dir, 1-17
-EH_BQD, 1-17
-EH_BQD, file, 1-17
-EH_CSV, 1-18
-EH_CSV, file, 1-18
-EH_CSVONLY, 1-18
-EH_FULLHTML, xx, 1-18
-EH_ICONS, dir, 1-18
-EH_IMAGES, dir, 1-19
-EH_KEEP, 1-19
-EH_LANGUAGE, xx, 1-19
-EH_PDF, 1-19

-EH_SCALE, nn, 1-19
-EH_XML, file, 1-19
-EH_ZIP, file, 1-19
ELSE, 2-151
ENCODE, 2-152
Encoding, 2-216 – 2-219, 2-250
END-DECLARE, 2-107, 2-153
END-DOCUMENT, 2-153
END-EVALUATE, 2-153
END-EXECUTE, 2-33
END-FOOTING, 2-153
END-HEADING, 2-153
END-IF, 2-156
ending a query, 2-168
END-PROCEDURE, 2-157
END-PROGRAM, 2-157
END-SELECT, 2-33, 2-157
END-SETUP, 2-157
END-SQL, 2-157
END-WHILE, 2-157
Enhanced-HTML, 5-21, 5-24
entering SQR commands, 1-6
environment variables, 5-9
Environment, Common, 5-9
ERASE-PAGE argument, NEXT-COLUMN command, 2-246
error message file, 1-24
EVALUATE, 2-58, 2-159
Examples, 2-4
exists, 2-206
EXIT-SELECT, 2-168
exp, 2-201
exponents, 2-201
expressions, 2-193
EXTENT argument, LOAD-LOOKUP command, 2-230
external source files, 2-187
EXTRACT, 2-169

F

-F, 1-20
FIELD argument, CREATE-ARRAY command, 2-75
fields See columns, 2-75
file number, 2-250
file-related functions
 delete, 2-206
 exists, 2-206
 rename, 2-206
files, reading, 2-300
FIND, 2-171
FIXED argument, OPEN command, 2-251
FIXED_NOLF argument, OPEN command, 2-251
flags, in the SQR command line, 1-15
FLOAT argument, DECLARE-VARIABLE command,
 2-136
floor, 2-201
FONT argument, DECLARE-PRINTER command,
 2-124, 8-12
fonts, 2-129
 HP Laser Jet, 2-126
 PostScript, 2-127
 Windows printers, 2-128
Fonts section, in SQR.INI file, 5-14
FONT-TYPE argument, DECLARE-PRINTER
 command, 2-124, 8-12
FOOTING, 2-39
 footings, 2-39
FOOTING section, 1-3
FOR-APPEND argument, OPEN command, 2-250
FORMFEED argument, DECLARE-LAYOUT
 command, 2-116
FOR-READING argument, OPEN command, 2-250
FOR-REPORTS argument
 DECLARE-PRINTER command, 2-125
 DECLARE-PROCEDURE command, 2-130
 DECLARE-TOC command, 2-134
 USE-PROCEDURE command, 2-327
FOR-REPORTS, BEGIN-HEADING command, 2-41
FOR-WRITING argument, OPEN command, 2-250

FROM PARAMETER, 2-35, 2-231
FROM ROWSET, 2-35, 2-231
functions
 date, 2-214
 file-related, 2-206
 miscellaneous, 2-220
 numeric, 2-199
 string, 2-207
 transcendental, 2-205
 unicode, 2-216
 writing custom functions, 2-225

G

general purpose procedures, in HTML, 3-2
GET, 2-173
GET-COLOR, 2-175
GETDATA, 2-33 – 2-34, 2-228 – 2-229, 2-231
getenv, 2-221
global variables, 1-8
GOTO, 2-177
GOTO-TOP argument, NEXT-COLUMN
 command, 2-246
GRAPHIC BOX, 2-178
GRAPHIC FONT, 8-18
GRAPHIC HORIZ-LINE, 2-179
GRAPHIC VERT-LINE, 2-180

H

halting SQR, 2-316
hardware/operating system, 1-12
HEADING, 2-41
heading procedures, in HTML, 3-5
Hebrew language support, 2-280
HEIGHT argument, PRINT-BAR CODE command,
 2-281
hex, 2-202
highlighting procedures, in HTML, 3-7
HORIZ-LINE argument, GRAPHIC command, 2-179
HPLASERJET, 2-125

HTML

- general purpose procedures, 3-2
- heading procedures, 3-5
- highlighting procedures, 3-7
- hypertext link procedures, 3-10
- list procedures, 3-11
- table procedures, 3-15

HTML-Images

- hyperbolic cosine, 2-200
- hyperbolic sine, 2-204
- hyperbolic tangent, 2-205
- Hyperion Consulting Services, 1-xvii
- Hyperion Download Center
 - accessing documents, 1-xvi
- Hyperion Education Services, 1-xvii
- Hyperion product information, 1-xvii
- Hyperion Solutions Web Site
 - accessing documents, 1-xv
- Hyperion support, 1-xvii
- Hyperion Technical Support, 1-xviii
- hypertext link procedures, in HTML, 3-10

I

- ID, 1-20
- IF, 2-183
 - IF nested within a WHILE loop, 2-331
- IFDEF, 2-182
- images
 - DECLARE-IMAGE command, 2-112
 - printing, 2-295
- IMAGE-SIZE argument
 - DECLARE-IMAGE command, 2-112
 - PRINT-IMAGE command, 2-295
- INDENTATION argument, DECLARE-TOC command, 2-134
- Informix, 1-14, 2-316, 8-9
- INIT-STRING argument, DECLARE-PRINTER command, 2-125
- INPUT, 2-188

instr, 2-208

instrb, 2-208

INTEGER argument, DECLARE-VARIABLE command, 2-137

INTO argument, EXECUTE command, 2-163

isblank, 2-208

isnull, 2-221

K

-KEEP, 1-20, 2-133

KEY argument, LOAD-LOOKUP command, 2-230

L

labels, 2-177

LAST-PAGE, 2-192

LAYOUT argument, DECLARE-REPORT command, 2-132

-ldir_list, 1-20

LEFT-MARGIN argument

DECLARE-LAYOUT command, 2-117

DECLARE-PRINTER command, 8-11

length, 2-209

lengthb, 2-209

lengthh, 2-216

lengthp, 2-216

LET, 2-193

LEVEL argument, TOC-ENTRY command, 2-320

LEVEL qualifier, ON-BREAK argument, 2-274

LINE-HEIGHT argument, DECLARE-LAYOUT command, 2-117

LINE-INCH argument, DECLARE-PRINTER command, 8-11

LINEPRINTER, 2-125

LINE-SIZE argument, DECLARE-PRINTER command, 8-11

LINE-WIDTH argument, DECLARE-LAYOUT command, 2-117

list procedures, in HTML, 3-11

literals, 1-7

-LL, 1-20
loading an internal table, 2-228
LOAD-LOOKUP, 1-20, 2-228, 2-235
local procedures, 2-43
local variables, 1-8
LOCALE, 5-12
LOCALE argument, ALTER-LOCALE command, 2-10
log, 2-202
log10, 2-202
logical expressions, 2-183
LOOKUP, 2-232, 2-235
LOOPS argument, BEGIN-SELECT command, 2-48
lower, 2-209
LOWERCASE, 2-236
lpad, 2-209
ltrim, 2-210

M

MATCH argument, PRINT command, 2-273
MAX-COLUMNS argument, DECLARE-LAYOUT command, 2-117
MAXLEN arguments, INPUT command, 2-188
MAX-LINES argument, DECLARE-LAYOUT command, 2-117
MBTOSBS, 2-237
-Mfile, 8-2
Microsoft SQL Server, 1-14
miscellaneous functions
 array, 2-220
 command_line, 2-220
 cond, 2-220
 getenv, 2-221
 isnull, 2-221
 nvl, 2-222
 range, 2-223
 roman, 2-223
 wrapdepth, 2-224
mod, 2-202
MONEY argument

DISPLAY command, 2-142
MOVE command, 2-238
PRINT command, 2-273
SHOW command, 2-310
MONEY-EDIT-MASK argument, ALTER-LOCALE command, 2-10
MONEY-SIGN argument, ALTER-LOCALE command, 2-10
MONEY-SIGN-LOCATION argument, ALTER-LOCALE command, 2-11
MONEY-SYMBOL, 8-16, 8-19
MONTHS-CASE argument, ALTER-LOCALE command, 2-12
MONTHS-FULL argument, ALTER-LOCALE command, 2-12, 5-13
MONTHS-SHORT argument, ALTER-LOCALE command, 2-12
MOVE, 2-238
multiple reports, 2-18
MULTIPLY, 2-243

N

NAME, 2-5
NAME argument, 2-66
 CREATE-ARRAY command, 2-75
 LOAD-LOOKUP command, 2-229
natural log base e raised to x power, 2-201
NEED argument, NEXT-LISTING command, 2-248
nesting
 arguments, 2-198
 IF command, 2-183
nesting levels, #INCLUDE command, 2-187
NewGraphics, 2-85, 5-5
NEW-PAGE, 2-244, 8-22
NEW-REPORT, 2-245
NEXT-COLUMN, 2-68, 2-246
NEXT-LISTING, 2-248, 8-23
no logon, 1-23
NO-ADVANCE argument, NEXT-LISTING command, 2-248

NO-DUPLICATE, 2-6, 2-110
NO-FORMFEED, 8-21 – 8-22
NOLINE argument
 DISPLAY command, 2-142
 SHOW command, 2-311
-NOLIS, 1-20, 2-133
non-Windows, 1-16, 1-23
NOPROMPT arguments, INPUT command, 2-188
NORMAL argument, SHOW command, 2-311
NOWAIT, CALL command, 2-60
NUMBER argument
 DISPLAY command, 2-142
 MOVE command, 2-238
 PRINT command, 2-273
 SHOW command, 2-310
NUMBER-EDIT-MASK argument, ALTER-LOCALE command, 2-10
numeric functions, 2-199
 10 raised to x power, 2-201
 absolute value, 2-199
 arccosine, 2-199
 arcsine, 2-199
 arctangent, 2-200
 cosine, 2-200
 degrees, 2-201
 hyperbolic cosine, 2-200
 hyperbolic sine, 2-204
 hyperbolic tangent, 2-205
 largest integer, 2-201
 log base 10, 2-202
 log base e, 2-202
 natural log base e raised to x power, 2-201
 power, 2-203
 radians, 2-203
 round, 2-203
 sign, 2-204
 sine, 2-204
 square root, 2-204
 tangent, 2-205

nvl, 2-222

0

-O, 1-21
ODBC, 1-14
ON-BREAK
 AFTER qualifier, 2-274
 BEFORE qualifier, 2-274
 in PRINT command, 2-273
 SET qualifier, 2-274
ON-ERROR argument, 6-2
BEGIN-EXECUTE command, 2-33
BEGIN-SELECT command, 2-49
BEGIN-SQL command, 2-55
CONNECT command, 2-73
DIVIDE command, 2-146
EXECUTE command, 2-162
OPEN, 2-250
open a new report, 2-245
operands, 2-194
operators, 2-195
Oracle, 1-14, 2-47, 8-9
ORIENTATION argument
 DECLARE-LAYOUT command, 2-116
 DECLARE-PRINTER command, 8-11
OUTPUT argument, EXECUTE command, 2-163
output file, 1-20

P

page numbering, 2-192
page overflow, 2-244
PAGE-DEPTH argument, DECLARE-LAYOUT command, 2-117
PAGE-NUMBER, 2-253
PAGE-SIZE, 8-22
PAPER-SIZE argument, DECLARE-LAYOUT command, 2-116, 8-22
PARAMETER_LIST, 2-34, 2-231

PARAMETERS, 2-5, 2-109
PASSWORD, 2-5, 2-109
-PB, 1-21
PC, 2-60
PDF, 5-18
PITCH argument, DECLARE-PRINTER command, 2-125
PL/SQL, 2-57
POINT-SIZE argument, DECLARE-PRINTER command, 2-125, 8-11
POSITION, 2-254
position, PRINT-IMAGE command, 2-295
POSTSCRIPT, 2-125
power, 2-203
precision, 2-194
prerequisites for using this guide, 1-xiii
PRINT
 BOLD format command, 2-257
 BOX format command, 2-257
 CENTER format command, 2-257
 CODE-PRINTER format command, 2-258
 DATE format command, 2-259
 definition, 2-255
 DELAY format command, 2-259
 EDIT format command, 2-260
 FILL format command, 2-271
 FONT format command, 2-271
 FOREGROUND/BACKGROUND format command, 2-271
 format commands, 2-255, 8-24
 MATCH format command, 2-272
 MONEY format command, 2-273
 NOP format command, 2-273
 NUMBER format command, 2-273
 ON-BREAK format command, 2-273
 POINT-SIZE format command, 2-277
 SHADE format command, 2-277
 UNDERLINE format command, 2-277
 URL format command, 2-277
 URL-TARGET format command, 2-278
WRAP format command, 2-278
PRINT qualifier, ON-BREAK argument, 2-274
PRINT...CODE, 8-24
PRINT-BAR-CODE, 2-281
PRINT-CHART, 2-95, 2-284
PRINT-DIRECT, 2-294
-PRINTER
 WP, 2-128
 xx, 1-21
PRINTER qualifier, 2-294
printer type, 1-21, 2-326
PRINTER-DEINIT, 8-25
PRINTER-INIT, 8-26
PRINTER-TYPE argument, DECLARE-REPORT command, 2-132
PRINT-IMAGE
 definition, 2-295
 position, 2-295
printing
 Hebrew, 2-260
 on Windows, 1-21, 2-128
PROCEDURE, 2-33 – 2-34, 2-228 – 2-229, 2-231
PROCEDURE section, 1-3
procedures
 running, 2-148
 USE-PROCEDURE command, 2-327
process ID, 1-12
PROGRAM section, 1-3
prompt, 2-189
PUT, 2-298

Q

QUIET argument
 LOAD-LOOKUP command, 2-231
 STOP command, 2-305, 2-316

R

rad, 2-203

range, 2-223
READ, 2-300
RECORD argument, OPEN command, 2-251
rename, 2-206
replace, 2-210
report arguments file on command line, 1-13
report layout, 2-115
report output file, 1-12
reserved variables
 #current-column, 1-10
 #current-line, 1-10
 #end-file, 1-10
 #INCLUDE, 1-20
 #max-columns, 1-12
 #page-count, 1-10
 #return-status, 1-10
 #sql-count, 1-10
 #sql-status, 1-11, 2-44
 #sqr-max-lines, 1-12
 #sqr-pid, 1-12
 #sqrpid, 1-12
 \$current-date, 1-10
 \$sql-error, 1-11, 2-44
 \$sqr-database, 1-11
 \$sqr-dbcS, 1-11
 \$sqr-encoding, 1-11
 \$sqr-encoding-console, 1-12
 \$sqr-local, 1-12
 \$sqr-platform, 1-12
 \$sqr-program, 1-12
 \$sqr-report, 1-12
 \$sqr-ver, 1-12
 \$username, 1-12, 2-74
RESET-STRING argument, DECLARE-PRINTER command, 2-125
RETURN_VALUE argument, LOAD-LOOKUP command, 2-230
REVERSE argument, SHOW command, 2-311
reversed characters, 2-280

RIGHT-MARGIN argument, DECLARE-LAYOUT command, 2-117
ROLLBACK, 2-303, 2-316
roman, 2-223
round, 2-203
ROUND argument
 ADD command, 2-2, 2-146, 2-243
 SUBTRACT command, 2-319
ROWS argument, LOAD-LOOKUP command, 2-230
ROWSET, 2-35, 2-231
rpad, 2-210
-RS, 1-22
RSV, 2-33 – 2-34
-RT, 1-22
rtrim, 2-210
run-time report files, 1-22

S

-S, 1-22
SAVE qualifier, ON-BREAK argument, 2-274
SBTOMBS, 2-304
SCHEMA, 2-34, 2-231
screen I/O, 2-310
SECURITY, 2-305
security issues, username/password, 2-74
sequential processing, FOR-READING, 2-250
SET qualifier, ON-BREAK argument, 2-274
SET-COLOR, 2-307
SETUP, 2-107, 2-109
SETUP section, 1-3, 2-52
SHADE argument, PRINT command, 2-277
SHOW, 2-310
sign, 2-204
sin, 2-204
sine, 2-204
sinh, 2-204
SIZE argument, CREATE-ARRAY command, 2-75
SKIPLINES argument, NEXT-LISTING command, 2-248

SKIPLINES qualifier, ON-BREAK argument, 2-274
SORT argument, LOAD-LOOKUP command, 2-230
SOURCE argument
 DECLARE-IMAGE command, 2-112
 PRINT-IMAGE command, 2-296
source field, 2-239
SPF files, 1-20
SQR banner suppression, 1-23
SQR Extension, 5-11
SQR messages, 7-1
SQR page buffer, 2-294
SQR program file, 1-12
SQR program structure, 1-3
SQR Remote section, in SQR.INI, 5-24
SQR version, 1-12
SQR.INI file, 5-1
SQRDIR, 5-9
SQRFLAGS, 5-9
sqrt, 2-204
SQRT, compiling reports, 1-22
square root, 2-204
starting, 8-7
startup file, 8-2
STARTUP-FILE argument, DECLARE-PRINTER
 command, 2-125, 8-12
STATUS, 2-33 – 2-34
STATUS argument
 INPUT command, 2-188
 OPEN command, 2-251
STOP, 2-316
stored procedures
 Oracle, 2-56 – 2-57
 -XP, 1-24
STRING, 2-317
string functions
 ascii, 2-207
 asciic, 2-207
 chr, 2-207
 edit, 2-207
 instr, 2-208
 instrb, 2-208
 isblank, 2-208
 length, 2-209
 lengthb, 2-209
 lower, 2-209
 lpad, 2-209
 ltrim, 2-210
 replace, 2-210
 rpad, 2-210
 rtrim, 2-210
 substr, 2-211
 substrb, 2-211
 to_char, 2-211
 to_multi_byte, 2-212
 to_number, 2-212
 to_single_byte, 2-212
 translate, 2-213
 upper, 2-213
 string values, 2-220
 strtotime, 2-215
 structure, 1-3
STRUCTURE, See List Variable, 1-8
subroutines
 calling, 2-59
 writing, 2-60
substitution variables
 #DEFINE command, 2-140
 #IF command, 2-181
 #IFDEF command, 2-185
 #IFNDEF command, 2-186
 ASK command, 2-31
substring, 2-169, 2-321
substringb, 2-211
substrp, 2-217
substrt, 2-217
SUBTRACT, 2-319
SYBASE, 2-47
Sybase
 Default Date-Time Format, 8-9
 Description, 1-14

server, 1-23
STOP command, 2-316
USE command, 2-324
Sybase DB-Lib, 2-48, 2-54
SYMBOL-SET argument, DECLARE-PRINTER command, 2-125, 8-12
syntax conventions, 1-4
syntax, conventions, 1-4
SYSTEM argument, CALL command, 2-63

T

-T{B}, 1-23
-T{Z}, 1-23
TABLE argument, LOAD-LOOKUP command, 2-229
Table of Contents file, 1-16
table procedures, in HTML, 3-15
tabular reports
 NEXT-LISTING command, 2-248
ON-BREAK argument, 2-273
tan, 2-205
tangent, 2-205
tanh, 2-205
technical support, 1-xviii
testing, 1-23
TEXT argument
 DECLARE-VARIABLE command, 2-137
 PRINT-BAR-CODE, 2-281
 TOC-ENTRY command, 2-320
text edit format characters, 2-260
THOUSAND-SEPARATOR argument, ALTER-LOCALE command, 2-11
time, 8-8
TIME-SEPARATOR argument, ALTER-LOCALE command, 2-11
-Tnn, 1-23
to_char, 2-211
to_multi_byte, 2-212
to_number, 2-212
to_single_byte, 2-212

TOC argument, DECLARE-REPORT command, 2-132
TOC-ENTRY, 2-320
TOP-MARGIN argument
 DECLARE-LAYOUT command, 2-117
 DECLARE-PRINTER command, 8-11
trailing blanks, 1-24
transcendental functions, 2-205
transform, 2-218
translate, 2-213
trunc, 2-205
truncate, 2-205
TYPE argument
 DECLARE-IMAGE command, 2-112, 2-295
 DECLARE-PRINTER command, 2-125, 8-11
 INPUT command, 2-188
 PRINT-BAR-CODE, 2-281

U

ucall, 5-11
UCALL.C, 2-61 – 2-62
Ufunc, 5-11
UFUNC.C, 2-225
UNDERLINE argument
 SHOW command, 2-311
unicode, 2-219
unicode functions
 lengthp, 2-216
 lengtht, 2-216
 substrp, 2-217
 substrt, 2-217
 transform, 2-218
 unicode, 2-219
Unix, 1-20, 2-60
UNSTRING, 2-321
upper, 2-213
UPPERCASE, 2-323
USE, 2-324
USE-COLUMN, 2-68, 2-325
USE-PRINTER-TYPE, 2-326

USE-PROCEDURE, 2-327

USER, 2-5, 2-109

USE-REPORT, 2-329

V

valid uom suffixes, 2-115

value determination, 2-159

values of STATUS argument, 2-190

variables, 1-7

 global, 1-8

 local, 1-8

 reserved, 1-10

 rules, 1-8

VARY argument, OPEN command, 2-251

VERT-LINE argument, GRAPHIC command, 2-180

-Vserver, 1-23

W

WAIT, CALL command, 2-60

WHEN argument, EVALUATE command, 2-159

WHEN-OTHER argument, EVALUATE command,
 2-160

WHERE argument, LOAD-LOOKUP command,
 2-230

WHILE, 2-58, 2-330

Windows

 -C, 1-16

 CALL command, 2-60

 log messages, 1-21

 -XCB, 1-23

WITH RECOMPILE argument, EXECUTE
 command, 2-166

WRAP argument

 ON, 2-279

wrapdepth, 2-224

WRITE, 2-332

writing to a page, 2-255

X

-XB, 1-23

-XC

 definition of, 1-23

 in the EXECUTE command, 2-162

-XCB, 1-23

-XI, 1-23

-XL, 1-23

-XLFF, 1-23

-XMB, 1-23

-XNAV, 1-24

-XP

 BEGIN-SELECT command, 2-48

 BEGIN-SQL command, 2-54

 definition of, 1-24

-XTB, 1-24

-XTOC, 1-24

Z

-ZIF{file}, 1-24

ZIP+4 Postnet, 2-283

-ZIV, 1-24

-ZMF{file}, 1-24