ORACLE
PEOPLESOFT ENTERPRISE

# Enterprise PeopleTools 8.49 PeopleBook: SQR Language Reference for PeopleSoft

**March 2007**

ORACLE

Enterprise PeopleTools 8.49 PeopleBook: SQR Language Reference for PeopleSoft
SKU PT849SQL-B 0307
Copyright © 1988-2007, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

**U.S. GOVERNMENT RIGHTS**

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

**Open Source Disclosure**

Oracle takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation. The following open source software may be used in Oracle's PeopleSoft products and the following disclaimers are provided.

*Apache Software Foundation*

This product includes software developed by the Apache Software Foundation (http://www.apache.org/). Copyright © 2000-2003. The Apache Software Foundation. All rights reserved. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

*OpenSSL*

Copyright © 1998-2005 The OpenSSL Project. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*Loki Library*

Copyright © 2001 by Andrei Alexandrescu. This code accompanies the book: Alexandrescu, Andrei. "Modern C++ Design: Generic Programming and Design Patterns Applied". Copyright © 2001 Addison-Wesley. Permission to use, copy, modify, distribute and sell this software for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

*Helma Project*

Copyright © 1999-2004 Helma Project. All rights reserved. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE HELMA PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Helma includes third party software released under different specific license terms. See the licenses directory in the Helma distribution for a list of these license.

*Sarissa*

Copyright © 2004 Manos Batsis.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

*ICU*

ICU License - ICU 1.8.1 and later COPYRIGHT AND PERMISSION NOTICE Copyright © 1995-2003 International Business Machines Corporation and others. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. THE SOFTWARE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

All trademarks and registered trademarks mentioned herein are the property of their respective owners.

*Sun's JAXB Implementation – JDSDK 1.5 relaxngDatatype.jar 1.0 License*

Copyright © 2001, Thai Open Source Software Center Ltd, Sun Microsystems. All rights reserved.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*W3C IPR SOFTWARE NOTICE*

Copyright © 2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

Note: The original version of the W3C Software Copyright Notice and License could be found at http://www.w3.org/Consortium/Legal/copyright-software-19980720.

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

# Contents

**General Preface**
**About This PeopleBook ..... xi**
PeopleSoft Enterprise Application Prerequisites ..... xi
Application Fundamentals ..... xi
Documentation Updates and Printed Documentation ..... xii
Obtaining Documentation Updates ..... xii
Downloading and Ordering Printed Documentation ..... xii
Additional Resources ..... xiii
Typographical Conventions and Visual Cues ..... xiv
Typographical Conventions ..... xiv
Visual Cues ..... xv
Country, Region, and Industry Identifiers ..... xv
Currency Codes ..... xvi
Comments and Suggestions ..... xvi
Common Elements Used in PeopleBooks ..... xvi

**Preface**
**SQR Language Reference for PeopleSoft Preface ..... xix**
SQR Language Reference for PeopleSoft ..... xix

**Chapter 1**
**Understanding SQR for PeopleSoft ..... 1**
SQR for PeopleSoft Tools ..... 1
The SQR Language ..... 1
Understanding the SQR Language ..... 2
SQR Program Structure ..... 2
SQR Syntax Conventions ..... 3
Rules for Entering SQR Commands ..... 4
SQR Command Line ..... 4
SQR Command-Line Arguments ..... 4
SQR Command-Line Flags ..... 5
SQR Data Elements ..... 15
Columns ..... 15

Copyright © 1988-2007, Oracle. All rights reserved.

Variables.....16
Literals.....20
Sample Reports.....20

**Chapter 2**
**SQR Command Reference.....21**
SQR Command Overview.....21
SQR Commands.....22
ADD.....22
ALTER-COLOR-MAP.....23
ALTER-LOCALE.....25
ALTER-PRINTER.....32
ALTER-REPORT.....33
ARRAY-ADD, ARRAY-DIVIDE, ARRAY-MULTIPLY, ARRAY-SUBTRACT.....34
ASK.....36
BEGIN-DOCUMENT.....37
BEGIN-EXECUTE.....38
BEGIN-FOOTING.....41
BEGIN-HEADING.....42
BEGIN-PROCEDURE.....43
BEGIN-PROGRAM.....46
BEGIN-SELECT.....46
BEGIN-SETUP.....49
BEGIN-SQL.....50
BREAK.....53
CALL, CALL SYSTEM.....53
CLEAR-ARRAY.....57
CLOSE.....58
COLUMNS.....59
COMMIT.....59
CONCAT.....60
CONNECT.....61
CREATE-ARRAY.....62
CREATE-COLOR-PALETTE.....64
#DEBUG.....65
DECLARE-CHART.....66
DECLARE-COLOR-MAP.....74
DECLARE-CONNECTION.....75
DECLARE-IMAGE.....77

 Copyright © 1988-2007, Oracle. All rights reserved.

DECLARE-LAYOUT……78
DECLARE-PRINTER……85
DECLARE-PROCEDURE……94
DECLARE-REPORT……96
DECLARE-TOC……97
DECLARE-VARIABLE……99
#DEFINE……101
DISPLAY……103
DIVIDE……106
DO……107
#ELSE……108
ELSE……108
ENCODE……108
END-DECLARE, END-DOCUMENT, END-EVALUATE, END-FOOTING, END-HEADING……109
#END-IF, #ENDIF……110
END-IF……110
END-PROCEDURE, END-PROGRAM, END-SELECT, END-SETUP, END-SQL, END-WHILE……110
EVALUATE……111
EXECUTE (Sybase and Microsoft SQL Server)……113
EXIT-SELECT……115
EXTRACT……115
FIND……117
GET……118
GET-COLOR……119
GOTO……121
GRAPHIC BOX, GRAPHIC HORZ-LINE, GRAPHIC VERT-LINE……121
#IF……124
IF……126
#IFDEF……127
#IFNDEF……127
#INCLUDE……128
INPUT……129
LAST-PAGE……131
LET……132
LOAD-LOOKUP……163
LOOKUP……167
LOWERCASE……167
MBTOSBS……168
MOVE……168
MULTIPLY……172

Copyright © 1988-2007, Oracle. All rights reserved.

NEW-PAGE.....173
NEW-REPORT.....173
NEXT-COLUMN.....174
NEXT-LISTING.....175
OPEN.....176
PAGE-NUMBER.....178
POSITION.....179
PRINT.....180
PRINT-BAR-CODE.....200
PRINT-CHART.....202
PRINT-DIRECT.....207
PRINT-IMAGE.....207
PUT.....209
READ.....210
ROLLBACK.....212
SBTOMBS.....213
SECURITY.....213
SET-COLOR.....214
SET-GENERATIONS.....216
SET-LEVELS.....216
SET-MEMBERS.....217
SHOW.....217
STOP.....222
STRING.....222
SUBTRACT.....223
TOC-ENTRY.....224
UNSTRING.....225
UPPERCASE.....226
USE.....226
USE-COLUMN.....227
USE-PRINTER-TYPE.....227
USE-PROCEDURE.....228
USE-REPORT.....230
WHILE.....230
WRITE.....232

**Chapter 3**
**Generating HTML Output.....235**
HTML General Purpose Procedures.....235

 Copyright © 1988-2007, Oracle. All rights reserved.

HTML Heading Procedures.....239
HTML Highlighting Procedures.....242
HTML Hypertext Link Procedures.....246
HTML List Procedures.....247
HTML Table Procedures.....251

**Chapter 4**
**Invoking SQR Execute.....255**
Running SQR Execute.....255
Using SQR Execute Flags.....255

**Chapter 5**
**Using SQR Print.....263**
Understanding SQR Print.....263
Generating Output from the Command Line.....263
Using SQR Print Command-Line Flags.....264
Generating Output in Microsoft Windows.....268

**Chapter 6**
**Avoiding Older SQR Commands.....269**
Understanding Older SQR Commands.....269
Using Older SQR Commands.....270
BEGIN-REPORT.....270
DATE-TIME.....270
DECLARE PRINTER.....272
DECLARE PROCEDURE.....277
DOLLAR-SYMBOL.....278
GRAPHIC FONT.....279
MONEY-SYMBOL.....280
NO-FORMFEED.....281
PAGE-SIZE.....282
PRINT … CODE.....283
PRINTER-DEINIT.....283
PRINTER-INIT.....284

Copyright © 1988-2007, Oracle. All rights reserved. 

**Chapter 7**
**Using the PSSQR.INI File and the PSSQR Command Line.....285**
Installing PSSQR.INI.....285
Default Settings Section.....286
Processing-Limits Section.....292
Environment Sections.....294
Locale Section.....296
Fonts Section.....298
Adding Fonts Entries.....298
Specifying Character Sets in Windows.....299
HTML-Images Section.....299
PDF Fonts Section.....300
PDF Fonts: Exclusion Ranges Section.....302
TrueType Font Section.....302
Enhanced-HTML Section.....303
Colors Section.....304
Using PSSQR.EXE Command-Line Options.....304

**Appendix A**
**SQR Messages.....309**
Unnumbered Messages.....309
Numbered Messages.....312

**Appendix B**
**Using SQR Sample Programs.....425**
Understanding SQR Samples.....425
SQR Sample Programs.....425

**Glossary of PeopleSoft Enterprise Terms.....431**

**Index .....457**


Copyright © 1988-2007, Oracle. All rights reserved.

# About This PeopleBook

PeopleSoft Enterprise PeopleBooks provide you with the information that you need to implement and use PeopleSoft Enterprise applications from Oracle.

This preface discusses:

- PeopleSoft Enterprise application prerequisites.
- Application fundamentals.
- Documentation updates and printed documentation.
- Additional resources.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common elements in PeopleBooks.

---

**Note.** PeopleBooks document only elements, such as fields and check boxes, that require additional explanation. If an element is not documented with the process or task in which it is used, then either it requires no additional explanation or it is documented with common elements for the section, chapter, PeopleBook, or product line. Elements that are common to all PeopleSoft Enterprise applications are defined in this preface.

---

## PeopleSoft Enterprise Application Prerequisites

To benefit fully from the information that is covered in these books, you should have a basic understanding of how to use PeopleSoft Enterprise applications.

You might also want to complete at least one introductory training course, if applicable.

You should be familiar with navigating the system and adding, updating, and deleting information by using PeopleSoft Enterprise menus, pages, or windows. You should also be comfortable using the World Wide Web and the Microsoft Windows or Windows NT graphical user interface.

These books do not review navigation and other basics. They present the information that you need to use the system and implement your PeopleSoft Enterprise applications most effectively.

## Application Fundamentals

Each application PeopleBook provides implementation and processing information for your PeopleSoft Enterprise applications.

For some applications, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals PeopleBook. Most product lines have a version of the application fundamentals PeopleBook. The preface of each PeopleBook identifies the application fundamentals PeopleBooks that are associated with that PeopleBook.

Copyright © 1988-2007, Oracle. All rights reserved.

The application fundamentals PeopleBook consists of important topics that apply to many or all PeopleSoft Enterprise applications. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of the appropriate application fundamentals PeopleBooks. They provide the starting points for fundamental implementation tasks.

# Documentation Updates and Printed Documentation

This section discusses how to:

- Obtain documentation updates.
- Download and order printed documentation.

## Obtaining Documentation Updates

You can find updates and additional documentation for this release, as well as previous releases, on Oracle's PeopleSoft Customer Connection website. Through the Documentation section of Oracle's PeopleSoft Customer Connection, you can download files to add to your PeopleBooks Library. You'll find a variety of useful and timely materials, including updates to the full line of PeopleSoft Enterprise documentation that is delivered on your PeopleBooks CD-ROM.

**Important!** Before you upgrade, you must check Oracle's PeopleSoft Customer Connection for updates to the upgrade instructions. Oracle continually posts updates as the upgrade process is refined.

### See Also

Oracle's PeopleSoft Customer Connection, http://www.oracle.com/support/support_peoplesoft.html

## Downloading and Ordering Printed Documentation

In addition to the complete line of documentation that is delivered on your PeopleBook CD-ROM, Oracle makes PeopleSoft Enterprise documentation available to you via Oracle's website. You can:

- Download PDF files.
- Order printed, bound volumes.

### Downloading PDF Files

You can download PDF versions of PeopleSoft Enterprise documentation online via the Oracle Technology Network. Oracle makes these PDF files available online for each major release shortly after the software is shipped.

See Oracle Technology Network, http://www.oracle.com/technology/documentation/psftent.html.

### Ordering Printed, Bound Volumes

You can order printed, bound volumes of selected documentation via the Oracle Store.

See Oracle Store, http://oraclestore.oracle.com/OA_HTML/ibeCCtpSctDspRte.jsp?section=14021


Copyright © 1988-2007, Oracle. All rights reserved.

# Additional Resources

The following resources are located on Oracle's PeopleSoft Customer Connection website:

| Resource | Navigation |
|---|---|
| Application maintenance information | Updates + Fixes |
| Business process diagrams | Support, Documentation, Business Process Maps |
| Interactive Services Repository | Support, Documentation, Interactive Services Repository |
| Hardware and software requirements | Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Hardware and Software Requirements |
| Installation guides | Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Installation Guides and Notes |
| Integration information | Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Pre-Built Integrations for PeopleSoft Enterprise and JD Edwards EnterpriseOne Applications |
| Minimum technical requirements (MTRs) | Implement, Optimize + Upgrade; Implementation Guide; Supported Platforms |
| Documentation updates | Support, Documentation, Documentation Updates |
| PeopleBooks support policy | Support, Support Policy |
| Prerelease notes | Support, Documentation, Documentation Updates, Category, Release Notes |
| Product release roadmap | Support, Roadmaps + Schedules |
| Release notes | Support, Documentation, Documentation Updates, Category, Release Notes |
| Release value proposition | Support, Documentation, Documentation Updates, Category, Release Value Proposition |
| Statement of direction | Support, Documentation, Documentation Updates, Category, Statement of Direction |
| Troubleshooting information | Support, Troubleshooting |
| Upgrade documentation | Support, Documentation, Upgrade Documentation and Scripts |

Copyright © 1988-2007, Oracle. All rights reserved.

# Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions.
- Visual cues.
- Country, region, and industry identifiers.
- Currency codes.

## Typographical Conventions

This table contains the typographical conventions that are used in PeopleBooks:

| Typographical Convention or Visual Cue | Description |
|---|---|
| **Bold** | Indicates PeopleCode function names, business function names, event names, system function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call. |
| *Italics* | Indicates field values, emphasis, and PeopleSoft Enterprise or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply.<br>We also use italics when we refer to words as words or letters as letters, as in the following: Enter the letter *O*. |
| KEY+KEY | Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press the W key. |
| `Monospace font` | Indicates a PeopleCode program or other code example. |
| “ ” (quotation marks) | Indicate chapter titles in cross-references and words that are used differently from their intended meanings. |
| . . . (ellipses) | Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax. |
| { } (curly braces) | Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe ( \| ). |


Copyright © 1988-2007, Oracle. All rights reserved.

| Typographical Convention or Visual Cue | Description |
|---|---|
| [ ] (square brackets) | Indicate optional items in PeopleCode syntax. |
| & (ampersand) | When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object.<br>Ampersands also precede all PeopleCode variables. |

# Visual Cues

PeopleBooks contain the following visual cues.

## Notes

Notes indicate information that you should pay particular attention to as you work with the PeopleSoft Enterprise system.

---

**Note.** Example of a note.

---

If the note is preceded by *Important!,* the note is crucial and includes information that concerns what you must do for the system to function properly.

---

**Important!** Example of an important note.

---

## Warnings

Warnings indicate crucial configuration considerations. Pay close attention to warning messages.

---

***Warning!*** Example of a warning.

---

## Cross-References

PeopleBooks provide cross-references either under the heading “See Also” or on a separate line preceded by the word *See.* Cross-references lead to other documentation that is pertinent to the immediately preceding documentation.

# Country, Region, and Industry Identifiers

Information that applies only to a specific country, region, or industry is preceded by a standard identifier in parentheses. This identifier typically appears at the beginning of a section heading, but it may also appear at the beginning of a note or other text.

Example of a country-specific heading: “(FRA) Hiring an Employee”

Example of a region-specific heading: “(Latin America) Setting Up Depreciation”

## Country Identifiers

Countries are identified with the International Organization for Standardization (ISO) country code.

Copyright © 1988-2007, Oracle. All rights reserved.

### Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in PeopleBooks:

- Asia Pacific
- Europe
- Latin America
- North America

### Industry Identifiers

Industries are identified by the industry name or by an abbreviation for that industry. The following industry identifiers may appear in PeopleBooks:

- USF (U.S. Federal)
- E&G (Education and Government)

## Currency Codes

Monetary amounts are identified by the ISO currency code.

# Comments and Suggestions

Your comments are important to us. We encourage you to tell us what you like, or what you would like to see changed about PeopleBooks and other Oracle reference and training materials. Please send your suggestions to your product line documentation manager at Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, U.S.A. Or email us at appsdoc@us.oracle.com.

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions.

# Common Elements Used in PeopleBooks

| | |
|---|---|
| **As of Date** | The last date for which a report or process includes data. |
| **Business Unit** | An ID that represents a high-level organization of business information. You can use a business unit to define regional or departmental units within a larger organization. |
| **Description** | Enter up to 30 characters of text. |
| **Effective Date** | The date on which a table row becomes effective; the date that an action begins. For example, to close out a ledger on June 30, the effective date for the ledger closing would be July 1. This date also determines when you can view and change the information. Pages or panels and batch processes that use the information use the current row. |


Copyright © 1988-2007, Oracle. All rights reserved.

| | |
|---|---|
| **Once, Always,** and **Don't Run** | Select Once to run the request the next time the batch process runs. After the batch process runs, the process frequency is automatically set to Don't Run.<br>Select Always to run the request every time the batch process runs.<br>Select Don't Run to ignore the request when the batch process runs. |
| **Process Monitor** | Click to access the Process List page, where you can view the status of submitted process requests. |
| **Report Manager** | Click to access the Report List page, where you can view report content, check the status of a report, and see content detail messages (which show you a description of the report and the distribution list). |
| **Request ID** | An ID that represents a set of selection criteria for a report or process. |
| **Run** | Click to access the Process Scheduler request page, where you can specify the location where a process or job runs and the process output format. |
| **SetID** | An ID that represents a set of control table information, or TableSets. TableSets enable you to share control table information and processing options among business units. The goal is to minimize redundant data and system maintenance tasks. When you assign a setID to a record group in a business unit, you indicate that all of the tables in the record group are shared between that business unit and any other business unit that also assigns that setID to that record group. For example, you can define a group of common job codes that are shared between several business units. Each business unit that shares the job codes is assigned the same setID for that record group. |
| **Short Description** | Enter up to 15 characters of text. |
| **User ID** | An ID that represents the person who generates a transaction. |

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

# SQR Language Reference for PeopleSoft Preface

This book is a language reference for Structured Query Reports (SQR) for PeopleSoft.

## SQR Language Reference for PeopleSoft

This reference describes and demonstrates the structure, command set, and syntax of the SQR language. It also provides a list of the sample SQR programs, an overview of the SQR initialization file, and a detailed list of SQR messages.

This reference is intended for SQR and SQL developers who must report on data from a wide range of enterprise data sources. Before using this reference, familiarize yourself with the data sources from which you are reporting and the connectivity between those data sources and your operating system.

### See Also

*Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Getting Started with SQR for PeopleSoft"

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

# CHAPTER 1

# Understanding SQR for PeopleSoft

This chapter discusses:

- SQR for PeopleSoft tools.
- The Structured Query Report (SQR) language.
- SQR command line.
- SQR data elements.
- Sample reports.

## SQR for PeopleSoft Tools

SQR for PeopleSoft is a powerful enterprise reporting system that provides direct access to multiple data sources. The SQR for PeopleSoft tools make it possible to create clear, professional reports from complex arrays of information systems.

This reference describes the following SQR for PeopleSoft tools:

- The SQR language, which is a flexible, fourth generation reporting language with a lexicon of over 110 commands.

  SQR's procedural design makes it easy to develop, implement, and distribute complex reports.
- SQR Execute, which enables you to run previously compiled SQR programs.
- SQR Print, which enables you to configure reports for most printers.
- SQR Samples, a library of SQR programs and output that provides a framework for creating configured reports.

## The SQR Language

This section provides an overview of the SQR language and discusses:

- SQR program structure.
- SQR syntax conventions.
- Rules for entering SQR commands.

Copyright © 1988-2007, Oracle. All rights reserved.

# Understanding the SQR Language

SQR is a specialized programming language for accessing, manipulating, and reporting enterprise data. With SQR, you build complex procedures that perform multiple calls to multiple data sources and implement nested, hierarchical, or object-oriented program logic.

SQR has several important benefits:

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

With SQR, you design reports by defining the page size, headers, footers, and layout . SQR enables you to generate a variety of output types , 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 configured output for laser printers and phototypesetters.

SQR's high-level programming capabilities enable you to add procedural logic and control to data source calls. You can use SQR to write other types of applications, such as those for database manipulation and maintenance, table loading and unloading, and interactive querying and displaying.

# SQR Program Structure

SQR for PeopleSoft processes source code from a standard text file and generates a report. The text file containing source code comprises a set of sections that you delimit with BEGIN-section and END-section commands. The following examples show the general structure of SQR:

- 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 runs the procedures in the report.

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

- The PROCEDURE section performs the tasks to produce the report.

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

 Copyright © 1988-2007, Oracle. All rights reserved.

## SQR Syntax Conventions

The following table describes the SQR syntax conventions:

| Syntax Convention | Description |
|---|---|
| { } | Braces enclose required items. |
| [ ] | Square brackets enclose optional items. |
| ... | Ellipses indicate that the preceding parameter can be repeated. |
| \| | A vertical bar separates alternatives inside brackets, braces, or parentheses. |
| ! | 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. |
| ’ | A single quote starts and ends a literal text constant or any argument with more than one word.<br>**Important!** If you are copying code directly from the sample programs, change the slanted quotes to regular quotes; otherwise, you will receive an error message. |
| , | A comma separates multiple arguments. |
| ( ) | Parentheses must enclose an argument or element. |
| UPPERCASE | SQR commands and arguments are uppercase within the text, but lowercase in the sample programs. (Note that these commands are case insensitive.) |
| *Variable* | Information and values that you must supply appear in *variable* style. |
| hyphen versus underscore | Many SQR commands, such as BEGIN-PROGRAM, contain a hyphen, whereas procedure and variable names contain an underscore. Procedure and variable names can contain either hyphens or underscores, but it is best to use underscores in procedure and variable names to distinguish them from SQR commands.<br>It also prevents confusion when you mix variable names and numbers in an expression, where hyphens could be mistaken for minus signs. |

Copyright © 1988-2007, Oracle. All rights reserved.

## Rules for Entering SQR Commands

Use these command rules as you develop SQR programs:

- You can enter SQR commands in either uppercase or lowercase; they are *not* case sensitive.

  Many SQR programmers use uppercase for SQR commands, but SQR ignores case as it compiles source code.
- You must separate command names and arguments by at least one space or tab character.
- You must begin each command on a new line; however, you can develop commands that extend beyond one line.
- You can break a line in any position between words *except* inside a quoted string.
- You can use a hyphen (-) at the end of a line to indicate that it continues on the next line; however, SQR ignores hyphens and carriage returns in commands.
- You must begin each comment line with an exclamation point (!).

**Note.** To display the exclamation point (!) or single quote (’) symbols in a report, type the symbol twice to indicate that it is text. For example, DON’T is typed DON’’T. This rule does not apply in the document paragraph of form-letter reports.

# SQR Command Line

SQR for PeopleSoft comprises SQR, SQR Execute, and SQR Print . Each has a command-line interface.

To begin running SQR, enter the following command:

```
SQR [program][connectivity][flags...][args...][@file...]
```

**Note.** The executable name for SQR is SQR (SQRW for Windows). The executable name for SQR Execute is SQRT (SQRWT for Windows). The executable for SQR Print is SQRP (SQRWP for Windows).

See Chapter 4, “Invoking SQR Execute,” page 255 and Chapter 5, “Using SQR Print,” page 263.

## SQR Command-Line Arguments

The following table describes the SQR command-line arguments:

| Argument | Description |
|---|---|
| *program* | Name of the text file containing the source code. The default file type or extension is .sqr. If you enter this value as a question mark (?) or omit it, SQR for PeopleSoft prompts you for the report program name. On UNIX/Linux-based systems, if your shell uses the question mark as a wildcard character, you must precede it with a backslash (\). |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Description |
|---|---|
| *connectivity* | Information that SQR for PeopleSoft needs to connect to the database. If you enter this value as a question mark (?) or omit it, SQR for PeopleSoft prompts you for the information.<br><br>IBM DB2: Subsystem name and Structured Query Language (SQL) authorization ID.<br><br>`Ssname/SQLid`<br><br>Microsoft Windows or IBM DB2: Name of the database, your username, and the password for the database.<br>`[Database] / [Username] / [Password]`<br><br>Informix: Name of the database.<br>`Database[/username/password]`<br><br>Open Database Connectivity (ODBC): Name that you give to the ODBC driver when you set up the driver, and your username and password for the database. This port has been certified against IBM DB2 and Microsoft SQL Server.<br>`Data_Source_Name/[Username]/[Password⇒`<br>`]`<br><br>Oracle: Your username and password for the database, and an optional connection string for the database (for example, @sales.2cme.com).<br>`[Username]/[Password[@Database]]`<br><br>Sybase: Your username and password for the database.<br>`Username/[Password]` |
| *flags* | See the following section, “SQR Command-Line Flags.” |
| *args...* | Arguments that SQR for PeopleSoft uses while the program is running. The ASK and INPUT commands use these arguments rather than prompting the user. You must enter arguments on the command line in the sequence that the program expects—first all ASK arguments, in order, followed by INPUT arguments. |
| *@file...* | File containing program arguments—one argument per line. Arguments that are listed in the file are processed one at a time: first all ASK arguments, in order, followed by INPUT arguments. You can specify the command-line arguments (*program, connectivity,* and *args*) in this file for non-Microsoft Windows platforms. |

## SQR Command-Line Flags

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

Copyright © 1988-2007, Oracle. All rights reserved. 

The following table describes the SQR command-line flags:

| Flag | Description |
|---|---|
| -A | Appends the output to an existing output file carrying the same name as the source of the output. If the file does not exist, a new one is created. This flag is useful when you want to run the same report more than once but want only a single output file. |
| -B*nn* | (Oracle, ODBC, and Sybase CT-Lib) Indicates the number of rows to buffer each time SQR for PeopleSoft 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. Inside a program, each BEGIN-SELECT command can also have its own -B flag for further optimization. |
| -BURST:{*xx*} | -BURST:T generates the table of contents file only.<br><br>-BURST:S generates the report output according to the symbolic table of contents entries that are set in the program with the *level* argument of the TOC-ENTRY command. In -BURST:S[ {l} ], {l} is the level on which to burst. The -BURST:S setting is equivalent to -BURST:S1.<br><br>See *Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Generating and Publishing HTML from an SQR Program," Bursting Reports.<br><br>**Note.** -BURST:P and -BURST:S require -PRINTER:EH or -PRINTER:HT.<br><br>The page range selection feature of - BURST:P requires -PRINTER:HT.<br><br>-BURST:T requires -PRINTER:HT. |
| -C | (Microsoft Windows) Specifies that the Cancel dialog box appears while the program runs so that you can easily stop the program. |
| -CB | (Microsoft Windows, Callable SQR) Forces the communication box to use. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -D*nn* | (Non-Microsoft Windows) Displays the report output on the terminal while it is being written to the output file. The value for *nn* is the maximum number of lines to display before pausing. If no number is entered after -D, the display scrolls continuously.<br><br>**Note.** The printer type must be LP; otherwise, the display is ignored. If the program is producing more than one report, the display is for the first report only. |
| -DBdatabase | (Sybase) Forces the SQR program to use the specified database, which overrides any USE command in the SQR program. |
| -DEBUG [*xxx*] | See Chapter 2, "SQR Command Reference," #DEBUG, page 65. |
| -DNT: {*xx*} | See Chapter 2, "SQR Command Reference," DECLARE-VARIABLE, page 99. |
| -E[*file*] | Directs error messages to the named file or to the default file program.err. If no errors occur, no file is created. |
| -EH_APPLETS:*dir* | Specifies the directory location of the enhanced HTML applets. If you include an applet, SQR for PeopleSoft must know where it resides. SQR for PeopleSoft usually checks for the applet in a default directory; the default directory for these applets is IMAGES.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_BQD | Generates a {report}.bqd file from the report data. Also associates a query format file (BQD) icon with {report}.bqd in the navigation bar.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_BQD:*file* | Associates the BQD icon with the specified file.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Flag | Description |
|---|---|
| -EH_BROWSER:*xx* | Generates HTML, determines the browser, and displays HTML.<br><br>When this flag is set to ALL, SQR for PeopleSoft generates frame.html, which contains JavaScript to determine the browser on the user's machine (that is, the person reading the report, not the person writing it).<br><br>When this flag is set to BASIC, SQR for PeopleSoft generates HTML that is suitable for all browsers.<br><br>When this flag is set to IE, SQR for PeopleSoft generates HTML that is designed for Microsoft Internet Explorer.<br><br>When this flag is set to NETSCAPE, SQR for PeopleSoft generates HTML that is designed for Netscape.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_CSV | Generates a {report}.csv file from the report data.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_CSV:*file* | Associates the CSV icon with the specified file.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_CSVONLY | Creates a .csv file, but does not create an HTML file.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_FULLHTML:*xx* | Specifies the level of the generated enhanced HTML code. This can be 30, 32, or 40.<br><br>**Note.** For upward compatibility, a value of TRUE is equivalent to 40 and FALSE is 30. |
| -EH_Icons:dir | Specifies the directory in which the HTML should find the referenced icons.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_IMAGES:dir | Specifies the directory path for the .gif files that are used by the navigation bar.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -EH_KEEP | Copies (does not move) the files when used in conjunction with -EH_ZIP.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_LANGUAGE:*xx* | Sets the language that is used for the HTML navigation bar. You can specify English, French, German, Portuguese, Spanish, Japanese, Simplified Chinese, or Korean.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_PDF | Associates a PDF icon with {report}.pdf in the navigation bar.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_Scale:{*nn*} | Sets the scaling factor to a value from 50 to 200.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_XML:*file* | Associates the XML icon with the specified file.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag. |
| -EH_ZIP[:file] | Moves the generated files to the specified file or {report}.zip if {file} is not specified.<br><br>**Note.** This flag is applicable only when you specify the -PRINTER:EH or the -PRINTER:EP flag. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -F[*file* \| *directory*] | Overrides the default output file name, program.lis. The default action places the program.lis file in the same directory as the program.sqr 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 is actually printed on the page. If no data is printed, no output file is created.<br><br>Specify these file names and directories for different operating systems:<br><br>UNIX/Linux<br><br>Directory character is /<br><br>`-F$HOME/reports/`<br><br>IBM MVS<br><br>Directory character is (<br>`-FDSN:SQR.REPORTS(` |
| -GPRINT=YES \| NO | (IBM MVS) -GPRINT=YES includes control characters in the first column of each record of the SQR report output file. -GPRINT=NO omits the control characters. |
| -Idir_list | Specifies the list of directories that SQR for PeopleSoft searches when processing the #INCLUDE directive when the include file does not exist in the current directory and no path is specified for the file. The directory names must be separated by commas (,) or semicolons (;).<br><br>For UNIX/Linux-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, sorted by operating system. |
| -ID | (Non-Microsoft Windows) Displays the copyright banner on the console. |
| -KEEP | See *Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, “Printing with SQR”. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -LL{s \| d}{c \| i} | Specifies the following LOAD-LOOKUP values:<br>• s: SQR<br>• d: DB<br>• c: Case sensitive<br>• i: Case insensitive<br>See Chapter 2, "SQR Command Reference," LOAD-LOOKUP, page 163. |
| -M*file* | Defines a startup file containing sizes to assign to internal parameters—extremely small, large, or complex reports. M*files* are text files that have individual switches in the INI files that are unique to a run. |
| -NOLIS | Prevents the creation of .lis files, creating .spfs instead. |
| -O[*file*] | Directs log messages to the specified file or to program.log if no file is specified. By default, the sqr.log file is created in the current working directory. |
| -olim | Displays the SQR resources that are used by the SQR report.<br>• Limit is the programmatic limit, the maximum amount of memory that can be allocated to the variable.<br>• Defined is the value defined in the pssqr.ini file; this is a user-defined value.<br>• Actual is the actual amount of memory/space used by the variable at runtime. |
| -P | (IBM MVS) Suppresses printer control characters from column 1. |
| -PB | (Informix) Causes column data to preserve trailing blanks. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -PRINTER:*xx* | Uses printer type *xx* when creating output files. The *xx* represents:<br>EH: Enhanced HTML<br>`-PRINTER:EH`<br>EP: Enhanced HTML or PDF<br>`-PRINTER:EP`<br>HP: HP LaserJet<br>`-PRINTER:HP`<br>HT: HTML 2.0<br>`-PRINTER:HT`<br>LP: Line printer<br>`-PRINTER:LP`<br>PD: PDF<br>`-PRINTER:PD`<br>PS: PostScript<br>`-PRINTER:PS`<br>WP: Microsoft Windows<br>`-PRINTER:WP`<br>LP, HP, and PS produce files .lis files. EH and HT produce .htm files. HT produces version 2.0 HTML files with the report content inside <PRE></PRE> tags. EH produces reports in which content is fully formatted with version 3.0 or 3.2 HTML tags. On Microsoft Windows systems, WP sends the output to the default Microsoft Windows printer. To specify a Microsoft Windows printer that is not the default, enter -PRINTER:WP:{printer name}, where {printer name} is the name that is assigned to your printer. For example, to send output to a Microsoft Windows printer named New Printer, use -PRINTER:WP:NewPrinter. If the printer name has spaces, enclose the entire argument in quotes. To create an .spf file also, use -KEEP. |
| -RS | Saves the program in a runtime file. The program is scanned, compiled, and checked for correct syntax. Queries are validated and compiled. The executable version is saved in a file named program.sqt.<br>**Note.** SQR for PeopleSoft does not prompt ASK variables after compilation. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -RT | Uses the runtime file that is saved with the -RS flag. This skips all syntax and query checking, and processing begins immediately.<br><br>**Note.** SQR for PeopleSoft does not prompt ASK variables after compilation. |
| -S | Requests that the status of all cursors be displayed at the end of the report run. Status includes the text of each SQL statement, the number of times each was compiled and run and the total number of rows that were selected. The output appears directly on the screen. This information can be used for debugging SQL statements, enhancing performance, and tuning. |
| -T*nn* | Specifies that you want to test your report for *nn* pages. To save time during testing, SQR for PeopleSoft ignores all ORDER BY clauses in SELECT statements. If the program is producing more than one report, SQR for PeopleSoft stops after producing the specified number of pages defined for the first report. |
| -T{B} | (Microsoft Windows, IBM DB2, Sybase CT-Lib, and ODBC) Trims trailing blanks from database character columns.<br><br>(IBM MVS and DB2) Prevents SQR for PeopleSoft from removing trailing blanks from database character columns.<br><br>**Note.** The -TB flag only has an effect if SQR is connecting to either a DB2, Sybase CT-Lib, or ODBC (MSS) database. Confusingly, the behavior of the -TB command-line flag varies depending on theplatform. If you are using one of the above mentioned databases and are running SQR on z/OS, the -TB flag will behave as follows:<br><br>If you do not use the -TB flag, trailing blanks are trimmed.<br><br>If you do use the -TB flag, trailing blanks are not trimmed.<br><br>If you are running SQR on any other platform, the behavior of -TB is the opposite:<br><br>If you do not use the -TB flag, trailing blanks are not trimmed.<br><br>If you do use the -TB flag, trailing blanks are trimmed. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -T{Z} | (IBM MVS and DB2) Prevents SQR for PeopleSoft from removing trailing zeros from the decimal portion of numeric columns. |
| -Vserver | (Sybase) Uses the named server. |
| -XB | (Non-Microsoft Windows) Suppresses the SQR banner and the *SQR.... End of Run* message. |
| -XC | (Callable SQR) Suppresses the database commit when the report has finished running. |
| -XCB | (Microsoft Windows) Does not use the communication box. Requests for input are made in Microsoft Windows dialog boxes. |
| -XI | Prevents user interaction during a program run. If an ASK or INPUT command requires user input, an error is produced and the program ends. |
| -XL | Prevents SQR for PeopleSoft from signing in to the database. Programs that you run in this mode cannot contain SQL statements. -XL enables you to run SQR for PeopleSoft without accessing the database. You still must supply at least an empty slash on the command line as a placeholder for the connectivity information.<br>For example:<br>`sqr myprog / -xl` |
| -XLFF | Prevents a trailing form feed. |
| -XMB | (Microsoft Windows) Disables the error message display so that you can run a program without interruption from error message boxes. Error messages are sent to an .err file. See the -E flag. |
| -XNAV | Prevents SQR for PeopleSoft from creating the navigation bar in .htm files that are generated with -PRINTER:HT. This occurs when only a single .htm file is produced. Multiple .htm files that are generated from a single report always contain the navigation bar. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -XP | (Sybase) Prevents SQR for PeopleSoft from creating temporary stored procedures.<br>See Chapter 2, "SQR Command Reference," BEGIN-SELECT, page 46. |
| -XTB | Preserves the trailing blanks in a .lis file. |
| -XTOC | Suppresses the table of contents for the report. This flag is ignored when you specify either -PRINTER:EH or -PRINTER:HT. |
| -ZIF{*file*} | Sets the full path and name of the SQR initialization file, sqr.ini. |
| -ZIV | Invokes the SPF Viewer after generating the program.spf file. This flag implicitly invokes the -KEEP flag to create program.spf. In the case of multiple output files, only the first report file is passed to the viewer. |
| -ZMF{*file*} | Specifies the full path and name of the SQR error message file, sqrerr.dat. |

# SQR Data Elements

Each SQR data element begins with a special character that denotes the type of data element.

This section discusses:

- Columns
- Variables
- Literals

## Columns

Columns are fields that are defined in the database.

The ampersand character (&) begins a database column or expression name. It can be any type of column, such as character , number , or date . Columns that are defined in a query are declared automatically, except for dynamic columns and database or aggregate functions.

Copyright © 1988-2007, Oracle. All rights reserved. 

# Variables

Variables are storage places for text or numbers that you define and manipulate. Variables begin with special characters:

- $ begins a text or date variable.
- # begins a numeric variable.
- % begins a list variable.
- @ begins a variable name for a marker location.

  Marker locations identify positions to begin printing in 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.
- Do not use an underscore (_) or colon (:) as the first character of a two-variable name.

  See *Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Using Break Logic," Using Hyphens and Underscores.
- Variable names are *not* case sensitive.

  That is, you can use a name in uppercase on one line and lowercase on the next; both refer to the same variable.
- SQR for PeopleSoft initializes variables to null (text and date) or zero (numeric).
- A command 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 Chapter 2, "SQR Command Reference," DECLARE-VARIABLE, page 99.
- Variables and columns are known globally throughout a report, except when 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 Chapter 2, "SQR Command Reference," BEGIN-PROCEDURE, page 43.

## SQR Reserved Variables

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

The following table describes the SQR reserved variables:

| Variable | Description |
|---|---|
| *#current-column* | Current column on the page. |
| *$current-date* | Current date and time on the local machine when SQR for PeopleSoft starts running the process. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Variable | Description |
|---|---|
| *#current-line* | Current line on the page. This value is the physical line on the page, not the line in the report body.<br><br>Line numbers are referenced in PRINT and other SQR commands that are used for positioning the data on the page. Optional page headers and footers, which are 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. |
| *#end-file* | See Chapter 2, "SQR Command Reference," READ, page 210. |
| *#page-count* | Current page number. |
| *#return-status* | Value to return to the operating system when SQR for PeopleSoft exits. This can be set in the report. *#return-status* is initialized to the "success" return value for the operating system. |
| *#sql-count* | Count of the rows that are affected by a SELECT paragraph (INSERT, UPDATE, or DELETE). This is equivalent to ROWCOUNT in Oracle and Sybase. |
| *$sql-error* | Text message from the database explaining an error. This variable is rewritten when a new error is encountered. |
| *#sql-status* | The value of *#sql-status* is set whenever a BEGIN-SELECT command is run. Normally this variable is checked from inside an ON-ERROR procedure, so its value describes the error condition (whereas the *$sql-error* variable contains the error message). The actual meaning of *#sql-status* is database dependent. Therefore, consult the proper database manual to fully interpret its meaning. |
| *$sqr-encoding-console*<br><br>*{sqr-encoding-console}* | Name of encoding for character data that is written to the log file or console. |
| *$sqr-encoding-database*<br><br>*{sqr-encoding-database}* | Character data that is retrieved from and inserted into the database. |
| *$sqr-encoding-file-input*<br><br>*{sqr-encoding-file-input}* | Name of encoding for character data that is read from files that are used with the OPEN command. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Variable | Description |
|---|---|
| *$sqr-encoding-file-output*<br>*{sqr-encoding-file-output}* | Name of encoding for character data that is written to files that are used with the OPEN command. |
| *$sqr-encoding-report-output*<br>*{sqr-encoding-report-output}* | Report that is generated by SQR for PeopleSoft (for example, a .lis file or a PostScript file). |
| *$sqr-encoding-source*<br>*{sqr-encoding-source}* | Name of encoding for SQR source files and include files. |
| *$sqr-database*<br>*{sqr-database}* | Database type for which SQR was compiled. Values are ORACLE, DB2, ODBC, SYBASE, and INFORMIX. |
| *$sqr-dbcs*<br>*{sqr-dbcs}* | Specifies whether SQR for PeopleSoft recognizes double-byte character strings. Values are YES and NO. |
| *$sqr-encoding*<br>*{sqr-encoding}* | Name of the default encoding as defined by the ENCODING environment variable when SQR for PeopleSoft is invoked. |
| *$sqr-hostname*<br>*{sqr-hostname}* | Name of the computer on which SQR for PeopleSoft is currently running. |
| *$sqr-locale* | Name of the current locale that is being used. A plus symbol (+) at the end of the name indicates that an argument that is used in the locale has changed. |
| *#sqr-max-lines* | 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. |
| *#sqr-max-columns* | 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. |
| *#sqr-pid* | 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. |
| *$sqr-platform,*<br>*{sqr-platform}* | The hardware or operating system type for which SQR was compiled. Values are MVS, Windows, and UNIX/Linux. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Variable | Description |
|---|---|
| *$sqr-program* | Name of the SQR process file . |
| *$sqr-ver* | Text string that is shown with the -ID flag. SQR version. |
| *$username* | Database username that is specified on the command line. |
| *$sqr-report* | Name of the report output file . $sqr-report reflects the actual name of the file to use, as specified by the -F flag or NEW-REPORT command. |

## List Variables

List variables contain an ordered collection of SQR variables and are nonrecursive—that is, you cannot nest lists inside lists.

Indicate list variables with the percent symbol (%). Create list variables with the LET command and a list of variables. For example:

```
let %list1 = list (num_var1|str_var1, num_var2|str_var2,...)
```

Working with list variables includes the following tasks:

- Defining a list variable:

  You can use a list variable to hold multiple rows of information. Before you assign a list variable, define it by 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'[,...])
  ```

- Assigning a list variable:

  Assign a list variable by using the following syntax:

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

- Accessing a list variable:

  Access a list variable by using the following syntax:

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

## List Variable Arguments

The value between the brackets indicates either the number of rows in the list for the definition case or the row in the list to modify or assign.

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 that are given in the definition.

Copyright © 1988-2007, Oracle. All rights reserved. 

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.

## Literals

Literals are text or numeric constants:

- A single quote begins and ends a text literal . For example:

```
'Hello'
```

- Numerals 0−9 begin 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.

# Sample Reports

To see a quick overview of how an SQR report looks, view the sample reports that are stored in the SQR for PeopleSoft directory <PS_HOME>\bin\sqr\<database_platform>\SAMPLE (or SAMPLEW, for Windows). You can modify these reports to meet your needs.

 Copyright © 1988-2007, Oracle. All rights reserved.

# CHAPTER 2

# SQR Command Reference

This chapter describes and demonstrates each command in the SQR lexicon.

## SQR Command Overview

The commands in this section follow the conventions listed in the section "SQR Syntax Conventions" in the previous chapter, and use the abbreviations described in the following table:

***Warning!*** If you are copying code directly from the examples in the PDF file, make sure you change the slanted quotes to regular quotes or you will receive an error message.

| Abbreviation | Description | Example |
|---|---|---|
| txt_col | Text column retrievable from a database. | &address |
| num_col | Numeric column retrievable from a database. | &price |
| date_col | Date or datetime column retrievable from a database. | &date1 |
| txt_var | String variable defined in a program. | $your_name |
| num_var | Numeric variable defined in a program. | #total_cost |
| date_var | A variable explicitly defined as a date variable. | $date1 |
| any_lit | A literal of any type. | 'abc' 12 |
| any_var | A variable of any type. | $string #number $date |
| any_col | A column of any type. | &string &number &date |

Copyright © 1988-2007, Oracle. All rights reserved.

| Abbreviation | Description | Example |
|---|---|---|
| txt_lit | Text literal defined in a program. | ’Company Confidential’ |
| num_lit | Numeric literal defined in a program. | 12345.67 |
| int_lit | Integer literal defined in a program. | 12345 |
| 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) |

# SQR Commands

The following sections discuss the SQR commands in alphabetical order.

## ADD

### Syntax

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

### Description

Adds one number to another.

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.

When dealing with money-related values, 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

### Parameters

| Parameter | Description |
|---|---|
| *src_num_lit* \| *_var* \| *_col* | Source number literal, variable, or column. |
| *dst_num_var* | A numeric destination variable that contains the result after execution. |
| **ROUND** | Rounds the result to the specified number of digits to the right of the decimal point. For float variables, this value can be from 0 to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate. |

### Example

To add 10 to #counter:

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

### See Also

The LET command for information about complex arithmetic expressions.

## ALTER-COLOR-MAP

### Syntax

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

### Description

Dynamically alters a defined color.

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

### Parameters

| Parameter | Description |
|---|---|
| NAME | Defines the name of the color you want to alter. For example, *light blue*. |
| VALUE | Defines the RGB value of the color you want to alter. For example, (193, 233, 230). |
| {color_name_lit \| _var \| _col} | The color_name is composed of alphanumeric characters (A-Z, 0-9), the underscore (_) character, and the hyphen (-) character. It must start with an alpha (A-Z) character and is not case-sensitive. 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. |

Copyright © 1988-2007, Oracle. All rights reserved. 

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)

## Example

The following example illustrates the ALTER-COLOR-MAP command:

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

begin-program
   alter-color-map name = 'light_blue' value = (193, 233, 230)

   print 'Yellow Submarine' ()
      foreground = ('yellow')
      background = ('light_blue')

   get-color print-text-foreground = ($print-foreground)
   set-color print-text-foreground = ('purple')
   print 'Barney' (+1,1)
   set-color print-text-foreground = ($print-foreground)
end-program
```

## See Also

DECLARE-COLOR-MAP, SET-COLOR, GET-COLOR

 Copyright © 1988-2007, Oracle. All rights reserved.

# ALTER-LOCALE

## 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})]
```

## Description

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.

The SYSTEM locale represents the behavior of older versions of SQR prior to Version 4.0. When you install SQR for PeopleSoft Version 4.0 or later, the default locale is set to SYSTEM. This provides upwards compatibility for older SQR programs. This table describes the SYSTEM locale settings:

Copyright © 1988-2007, Oracle. All rights reserved. 

| 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 justify 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 format. See the Date Time section for more details. |
| INPUT-DATE-EDIT-MASK | SQR uses a default date edit mask with the INPUT command. See the Sample Date Edit Masks table for a listing of the date edit mask. |
| 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 |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Keyword | Value |
|---|---|
| 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') |

## Parameters

The following table list and describes the parameters:

**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 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.

| Parameter | Description |
|---|---|
| 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 not case-sensitive and is limited to the following character set: A-Z, 0-9, underscore, and 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 information about Edit Masks, see *PRINT.* |
| MONEY-SIGN | Specifies the characters that replace the $ or other currency symbol used in edit masks. |
| MONEY-SIGN-LOCATION | Specifies where to place the MONEY-SIGN characters. Valid values are LEFT and RIGHT. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Parameter | Description |
|---|---|
| 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 characters to use with the 'na' option. |
| EDIT-OPTION-AM | Specifies the characters to replace 'AM'. |
| EDIT-OPTION-PM | Specifies the characters to replace 'PM'. |
| EDIT-OPTION-BC | Specifies the characters to replace 'BC'. |
| EDIT-OPTION-AD | Specifies the characters to replace 'AD'. |
| DAY-OF-WEEK-CASE | Specifies how the case for the DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT entries is 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 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 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 for PeopleSoft 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 for PeopleSoft considers the first month of the year to be January. You must specify all 12 abbreviations. |

## Example

The following example illustrates the ALTER-LOCALE command:

```
!
! The following program segments will illustrate the various
! ALTER-LOCALE features.
!
begin-setup
 declare-variable
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
  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 2004 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, 2004
Without DATE option   $Date  = 01-JAN-04
```

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

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

Produces the following output:

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

```
 !
 ! 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
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
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$
```

```
!
! 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
```

```
!
! 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/01/2004 11:22:33', $edit)
show $date edit :$edit
```

Produces the following output:

```
Jan-01-2004 11.22.33
```

```
!
! 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.'
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
let $value = ''
let $edit = 'Mon DD YYYY HH:MI'
let $date1 = strtodate('Jan 01 2004 11:59', $edit)
let $date2 = strtodate('Feb 28 2004 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 2004 11:59:00 A.M.
Feb 28 2004 12:01:00 p.m.
```

```
!
! 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 2004', 'Mon DD YYYY')
let $date2 = strtodate('Feb 28 2004', 'Mon DD YYYY')
let $date3 = strtodate('Mar 15 2004', '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 2004 is Thursday
February 28 2004 is Saturday
March 15 2004 is Monday
```

Copyright © 1988-2007, Oracle. All rights reserved.

```
German:

01 Januar 2004 ist Donnerstag
28 Februar 2004 ist Samstag
15 März 2004 ist Montag

Spanish:

01 enero 2004 es jueves
28 febrero 2004 es sábado
15 marzo 2004 es lunes
```

### See Also

DISPLAY, LET, MOVE, PRINT, SHOW

Chapter 7, "Using the PSSQR.INI File and the PSSQR Command Line," page 285

## ALTER-PRINTER

### 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}]
```

### Description

Alters printer parameters at runtime.

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 is creating multiple reports and the printer is shared by another report, the attributes are changed for that report as well.

 Copyright © 1988-2007, Oracle. All rights reserved.

### Parameters

| Parameter | Description |
|---|---|
| POINT-SIZE | Specifies the new font point size. |
| FONT-TYPE | Specifies the new font type. Values are PROPORTIONAL or FIXED. |
| SYMBOL-SET | Specifies the new symbol set identifier. |
| FONT | Specifies the new font as a number. (For example, font=3 for Courier and font=4 for Helvetica.) |
| PITCH | Specifies the new pitch in characters per inch.<br>See the DECLARE-PRINTER arguments table for information about these arguments. |

### Example

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 (-KEEP) and later printed on an HP LaserJet, the symbol set 12U can be used.

### See Also

The DECLARE-PRINTER command and the -KEEP command-line flag.

## ALTER-REPORT

### Syntax

```
ALTER-REPORT
 [HEADING={heading_name_txt_lit|_var|_col}]
 [HEADING-SIZE={heading_size_int_lit|_var|_col}]
 [FOOTING={footing_name_txt_lit|_var|_col}]
 [FOOTING-SIZE={heading_size_int_lit|_var|_col}]
```

### Description

Alters some of the report-specific functionality.

This command gives you the ability to dynamically change the heading or footing sections that are active for the current report. You can also change how much space the heading or footing sections occupy.

If the HEADING or FOOTING value is set to NONE, the section is disabled for the current report.

If the HEADING or FOOTING value is set to DEFAULT, the section reverts to the setting that were in effect when the report was initiated.

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

Copyright © 1988-2007, Oracle. All rights reserved. 

If the ALTER-REPORT command was not invoked from within a BEGIN-HEADING or BEGIN-FOOTING section and the page has not been written to, the assignment takes effect immediately; otherwise it takes effect for the next page.

### Parameters

| Parameter | Description |
|---|---|
| HEADING | Specifies the name of the BEGIN-HEADING section to use. |
| HEADING-SIZE | Specifies the amount of space the BEGIN-HEADING section occupies on the page. |
| FOOTING | Specifies the name of the BEGIN-FOOTING section to use. |
| FOOTING-SIZE | Specifies the amount of space the BEGIN-FOOTING section occupies on the page |

### Example

The following example illustrates the ALTER-REPORT command:

```
begin-footing 2 name=confidental
  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-report
 alter-report
   footing = 'Proprietary'
   footing-size = 6                     ! Increase depth

.
.
.
end-report
```

### See Also

The BEGIN-FOOTING and BEGIN-HEADING commands in this section.

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

### 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*

 Copyright © 1988-2007, Oracle. All rights reserved.

```
[(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)]]...
```

## Description

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

The following information applies to the array arithmetic commands:

- The array must first be created using the CREATE-ARRAY command.
- The four array arithmetic commands perform on one or more source numbers, placing the result into the corresponding field in the array.
- 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 type number, decimal, float, or integer. They cannot be of type date, char, or text.
- If division by zero is attempted, a warning message appears, the result field is unchanged, and SQR continues executing.

## Parameters

| Parameter | Description |
|---|---|
| src_num_lit\|_var\|_col | Source values 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 fields contain the results after the operation. All variables must be numeric in type. |

## Example

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:

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

The following example subtracts #lost, #count, and 1 from the fields loses, tot, and sequence of the #j2'th element of the stats array:

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

The following example multiplies occurrences 0 through 2 of the field p in the #i'th element of the percentages array by 2:

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

Copyright © 1988-2007, Oracle. All rights reserved. 

The following example divides the #i2'th occurrence of the salesman field of the #j'th element of the commissions array by 100:

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

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

## ASK

### Syntax

ASK *substitution_variable* [*prompt*]

 Copyright © 1988-2007, Oracle. All rights reserved.

### Description

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.

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}'. When the substitution variable is text or date, surround the brackets by single quotes. Substitutions are made when 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.

### Parameters

| Parameter | Description |
|---|---|
| Substitution_variable | The variable to be used as the substitution variable. |
| Prompt | An optional, literal text string to be displayed as a prompt if the value for the substitution variable is not entered on the command line or in an argument file. |

### Example

In the following example, state takes the value entered by the user in response to the prompt *Enter state for this report:*

```
begin-setup
  ask state 'Enter state for this report'
end-setup
  ...
begin-select
name, city, state, zip
from customers where state = '{state}'
end-select
```

### See Also

The INPUT command for information about input at runtime.
*Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Compiling Programs and Using SQR Execute"

## BEGIN-DOCUMENT

### Syntax

BEGIN-DOCUMENT *position*

END-DOCUMENT

### Description

Begins a document paragraph . A document paragraph enables you to write free-form text to create form letters, invoices, and so on.

Copyright © 1988-2007, Oracle. All rights reserved. 

You can reference database columns, SQR variables, and document markers within a document. Their locations in the document determine where they print on the page. You should not use tabs inside a document paragraph. To indent text or fields, use the spacebar. Note also that if the variables being printed inside a document paragraph are variable in length, you might needto manipulate the variable outside the document paragraph.

---

**Note.** A document must be executed before you can reference its document markers. Because 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.

---

### Parameters

| Parameter | Description |
|---|---|
| position | The location on the page where the document begins. The position can be fixed or relative to the current position. See the POSITION command for a description and examples of the *position* parameter. |

### Example

The following example illustrates the BEGIN-DOCUMENT command

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

### See Also

END-DOCUMENT

*Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, “Creating Form Letters”

## BEGIN-EXECUTE

### Syntax

```
BEGIN-EXECUTE
[CONNECTION=uq_txt_lit]
[ON-ERROR=procedure[(arg1[,argi]...)]]
[RSV=num_var]
[STATUS=list_var|num_var|txt_var]
[SCHEMA=txt_lit|txt_var]
[PROCEDURE=txt_lit|txt_var
[PARAMETERS=(arg1[IN|INOUT|NULL[,argi[IN|INOUT]]...]])]
|COMMAND=txt_lit|txt_var
|GETDATA=txt_lit|txt_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,-n,n-m,m-|all})
|FROM PARAMETER=txt_lit|txt_var
END-SELECT]
```

 Copyright © 1988-2007, Oracle. All rights reserved.

END-EXECUTE

## Description

Begins a new construct. In a BEGIN-EXECUTE paragraph, the syntax of BEGIN-SELECT varies as shown in the following syntax:

## Parameters

| Parameter | Description |
|---|---|
| CONNECTION | Identifies a name previously specified using the DECLARE-CONNECTION construct. If you do not specify a name, SQR Server 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 the 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. |
| SCHEMA | Identifies the location in the datasource of the object being queried. |
| PROCEDURE | The name of the datasource-stored procedure to be executed. The name may include spaces. If the datasource is SAP R/3, this procedure is a BAPI. |
| PARAMETER_LIST | 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, the default value is IN. 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. |
| BEFORE/AFTER | Names an SQR procedure to be executed before or after the row set. The procedure is not performed unless at least one row is returned from the specified row sets. |
| FROM ROWSET | Special case addition to the BEGIN-SELECT syntax. Available for use with all datasource types, including SAP R/3 and JDBC. Names the row sets 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. |
| FROM PARAMETER | Special case addition to the BEGIN-SELECT syntax. Available only for SAP R/3 datasources. Use only with the PROCEDURE keyword. This argument names an output parameter containing one or more rows from which the column variables are to be retrieved. |
| PROPERTIES = (*txt_var* \| *strlit* = *txt_var* \| *strlit* \| *num_var* \| *num_lit* \| *any_col, …* ) | Specifies a set of keyword-value pairs 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. |

Copyright © 1988-2007, Oracle. All rights reserved. 

**Note.** This is a similar concept to the PARAMETERS = statement in DECLARE-CONNECTION, with the minor difference being that the properties specified here alter the flow of returned information, as opposed to just 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 )

## Example

The following example illustrates the BEGIN-EXECUTE command

```
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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
End-Execute
```

### See Also

EXECUTE

## BEGIN-FOOTING

### Syntax

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

### Description

Begins the FOOTING section.

The FOOTING section defines and controls the information to be printed at the bottom of each page.

You must define the *report_name* 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.

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.

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 section for each table of contents. A BEGIN-FOOTING section with FOR-TOCS=(ALL) can be followed by other BEGIN-FOOTING sections for a specific table of contents, which override the ALL setting.

The BEGIN-FOOTING section can be shared between reports and tables of contents.

You can print outside the footing area of the report—that is, in the body area—from the footing, but you cannot print in the footing area from the body.

### Parameters

| Parameter | Description |
|---|---|
| footing_lines_int_lit | The number of lines to be reserved at the bottom of each page. |
| 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. |
| FOR-TOCS | Specifies the table of contents to which this heading applies. |
| NAME | Specifies the name to be associated with this footing section. Use this parameter with the ALTER-REPORT command. The name cannot be NONE or DEFAULT. |

Copyright © 1988-2007, Oracle. All rights reserved. 

### Example

The following example illustrates the BEGIN-FOOTING command

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

### See Also

The ALTER-REPORT command for information about dynamic headings and footings.
The DECLARE-LAYOUT command for information about page layout.
The DECLARE-REPORT command for information about programs with multiple reports.
The DECLARE-TOC command for information about the table of contents.
The END-FOOTING command.

## BEGIN-HEADING

### Syntax

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

### Description

Begins a HEADING section.

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

You must define the *report_name* 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.

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.

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

 Copyright © 1988-2007, Oracle. All rights reserved.

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

The BEGIN-HEADING section can be shared between reports and a table of contents.

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

### Parameters

| Parameter | Description |
|---|---|
| heading_lines_int_lit | The number of lines to be reserved at the top of each page. |
| 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. |
| FOR-TOCS | Specifies the table of contents to which this heading applies. |
| 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. Use this parameter with the ALTER-REPORT command. The name cannot be NONE or DEFAULT. |

### Example

The following example illustrates the BEGIN-HEADING command

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

## BEGIN-PROCEDURE

### Syntax

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

Copyright © 1988-2007, Oracle. All rights reserved. 

## Description

Begins a procedure. A procedure is one of the most powerful parts of the SQR language. It enables modularized functions and provides standard execution control.

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

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

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

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

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, or the database-independent format SYYYYMMDD[HH24[MI[SS[NNNNNN]]]].

  See the Default Database Formats table in the PRINT command description.
- 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 a date variable or column is passed 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 has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
  - For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

    If this has not been set, SQR uses the format listed in the Default Database Formats table.
  - For TIME columns, the format specified by the SQR_DB_TIME_ONLY_FORMAT setting is used.

    If this has not been set, SQR uses the format listed in the TIME Column Formats table.

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*.

---

 Copyright © 1988-2007, Oracle. All rights reserved.

## Parameters

| Parameter | Description |
|---|---|
| procedure_name | Specifies a unique name for this procedure. Procedure names are not case-sensitive. |
| LOCAL | Specifies that this is a local procedure. |
| arg1 [, argi]... | Specifies the arguments to be passed to or returned from the procedure. Arguments can be string variables ($arg), numeric variables (#arg), or date variables ($arg). 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. |

## Example

The following example shows a 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
```

Copyright © 1988-2007, Oracle. All rights reserved. 

### See Also

DO, END-PROCEDURE
The Default Database Formats table in the
Chapter 2, "SQR Command Reference," PRINT, page 180
command description.

## BEGIN-PROGRAM

### Syntax

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

### Description

Begins the PROGRAM section of an SQR program.

After processing the any commands in the SETUP section, 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.

### Example

The following example illustrates the BEGIN-PROGRAM command

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

### See Also

BEGIN-REPORT, BEGIN-SETUP, END-PROGRAM

## BEGIN-SELECT

### 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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

## Description

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.

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

## Parameters

The table describes the parameters:

---

**Note.** The arguments can span multiple lines; however, the first character position cannot be used unless the continuation character terminated the previous line. If the first character position is used with arguments spanning multiple lines, the argument will be misconstrued as a Select column.

---

Copyright © 1988-2007, Oracle. All rights reserved. 

| Parameter | Description |
|---|---|
| DISTINCT | Specifies that duplicate rows be eliminated from your query. |
| -C*nn* | (Oracle) Sets the context area size (buffer size for query) to larger or smaller than the default. This option is rarely needed. |
| -B*nn* | (Oracle, ODBC, Sybase CT-Lib) 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. Use this flag if you change the variables frequently during execution and you do not want SQR to automatically create stored procedure. You can also use -XP if the username/password to your program does not have permission to create stored procedures.<br>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.<br>-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. |
| -DB*connectionstring* | (ODBC) Specifies the ODBC connection string for this SELECT paragraph only. A connection string has the following syntax:<br>DSN=data_source_name[;keyword=value[;keyword=value [;...]]]<br>This option enables you to combine data from multiple databases in one program. For example, a connection string for an Oracle database named ora8 might look like this:<br>`'DSN=ora8;UID=scott;PWD=tiger'`<br>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 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.<br>Optionally, you can specify arguments to be passed to the ON-ERROR procedure. Arguments can be any variable, column, or literal. |

## Example

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 *Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Using Column Variables".

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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

*Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Selecting Data from the Database" and *Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Using Dynamic SQL and Error Checking".
END-SELECT, EXIT-SELECT

## BEGIN-SETUP

### Syntax

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

### Description

Begins a SETUP section. This optional section is processed prior to the BEGIN-PROGRAM, BEGIN-HEADING, and BEGIN-FOOTING sections.

The SETUP section should be the first section in the program.

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:

```
ASK
BEGIN-SQL
```

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

```
CREATE-ARRAY
```

(The CREATE-ARRAY command can also be used in the other sections of an SQR program.)

```
DECLARE-CHART
DECLARE-IMAGE
DECLARE-LAYOUT
DECLARE-PRINTER
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
DECLARE-PROCEDURE
DECLARE-REPORT
DECLARE-VARIABLE
```

(The DECLARE-VARIABLE command can also be used in LOCAL procedures.)

```
DECLARE-TOC
LOAD-LOOKUP
```

(The LOAD-LOOKUP command can also be used in the other sections of an SQR program.)

```
USE
```

(Sybase and Microsoft SQL Server only.)

### Example

The following example illustrates the BEGIN-SETUP command

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

### See Also

ASK, BEGIN-SQL, CREATE-ARRAY, LOAD-LOOKUP, USE

## BEGIN-SQL

### Syntax

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

### Description

Begins an SQL paragraph . This paragraph can reside in a BEGIN-PROCEDURE, BEGIN-SETUP, or BEGIN-PROGRAM section.

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 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). Columns and variables can be referenced in the SQL statements.

 Copyright © 1988-2007, Oracle. All rights reserved.

## Stored Procedures

For Oracle, stored procedures are implemented using PL/SQL in the BEGIN-SQL paragraph. For Sybase and Microsoft SQL Server, SQR supports stored procedures with the EXECUTE command.

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, ensure that these are done in the proper order or unpredictable results may occur.

## Oracle PL/SQL

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

For Oracle PL/SQL:

```
begin-sql
declare
   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
```

Copyright © 1988-2007, Oracle. All rights reserved. 

## Parameters

| Parameter | Description |
|---|---|
| -C*nn* | (Oracle) Sets the context area size (buffer size for query) to larger or smaller than the default. This option is rarely needed. |
| -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. Use this flage if you change the variables frequently during execution and you do not want SQR to automatically create stored procedures. You may also use it if your program does not have permission to create stored procedures.<br>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.<br>-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. |
| -DB*connectionstring* | (ODBC) Specifies the ODBC connection string for this SQL paragraph only. A connection string has the following syntax:<br>`DSN=data_source_name[;keyword=value[;keyword=value[;...]]]`<br>This option enables 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:<br>`'DSN=ora8;UID=scott;PWD=tiger'`<br>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. |
| 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.<br>Optionally, you can specify arguments to be passed to the ON-ERROR procedure. Arguments can be any variable, column, or literal.<br>If ON-ERROR is used in the SETUP section, it is a condition flag supporting the following conditions:<br>STOP: Do not run the program.<br>WARN: Run the program but with a warning message.<br>SKIP: Ignore any errors and run the program. |

## Example

The following example illustrates the BEGIN-SQL command

```
begin-sql
update orders set invoice_num = #next_invoice_num
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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
```

### See Also

*Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Using Dynamic SQL and Error Checking" and *Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Using Additional SQL Statements with SQR".
END-SQL, BEGIN-PROCEDURE, EXECUTE
The -S command-line flag.

## BREAK

### Syntax

BREAK

### Description

Causes an exit from within an EVALUATE or WHILE command. Execution then continues to the command immediately following the END-WHILE or END-EVALUATE.

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

### See Also

WHILE, EVALUATE

## CALL, CALL SYSTEM

### Syntax

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

To issue operating system commands from within an SQR program, use the following syntax:

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

### Description

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

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

---

***Warning!*** PeopleSoft recommends 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:

Copyright © 1988-2007, Oracle. All rights reserved. 

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

The CALL SYSTEM is a special subroutine that is provided as part of SQR to enable the program to issue operating system commands. Its arguments, *command, status,* and WAIT|NOWAIT are described below.

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:

| 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, your arguments are processed 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 and passing along the arguments you need.

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

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

 Copyright © 1988-2007, Oracle. All rights reserved.

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.

## Parameters

| Parameter | Description |
|---|---|
| subroutine | Specifies the name of your subroutine. |
| src_txt_lit\|_var\|_col | Specifies a text column, variable, or literal which is to be input to the called subroutine. |
| src_num_lit\|_var\|_col | Specifies a numeric column, variable (decimal, float, or integer), or literal that is to be input to the called subroutine. |
| dst_txt_var\|_num_var | Specifies a text or numeric variable (decimal, float, or integer) into which the called subroutine is to place the return result. |
| 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 follows:<br>UNIX/Linux: Zero (0) indicates success. Any other value is the system error code.<br>PC/ Microsoft Windows: A value less than 32 indicates an error. |
| WAIT\|NOWAIT | (Microsoft Windows only): 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.<br>For Microsoft Windows, the default is NOWAIT. On UNIX\Linux operating systems the behavior is always WAIT. |

## Example

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
```

Copyright © 1988-2007, Oracle. All rights reserved. 

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 (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 makes the first letter of a string uppercase, 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 that 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 lowercase; however, in your SQR program it can be referenced either uppercase or lowercase.

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

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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 the programming manual for your particular machine 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 lowercase
   call initcap using $name $capname  ! Now set the first character to uppercase
   input $last 'Enter the last name ' ! Get the last name from the user
   lowercase $last                    ! Set the last name to all lowercase
   call initcap using $last $caplast  ! Now set the first character to uppercase
   .
   .
   .
```

### See Also

The LET command for information about 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

### Syntax

CLEAR-ARRAY NAME=*array_name*

Copyright © 1988-2007, Oracle. All rights reserved. 

### Description

Resets each field of an array to its initial value.

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.

### Parameters

| Parameter | Description |
|---|---|
| NAME | Specifies the name of the array to be cleared. |

### Example

The following example illustrates the CLEAR-ARRAY command

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

### See Also

CREATE-ARRAY

## CLOSE

### Syntax

CLOSE {*filenum_lit*|*_var_col*}

### Description

Closes a file, specified by its file number.

Closes a flat file that has been previously opened using the OPEN command.

### Parameters

| Parameter | Description |
|---|---|
| filenum_lit\|_var_col | Specifies the number assigned to the file in the OPEN command. |

### Example

The following example illustrates the CLOSE command

```
close 5
close #j
```

### See Also

OPEN, READ, WRITE

 Copyright © 1988-2007, Oracle. All rights reserved.

# COLUMNS

## Syntax

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

## Description

Defines logical columns to be used for PRINT commands.

COLUMNS defines the leftmost position of one or more columns within the current page layout. It sets the first column as current.

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

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.

The USE-COLUMN 0 turns off columns. See USE-COLUMN.

## Parameters

| Parameter | Description |
|---|---|
| int_lit\|_var\|_col | Specifies the left margin position of each column. |

## See Also

NEXT-COLUMN, NEXT-LISTING, NEW-PAGE, USE-COLUMN, USE-REPORT.

# COMMIT

## Syntax

```
COMMIT
```

## Description

Causes a database commit.

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.

When the application completes, a commit is performed automatically unless a ROLLBACK was done or, for callable SQR, the -XC flag was set.

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.

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

---

**Note.** The COMMIT command can be used with SQR servers for Oracle, DB2, Informix, and ODBC. For Sybase and Microsoft SQL Server, use BEGIN TRANSACTION and COMMIT TRANSACTION within SQL paragraphs as in the following code segment.

---

Copyright © 1988-2007, Oracle. All rights reserved. 

### Example

The following example illustrates the COMMIT command

```
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
```

***Warning!*** 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 or Microsoft SQL Server).

## CONCAT

### Syntax

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

### Description

Concatenates a variable, column, or literal with a string variable.

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

CONCAT can optionally edit the source field before appending it. Edit masks can be changed dynamically by placing them in a string variable and referencing the variable name preceded by a colon (:).

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:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.

 Copyright © 1988-2007, Oracle. All rights reserved.

If this has not been set, the first database-dependent format as listed in the Default Database Formats table is used.

- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT.

  If this has not been set, the format listed in the Default Database Formats table is used.

- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT.

  If this has not been set, the format as listed in the Time Column Formats table is used.

### Parameters

| Parameter | Description |
|---|---|
| src_any_lit\|_var\|_col | Specifies the source field to be concatenated with the dst_txt_var field. |
| dst_txt_var | Contains the result after execution |
| edit_mask | Specifies an optional edit mask. |

### Example

The following example illustrates the CONCAT command

```
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 about the Default Database Formats table, the Time Column Formats table, and edit masks.
The LET command for string functions.
STRING, UNSTRING

## CONNECT

### Syntax

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

### Description

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

The new username and password can be stored in a string variable, column, or literal.

---

***Warning!*** The username/password is not encrypted, so beware of security issues.

After each CONNECT, the reserved variable *$username* 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.

---

Copyright © 1988-2007, Oracle. All rights reserved. 

### Parameters

| Parameter | Description |
|---|---|
| txt_lit\|_var\|_col | Specifies a username 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. |

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

### Example

The following example illustrates the CONNECT command

```
connect  $new-user  on-error=bad-logon($new_user)
connect  'sqr/test'
```

## CREATE-ARRAY

### Syntax

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

### Description

Creates an array of fields to store and process data.

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 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.

The following code is a representation of an array *emps* with three fields where the CREATE-ARRAY command defines the array:

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

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   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.

## Parameters

| Parameter | Description |
|---|---|
| NAME | Names the array. The name is referenced in other array commands. |
| SIZE | Defines the number of elements in the array. |
| FIELD | Defines each field or column in the array. Each field must be defined as type:<br>DECIMAL[(p)]: Decimal numbers with an optional precision (p).<br>FLOAT: Double precision floating point numbers.<br>INTEGER: Whole numbers.<br>NUMBER: Uses the DEFAULT-NUMERIC type. See the DECLARE-VARIABLE command.<br>CHAR (or TEXT): Character string.<br>DATE: Same as date variable.<br>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 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'. |
| OCCURS | Fields can optionally have a number of occurrences (occurs ), that is, they can be repeated any number of times. |

## Example

The following example illustrates the CREATE-ARRAY command

```
create-array  name=custs size=100
   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.
DECLARE-VARIABLE, ARRAY-ADD, ARRAY-DIVIDE, ARRAY-MULTIPLY, ARRAY-SUBTRACT, GET, PUT, LET, CLEAR-ARRAY.
The LOAD-LOOKUP command for an alternative way to store database tables in memory.

Copyright © 1988-2007, Oracle. All rights reserved. 

# CREATE-COLOR-PALETTE

## Syntax

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

## Description

Create a color palette.

This command enables you 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.

## Parameters

| Parameter | Description |
|---|---|
| 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. |

## Example

The following example illustrates the CREATE-COLOR-PALETTE command

```
begin-report
 create-color-palette
   name = 'funky'
   color_1 = ('blue')
   color_2 = ('red')
   color_3 = ('orange')

  Print-Chart Groovy
   Color-Palette = 'Funky'
 end-report
```

## See Also

DECLARE-CHART, PRINT-CHART

 Copyright © 1988-2007, Oracle. All rights reserved.

# #DEBUG

## Syntax

`#DEBUG[x...]SQR_Command`

## Description

Causes the current command to be processed during a debugging session.

A -DEBUG[*xx*] flag in the SQR command line enables conditional compilation of SQR commands. When this flag is used, any command (including other compiler directives) preceded by the word #DEBUG is processed; other commands are ignored.

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

The -DEBUG flag can contain a suffix up to 10 letters or digits. These characters are used to match any letters or digits appended to the #DEBUG preprocess command inside the program. #DEBUG commands with one or more matching suffix characters are processed; other commands are ignored. Commands without any suffix always match.

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

## Parameters

| Parameter | Description |
|---|---|
| x | Represents any letter or digit. |

## Example

The following SQR command line contains the -DEBUG flag with no suffixes:

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

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

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

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

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

In the following code example, the first three #DEBUG commands are compiled, but the fourth, beginning *#debuge,* is not because 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 code example shows the use of an #IF with a #DEBUG:

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

Copyright © 1988-2007, Oracle. All rights reserved. 

### See Also

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

## DECLARE-CHART

### Syntax

```
DECLARE-CHART chart_name
[DATA-LABELS=data_labels_lit]
[COLOR-PALETTE=color_palette_lit]
[ITEM-COLOR=(chart_item_keyword_lit, color_value_lit |(r,g,b)]
[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]
[POINT-MARKERS=point_markers_lit]
[TYPE=chart_type_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_segement_quantity_display_lit]
[PIE-SEGMENT-PERCENT-DISPLAY=
pie_segement_percent_display_lit]
[PIE-SEGMENT-EXPLODE=pie_segement_explode_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-SCALE=x_axis_scale_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-MAJOR-INCREMENT=
{x_axis_major_increment_lit|_num_lit}]
[X-AXIS-MINOR-INCREMENT=
x_axis_minor_increment_num_lit]
[X-AXIS-TICK-MARK-PLACEMENT=
x_axis_tick_mark_placement_lit]
[X-AXIS-GRID=x_axis_grid_lit]
[Y-AXIS-LABEL=y_axis_label_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-SCALE=y_axis_scale_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-MAJOR-INCREMENT=
{y_axis_major_increment_lit|_num_lit}]
[Y-AXIS-MINOR-INCREMENT=
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
y_axis_minor_increment_num_lit]
[Y-AXIS-TICK-MARK-PLACEMENT=
y_axis_tick_mark_placement_lit]
[Y-AXIS-GRID=y_axis_grid_lit]
END-DECLARE
```

**Note.** If CHART-SIZE is not defined, it must be defined in PRINT-CHART.

## Description

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

The DECLARE-CHART command can define the attributes of a chart to be printed as part of a report.

This command may only appear in the SETUP section.

A chart defined using DECLARE-CHART is printed by referencing its name in the PRINT-CHART command. Some or all of the chart attributes may be overridden at runtime 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.

**Note.** All DECLARE-CHART attributes can be overridden as part of the PRINT-CHART command. Columns are not supported within the DECLARE-CHART command or the PRINT-CHART command. Attributes that are specified more than once produce a warning, and the first instance is regarded as the actual value. Attributes can be used in any order, with the exception of *chart-name,* which must follow the DECLARE-CHART keyword.

Also, the FILL specification in the DECLARE-PRINTER command can influence the appearance of the chart. The following table lists the final appearance of the chart with a combination of values for PRINTER.COLOR and CHART.FILL options.

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

## Specifying Chart Data Series Colors

Color palettes are used in the new graphics to set the colors of each data point in a data series. You specify the color palette to be used in a business chart by creating an SQR COLOR-PALETTE using the CREATE-COLOR-PALETTE command. The following code demonstrates how to create the color palette:

```
Create-Color-Palette
   Name = 'Test-Palette'
   Color_1 = (100,133,238)
   Color_2 = (0, 0, 255)
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
Color_3 = (0,255,0)
Color_4 = (0,0,255)
Color_5 = (0,0,0)
```

Users can specify any number of palettes, with up to 64 colors defined in each palette. If there are more data points in the data sets than defined colors in the palette, the palette resets and continues to set the data point colors from Color_1 to Color_n.

After a color palette has been defined, it can be used within the DECLARE-CHART and PRINT-CHART commands to specify the color palette to be used. The following code example demonstrates the use of a color palette:

```
Print-Chart test_Chart
   COLOR-PALETTE = 'Test-Palette'
```

## Specifying Chart Item Colors

Users can specify the foreground and background colors of the individual areas within a business chart using ITEM-COLOR = (*rgb-value*) within the DECLARE-CHART and PRINT-CHART commands. The following is a list of chart item keywords that are valid for 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 Subtitle.
- HeaderForeground—Text color of the Title and Subtitle.
- FooterBackground—Area within the text box specified for the X Axis label.
- FooterForeground—Text color of the X Axis label.
- 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.
- PlotAreaForeground—Text and line colors of the plot area.

## Parameters

| Parameter | Description |
|---|---|
| chart_name | A unique name to be used for referencing a chart. |
| CHART-SIZE | The size of the chart frame in standard SQR coordinate units. |

The following DECLARE-CHART Command Arguments table describes other arguments for the DECLARE-CHART command.

---

**Note.** PeopleSoft does not currently support setting NewGraphics to Yes. You should not use the DATA-LABELS, COLOR-PALETTE, and ITEM-COLOR attributes listed below as they are only valid when NewGraphics=Yes.

---

 Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Values | Description |
|---|---|---|
| DATA-LABELS | Yes\|No | If NewGraphics is set to Yes, use this argument to specify whether SQR prints the numeric value above the individual data points. Specify NO to suppress the numeric values. |
| COLOR-PALETTE | palette_name | If NewGraphics is set to Yes, use this argument to specify the name of the color palette to be used to color the individual data points in each chart (example: bar, slice, point). A valid SQR color-palette must be defined to use COLOR-PALETTE. |
| ITEM-COLOR | ChartBackground\|ChartForeground<br>HeaderBackground\|HeaderForeground<br>FooterBackground\|FooterForeground<br>LegendBackground\|LegendForeground<br>ChartAreaBackground\|ChartAreaForeground<br>PlotAreaBackground | If NewGraphics is set to Yes, use this argument to specify the color of 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. |
| TITLE | NONE \| text | Specifies a title for the chart. That text is placed at the top of the chart. |
| SUB-TITLE | NONE \| text | Specifies a subtitle for the chart. That text is placed below the title regardless of whether TITLE is specified. |
| FILL | GRAYSCALE \| COLOR \| CROSSHATCH \| NONE | Specifies the type of filling that is applied to the shapes (bars, pie-segments, and so on.) in the chart. GRAYSCALE varies the density of black dots. COLOR sends color instructions to the current printer. If the current printer does not support color, then color can 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. |
| 3D-EFFECTS | YES \| NO | Specifies whether the chart depth appears with 3-D effects. If set to NO, the chart is displayed in the default 2D mode. |
| BORDER | YES \| NO | If set to YES, a border is drawn around the chart. If set to NO, no border is displayed around the chart. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Values | Description |
|---|---|---|
| POINT-MARKERS | YES \| NO | Specifies whether point markers appear on line charts. If set to YES, point markers appear on line charts. If set to NO, point markers do not appear. |
| TYPE | LINE \| PIE \| BAR, STACKED-BAR \| 100%-BAR \| OVERLAPPED-BAR \| FLOATING-BAR \| HISTOGRAM \| AREA \| STACKED-AREA \| 100%-AREA \| XY-SCATTER-PLOT \| HIGH-LOW-CLOSE | Specifies the type of chart.<br>See *Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Using Business Charts". |
| LEGEND | YES \| NO | Specifies whether to display a legend. |
| LEGEND-TITLE | NONE \| text | Specifies the title for the legend. If set to NONE, no title is displayed in the legend box. |
| 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 (RIGHT, LEFT, or CENTER) is the horizontal. |
| LEGEND-PRESENTATION | INSIDE \| OUTSIDE | Specifies where the legend appears on the chart. If set to INSIDE, the legend is presented inside the area defined by the two axes. If set to OUTSIDE, the legend is presented within the chart border, but outside of the region represented by the two axes. |
| PIE-SEGMENT-QUANTITY-DISPLAY | YES \| NO | Specifies whether quantity is presented for each pie segment. If set to YES, the quantity is presented. |
| PIE-SEGMENT-PERCENT-DISPLAY | YES \| NO | Specifies whether the percent-of-total number is presented for each pie segment. If set to YES, the percent-of-total figures is presented. |


Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Values | Description |
|---|---|---|
| PIE-SEGMENT-EXPLODE | NONE \| MAX \|MIN \| USE-3RD-DATA-COLUMN | Controls which pie segments are exploded (selected) within the pie chart. MAX selects the largest segment. MIN selects the smallest segment. USE-3RD-DATA-COLUMN uses the third field in the DATA-ARRAY 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. |
| X-AXIS-LABEL or Y-AXIS-LABEL | NONE \| text | Specifies a line of text to be presented below (or alongside) the tick-mark labels on the axis. |
| X-AXIS-MIN-VALUE | AUTOSCALE \| number | Specifies the minimum value on the axis. If there are data values that are less than X-AXIS- MIN-VALUE, they are not presented. AUTOSCALE directs SQR to calculate an appropriate minimum value. |
| Y-AXIS-MIN-VALUE | AUTOSCALE \| number | Specifies the minimum value on the axis. If there are data values that are less than Y-AXIS- MIN-VALUE, then they are not presented. AUTOSCALE directs SQR to calculate an appropriate minimum value. |
| X-AXIS-MAX-VALUE | AUTOSCALE \| number | Specifies the maximum value on the axis. If there are data values that are greater than X-AXIS-MAX-VALUE, they are not presented. AUTOSCALE directs SQR to calculate an appropriate maximum value. |
| Y-AXIS-MAX-VALUE | AUTOSCALE \| number | Specifies the maximum value on the axis. If there are data values that are greater than Y-AXIS-MAX-VALUE, they are not presented. AUTOSCALE directs SQR to calculate an appropriate maximum value. |
| X-AXIS-SCALE or Y-AXIS-SCALE | LOG \| LINEAR | Specifies the scale for the axis. LOG specifies a logarithmic scale for the axis. Otherwise, the scale is linear. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Argument | Values | Description |
|---|---|---|
| X-AXIS-MAJOR-TICK- MARKS | YES \| NO | Specifies whether to display tick-marks for major increments on the X-axis. If set to YES, tick-marks appear on the axis between X-AXIS-MIN-VALUE and X-AXIS-MAX-VALUE, according to the X-AXIS-SCALE setting spaced by X-AXIS-MAJOR-INCREMENT. |
| Y-AXIS-MAJOR-TICK- MARKS | YES \| NO | Specifies whether to display tick-marks for major increments on the Y-axis. If set to YES, tick-marks appear on 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. |
| X-AXIS-MINOR-TICK- MARKS | YES \| NO | Specifies whether to display tick-marks for minor increments on the X-axis. If set to YES, tick-marks appear on the axis between X-AXIS-MIN-VALUE and X-AXIS-MAX-VALUE, according to the X-AXIS-SCALE setting spaced by X-AXIS-MINOR-INCREMENT. |
| Y-AXIS-MINOR-TICK- MARKS | YES \| NO | Specifies whether to display tick-marks for minor increments on the Y-axis. If set to YES, tick-marks appear on 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. |
| X-AXIS-MAJOR-INCREMENT or Y-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. |
| X-AXIS-MINOR-INCREMENT or Y-AXIS-MINOR-INCREMENT | number | Specifies, for SQR, the increment used for spacing the minor tick-marks on the axis. These arguments must be set for the X-AXIS-MINOR-TICK-MARKS and the Y-AXIS-MINOR-TICK-MARKS to appear. |


Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Values | Description |
|---|---|---|
| X-AXIS-TICK-MARK-PLACEMENT or Y-AXIS-TICK-MARK-PLACEMENT | INSIDE \| OUTSIDE \| BOTH | Specifies where to place the tick-marks on the axis. 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. |
| X-AXIS-GRID or Y-AXIS-GRID | YES \| NO | Specifies whether a grid line is drawn for each major tick-mark on the axis. If set toYES, a dashed grid line is drawn for each major tick-mark. If set to NO, no grid line is drawn on the axis. |

## Example

This code example declares a basic sales chart using DECLARE-CHART. 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
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
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
```

### See Also

The PRINT-CHART command.

## DECLARE-COLOR-MAP

### Syntax

In the SETUP section:

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

 Copyright © 1988-2007, Oracle. All rights reserved.

### Description

Defines colors in an SQR report.

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.

### Parameters

| Parameter | Description |
|---|---|
| color_name | A *color_name* is composed of alphanumeric characters (A-Z, 0-9), the underscore (_) character, and the hyphen (-) character. It must start with an alphabetical (A-Z) character and is not case-sensitive. 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 cannot 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.<br>The default colors implicitly installed with SQR include:<br>black=(0,0,0)<br>white=(255,255,255)<br>gray=(128,128,128)<br>silver=(192,192,192)<br>red=(255,0,0)<br>green=(0,255,0)<br>blue=(0,0,255)<br>yellow=(255,255,0)<br>purple=(128,0,128)<br>olive=(128,128,0)<br>navy=(0,0,128)<br>aqua=(0,255,255)<br>lime=(0,128,0)<br>maroon=(128,0,0)<br>teal=(0,128,128)<br>fuchsia=(255,0,255) |

### Example

The following example illustrates the DECLARE-COLOR-MAP command

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

### See Also

The ALTER-COLOR-MAP, SET-COLOR, and GET-COLOR commands in this section.

## DECLARE-CONNECTION

### Syntax

In the SETUP section:

Copyright © 1988-2007, Oracle. All rights reserved. 

```
DECLARE-CONNECTION connection_name_txt_lit
DSN={uq_txt_lit}
[USER={uq_txt_lit}]
[PASSWORD={uq_txt_lit}]
[PARAMETERS=keyword_str=attr_str;[,keyword_str=attr_str ;...]]
END-DECLARE
```

In the body of the report:

```
DECLARE-CONNECTION 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;...]]
END-DECLARE
```

## Description

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

## Parameters

| Parameter | Description |
|---|---|
| *connection_name* | A user-defined name for describing a datasource connection. |
| 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 semicolons (;) 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. The behavior, while allowing dynamic logins as-needed, causes problems 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. |

## Example

The following example illustrates the DECLARE-CONNECTION command

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

 Copyright © 1988-2007, Oracle. All rights reserved.

# DECLARE-IMAGE

## Syntax

```
DECLARE-IMAGE image_name
[TYPE=image_type_lit]
[IMAGE-SIZE=(width_num_lit,height_num_lit)]
[SOURCE=file_name_lit]
END-DECLARE
```

**Note.** If TYPE, IMAGE-SIZE, and SOURCE are not defined in DECLARE-IMAGE, they must be defined in PRINT-IMAGE.

## Description

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

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.

**Note.** If the image file is unrecognizable, or has incomplete header information, a box (either shaded, for HP printers, or having a diagonal line through it in the case of postscript) appears where the image is expected.

## Parameters

| Parameter | Description |
|---|---|
| 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, or BMP-FILE (for Windows). |
| 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.<br>**Note.** If the file is not in the SQRDIR directory, the full path or no path should be given. You cannot specify a relative path, because you must know where to execute the file from. |

## Example

The following example illustrates the DECLARE-IMAGE command

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

## See Also

PRINT-IMAGE.

Copyright © 1988-2007, Oracle. All rights reserved. 

# DECLARE-LAYOUT

## 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
```

## Description

Defines the attributes for the layout of an output file .

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. If no DECLARE-LAYOUT is defined or if a DECLARE-REPORT does not reference a defined layout, a layout named DEFAULT is created with the default attribute values shown in the DECLARE-LAYOUT Command Arguments table. For an example of how DECLARE-LAYOUT relates to DECLARE-REPORT, see the DECLARE-REPORT examples in this document.

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 boldfaced 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 DECLARE-LAYOUT command 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, 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, the ORIENTATION value overrides the PAPER-SIZE value.

 Copyright © 1988-2007, Oracle. All rights reserved.

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

## Parameters

| Parameter | Description |
|---|---|
| layout_name | A unique layout name to be used for referencing the layout and its attributes. |
| uom (unit of measure) | An optional suffix which denotes the unit of measure to apply to the preceding value. |
| paper_name | An option of PAPER-SIZE. This name is associated with predefined dimensions. |

This table lists valid unit of measure 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 |

This table lists valid paper names for the *paper_name* parameter.

| 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 |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Name | Width | Depth | Orientation |
|---|---|---|---|
| DL | 4.33 in | 8.66 in | Landscape |
| C5 | 6.378 in | 9.016 in | Landscape |

This table describes the arguments of the DECLARE-LAYOUT command:

| 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 the table of valid paper names..) Note that when ORIENTATION= LANDSCAPE the default values are 11 in, 8.5 in. |
| 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. |
| 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Choice or Default UOM | Default Value | Description |
|---|---|---|---|
| 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Choice or Default UOM | Default Value | Description |
|---|---|---|---|
| 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). |

## Example

This example illustrates the ability to specify these parameters using a different measurement system, such as 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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
                                  !   orientation=portrait

                                  !   columns=67

                                  !   lines=64
```

This 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
```

This 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
```

This example changes the orientation to landscape; the default page dimensions of (8.5in and 11in) are swapped, the columns and rows are recalculated, and all other values remain the same:

```
declare-layout default         ! Results in:

orientation=landscape          !   paper-size=(11in, 8.5in)
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
end-declare                    !   top-margin=0.5in

                               !   left-margin=0.5in

                               !   right-margin=0.5in

                               !   bottom-margin=0.5in

                               !   orientation=landscape

                               !   columns=100

                               !   lines=45
```

This example changes the orientation to landscape; the default page dimensions of (8.5in and 11in) are swapped, and 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
```

This example illustrates how to specify the page dimensions using one of the predefined names (note that the orientation has also changed because 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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
!   columns=85

!   lines=18
```

### See Also

DECLARE-REPORT

## DECLARE-PRINTER

### 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
```

### Description

Overrides the printer defaults for the specified printer type.

Each printer has a set of defaults as listed in the DECLARE-PRINTER Command Arguments table. 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 runtime 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. The names are:

- DEFAULT-LP for line printer.
- DEFAULT-HP for HP LaserJet.
- DEFAULT-HT for HTML.
- DEFAULT-PS for PostScript

This table lists the arguments, provides the possible choices or measure, lists the default values, and describes the arguments.

Copyright © 1988-2007, Oracle. All rights reserved. 

| Argument | Choice or Measure | Default | Description |
|---|---|---|---|
| FOR-REPORTS | | ALL | The name of the reports that use this printer definition. The default is ALL, for 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. |
| TYPE | LINEPRINTER, POSTSCRIPT, HPLASERJET, HTML, LP, PS, HP, HT | LP | The output type specific to each printer. LINEPRINTER (LP) files can be viewed by a text editor. POSTSCRIPT (PS) files require you 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 can be viewed by a browser. |
| INIT-STRING | | (none) | Sends control or other characters to the printer at the beginning of the report. This parameter is designed primarily for the LINEPRINTER and has limited use with other printer types. Specify nondisplay characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Choice or Measure | Default | Description |
|---|---|---|---|
| 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 nondisplay characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character. |
| COLOR | Yes, No | No | Specifies whether this printer can print in color. |
| POINT-SIZE | points | 12 | This argument does not apply to LINEPRINTER printers. The beginning size of the selected font. |
| FONT-TYPE | PROPORTIONAL, FIXED | Depends on the font | This argument applies only to HPLASERJET printers and needs to be specified only for font types not defined in the Fonts Available for HP LaserJet Printers in SQR table. |
| SYMBOL-SET | HP defined sets | 0U | This argument applies only to HPLASERJET printers. The default value of 0U is for the USASCII symbol set. For a complete list of the symbol sets, see the *HP LaserJet Technical Reference Manual*. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Choice or Measure | Default | Description |
|---|---|---|---|
| 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. |
| PITCH | characters/inch | 10 | This argument is for HPLASERJET printers only. If you specify a fixed pitch font, you should also indicate the pitch. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Choice or Measure | Default | Description |
|---|---|---|---|
| FONT | font_number | 3 | This is the font number of the typeface to use. For HPLASERJET printers, this is the typeface value as defined by Hewlett-Packard. For a complete list of the typeface numbers, see the *HP LaserJet Technical Reference Manual.*<br><br>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. The Fonts Available for HP LaserJet Printers in SQR table lists the fonts available in SQR internally. The Fonts Available for PostScript Printers table lists the fonts available in the SQR POSTSCRI.STR file. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Choice or Measure | Default | Description |
|---|---|---|---|
| BEFORE-BOLD | any string | (none) | The BEFORE-BOLD and AFTER-BOLD arguments are for LINEPRINTER printers only. They specify the character string to turn boldfacing on and off. If the string contains blank characters, enclose it in single quote marks ('...'). To specify nonprintable characters, such as ESC, enclose the decimal value inside angle brackets as shown: `BEFORE-BOLD=<27>[r ! Turn on bold` `AFTER-BOLD=<27>[u ! Turn it off`<br>These arguments work with the BOLD argument of the PRINT command. |
| AFTER-BOLD | any string | (none) | See BEFORE-BOLD. |

This table lists the fonts available in SQR for use with the FONT argument for HPLASERJET printer types.

| 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 |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Value | Typeface | Style |
|---|---|---|
| 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 the Fonts Available for HP LaserJet Printers in SQR table, you must indicate the font style using the FONT-TYPE argument to ensure that the correct typeface is selected by the printer.

This table lists the fonts available in SQR for use with the FONT argument for PostScript printer types:

| Value | Typeface | Boldface Type Available |
|---|---|---|
| 3 | Courier | Y |
| 4 | Helvetica | Y |
| 5 | Times Roman | Y |
| 6 | Avant Garde Book | N/A |
| 8 | Palatino Roman | Y |
| 11 | Symbol | N/A |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Value | Typeface | Boldface Type Available |
|---|---|---|
| 12 | Zapf Dingbats | N/A |
| 17 | Zapf Chancery Medium Italic | N/A |
| 18 | Bookman Light | N/A |
| 23 | New Century Schoolbook Roman | Y |
| 30 | Courier Oblique | Y |
| 31 | Helvetica Oblique | Y |
| 32 | Times Italic | Y |
| 33 | Avant Garde Demi | N/A |
| 34 | Avant Garde Book Oblique | N/A |
| 35 | Avant Garde Demi Oblique | N/A |
| 36 | Palatino Oblique | Y |
| 37 | New Century Schoolbook Italic | Y |
| 38 | Helvetica Narrow | Y |
| 39 | Helvetica Narrow Oblique | Y |
| 40 | Bookman Demi | N/A |
| 41 | Bookman Light Italic | N/A |
| 42 | Bookman Demi Italic | N/A |

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

 Copyright © 1988-2007, Oracle. All rights reserved.

There are different fonts available in SQR for Windows when printing with Microsoft 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 Microsoft Windows printer. To specify a nondefault Microsoft Windows printer, enter -PRINTER:WP:{printer name}. The {printer name} is the name assigned to your printer. For example, to send output to a Windows printer named *NewPrinter,* you would use -PRINTER:WP:NewPrinter. If your printer name has spaces, enclose the entire argument in quotes.

Fonts are specified by number in the FONT qualifier of the ALTER-PRINTER command.

This table lists the fonts available when printing with Microsoft Windows printer drivers:

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

**Note.** Fonts 6, 8, and 800 are not supplied with Microsoft 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 Chapter 7, “Using the PSSQR.INI File and the PSSQR Command Line,” page 285.

## Parameters

| Parameter | Description |
|---|---|
| printer_name | A unique name to be used for referencing a printer definition and its attributes. |

Copyright © 1988-2007, Oracle. All rights reserved. 

**Note.** The DECLARE-PRINTER Command Arguments table describes the other arguments of the DECLARE-PRINTER command. The table lists the options, default values, and description of each of the arguments.

### Example

The following example illustrates the DECLARE-PRINTER command

```
declare-printer HP-definition   ! Default HP definition

type=HP                         ! for all reports

font=4                          ! Helvetica

symbol-set=12U                  ! PC-850 Multilingual
end-declare
declare-printer PS-Sales        ! PS definition

for-reports=(sales)             ! for the Sales report

type=PS

font=5                          ! Times-Roman
end-declare
```

### See Also

ALTER-PRINTER, DECLARE-REPORT

## DECLARE-PROCEDURE

### 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
```

### Description

Declares procedures that are triggered when a specified event occurs.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

If there are no FOR-REPORTS 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 runtime. To turn a procedure off, specify NONE in the USE-PROCEDURE statement.

## Parameters

| Parameter | Description |
|---|---|
| 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.<br>You can optionally specify arguments to be passed to any of the procedures. Arguments can be any variable, column, or literal. |

## Example

The following example illustrates the DECLARE-PROCEDURE command

```
declare-procedure              ! These procedures will

before-report=report_heading   ! be used by all reports

after-report=report_footing
end-declare
declare-procedure             ! These procedures will

for-reports=(customer)        ! be used by the customer

before-page=page_setup        ! report

after-page=page_totals
end-declare
```

## See Also

USE-PROCEDURE

Copyright © 1988-2007, Oracle. All rights reserved. 

# DECLARE-REPORT

## Syntax

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

## Description

Defines reports and their attributes.

Issue the DECLARE-REPORT in the SETUP section.

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

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

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

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

Additional reports are generated with names conforming to the rules dictated by the OUTPUT-FILE-MODE setting in the sqr.ini file.

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 OUTPUT-FILE-MODE setting in the sqr.ini file..

## Parameters

| Parameter | Description |
|---|---|
| report_name | Specifies the name of the report. |
| TOC | Specifies the name of the table of contents for this report. |
| LAYOUT | Specifies the name of the layout for this report. If no layout is specified, the default layout is used. |
| PRINTER-TYPE | Specifies the type of printer to be used for this report. If no printer type is specified, the default, LINEPRINTER, is used for this report. 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 |

## Example

The following example illustrates the DECLARE-REPORT command

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

 Copyright © 1988-2007, Oracle. All rights reserved.

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

declare-report customer_detail

toc=detailed

layout=customer_layout

printer-type=postscript
end-declare
declare-report customer_summary

layout=summary_layout

printer-type=postscript
end-declare
.
.
.
use-report customer_detail

...print customer detail...
use-report customer_summary

...print customer summary...
```

### See Also

USE-REPORT, DECLARE-LAYOUT, DECLARE-PRINTER, DECLARE-TOC

## DECLARE-TOC

### 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
```

### Description

Defines the table of contents and its attributes.

Use DECLARE-TOC in the SETUP section.

Copyright © 1988-2007, Oracle. All rights reserved. 

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.

## Parameters

| Parameter | Description |
|---|---|
| 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 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:<br>#SQR-TOC-LEVEL contains the level.<br>#SQR-TOC-TEXT contains the text.<br>#SQR-TOC-PAGE contains the page number.<br>These are global variables. If the procedure is local, you must precede it with an underscore (for example, #_sqr-toc-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. |

## Example

The following example illustrates the DECLARE-TOC command

```
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
.
.
```

 Copyright © 1988-2007, Oracle. All rights reserved.

### See Also

BEGIN-FOOTING, BEGIN-HEADING, DECLARE-REPORT, TOC-ENTRY

# DECLARE-VARIABLE

## 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
```

## Description

Enables the user to explicitly declare a variable type.

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.

In addition to FLOAT, INTEGER, or DECIMAL, you can set DEFAULT-NUMERIC in the sqr.ini file and -DNT command-line flag to V30. With V30, the program acts in the same manner as in pre-version 4.0 releases; that is, all variables are FLOAT. V30 is not a valid setting for the DEFAULT-NUMERIC setting in the DECLARE-VARIABLE command.

The DECLARE-VARIABLE command enables the user to determine the type of variables to use. 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, this command enables 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, use the DECIMAL qualifier.

The DECLARE-VARIABLE command, the -DNT command-line flag, and the DEFAULT-NUMERIC setting in the sqr.ini file affect 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.

Copyright © 1988-2007, Oracle. All rights reserved. 

## Parameters

| Parameter | Description |
|---|---|
| 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 -9.999999999999999999999999999999999999E±4096 to +9.999999999999999999999999999999999999E±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 January1, 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 variables. |

## Example

The following example illustrates the DECLARE-VARIABLE command

```
begin-setup
  declare-variable
    default-numeric=float
    decimal  #decimal(10)
    integer  #counter
    date     $date
  end-declare
end-setup
.
.
let $date = strtodate('Jan 01 2004','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)
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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
.
.
```

# #DEFINE

## Syntax

`#DEFINE` *`substitution_variable value`*

## Description

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

#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 the #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 enables you 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 does not need to be enclosed within quotes. The entire string is used as is.

The #DEFINE command cannot be broken across program lines.

Copyright © 1988-2007, Oracle. All rights reserved. 

## Parameters

| Parameter | Description |
|---|---|
| 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. |

## Example

This code example 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
```

This code example from a report uses the definitions from the preceding example:

```
  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.
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
end-procedure
```

### See Also

ASK

## DISPLAY

### Syntax

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

### Description

Displays the specified column, variable, or literal.

The DISPLAY command can display data to a terminal. The data is displayed to the current location on the screen. If you want 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 has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the TIME Column Formats table.

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 the Default Database Formats table, or the database-independent format SYYYYMMDD[HH24[MI[SS[NNNNNN]]]].

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

Copyright © 1988-2007, Oracle. All rights reserved. 

## Parameters

| Parameter | Description |
|---|---|
| any_lit\|_var\|_col | The text, number, or date to be displayed. |
| edit_mask | Causes the field to be edited before being displayed. 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 of the current locale. This option is not valid with date variables. |
| MONEY | Indicates that any_lit\|_var\|_col is to be formatted using the MONEY-EDIT-MASK of the current locale. This option is not valid with date variables. |
| DATE | Indicates that any_lit\|_var\|_col is to be formatted using the DATE-EDIT-MASK of the current locale. This option is not valid with numeric variables. If DATE-EDIT-MASK has not been specified, the date is displayed using the default format for that database (see the Default Database Formats table). |
| NOLINE | Suppresses the carriage return after the field is displayed. |

## Example

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

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

Produces the following output:

```
123-45-6789
```

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

Produces the following output:

```
1,234,567.89
```

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

Produces the following output:

```
123.780000
```

```
!
 ! Display a number using the locale default numeric edit mask
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
 !
 alter-locale  number-edit-mask = '99,999,999.99'
 display 123456.78 number
```

Produces the following output:

```
123,456.78
```

```
!
 ! 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
```

```
!
 ! 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-2004
```

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

Produces the following output:

```
Hello World
```

```
!
 ! 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.
```

Copyright © 1988-2007, Oracle. All rights reserved. 

### See Also

The SHOW command for information about screen control.
The LET command for information about 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

### Syntax

```
DIVIDE {src_num_lit|_var|_col} INTO dst_num_var
[ON-ERROR={HIGH|ZERO}][ROUND=nn]
```

### Description

Divides one number into another.

The source field is divided into the destination field and the result is placed in the destination. The source is always first, the destination always second.

When dealing with money-related values (dollars and cents), use decimal variables rather than float variables. Float variables are stored as double precision floating point numbers, and small inaccuracies can appear when dividing many numbers in succession. These inaccuracies can appear due to the way different hardware and software implementations represent floating point numbers.

### Parameters

| Parameter | Description |
|---|---|
| src_num_lit\|_var\|_col | Divided into the contents of dst_num_var. |
| dst_num_var | Contains the result after execution. |
| ON-ERROR | Sets the result to the specified number when a division by zero is attempted. If ON-ERROR is omitted and a division by zero is attempted, SQR halts with an error message. |
| ROUND | Rounds the result to the specified number of digits to the right of the decimal point. For float variables, this value can be from 0 to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate. |

### Example

The following example illustrates the DIVIDE command

```
divide  37.5  into  #price   ! #price / 37.5
divide  &rate  into  #tot  on-error=high
divide  #j  into  #subtot  on-error=zero
```

**Note.** In the preceding example, `High` is the maximum value and zero is the lowest value.

### See Also

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

 Copyright © 1988-2007, Oracle. All rights reserved.

# DO

## Syntax

```
DO procedure_name[(arg1[, argi]...)]
```

## Description

Invokes the specified procedure.

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:

- 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 is 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 has not been set, SQR ues the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

## Parameters

| Parameter | Description |
|---|---|
| procedure_name | Specifies the name of the procedure to be executed. |
| arg1 [, argi ] | Specifies the arguments to be passed to the procedure. Arguments can be any type of variable or constant value. |

## Example

The following example illustrates the DO command

```
do get_names
do add_to_list ($name)
```

Copyright © 1988-2007, Oracle. All rights reserved. 

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

### See Also

The BEGIN-PROCEDURE command for information about passing arguments.
The PRINT command for information about date and time formats.

## #ELSE

### Syntax

```
#ELSE
```

### Description

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.)

### See Also

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

## ELSE

### Syntax

```
ELSE
```

### Description

ELSE is an optional command in an IF command.

### See Also

The IF command for a description and example.

## ENCODE

### Syntax

ENCODE *src_code_string_lit* INTO *dst_txt_var*

### Description

Assigns a nondisplay or display character to a string variable.

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.

The encode characters can be included in a report at the appropriate location using a PRINT or PRINT-DIRECT command.

Unicode (UCS-2) code points from <1> to <65535> can be defined in ENCODE command.

 Copyright © 1988-2007, Oracle. All rights reserved.

### Parameters

| Parameter | Description |
|---|---|
| src_code_string_lit | Specifies a string of characters to be encoded and placed in *dst_txt_var*. |
| dst_txt_var | Contains the result after execution. |

### Example

The following example illustrates the ENCODE command

```
encode '<27>L11233' into $bold            ! Code sequence to turn bold on.
print $bold () code-printer=lp
```

### See Also

The chr function described in the Miscellaneous Functions table under the LET command.
PRINT, PRINT-DIRECT

## END-DECLARE, END-DOCUMENT, END-EVALUATE, END-FOOTING, END-HEADING

### Syntax

```
END-DECLARE
END-DOCUMENT
END-EVALUATE
END-FOOTING
END-HEADING
```

### Description

Completes a section or paragraph.

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- commands complete the corresponding BEGIN- command:

```
BEGIN-DOCUMENT
EVALUATE
BEGIN-FOOTING
BEGIN-HEADING
```

Each command must begin on its own line.

### Example

The following example illustrates the BEGIN-FOOTING and END-FOOTING commands

Copyright © 1988-2007, Oracle. All rights reserved. 

```
begin-footing  2
   print  'Company Confidential'  (1)  center
end-footing
```

### See Also

DECLARE-paragraph, BEGIN-section

## #END-IF, #ENDIF

### Syntax

```
#END-IF
```

### Description

Ends an #IF, #IFDEF, or #IFNDEF command. (#END-IF is a compiler directive.)

#ENDIF (without the dash) is a synonym for #END-IF.

### Example

The following example illustrates the #END-IF compiler directive:

```
#ifdef debuga
  show  'DebugA: #j = '  #j  edit  9999.99
  show  'Cust_num   = '  &cust_num
#end-if
```

### See Also

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

## END-IF

### Syntax

```
END-IF
```

Ends an IF command.

### See Also

The IF command for a description and example.

## END-PROCEDURE, END-PROGRAM, END-SELECT, END-SETUP, END-SQL, END-WHILE

### Syntax

```
END-PROCEDURE
END-PROGRAM
END-SELECT
END-SETUP
END-SQL
END-WHILE
```

 Copyright © 1988-2007, Oracle. All rights reserved.

### Description

Completes the corresponding section or paragraph.

Each END- command completes the corresponding BEGIN- command:

```
BEGIN-PROCEDURE
BEGIN-PROGRAM
BEGIN-SELECT
BEGIN-SETUP
BEGIN-SQL
WHILE
```

Each command must begin on its own line.

### Example

The following example illustrates the END-PROGRAM command

```
begin-program
   do main
end-program
```

### See Also

BEGIN-section, WHILE

## EVALUATE

### 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_Command...
[BREAK]
[WHEN comparison_operator {any_lit|_var|_col}
SQR_Command...
[BREAK]]
[WHEN-OTHER
SQR_Command...
[BREAK]]
END-EVALUATE
```

### Description

Determines the value of a column, literal, or variable and takes action based on that value.

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.

Copyright © 1988-2007, Oracle. All rights reserved. 

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 in the format specified by the SQR_DB_DATE_FORMAT setting, SQR uses one of the database-dependent formats (see the Default Database Formats table), 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, or the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format as listed in the TIME Column Formats table.

## Parameters

| Parameter | Description |
|---|---|
| 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 the list of operators in the #IF command. .<br><br>See Chapter 2, "SQR Command Reference," #IF, page 124. |
| 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.<br>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. |

## Example

The following example illustrates the EVALUATE command

```
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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
    when-other
       if isnull (&code)
           do null_code
        else
           move 0 to #j  ! Unknown code.
       end-if
       break
end-evaluate
```

### See Also

The commands IF and LET for comparison operators.

## EXECUTE (Sybase and Microsoft SQL Server)

### Syntax

```
EXECUTE [-XC][ON-ERROR=procedure[(arg1[,argi]...)]]
[DO=procedure[(arg1[,argi]...)]]
[@#status_var=]stored_procedure_name
[[@param=]{any_col|_var|_lit}[,...]]
[INTO any_coldata_type[(length_int_lit)]
[,...]][WITH RECOMPILE]
```

The syntax of this command generally follows that of the Sybase Transact-SQL EXECUTE command, with the exception of optional arguments and the INTO argument.

### Description

Executes a stored procedure in Sybase or Microsoft SQL Server database.

If the stored procedure specified in **stored_procedure_name** contains a SELECT query, the EXECUTE command must specify an INTO argument 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.

Copyright © 1988-2007, Oracle. All rights reserved. 

## Parameters

| Parameter | Description |
|---|---|
| -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. |
| 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.<br>You can specify arguments to be passed to the ON-ERROR procedure. Arguments can be any variable, column, or literal. |
| DO | Specifies an SQR procedure to execute for each row selected in the query. Processing continues until all rows have been retrieved. You can specify arguments to be passed to the procedure. Arguments can be any variable, column, or literal. |
| *@#status_variable* | Returns the procedure status in the specified numeric variable. The status is returned only after selected rows are retrieved. |
| *stored_procedure_name* | Names the stored procedure 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. |
| OUT[PUT] | 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 that are 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.<br>If the stored procedure contains more than one SELECT query, only the first query is described with the INTO argument. Rows from subsequent queries are ignored. |
| WITH RECOMPILE | Causes the query to recompile each time it is executed rather than using the plan stored with the procedure. Normally, this is not required or recommended. |

## Example

The following code 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 code 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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
   @#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

## Syntax

EXIT-SELECT

## Description

Exits a SELECT paragraph immediately.

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.

## Example

The following example illustrates the EXIT-SELECT command

```
begin-select
cust_num, co_name, contact, city, state, zip, employees
  add &employees to #tot_emps
  if #tot_emps >= 5000
    exit-select         ! Have reached required total emps.
  end-if
  do print_company
from customers order by employees desc
end-select
```

## See Also

BEGIN-SELECT

# EXTRACT

## Syntax

EXTRACT {*dst_txt_var*|*date_var*} FROM
{{*src_txt_lit*|*_var*|*_col*}|{*src_date_var*|*_col*}}

Copyright © 1988-2007, Oracle. All rights reserved. 

{*start_num_lit*|*_var*}{*length_num_lit*|*_var*}

## Description

Copies a portion of a string into a string variable.

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 leftmost character; an offset of 1 begins one character beyond that, and so on.

If the source is a date variable or column, it is converted to a string before the extraction according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR specifies the SQR_DB_DATE_FORMAT setting.

  If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

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 the Default Database Formats table).
- The database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.

## Parameters

| Parameter | Description |
|---|---|
| dst_txt_var \| date_var | Specifies a text or date variable into which the extracted string is placed. |
| {src_txt_lit\|_var\|_col}\| {src_date_var\|_col} | Specifies a text or date variable, column, or literal from which the string is to be extracted. |
| start_num_lit\|_var | Specifies starting location of the string to be extracted. |
| length_num_lit\|_var | Specifies length of the string to be extracted. |

## Example

The following example illustrates the EXTRACT command

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

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

**Note.** PeopleSoft recommends that you do not use the EXTRACT command when processing strings.

 Copyright © 1988-2007, Oracle. All rights reserved.

### See Also

The substr function described in Miscellaneous Functions table under the LET command.
FIND
The PRINT command for information about default date and time formats

## FIND

### Syntax

```
FIND {{obj_txt_lit|_var|_col}|{date_var|_col}} IN
{{src_txt_var|_col}|{date_var|_col}}
{start_int_lit|_var} dst_location_int_var
```

### Description

Determines the location of a character sequence within a string.

FIND searches the specified string for a character sequence and, if the string is found, returns its location as an offset from the beginning of the specified string. If the sequence is not found, FIND returns -1 in *dst_location_int_var*.

You must specify an offset from which to begin the search and supply a numeric variable for the return of the location.

If the source or search object is a date variable or column, it is converted to a string before the search according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.

  If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting is used.

  If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

### Parameters

| Parameter | Description |
|---|---|
| {obj_txt_lit\|_var\|_col}\|{date_var\|_col} | Specifies a text variable, column, or literal that is to be sought in *src_txt_var\|_col*. |
| {src_txt_var\|_col}\|{date_var\|_col} | Specifies a text variable or column to be searched. |
| start_int_lit\|_var | Specifies the starting location of the search. |
| dst_location_int_var | Specifies the returned starting location of the leftmost character of the matching text in {*src_txt_var\|_col\| date_var\|_col* }. |

Copyright © 1988-2007, Oracle. All rights reserved. 

### Example

The following example illustrates the FIND command

```
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 the Miscellaneous Functions table under the LET command.
EXTRACT
The PRINT command for information about default date and time formats.

## GET

### Syntax

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

### Description

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

### Parameters

| Parameter | Description |
|---|---|
| dst_any_var | Date, string, or numeric variables (not database columns) can be destination variables. Numeric variables (decimal, float, integer) are copied from number fields. String variables are copied from char, text, or date fields. Date variables are copied from char, text, or date fields.<br>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 the Default Database Formats table.<br>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 (see the Default Database Formats table), or the database-independent format 'SYYYYMMDD[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. |

### Example

The following code 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)
```

 Copyright © 1988-2007, Oracle. All rights reserved.

The following code 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 code 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 PUT command for information about moving data into an array.

## GET-COLOR

### Syntax

```
GET-COLOR
[PRINT-TEXT-FOREGROUND=({color_name_var |{rgb})]
[PRINT-TEXT-BACKGROUND=({color_name_var |{rgb})]
```

### Description

Retrieves the current colors.

The GET-COLOR command is allowed wherever the PRINT command is allowed. If the requested color settings do not map to a defined name, 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.

Copyright © 1988-2007, Oracle. All rights reserved. 

## Parameters

| Parameter | Description |
|---|---|
| PRINT-TEXT-FOREGROUND | Defines the color in which the text prints. |
| PRINT-TEXT-BACKGROUND | Defines the color to print as a background behind the text. |
| {*color_name_var*} | A color_name is composed of alphanumeric characters (A-Z, 0-9), the underscore (_) character, and the hyphen (-) character. The name must start with an alphabetical (A-Z) character and is not case-sensitive. 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.<br>The default colors implicitly installed with SQR include:<br>black=(0,0,0)<br>white=(255,255,255)<br>gray=(128,128,128)<br>silver=(192,192,192)<br>red=(255,0,0)<br>green=(0,255,0)<br>blue=(0,0,255)<br>yellow=(255,255,0)<br>purple=(128,0,128)<br>olive=(128,128,0)<br>navy=(0,0,128)<br>aqua=(0,255,255)<br>lime=(0,128,0)<br>maroon=(128,0,0)<br>teal=(0,128,128)<br>fuchsia=(255,0,255) |

## Example

The following example illustrates the GET-COLOR command

```
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)
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
    set-color print-text-foreground = ('purple')
    print 'Barney' (+1,1)
    set-color print-text-foreground = ($print-foreground)
end-program
```

### See Also

DECLARE-COLOR-MAP, ALTER-COLOR-MAP, SET-COLOR

## GOTO

### Syntax

`GOTO` *`label`*

### Description

Skips to the specified label.

Labels must end with a colon (:) and can appear anywhere within the same section or paragraph as the GOTO command.

### Parameters

| Parameter | Description |
|---|---|
| label | Specifies a label within the same section or paragraph. |

### Example

The following example illustrates the GOTO command

```
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
```

## GRAPHIC BOX, GRAPHIC HORZ-LINE, GRAPHIC VERT-LINE

### Syntax

The GRAPHIC commands have the following syntax:

```
GRAPHIC ({line_int_lit|_var},{column_int_lit|_var},
{width_int_lit|_var})

BOX {depth_int_lit|_var}
[rule_width_int_lit|_var[shading_int_lit|_var]]
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
GRAPHIC ({line_int_lit|_var},{column_int_lit|_var},
{length_int_lit|_var})

HORZ-LINE [rule_width_int_lit|_var]

GRAPHIC ({line_int_lit|_var},{column_int_lit|_var},
{length_int_lit|_var})

VERT-LINE [rule_width_int_lit|_var]
```

## Description

Draws a box or line.

After GRAPHIC commands execute, SQR changes the current print location to the starting location of the graphic. This is different from the way the PRINT command works.

The GRAPHIC command has the following variations:

- BOX
- HORZ-LINE
- VERT-LINE

The following sections describe the individual GRAPHIC commands:

 Copyright © 1988-2007, Oracle. All rights reserved.

## Parameters

| Parameter | Description |
|---|---|
| BOX | BOX 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. |
| *width* and *depth* | The width is the horizontal size in character columns; depth is the vertical size in lines. The top left corner of the box is drawn at the line and column specified. The bottom right corner is calculated using the width and depth. You can specify relative placement with (+), (-), or numeric variables, as with regular print positions. |
| *rule_width* | The default rule width is 2 decipoints (there are 720 decipoints per inch). The top horizontal line is drawn just below the base of the line above the starting point. The bottom horizontal line is drawn just below the base of the ending line. Therefore, a one-line deep box surrounds a single line. |
| *shading* | A number between 1 and 100, specifying the percentage of shading to apply. 1 is very light, and 100 is black. If no shading is specified, the box is blank. Specify a rule-width of zero, if a border is not desired. |
| HORZ-LINE | HORZ-LINE draws a horizontal line from the location specified, for the length specified. Horizontal lines are drawn just below the base. |
| *rule_width* | The default rule width is 2 decipoints. |
| VERT-LINE | VERT-LINE 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. |
| *rule_width* | The default rule width is 2 decipoints. |

## Example

The following code example shows the GRAPHIC BOX command:

```
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
```

The following code example shows the GRAPHIC HORZ-LINE command:

```
graphic (4,1,66) horz-line 10  ! Put line under page heading
graphic (+1,62,12) horz-line   ! Put line under final total
```

The following code example shows the GRAPHIC VERT-LINE command:

```
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
```

Copyright © 1988-2007, Oracle. All rights reserved. 

### See Also

The ALTER-PRINTER and DECLARE-PRINTER commands for information about setting and changing the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

## #IF

### Syntax

`#IF {`*`txt_lit`*`|`*`num_lit`*`}`*`comparison_operator`*
`{`*`txt_lit`*`|`*`num_lit`*`}`

### Description

Indicates that the commands following are to be compiled when the expression is TRUE. (#IF is a compiler directive.)

SQR has five compiler directives that enable 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).

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.

You can nest #IF, #IFDEF, or #IFNDEF directives to a maximum of 10 levels.

The #IF, #IFDEF, or #IFNDEF directives cannot be broken across program lines.

The following table lists the 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 not case-sensitive. 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). |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Directive | Example | Description |
|---|---|---|
| #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. |

## Parameters

| Parameter | Description |
|---|---|
| txt_lit \| num_lit | Any text or numeric literal. |
| comparison_operator | Any of the comparison operators as follows:<br>=Equal<br>!=Not Equal<br><>Not Equal<br><Less than<br>>Greater than<br><=Less than or equal<br>>=Greater than or equal |

## Example

The following example illustrates the #IF compiler directive:

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

#ifdef  debug
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
  show 'DEBUG:  Cust_num = '  &cust_num  edit  099999
#endif

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

### See Also

The #DEBUG command for information about the -DEBUG command-line flag.

## IF

### Syntax

`IF logical_expression`

IF commands have the following structure:

```
IF logical_expression
SQR Command...
[ELSE
SQR Command...]
END-IF
```

### Description

Executes commands depending on the value of a condition.

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 command.

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 the Default Database Formats table), 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, or the format listed in the Date Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format as listed in Time Column Formats table.

### Parameters

| Parameter | Description |
|---|---|
| logical_expression | Any valid logical expression. See the LET command for a description of logical expressions. |

 Copyright © 1988-2007, Oracle. All rights reserved.

### Example

The following example illustrates the IF command

```
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 2004 23:59'
   do past_due
end-if
```

### See Also

The LET command for a description of logical expressions.
EVALUATE

## #IFDEF

### Syntax

`#IFDEF` *`substitution_variable`*

### Description

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.)

### Parameters

| Parameter | Description |
|---|---|
| substitution_variable | Is the variable to be used as the substitution variable. |

### See Also

The #IF command for a description of each compiler directive.

## #IFNDEF

### Syntax

`#IFNDEF` *`substitution_variable`*

Copyright © 1988-2007, Oracle. All rights reserved. 

### Description

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.)

### Parameters

| Parameter | Description |
|---|---|
| substitution_variable | Is the variable to be used as the substitution variable. |

### See Also

The #IF command for a description of each compiler directive.

## #INCLUDE

### Syntax

`#INCLUDE` *`filename_lit`*

### Description

Includes an external source file into the SQR report specification.

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 enables you to substitute all or part of the include file name at runtime, adding flexibility to controlling which file is included for the run.

### Parameters

| Parameter | Description |
|---|---|
| filename_lit | A file name that is valid for the platform on which the application is to be compiled. |

### Example

The following example illustrates the #INCLUDE command

```
#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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
                                        ! in the SETUP section, preceding
                                        ! this #include statement.
```

# INPUT

## Syntax

```
INPUT input_var[MAXLEN=nn] [prompt]
[TYPE={CHAR|TEXT|NUMBER|INTEGER|DATE}]
[STATUS=num_var] [NOPROMPT] [BATCH-MODE]
[FORMAT={txt_lit|_var|_col}]
```

## Description

Accepts data entered by the user at a terminal.

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 the length 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 either 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 data types supported are described in the following table:

| Data Type | 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:<br>• MM/DD/YYYY [BC\|AD] [HH:MI[:SS[.NNNNNN]] [AM\|PM]]<br>• MM-DD-YYYY [BC\|AD] [HH:MI[:SS[.NNNNNN]] [AM\|PM]]<br>• MM.DD.YYYY [BC\|AD] [HH:MI[:SS[.NNNNNN]] [AM\|PM]]<br>• SYYYYMMDD[HH24[MI[SS[NNNNNN]]]] |

Copyright © 1988-2007, Oracle. All rights reserved. 

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.

The following table lists the values of the INPUT command's STATUS argument:

| 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 the datatypes table.

## Parameters

| Parameter | Description |
|---|---|
| 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 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. The Date Edit Format Codes table lists date edit format codes. |

## Example

The following example shows several INPUT commands:

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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

ALTER-LOCALE

The INPUT-DATE-EDIT-MASK setting in the chapter "Using the PSSQR.INI File and the PSSQR Command Line."

## LAST-PAGE

### Syntax

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

### Description

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

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.

Copyright © 1988-2007, Oracle. All rights reserved. 

### Parameters

| Parameter | Description |
|---|---|
| position | Specifies the position for printing the last page number. See the POSITION command for a description of the *position* parameter. |
| 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. |

### Example

The following example illustrates the LAST-PAGE command

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

### See Also

PAGE-NUMBER, BEGIN-HEADING, BEGIN-FOOTING

## LET

### Syntax

`LET` *`dst_var=expression`*

### Description

Assigns the value of an expression to a string, numeric, or date variable.

Valid expressions are formed as a combination of operands, operators, and functions. String, numeric, date, and array field operands can be used in an expression and 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 a rich set of mathematical, string, date, and file manipulation functions along with a number of special purpose utility functions. All combined, the SQR expression provides a powerful tool that can be tailored to suit any information processing need. The following detail outlines the specific behavior of each expression component: (1) the operand, (2) the operator, and (3) the function.

### Parameters

| Parameter | Description |
|---|---|
| dst_var | A string, numeric, or date variable or array field to which the result of the expression is assigned. |
| expression | The expression to evaluate. |

 Copyright © 1988-2007, Oracle. All rights reserved.

## 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 code example:

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

Because 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 2004', 'MON DD YYYY')
```

The following sections list operators and functions supported in expressions.

## Operators

Operators of the same precedence are processed in the sequence in which 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.

This table lists operators in descending order of precedence (operators listed in the same row within the table have the same precedence):

| Operator | Explanation |
|---|---|
| \|\| | Concatenate two strings or dates |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Operator | Explanation |
|---|---|
| +, - | Sign prefix (positive or negative) |
| ^ | Exponent |
| *, /, % | Multiply, divide, remainder: a % b = mod(a,b) for integers |
| +, - | Plus, minus<br>**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 |
| not | Logical NOT |
| and | Logical AND |
| or, xor | Logical OR, XOR (exclusive OR) |

## Functions

This section lists numeric, file-related, and miscellaneous functions. The functions are listed in alphabetical order.

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 operands to the float type before the function is evaluated. The following table annotates the functions that directly support the decimal type and the ones that do not.

Use parentheses to override the normal precedence rules.

This table describes numeric functions:

 Copyright © 1988-2007, Oracle. All rights reserved.

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

Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Description |
|---|---|
| `atan` | Returns the arctangent of *num_value* in the range of -p/2 to p/2 radians. The value of *num_value* must be between -1 and 1. This function returns a float value.<br><br>Syntax:<br><br>`dst_var = atan(num_value)`<br><br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br><br>*dst_var* = decimal, float, or integer variable.<br><br>Example:<br>`let #fatanvar = atan(#fvar)` |
| `ceil` | Returns a value representing the smallest integer that is greater than or equal to *num_value*. This function returns a value of the same type as *num_value*.<br><br>Syntax:<br><br>`dst_var = ceil(num_value)`<br><br>*num_value* = decimal, float, or integer literal, column, variable, or expression.<br><br>*dst_var* = decimal, float, or integer variable.<br><br>Example:<br>`let #fceilvar = ceil(#fvar)` |
| `cos` | Returns the cosine of *num_value*. This function returns a float value.<br><br>Syntax:<br><br>`dst_var = cos(num_value)`<br><br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br><br>*dst_var* = decimal, float, or integer variable.<br><br>Example:<br>`let #fcosvar = cos(#fvar)` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Description |
|---|---|
| `cosh` | Returns the hyperbolic cosine of *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `cosh(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #fcoshvar = cosh(#fvar)` |
| `deg` | Returns a value expressed in degrees of *num_value* which is expressed in radians. This function returns a float value.<br>Syntax:<br>*dst_var* = `deg(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #fdegvar = deg(#fvar)` |
| `e10` | Returns the value of 10 raised to *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `e10(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #fe10var = e10(#fvar)` |

Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Description |
|---|---|
| `exp` | Returns the value of e raised to *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `exp(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #fexpvar = exp(#fvar)` |
| `floor` | Returns a value representing the largest integer that is less than or equal to *num_value*. This function returns a value of the same type as *num_value*.<br>Syntax:<br>*dst_var* = `floor(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #ffloorvar = floor(#fvar)` |
| `log` | Returns the natural logarithm of *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `log(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #flogvar = log(#fvar)` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Description |
|---|---|
| `log10` | Returns the base-10 logarithm of *num_value*. This function returns a float value.<br><br>Syntax:<br><br>*dst_var* = `log10(`*num_value*`)`<br><br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br><br>*dst_var* = decimal, float, or integer variable.<br><br>Example:<br>`let #flog10var = log10(#fvar)` |
| `mod` | Returns the fractional remainder, f, of *x_value*/ *y_value* such that *x_value* = i * *y_value* + f, where i is an integer, f has the same sign as *x_value*, and the absolute value of f is less than the absolute value of *y_value*. The arguments are promoted to the type of the greatest precision and the function returns a value of that type.<br><br>Syntax:<br><br>*dst_var* = `mod(`*x_value*`, `*y_value*`)`<br><br>*x_value* = decimal, float, or integer literal, column, variable, or expression.<br><br>*y_value* = decimal, float, or integer literal, column, variable, or expression.<br><br>*dst_var* = decimal, float, or integer variable.<br><br>Example:<br>`let #fmodvar = mod(#fxvar, #fyvar)` |
| `power` | Returns the value of *x_value* raised to the power of *y_value*. This function returns a float value.<br><br>Syntax:<br><br>*dst_var* = `power(`*x_value*`, `*y_value*`)`<br><br>*x_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br><br>*y_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br><br>*dst_var* = decimal, float, or integer variable.<br><br>Example:<br>`let #fpowervar = power(#fxvar, #fyvar)` |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Function | Description |
|---|---|
| `rad` | Returns a value expressed in radians of *num_value* which is expressed in degrees. This function returns a float value.<br><br>Syntax:<br><br>`dst_var = rad(num_value)`<br><br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br><br>place_value = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br><br>*dst_var* = decimal, float, or integer variable.<br><br>Example:<br>`let #fradvar = rad(#fvar)` |
| `round` | Returns a value that is *num_value* rounded to place_value digits after the decimal separator. This function returns a value of the same type as *num_value*.<br><br>Syntax:<br><br>`dst_var = round(num_value, place_value⇒`<br>`⇒`<br>`⇒`<br>`)`<br><br>*num_value* = decimal, float, or integer literal, column, variable, or expression.<br><br>*place_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br><br>*dst_var* = decimal, float, or integer variable.<br><br>Example:<br>`let #frndvar = round(#fvar, #fplace) ⇒`<br>`(#x, #y)` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Description |
|---|---|
| `sign` | Returns a -1, 0, or +1 depending on the sign of *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `sign(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #fsignvar = sign(#fvar)` |
| `sin` | Returns the sine of *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `sin(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #fsinvar = sin(#fvar)` |
| `sinh` | Returns the hyperbolic sine of *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `sinh(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #fsinhvar = sinh(#fvar)` |

Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Description |
|---|---|
| `sqrt` | Returns the square root of *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `sqrt(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #fsqrtvar = sqrt(#fvar)` |
| `tan` | Returns the tangent of *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `tan(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #ftanvar = tan(#fvar)` |


Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Description |
|---|---|
| `tanh` | Returns the hyperbolic tangent of *num_value*. This function returns a float value.<br>Syntax:<br>*dst_var* = `tanh(`*num_value*`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #ftanhvar = tanh(#fvar)` |
| `trunc` | Returns a value that is *num_value* truncated to place_value digits after the decimal separator. This function returns a value of the same type as *num_value*.<br>Syntax:<br>*dst_var* = `trunc(`*num_value*`,` *place_value*⇒<br>⇒<br>⇒<br>`)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression.<br>*place_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = decimal, float, or integer variable.<br>Example:<br>`let #ftruncvar = trunc(#fvar,⇒`<br>`#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 the Miscellaneous Functions table to prevent division by zero or other invalid function or operator argument values.

Copyright © 1988-2007, Oracle. All rights reserved. 

The following table lists file-related functions. These functions return zero (0) when successful; otherwise, they return the system error code.

| Function | Description |
|---|---|
| delete | Deletes the file *filename*. The function returns either a zero (0) to indicate success or the value returned from the operating system to indicate an error.<br>Syntax:<br>`stat_var = delete(filename)`<br>*filename* = text literal, column, variable, or expression.<br>*stat_var* = decimal, float, or integer variable.<br>Example:<br>`let #fstatus = delete($filename)` |
| exists | Determines if the file *filename* exists. The function returns either a zero (0) to indicate success or the value returned from the operating system to indicate an error.<br>Syntax:<br>`stat_var = exists(filename)`<br>*filename* = text literal, column, variable, or expression.<br>*stat_var* = decimal, float, or integer variable.<br>Example:<br>`let #fstatus = exists($filename)` |
| rename | Renames *old_filename* to *new_filename*. The function returns either a zero (0) to indicate success or the value returned from the operating system to indicate an error.<br>Syntax:<br>`stat_var = rename(old_filename, new_filename)`<br>*old_filename* = text literal, column, variable, or expression.<br>*new_filename* = text literal, column, variable, or expression.<br>*stat_var* = decimal, float, or integer variable.<br>Example:<br>`let #fstatus = rename($old_filename, $new_⇒ filename)` |

The following table lists miscellaneous functions. These functions return a string value unless otherwise indicated.

In these 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:

 Copyright © 1988-2007, Oracle. All rights reserved.

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.

  If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

Except where noted in an individual function, if a string variable, column, or expression is entered where a date argument is expected, the string must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats listed in the Default Database Formats table, or the database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.

| Function | Explanation |
|---|---|
| array | 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.<br><br>Syntax:<br><br>`array_var = array(array_name, field_name)`<br><br>*array_name* = text literal, column, variable, or expression<br><br>*field_name* = text literal, column, variable, or expression<br><br>*array_var* = text variable<br><br>Example:<br>`let #fstatus = printarray(array('products',⇒ 'name'), 10, 2, 'c')` |
| ascii | Returns the numeric value for the first character in *str_value*. This function returns a float value.<br><br>Syntax:<br><br>`ascii_var = ascii(str_value)`<br><br>*str_value* = date or text literal, column, variable, or expression<br><br>*ascii_var* = decimal, float, or integer variable<br><br>Example:<br>`let #fascii = ascii($filename)` |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Function | Explanation |
|---|---|
| asciic | Returns the numeric value for the first character (rather than byte) of the specified string.<br>Syntax:<br>`ascii_var = asciic(str_value)`<br>*str_value* = date or text literal, column, variable, or expression<br>*ascii_var* = decimal, float, or integer variable<br>Example:<br>`let #fascii = asciic($filename)` |
| chr | Returns a string that is composed of a character with the numeric value of *num_value*.<br>Syntax:<br>`dst_var = chr(num_value)`<br>*num_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = text variable.<br>Example:<br>`let $svar = chr(#num)` |
| cond | Returns *y_value* if the *x_value* is nonzero (0); otherwise returns *z_value*. If *y_value* is numeric, the *z_value* must also be numeric; otherwise, date and textual arguments are compatible. If either the *y_value* or *z_value* is a date variable, column, or expression, a date is returned. The return value of the function depends on which value is returned.<br>Syntax:<br>`dst_var = cond(x_value, y_value, z_value)`<br>*x_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*y_value* = Any literal, column, variable, or expression<br>*z_value* = Any literal, column, variable, or expression<br>*dst_var* = Any variable<br>Example:<br>`let #avg = #total / cond(&rate != 0, &rate, 1)` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| dateadd | Returns a date after adding (or subtracting) the specified units to the *date_value*.<br>Syntax:<br>`dst_var = dateadd(date_value, units_value, ⇒ quantity_value)`<br>*date_value* = date variable or expression<br>*units_value* = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second'<br>*quantity_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.<br>*dst_var* = date variable<br>Example:<br>`let $date = dateadd($startdate, 'day', 7.5)` |
| datediff | Returns the difference between the specified dates expressed in *units_value*. The function returns a float value. The result can be negative if the first date is earlier than the second date.<br>Syntax:<br>`dst_var = datediff(date1_value, date2_value, ⇒ units_value)`<br>*date1_value* = date variable or expression<br>*date2_value* = date variable or expression<br>*units_value* = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second'<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`let #diff = datediff($date1, $date2, 'hour')` |
| datenow | Returns the current local date and time from the client machine.<br>Syntax:<br>`dst_var = datenow()`<br>*dst_var* = date variable<br>Example:<br>`let $date = datenow()` |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Function | Explanation |
|---|---|
| datetostr | Converts the date *date_value* to a string in the format format_mask.<br>Syntax:<br>`dst_var = datetostr(date_value [, format_mask])`<br>*date_value* = date variable or expression<br>*format_mask* = 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, the format specified by the SQR_DB_DATE_FORMAT setting is used. If this has not been set, the first database-dependent format listed in the Default Database Formats table is used.<br>*dst_var* = text variable<br>Example: let $formdate = datetostr($date, 'Day Mon DD, YYYY') let $localedate = datetostr($date, DATE) |
| edit | Formats *source_value* according to edit_mask and returns a string containing the result.<br>Syntax:<br>`dst_var = edit(source_value, edit_mask)`<br>*source_value* = Any literal, column, variable, or expression<br>*edit_mask* = text literal, column, variable, or expression<br>*dst_var* = text variable<br>Example: )<br>`let $phone = edit(&phone, '(xxx) xxx-xxxxx')⇒`<br>`let $price = edit(#price, '999.99')⇒`<br>`let $today = edit($date, 'DD/MM/YYYY'` |
| getenv | Returns the value of the specified environment variable. If the environment variable does not exist, an empty string is returned.<br>Syntax:<br>`dst_var = getenv(env_value)`<br>*env_value* = text literal, column, variable, or expression<br>*dst_var* = text variable<br>Example:<br>`let $myuser = getenv('USER')` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| instr | Returns the numeric position of *sub_value* in *source_value* or zero (0) if not found. The search begins at offset *offset_value*. This function returns a float value.<br>Syntax:<br>`dst_var = instr(source_value, sub_value, offset_⇒ value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*sub_value* = text literal, column, variable, or expression<br>*offset_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`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.<br>Syntax:<br>`dst_var = instrb(source_value, sub_value, offset_⇒ value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*sub_value* = text literal, column, variable, or expression<br>*offset_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`let #offset = instrb(&description, 'auto', 10)` |
| isblank | Returns a value of one (1) if source_val is an empty string, null string, or composed entirely of whitespace characters; otherwise, returns a value of zero (0).<br>Syntax:<br>`dst_var = isblank(source_value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`let #blank = isblank(&description)` |

Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| isnull | Returns a value of one (1) if source_val is null; otherwise, returns a value of zero (0).<br>Syntax:<br>`dst_var = isnull(source_value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`let #null = isnull($date)` |
| length | Returns the number of characters in *source_value*.<br>Syntax:<br>`dst_var = length(source_value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`let #length = length(&description)`<br>**Note.** PeopleSoft recommends that you use either the lengthp or lengtht function instead of the length function. |
| lengthb | (Multibyte versions of SQR only.) Has the same functionality as the length function except that the return value is expressed in bytes, rather than in characters.<br>Syntax:<br>`dst_var = lengthb(source_value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`let #length = lengthb(&description)`<br>**Note.** PeopleSoft recommends that you use either the lengthp or lengtht function instead of the lengthb function. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| lengthp | Returns the length of a given string in print positions.<br>Syntax:<br>`dst_var = lengthp(source_value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`let #printlen = lengthp(&string)` |
| lengtht | Returns the length of a given string in bytes when converted (transformed) to a specified encoding.<br>Syntax:<br>`dst_var = lengtht(source_value, encoding_value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*encoding_value* = text literal, column, variable, or expression.<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`let #sjislen = lengtht($string, 'Shift-JIS')` |
| lower | Converts the contents of *source_value* to lowercase and returns the result.<br>Syntax:<br>`dst_var = lower(source_value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*dst_var* = text variable<br>Example:<br>`let $lower = lower(&description)` |

Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| lpad | Pads the *source_value* on the left to a length of *length_value* using *pad_value* and returns the result.<br>Syntax:<br>`dst_var = lpad(source_value, length_value, pad_⇒ value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*length_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br>*pad_value* = text literal, column, variable, or expression<br>*dst_var* = text variable<br>Example:<br>`let $lpad = lpad($notice, 25, '.')` |
| ltrim | Trims characters in *source_value* from the left until a character is not in *set_value* and returns the result.<br>Syntax:<br>`dst_var = ltrim(source_value, set_value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*set_value* = text literal, column, variable, or expression<br>*dst_var* = text variable<br>Example:<br>`let $ltrim = ltrim(&description, '.')` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| nvl | Returns *y_value* if the *x_value* is null; otherwise, returns *x_value*. If *x_value* is numeric, *y_value* must also be numeric; otherwise, date and textual arguments are compatible. In any case, the *x_value* determines the type of expression returned. The return value of the function depends on which value is returned.<br><br>Syntax:<br><br>`dst_var = nvl(x_value, y_value)`<br><br>*x_value* = Any literal, column, variable, or expression<br><br>*y_value* = Any literal, column, variable, or expression<br><br>*dst_var* = Any variable<br><br>Example:<br>`let $city = nvl(&city, '-- not city --')`<br><br>If *x_value* is a date and *y_value* is textual, *y_value* is validated according to the following rules:<br><br>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 the Default Database Formats table), or the database-independent format 'SYYYYMDD[HH24[MI[SS[NNNNNN]]]]'.<br><br>For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting, or the format listed in the DATE Column Formats table.<br><br>For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format as listed in the TIME Column Formats table. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Function | Explanation |
|---|---|
| range | Returns a value of one (1) if *x_value* is between *y_value* and *z_value*; 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 *x_value* is a date and *y_value* and *z_value* are text arguments. In a comparison of this sort, *y_value* must represent a date that is earlier than that of *z_value*.<br><br>Syntax:<br><br>`dst_var = range(x_value, y_value, z_value)`<br><br>*x_value* = Any literal, column, variable, or expression<br><br>*y_value* = Any literal, column, variable, or expression<br><br>*z_value* = Any literal, column, variable, or expression<br><br>*dst_var* = decimal, float, or integer variable<br><br>Example:<br>`let #inrange = range(&grade, 'A', 'D')⇒`<br>`let #inrange = range($date, $startdate,⇒`<br>`$enddate)⇒`<br>`let #inrange = range($date, $startdate, '15-Apr-⇒`<br>`04')⇒`<br>`let #inrange = range(#price, #low, #high)`<br><br>If *x_value* is a date and *y_value* and/or *z_value* is textual, then *y_value* and/or *z_value* is validated according to the following rules:<br><br>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 the Default Database Formats table), or the database-independent format 'SYYYYM-MDD[HH24[MI[SS[NNNNNN]]]]'.<br><br>For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting, or the format listed in the table DATE Column Formats.<br><br>For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format as listed in table TIME Column Formats. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| replace | Inspects the contents of *source_value* and replaces all occurrences of from_string with to_string and returns the modified string.<br>Syntax:<br>`dst_var = replace(source_value, from_string, to_⇒ string)`<br>*source_value* = date or text literal, column, variable, or expression<br>*from_string* = text literal, column, variable, or expression<br>*to_string* = text literal, column, variable, or expression<br>*dst_var* = text variable<br>`Example:`<br>`let $replaced = replace($paragraph, 'good',⇒ 'excellent')` |
| roman | Returns a string that is the character representation of *source_value* expressed in lowercase roman numerals.<br>Syntax:<br>`dst_var = roman(source_value)`<br>*source_value* = text literal, column, variable, or expression.<br>*dst_var* = text variable<br>Example:<br>`let $roman = roman(#page-count)` |
| rpad | Pads the *source_value* on the right to a length of *length_value* using *pad_value* and returns the result.<br>Syntax:<br>`dst_var = rpad(source_value, length_value, pad_⇒ value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*length_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br>*pad_value* = text literal, column, variable, or expression<br>*dst_var* = text variable<br>Example:<br>`let $rpad = rpad($notice, 25, '.')` |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Function | Explanation |
|---|---|
| rtrim | Trims characters in *source_value* from the right until a character is not in *set_value* and returns the result.<br><br>Syntax:<br><br>`dst_var = rtrim(source_value, set_value)`<br><br>*source_value* = date, or text literal, column, variable, or expression<br><br>*set_value* = text literal, column, variable, or expression<br><br>*dst_var* = text variable<br><br>Example:<br>`let $rtrim = rtrim(&description, '.')` |
| strtodate | Converts the string *source_value* in the format format_mask to a date type.<br><br>Syntax:<br><br>`dst_var = strtodate(source_value [, format_mask])`<br><br>*source_value* = text literal, column, variable, or expression<br><br>*format_mask* = text literal, column, variable, or expression that describes the exact format of the *source_value*. The keyword DATE can be used to specify the DATE-EDIT-MASK setting from the current locale. If this argument is not specified, then *source_value* must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats (see the Default Database Formats table), or the database-independent format 'SYYYYMDD[HH24[MI[SS[NNNNNN]]]]'. Valid format codes are specified in the Date Edit Format Codes table.<br><br>*dst_var* = date variable<br><br>Example: let $date = strtodate($str_date, 'Mon DD, YYYY') let $date = strtodate($str_date, DATE) |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| substr | Extracts the specified portion *source_value*. The extraction begins at *offset_value* (origin is 1) for a length of *length_value* characters.<br><br>Syntax:<br><br>`dst_var = substr(source_value, offset_value, ⇒ length_value)`<br><br>*source_value* = date or text literal, column, variable, or expression.<br><br>*offset_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br><br>*length_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br><br>*dst_var* = text variable.<br><br>Example:<br>`let $piece = substr(&record, 10, #len)`<br><br>**Note.** PeopleSoft recommends that you use either the substrp or substrt function instead of the substr function. |
| substrb | Has the same functionality as the substr function except that the starting point and length are expressed in bytes, rather than in characters.<br><br>Syntax:<br><br>`dst_var = substrb(source_value, offset_value, ⇒ length_value)`<br><br>*source_value* = date or text literal, column, variable, or expression.<br><br>*offset_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br><br>*length_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br><br>*dst_var* = text variable.<br><br>Example:<br>`let $piece = substrb(&record, 10, #len)`<br><br>**Note.** PeopleSoft recommends that you use either the substrp or substrt function instead of the substrb function. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Function | Explanation |
|---|---|
| substrp | Returns a substring of a given string starting at a specified print position into the string and of a specified print length.<br><br>Syntax:<br><br>`dst_var = substrp(source_value, offset_value, ⇒ length_value)`<br><br>*source_value* = date or text literal, column, variable, or expression.<br><br>*offset_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br><br>*length_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br><br>*dst_var* = decimal, float, or integer variable.<br><br>Example:<br>`let $sub = substrp(&string, #printpos, #printlen)` |
| substrt | Returns a substring of a given string starting at a specified byte and byte length in a given encoding.<br><br>Syntax:<br><br>`dst_var = substrt(source_value, offset_value, ⇒ length_value, encoding_value)`<br><br>*source_value* = date or text literal, column, variable, or expression.<br><br>*offset_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br><br>*length_value* = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.<br><br>*encoding_value* = text literal, column, variable, or expression.<br><br>*dst_var* = text variable.<br><br>Example:<br>`let $sjisPrep = substrt(&string, 1, 10, 'Shift-⇒ JIS')` |
| to_char | Converts *source_value* to a string, using maximum precision.<br><br>Syntax:<br><br>`dst_var = to_char(source_value)`<br><br>*source_value* = decimal, float, or integer literal, column, variable, or expression<br><br>*dst_var* = text variable<br><br>Example:<br>`let $string = to_char(#number)` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| to_multi_byte | 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.<br><br>Syntax:<br><br>`dst_var = to_multi_byte (source_value)`<br><br>*source_value* = date or text literal, column, variable, or expression<br><br>Example:<br>`let $multi = to_multi_byte (&text)` |
| to_number | Converts *source_value* to a number. This function returns a float value.<br><br>Syntax:<br><br>`dst_var = to_number(source_value)`<br><br>*source_value* = date or text literal, column, variable, or expression.<br><br>*dst_var* = decimal, float, or integer variable<br><br>Example:<br>`let #value = to_number($number)` |
| to_single_byte | Converts the specified string as follows: For SJIS, EBCDIK290, and EBCDIK1027, any occurrence of a single-byte character that also has a multi-byte representation (numerals, punctuation, roman characters, and katakana) is converted. This function also converts a sequence of kana characters followed by certain grammatical marks into a single-byte character that combines the two elements. For all other encodings, the string is not modified.<br><br>Syntax:<br><br>`dst_var = to_single_byte(source_value)`<br><br>*source_value* = date or text literal, column, variable, or expression<br><br>Example:<br>`let $single = to_single_byte (&text)` |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Function | Explanation |
|---|---|
| translate | Inspects the contents of *source_value* and converts characters that match those in *from_set* to the corresponding character in *to_set* and returns the translated string.<br>Syntax:<br>`dst_var = translate(source_value, from_set, to_⇒ set)`<br>*source_value* = date or text literal, column, variable, or expression<br>*from_set* = text literal, column, variable, or expression<br>*to_set* = text literal, column, variable, or expression<br>*dst_var* = text variable<br>Example:<br>`let $translated = translate(edit(&price,⇒ '999,999.99'),',',',')` |
| transform | Returns a Unicode string which is a specified transform of a given string.<br>Syntax:<br>`dst_var = transform(source_value, transform_value)`<br>*source_value* = date or text literal, column, variable or expression<br>transform_value = text literal, column, variable, or expression<br>*dst_var* = text variable<br>Example:<br>`let $hiragana = transform($string, 'ToHiragana')` |
| unicode | Returns a Unicode string from the string of hexadecimal values provided.<br>Syntax:<br>`dst_var = unicode(source_value)`<br>*source_value* = text literal, column, variable or expression<br>*dst_var* = text variable<br>Example:<br>`let $uniStr = unicode('u+5e73 u+2294')` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Function | Explanation |
|---|---|
| upper | Converts the contents of *source_value* to uppercase and returns the result.<br>Syntax:<br>`dst_var = upper(source_value)`<br>*source_value* = date or text literal, column, variable, or expression<br>*dst_var* = text variable<br>Example:<br>`let $upper = upper(&description)` |
| wrapdepth | Returns the number of print lines required by *source_value*. See the PRINTWRAP command for detailed descriptions of the parameters to this function. This function returns a float value.<br>Syntax:<br>`dst_var = wrapdepth(source_value, wrap_width, ⇒ line_height, on, strip)`<br>*source_value* = text literal, column, variable, or expression<br>*wrap_width* = decimal, float, or integer literal, column, variable, or expression<br>*line_height* = decimal, float, or integer literal, column, variable, or expression<br>*on* = text literal, column, variable, or expression.<br>*strip* = text literal, column, variable, or expression<br>*dst_var* = decimal, float, or integer variable<br>Example:<br>`let #depth = wrapdepth(&description,40,1,'<13>','')` |

## Writing Custom Functions

In addition to using the preceding 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, keep in mind the following considerations:

- String functions require the following arguments:

Copyright © 1988-2007, Oracle. All rights reserved. 

- (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 that was 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 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 uppercase or lowercase.

```
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();
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
return;
}
```

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

This is 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
    .
    .
    .
```

### Example

These 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 your code and saves the result in an internal format so that repetitive execution is at maximum speed.

## LOAD-LOOKUP

### Syntax

In the SETUP section:

Copyright © 1988-2007, Oracle. All rights reserved. 

```
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_name]
[PROCEDURE=procedure_name]
[COMMAND=command_string]
[GETDATA=getdata_string]
[PARAMETERS=(...)]
[FROM-ROWSET=(m,n,-n,m-,all)]
[FROM-PARAMETER=rowset_name]
```

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=schema_name]
[PROCEDURE=procedure_name]
[COMMAND=command_string]
[GETDATA=getdata_string]
[PARAMETERS=(...)]
[FROM-ROWSET=(m,n,-n,m-,all)]
[FROM-PARAMETER=rowset_name]
```

## Description

Loads an internal table with columns from the database. Enables a quick search using the LOOKUP command.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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, a lookup table can be slower because 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 enables 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.

## Parameters

| Parameter | Description |
|---|---|
| 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. |
| 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 with 12 digits or fewer 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. The following example is for ORACLE. See your database manual for the correct syntax.<br>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. |
| SORT | The sorting method to be used. The following values are permitted:<br>DC: Database sorts data, case-sensitive sort<br>DI: Database sorts data, case-insensitive sort<br>SC: SQR sorts data, case-sensitive sort<br>SI: SQR sorts data, case-insensitive sort<br>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. |

Copyright © 1988-2007, Oracle. All rights reserved. 

## Example

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'
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
key=abbr

return_value=name

where='s.abbr = r.abbr and r.location = ''ne'''
```

# LOOKUP

## Syntax

`LOOKUP` *`lookup_table_name`*{*`key_any_lit`*|*`_var`*|*`_col`*}
{*`ret_txt_var`*|*`_date_var`*}

## Description

Searches a lookup table (an array) for a key value and returns the corresponding text string.

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.

## Parameters

| Parameter | Description |
|---|---|
| lookup_table_name | Specifies the lookup table. This table must be previously loaded with a LOAD-LOOKUP command. |
| key_any_lit \| _var \| _col | The key used for the lookup. |
| ret_txt_var \| _date_var | A string variable into which the corresponding value is returned. |

## Example

The following example works in conjunction with the example for the LOAD-LOOKUP command:

```
lookup  states  &state_abbr  $state_name
```

This example searches the states lookup table for a matching *&state_abbr* value; if found, it returns the corresponding state name in *$state_name*. If not found, a null is placed in *$state_name*.

## See Also

The LOAD-LOOKUP command.

# LOWERCASE

## Syntax

`LOWERCASE` *`txt_var`*

## Description

Converts the contents of a text variable to lowercase.

Copyright © 1988-2007, Oracle. All rights reserved. 

### Parameters

| Parameter | Description |
|---|---|
| txt_var | Specifies a text variable to be converted to lowercase. |

### Example

The following example illustrates the LOWERCASE command

```
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 the Miscellaneous Functions table.

## MBTOSBS

### Syntax

MBTOSBS {*txt_var*}

### Description

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

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.

### Parameters

| Parameter | Description |
|---|---|
| txt_var | Specifies the string to be converted. |

### See Also

The TO_SINGLE_BYTE function of the LET command.

## MOVE

### Syntax

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

### 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

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 has not been set, SQR uses the first database-dependent format as listed the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

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

## Parameters

| Parameter | Description |
|---|---|
| src_any_lit \| _var \| _col | Specifies any source column, variable, or literal. Note that 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 the Default Database Formats table, or the database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'. |
| dst_any_var | Specifies a destination variable. |
| format_mask | Specifies an optional format mask. For additional information regarding edit masks, see the PRINT command. |
| 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 ALTER_LOCALE command.) |
| 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 ALTER_LOCALE command.) |
| 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 ALTER_LOCALE command.) |

## Example

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

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

Copyright © 1988-2007, Oracle. All rights reserved. 

Produces the following output:

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

```
 !
 ! 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
```

```
 !
 ! 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
```

```
 !
 ! 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
```

```
 !
 ! 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:

 Copyright © 1988-2007, Oracle. All rights reserved.

```
$NVar =    123,456.78

 !
 ! 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

  !
 ! 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-01-2004

  !
 ! 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 2004:

```
$Date2 = 10/01/04 00:00

  !
 ! 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
```

Copyright © 1988-2007, Oracle. All rights reserved. 

Produces the following output:

```
$Str_Date = 01-DEC-2004
```

```
 !
! Convert string (in partial format of SYYYYMMDDHHMISSNNN) to a
! date
!
move '20041129' to $date1
show '$Date1 = ' $date1 edit 'Mon DD YYYY HH:MI'
```

Produces the following output:

```
$Date1 = Nov 29 2004 00:00
```

### See Also

The LET command for information about 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

### Syntax

MULTIPLY {*src_num_lit*|*_var*|*_col*} TIMES *dst_num_var*
[ROUND=*nn*]

### Description

Multiplies the first field by the second and places the result into the second field.

When dealing with money-related values (dollars and cents), use decimal variables rather than float variables. Float variables are stored as double precision floating point numbers, and small inaccuracies can appear when multiplying many numbers in succession. These inaccuracies can appear due to the way different hardware and software implementations represent floating point numbers.

### Parameters

| Parameter | Description |
|---|---|
| src_num_lit \| _var \| _col | Specifies a numeric source column, variable, or literal. |
| dst_num_var | Specifies a destination numeric variable. |
| ROUND | Rounds the result to the specified number of digits to the right of the decimal point. For float variables, this value can be from 0 to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate. |

### Example

The following example illustrates the MULTIPLY command

 Copyright © 1988-2007, Oracle. All rights reserved.

```
multiply  &quantity  times  #cost
multiply  1.5  times  #result
```

### See Also

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

## NEW-PAGE

### Syntax

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

### Description

Writes the current page and begins a new one.

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.

---

### Parameters

| Parameter | Description |
|---|---|
| erase_from_line_num_lit \| _var \| _col | Specifies a numeric column, variable, or literal for line printers. |

### Example

The following example illustrates the NEW-PAGE command

```
! Write current page, then erase it
! beginning at line 5.
new-page  5
```

## NEW-REPORT

### Syntax

NEW-REPORT {*report_filename_txt_lit*|*_var*|*_col*}

### Description

Closes the current report output file and opens a new one with the specified file name.

Copyright © 1988-2007, Oracle. All rights reserved. 

This command is normally used with single reports only. 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 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.

---

### Parameters

| Parameter | Description |
|---|---|
| {report_filename_txt_lit \| _var \| _col} | Specifies a new file name. |

### Example

The following example shows the NEW-REPORT command:

```
new-report  'rep2a.lis'
new-report  $next-file
```

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

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

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

### See Also

DECLARE-REPORT, USE-REPORT
The -F command-line flag.

## NEXT-COLUMN

### Syntax

```
NEXT-COLUMN [AT-END={NEWLINE|NEWPAGE}]
[GOTO-TOP={num_lit|_var|_col}]
[ERASE-PAGE={num_lit|_var|_col}]
```

### Description

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

 Copyright © 1988-2007, Oracle. All rights reserved.

### Parameters

| Parameter | Description |
|---|---|
| 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. |

### Example

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

COLUMNS, USE-COLUMN

## NEXT-LISTING

### Syntax

NEXT-LISTING[NO-ADVANCE]
[SKIPLINES={*num_lit*|*_var*|*_col*}]
[NEED={*num_lit*|*_var*|*_col*}]

Copyright © 1988-2007, Oracle. All rights reserved.

### Description

Ends the current set of detail lines and begins another.

NEXT-LISTING is used in tabular reports. This command causes a new vertical offset in the page to begin.

After NEXT-LISTING is executed, 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.

The value of SKIPLINES must be a nonnegative integer. If it is less than 0, then 0 is assumed.

The value of NEED must be an integer greater than 0. If it is less than or equal to 0, then 1 is assumed.

### Parameters

| Parameter | Description |
|---|---|
| NO-ADVANCE | Suppresses any line movement when no printing has occurred since the previous NEXT-LISTING or NEW-PAGE. The default increments the line position even when nothing was printed. |
| SKIPLINES | Causes the specified number of lines to be skipped before setting up the new offset. |
| 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. |

### Example

The following example shows the NEXT-LISTING command:

```
begin-select
cust_num  (1,1)      edit 099999          ! Each detail group prints
city   (,+3)                              ! starting on line 1 since
name   (2,10,30)                          ! NEXT-LISTING keeps
address   (,+2)                           ! moving line 1 down the
  next-listing  skiplines=1  need=2       ! page. NEED=2 keeps 2
from customers order by cust_num          ! line detail groups from
end-select                                ! breaking across pages.
```

**Note.** The NEXT-LISTING command automatically issues a Use-Column 1 command if columns are active.

## OPEN

### Syntax

```
OPEN {filename_lit|_var|_col} AS
{filenum_num_lit|_var|_col}
{FOR-READING|FOR-WRITING|FOR-APPEND}
{RECORD=length_num_lit[:FIXED|:FIXED_NOLF|:VARY]}
[STATUS=num_var]
```

 Copyright © 1988-2007, Oracle. All rights reserved.

## Description

Opens an operating system file for reading or writing. 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.

## Parameters

| Parameter | Description |
|---|---|
| {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 runtime. |
| {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 and 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. |

## Example

This section contains two examples: a regular open command and an expanded command:

```
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
```

Copyright © 1988-2007, Oracle. All rights reserved. 

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

An encoding directive added to the OPEN command allows differently encoded files to be managed in a single run of SQR. When no ENCODING is specified on the OPEN command, 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. The complete syntax of the OPEN command is:

```
OPEN {filename_lit | _var | _col} AS {filenum_num_lit |
_var | _col}
{ FOR-READING | FOR-WRITING | FOR-APPEND }
{ RECORD = length_num_lit[:FIXED | :FIXED_NOLF |
:VARY]}
[ STATUS = num_var ]
[ ENCODING = { _var | _col | ASCII | ANSI | SJIS | JEUC
| EBCDIC |
EBCDIK290 | EBCDIK1027 | UCS-2 | UTF-8 | others... }
```

The ENCODING directive is only allowed when converting to Unicode internally.

### See Also

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

## PAGE-NUMBER

### Syntax

PAGE-NUMBER *position*[*pre_txt_lit*[*post_txt_lit*]]

### Description

Places the current page number on the page.

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

### Parameters

| Parameter | Description |
|---|---|
| position | Specifies the position of the page number. See the POSITION command for a description and examples of the *position* parameter. |
| pre_txt_lit | Specifies a text string to be printed before the page number. |
| post_txt_lit | Specifies a text string to be printed after the page number. |

### Example

The following example shows the PAGE-NUMBER command:

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

 Copyright © 1988-2007, Oracle. All rights reserved.

```
end-footing
```

### See Also

LAST-PAGE

## POSITION

### Syntax

```
POSITION position
[@document_marker[COLUMNS{num_lit|_var|_col}
[num_lit|_var|_col]...]]
```

### Description

Sets the current position on a page.

### Parameters

| Parameter | Description |
|---|---|
| position | A position qualifier consisting of *(line, column, width),* where *column* and *width* are numbers that denote characters and spaces. *Line, column, and width* are all optional. If line or column are omitted, the print position is set by default to the current position, which is the position following the last printed item. Width is set by default to the width of the text that is being printed. A plus sign or minus sign indicates relative positioning in SQR. A plus sign moves the print position forward from the current position, and a minus sign moves it back. |
| @document_markerg | 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. |
| COLUMNS | Defines columns beginning at the location of the document marker. The columns defined are relative to the position of the document marker.<br>When COLUMNS is used, the entire command cannot be broken across more than one program line. |

### Example

The following example shows the POSITION command:

```
position  (12,5)          ! Set current position to line 12, column 5.
position  (+2,25)         ! Set position 2 lines down, at 25th column.
position  ()  @total_location              ! Set position to document
print #total  ()  edit  999,999,999        ! marker @total_location.
position  ()  @name_loc  columns 1 30
print name  ()            !  Columns are defined at @name_loc and
next-column               !  29 characters to the right of @name_loc
print title ()
```

### See Also

The COLUMNS command for more information.
*Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, “Creating Form Letters”

Copyright © 1988-2007, Oracle. All rights reserved. 

# PRINT

## Syntax

```
PRINT {any_lit|_var|_col} position[format_command[format_cmd_params]...]...
```

## Description

Puts data on the page at a specified position.

See *Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, “Changing Fonts”.

## Parameters

| Parameter | Description |
|---|---|
| {any_lit \| _var \| _col} | Specifies the data to be printed.<br>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 of *EDIT* for further details. |
| position | A position qualifier consisting of *(line, column, width),* where *column* and *width* are numbers that denote characters and spaces. *Line, column, and width* are all optional. If line or column are omitted, the print position is set by default to the current position, which is the position following the last printed item. Width is set by default to the width of the text that is being printed. Position can be relative. See the POSITION command for a full description and examples of relative positioning. |
| format_command [format_cmd_params] | Specifies optional formatting commands and parameters. |

## Format Commands

The PRINT command has the following format commands:

- BACKGROUND
- BOLD
- BOX
- CENTER
- CODE-PRINTER
- DATE
- EDIT
- FILL
- FOREGROUND
- MONEY
- NOP
- NUMBER
- ON-BREAK
- SHADE

 Copyright © 1988-2007, Oracle. All rights reserved.

- UNDERLINE
- WRAP

Some of these format commands can be used in combination with others and some are mutually exclusive. The following tables shows which can be used together. An X indicates that they can be used together.

Copyright © 1988-2007, Oracle. All rights reserved.

| | BACKGROUND/ FOREGROUND | BOLD | BOX | CENTER | CODE-PRINTER | DATE/ EDIT/ MONEY/ NUMBER |
|---|---|---|---|---|---|---|
| BACKGROUND/ FOREGROUND | | X | | X | | X |
| BOLD | X | | X | X | | X |
| BOX | X | X | | X | | X |
| CENTER | X | X | X | | | X |
| CODE-PRINTER | | | | | | |
| DATE/ EDIT/ MONEY/ NUMBER | X | X | X | X | | |
| FILL | X | X | X | X | | |
| MATCH | X | X | X | X | | |
| NOP | X | X | X | X | X | X |
| ON-BREAK | X | X | X | X | | X |
| SHADE | X | X | X | X | | X |
| UNDERLINE | X | X | | X | | X |
| WRAP | X | X | X | | | |


Copyright © 1988-2007, Oracle. All rights reserved.

| | FILL | MATCH | NOP | ON-BREAK | SHADE | UNDERLINE | WRAP |
|---|---|---|---|---|---|---|---|
| BACKGROUND/ FOREGROUND | X | X | X | X | X | | X |
| BOLD | X | X | X | X | X | X | X |
| BOX | X | X | X | X | X | | X |
| CENTER | X | X | X | X | X | X | |
| CODE- PRINTER | | | X | | | | |
| DATE/ EDIT/ MONEY/ NUMBER | | | X | X | X | X | |
| FILL | | | X | X | X | X | X |
| MATCH | | | X | X | X | X | |
| NOP | X | X | | X | X | X | X |
| ON- BREAK | X | X | X | | X | X | X |
| SHADE | X | X | X | X | | X | X |
| UNDERLINE | X | X | X | X | X | | X |
| WRAP | X | | X | X | X | X | |

Copyright © 1988-2007, Oracle. All rights reserved.

The following sections describe the PRINT format commands.

## BOLD

BOLD causes the string or number to print in **bold** type.

For HP LaserJet printers, 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 the DECLARE-PRINTER Command Arguments table for information about which font can be boldfaced.

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 boldfaced. If BEFORE-BOLD and AFTER-BOLD are not specified, BOLD has no effect.

For example:

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

## BOX

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 LaserJet printers using proportional fonts, BOX and SHADE are not able to determine the correct length of the box because it varies with the width of the characters printed. BOX and SHADE work well with fixed-pitch fonts and with all PostScript fonts.

---

## CENTER

CENTER centers the field on a line. The position qualifier for column is ignored. For example:

```
print  'Quarterly Sales'  (1)  center
```

```
CODE-PRINTER
```

CODE-PRINTER has the following syntax:

```
CODE-PRINTER = printer_type
```

Valid values for *printer_type* are HT, HP, PS, LP, HTML, HPLASERJET, POSTSCRIPT, and LINEPRINTER.

```
CODE-PRINTER
```

Adds nondisplay characters to the program for the purpose of sending a sequence to the printer. CODE-PRINTER causes the string to be placed *behind* the page buffer, rather than within it, so alignment of printed data is not affected by the white space consumed by the nondisplay characters. Only strings can be printed using CODE-PRINTER.

Because 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:

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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

You cannot use DATE with numeric columns or variables. DATE indicates that the field is to be formatted using the DATE-EDIT-MASK from the current locale. (See the ALTER_LOCALE command.) If this entry is not set, then the date is printed according to the following rules shown in the Date table.

| Column Type | Default Mask | If not set |
|---|---|---|
| DATETIME | SQR_DB_DATE_FORMAT | See the Default Database Formats table for the format that is used. |
| DATE | SQR_DB_DATE_ONLY_FORMAT | See the DATE Column Formats table for the format that is used. |
| TIME | SQR_DB_TIME_ONLY_FORMAT | See the TIME Column Formats table for the format that is used. |

## EDIT

EDIT has the following syntax:

```
EDIT  edit_format
```

EDIT causes each field to be edited before it is printed. The three types of edits are:

- Text edit
- Numeric edit
- Date edit

The following table lists the text edit format characters.

Copyright © 1988-2007, Oracle. All rights reserved. 

| 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, such as 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.

The following table lists the 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 uppercase or lowercase; the display follows the case of the mask. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Character | Description |
|---|---|
| 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 be displayed around the number if the number is negative. |
| PS | Entered at the end of the mask, causes parentheses to be displayed around the number if the number is negative. |
| PF | Entered at the end of the mask, causes floating parentheses to be displayed around the number if the number is negative. |
| NA | Entered at the end of the mask, causes N/A to be displayed 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 be displayed if the numeric column variable is null. |
| . | Decimal point. |
| , | Comma. |

Characters other than those listed in the Numeric Edit Format Characters table are illegal for numeric edit masks and cause errors during processing.

The following table shows sample edit masks and resulting fields.

| Mask | Value | Display |
|---|---|---|
| 999.99 | 34.568 | 34.57 |
| 9,999,999V9999 | 123,456.7890 | 123,4567890 |
| 8,888,888.888 | 123,456.789 | 123,456.789 |
| 9,999 | 1234 | 1,234 |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Mask | Value | Display |
|---|---|---|
| 9,999 | 123 | 123 |
| 09999 | 1234 | 01234 |
| 9999 | -123 | -123 |
| 9999 | -1234 | **** |
| 9999 | 12345 | **** |
| 9999mi | -123 | 123- |
| 9999pr | -123 | < 123> |
| 999999ps | -123 | ( 123) |
| 999999pf | -123 | (123) |
| 9999na | (null) | n/a |
| 9999nu | (null) | (blank) |
| $$9,999.99c | 1234.56 | $1.234,56 |
| $$9,999.99 | 1234.56 | $1,234.56 |
| $$9,999.99 | 12.34 | $ 12.34 |
| $$$,$$9.99 | 12.34 | $12.34 |
| 9.999e | 123456 | 1.235e+05 |
| B9,999 | 0 | (blank) |
| B9,999 | 12345 | 12,345 |
| (xxx)bxxx-xxxx | 2169910551 | (216) 991-0551 |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Mask | Value | Display |
|---|---|---|
| xxx-xx-xxxx | 123456789 | 123-45-6789 |
| ~~xx~xx | ABCDEFGHIJ | CDFG |
| r10 | ABCDEFG | GFEDCBA |

The following example shows some uses of edit masks:

```
print #total (7,55,0)  edit $999,999.99          ! $ 12,345.67
print #total (7,55,0)  edit $$$9,999.99          ! $12,345.67
print #total (7,55,0)  edit 999,999.99pr         ! < 12,345.67>(if neg)
print #comm  (7,55,0)  edit b99,999.99           ! Blank if zero
print &cnum  (16,1,0)  edit 099999               ! 001234
print #cat   (5,10,0)  edit 9.999E               ! 1.235E+04
print #phone (16,60,0) edit (xxx)bxxx-xxxx       ! (216) 397-0551
print #total (7,55,0)  edit £££9,999.99          ! Dollar-Symbol £
```

The following table lists the date edit format codes:

| Character | Description |
|---|---|
| YYY YY Y | Last 3, 2, or 1 digits of the year. On input, for calculating the 4-digit year, the current century and/or decade is used. For example, a 9 using the Y mask would result in the year 1999 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 the Date Edit Format Code-RR table. |
| 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. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Character | Description |
|---|---|
| W | Week of the month (1–5) where week 1 starts on the first day of the month and ends on the seventh. |
| DDD | Day of year (1–366). |
| DD | Day of month (1–31). |
| D | Day of week (1–7). Sunday is first day of week. |
| DAY | Name of day. |
| DY | Abbreviated name of day. |
| ER | Japanese Imperial Era. Returns the name of the of the Japanese Imperial Era in the appropriate kanji (Heisei is the current era). |
| EY | Year of the Japanese Imperial Era. Returns the current year within the Japanese Imperial Era.<br>**Note.** The common Japanese date format is: ’YYYY<nen>MM<gatsu>DD<nichi>’ where <nen>, <gatsu>, and <nichi> are the kanji strings for year, month, and day respectively. |
| J | Julian day; the number of days since Jan 1, 4713 BC. Numbers specified with ’J’ must be integers. |
| AM PM | Meridian indicator. |
| HH | Assumes 24-hour clock unless meridian indicator specified. |
| HH12 | Hour of day (1–12). |
| HH24 | Hour of day (0–23). |
| SSSSS | Seconds past midnight (0–86399). |
| N NN NNN NNNN NNNNN NNNNNN | Fractions of a second. Precise to microseconds; however, for most hardware and databases, this much accuracy is not attainable. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Character | Description |
|---|---|
| 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. |

**Note.** If the last two digits of the year are between 00 and 49, the return date is in the current century. If the last two digits of the year are between 50 and 99, the return date is in the century after the current one.

| 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. |

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\rre\nt \mo\nth is Month* results in the output string of *The current month is January.* Without the backslashes the output string would be *The cu04e7t january is January.*

A vertical bar can be used 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, such as punctuation, in a date edit mask is treated as a constant and is included in the edited field. If the edit mask contains spaces, it must be enclosed 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. In addition, National Language Support is provided for the following masks: MON, MONTH, DAY, DY, AM, PM, BC, and AD.

See the ALTER-LOCALE command in the SQR Samples section for additional information.

If the value of the date field being edited is "Mar 14 2004 9:35", the edit masks produce the results in the following table.

Copyright © 1988-2007, Oracle. All rights reserved. 

| Edit Mask | Result |
|---|---|
| dd/mm/yy | 14/03/04 |
| DD-MON- YYYY | 14-MAR-2004 |
| ’Month dd, YYYY’ | March 14, 2004 |
| MONTH-YYYY | MARCH-2004 |
| HH:MI | 09:35 |
| ’HH:MI PM’ | 09:35 AM |
| YYYYMMDD | 20040314 |
| MM.DD.YYYY | 03.14.2004 |
| Mon | Mar |
| DD\|D\|DDD | 143073 |

In addition to being used with the EDIT argument, edit masks can be used 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 is displayed or printed, 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, 2004, the following assignment would produce an equivalent date-time of September 1, 2004 13:21:00.000000:

```
let $date1 = strtodate('13:21','HH:MI')
```

Edit masks can be changed dynamically 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 that is stored in a string literal, column, or variable is printed with an edit mask, it must be in one of the following formats:

 Copyright © 1988-2007, Oracle. All rights reserved.

- The format specified by the environment variable SQR_DB_DATE_FORMAT, or the corresponding setting in the sqr.ini file.
- One of the database-dependent formats, as listed in the Default Database Formats table.
- The database-independent format, ’SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]’.

When a date column or variable is printed without an edit mask, the date is printed 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 is printed in the primary database format (the first entry) listed in the Default Database Formats table.

This applies to DISPLAY, MOVE, and SHOW commands as well as PRINT.

The following table lists default date formats for each database.

| Database | Default Database Formats |
|---|---|
| Oracle | DD-MON-YY |
| Informix | YYYY-MM-DD HH:MI:SS.NNN MM/DD/YYYY<br>MM-DD-YYYY MM.DD.YYYY |
| ODBC | MM-DD-YYYY |
| DB2 | YYYY-MM-DD- HH:MI:SS.NNNNNN<br>YYYY-MM-DD |
| Sybase | MON DD YYYY HH:MIPM MON DD YYYY<br>[HH:MI[:SS[:NNN]][PM] MON DD YYYY<br>[HH:MI[:SS[.NNN]][PM] YYYYMMDD<br>[HH:MI[:SS[:NNN]]PM] YYYYMMDD<br>[HH:MI[:SS[.NNN]]PM] |

| Database | DATE Column Formats |
|---|---|
| DB2 | YYYY-MM-DD |
| Informix | MM/DD/YYYY |
| ODBC | DD-MON-YYYY |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Database | TIME Column Formats |
|---|---|
| DB2 | HH24.MI.SS |
| ODBC | HH24:MI:SS |

## FILL

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.
```

## 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* is used. Use the color name *none* to turn off color for the specified area.

```
PRINT {any_lit|_var|_col}
[FOREGROUND =({color_name_lit|_var|_col}|{rgb})]
[BACKGROUND =({color_name_lit|_var|_col}|{rgb})]
```

## MATCH

MATCH has the following syntax:

```
MATCH  match_text  { line_num_lit|_var|_col }
{ column_num_lit|_var|_col }  print_text ...
```

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. 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'
```

 Copyright © 1988-2007, Oracle. All rights reserved.

The column positions are treated as fixed locations due to the fixed 25 position declared in parentheses.

### MONEY

MONEY indicates that the column or variable is to be formatted using the MONEY-EDIT-MASK from the current locale. (See the ALTER_LOCALE command.) This can only be used with a numeric column or variable.

### NOP

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

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

ON-BREAK has the following 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 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). 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 qualifier.
- AFTER and BEFORE procedures specify procedures to invoke either after or before the value changes.

Copyright © 1988-2007, Oracle. All rights reserved. 

If no rows are fetched, neither procedure is executed. AFTER and BEFORE can be used only within a SELECT paragraph.

- The sequence of events is shown here:
  - 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.

Following is the sequence of events for a query containing ON-BREAK fields:

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:
   - AFTER procedures are processed in descending sequence from the highest level to the level of the current break field.
   - SAVE variables are set with the new value.
   - BEFORE procedures are processed in ascending sequence from the current level to the highest level break.
   - Any breaks with the same or higher level numbers are cleared so they do not break on the next value.
   - If a PROCEDURE has been declared, the procedure is invoked.
   - If SKIPLINES was specified, the current line position is advanced.
   - The value is printed (unless PRINT=NEVER was specified).
3. After the query finishes (at END-SELECT) any AFTER procedures are processed in descending level sequence, for example:

```
begin-select
state  (+1,1,2)  on-break level=1  after=state-tot  skiplines=2
county (,+2,14)  on-break level=2  after=cnty-tot  skiplines=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 cnty-tot procedure is executed.

 Copyright © 1988-2007, Oracle. All rights reserved.

- When state breaks, the city-tot, cnty-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, this sequence of procedures is executed:

- City-tot
- Cnty-tot
- State-tot
- Bef-state
- Bef-cnty
- Bef-city

Upon entering the query at BEGIN-SELECT, the three BEFORE procedures are executed in sequence:

- Bef-state
- Bef-cnty
- Bef-city

After the last row is retrieved, at END-SELECT, the three AFTER procedures are executed in sequence:

- City-tot
- Cnty-tot
- 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 enables you to have sub-reports with leveled breaks. By separating the ON-BREAKs into sets, the associated leveled breaks in each set do 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.

## 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
```

Copyright © 1988-2007, Oracle. All rights reserved. 

**Note.** For HP LaserJet printers using proportional fonts, BOX and SHADE are not able to determine the correct length of the box because it varies with the width of the characters printed. BOX and SHADE work well with fixed pitch fonts and with all PostScript fonts.

## UNDERLINE

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
```

## WRAP

WRAP wraps text at word spaces. Additional text is moved to a new line. WRAP has the following 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.

In this example, the line position is 1 for each of the three wrapped fields:

```
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:

 Copyright © 1988-2007, Oracle. All rights reserved.

```
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 30 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 decimal values are surrounded by angled brackets, <nn>, where *nn* is a decimal number between 1 and 255, representing the character in the current character set of the operating system. 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 about 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.
DISPLAY, SHOW

Copyright © 1988-2007, Oracle. All rights reserved. 

# PRINT-BAR-CODE

## Syntax

```
PRINT-BAR-CODE position
{TYPE={bar_code_type_num_lit|_var|_col}}
{HEIGHT={bar_code_height_num_lit|_var|_col}}
{TEXT={bar_code_txt_lit|_var|_col}}
[CAPTION={bar_code_caption_txt_lit|_var|_col}]
[CHECKSUM={checksum_lit}]
```

## Description

Prints industry standard bar codes. SQR supports the bar code types listed in the following table.

| 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 30 | 9, X, p | y |
| 6 | Extended 3 of 9 | 1 to 30 | 9, X, x, p, c | y |
| 7 | Interleaved 2 of 5 | 2 to 30 | 9 | y |
| 8 | Code 128 | 1 to 30 | 9, X, x, p, c | |
| 9 | Codabar | 1 to 30 | 9 | y |
| 10 | Zip+4 Postnet | 5, 9, or 11 | 9 | |
| 11 | MSI Plessey | 1 to 30 | 9 | y |
| 12 | Code 93 | 1 to 30 | 9, X, p | y |
| 13 | Extended 93 | 1 to 30 | 9, X, x, p | y |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Type | Description | Text Length | Text Type | CHECKSUM RECOGNIZED |
|---|---|---|---|---|
| 14 | UCC-128 | 19 | 9 | |
| 15 | HIBC | 1 to 30 | 9 | y |

## Parameters

| Parameter | Description |
|---|---|
| position | 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.) |
| TYPE | Specifies the type of bar code to be printed. Types are shown in the Bar Code Types table. |
| HEIGHT | 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. |
| TEXT | 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 the Bar Code Types table for specifications. |
| CAPTION | 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.<br>CAPTION is not valid for Zip+4 Postnet. If specified, it is ignored. |
| 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.<br><br>**Note.** Some barcode types ignore the CHECKSUM qualifier. See the Bar Code Types table for those barcode types for which CHECKSUM is relevant. |

## Example

This example shows how to use the PRINT-BAR-CODE command to create a UPC-A bar code:

```
begin-program
   print 'Sample UPC-A Barcode' (1,1)
   print-bar-code (3,1)
      type=1                  ! UPC-A
      height=0.3
      text='01234567890'
      caption='0 12345 67890'
end-program
```

This example shows how to use the PRINT-BAR-CODE command to create a ZIP+4 Postnet code:

Copyright © 1988-2007, Oracle. All rights reserved. 

```
begin-program
   print 'Sample Zip+4 Postnet'       (1,1)
   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
```

**Note.** SQR does not check bar code syntax. See your bar code documentation for the proper formatting of certain bar codes.

# PRINT-CHART

## Syntax

```
PRINT-CHART[chart_name]position
DATA-ARRAY-ROW-COUNT={x_num_lit|_var|_col}
DATA-ARRAY-COLUMN-COUNT={x_num_lit|_var|_col}
DATA-ARRAY=array_name
[DATA-LABELS=data_labels_lit | _var | _col]
[COLOR-PALETTE=color_palette_lit | _var | _col]]
[ITEM-COLOR=(Chart_item_keyword, txt_lit |var | (r,g,b))]
[DATA-ARRAY-COLUMN-LABELS={NONE|array_name|
{({txt_lit|_var}[,{txt_lit|_var}]...)}}]
[CHART-SIZE=(chart_width_num_lit|_var,
chart_depth_num_lit|_var)]
[TITLE={title_txt_lit|_var|_col}]
[SUB-TITLE={subtitle_txt_lit|_var|_col}]
[FILL={fill_lit|txt_var|_col}]
[3D-EFFECTS={3d_effects_lit|txt_var|_col}]
[BORDER={border_lit|txt_var|_col}]
[POINT-MARKERS={point_markers_lit|txt_var|_col}]
[TYPE={chart_type_lit|txt_var|_col}]
[LEGEND={legend_lit|txt_var|_col}]
[LEGEND-TITLE={legend_title_txt_lit|_var|_col}]
[LEGEND-PLACEMENT={legend_placement_lit|txt_var|_col}]
[LEGEND- PRESENTATION={legend_presentation_lit|txt_var|_col}]
[PIE-SEGMENT-QUANTITY-DISPLAY=    {pie_segment_quantity_display_lit|txt_var|_col}]
[PIE-SEGMENT-PERCENT- DISPLAY={pie_segment_percent_display_lit
|txt_var|_col}]
[PIE-SEGMENT-EXPLODE={pie_segment_explode_lit
|txt_var|_col}]
[X-AXIS-LABEL={x_axis_label_txt_lit|_var|_col}]
[X-AXIS-MIN-VALUE={x_axis_min_value_lit|_num_lit|_var
|_col}]
[X-AXIS-MAX-VALUE={x_axis_max_value_lit|_num_lit|_var
 |_col}]
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
[X-AXIS-SCALE={x_axis_scale_lit|txt_var|_col}]
[X-AXIS-MAJOR-TICK-MARKS={x_axis_major_tick_marks_lit
|txt_var|_col}]
[X-AXIS-MINOR-TICK-MARKS={x_axis_minor_tick_marks_lit
|txt_var|_col}]
[X-AXIS-MAJOR-INCREMENT={x_axis_major_increment_lit
|_num_lit|_var|_col}]
[X-AXIS-MINOR-INCREMENT={x_axis_minor_increment_lit
|_num_lit|_var|_col}]
X-AXIS-TICK-MARK-PLACEMENT=
{x_axis_tick_mark_placement_lit|txt_var|_col}]
[X-AXIS-GRID={x_axis_grid_lit|txt_var|_col}]
[Y-AXIS-LABEL={y_axis_label_lit|txt_var|_col}]
[Y-AXIS-MIN-VALUE={y_axis_min_value_lit|_num_lit
|_var|_col}]
[Y-AXIS-MAX-VALUE={y_axis_max_value_lit|_num_lit
|_var|_col}]
[Y-AXIS-SCALE={y_axis_scale_lit|txt_var|_col}]
[Y-AXIS-MAJOR-TICK-MARKS={y_axis_major_tick_marks_lit
|txt_var|_col}]
[Y-AXIS-MINOR-TICK-MARKS={y_axis_minor_tick_marks_lit
|txt_var|_col}]
[Y-AXIS-MAJOR-INCREMENT={y_axis_major_increment_lit
|_num_lit|_var|_col}]
[Y-AXIS-MINOR-INCREMENT={y_axis_minor_increment_lit
|_num_lit|_var|_col}]
[Y-AXIS-TICK-MARK-PLACEMENT=
   {y_axis_tick_mark_placement_lit|txt_var|_col}]
[Y-AXIS-GRID={y_axis_grid_lit|txt_var|_col}]
```

**Note.** If you do not define CHART-SIZE with this command, you must define it with DECLARE-CHART.

## Description

Prints a chart. Only PostScript printers or HP printers that support HPGL (generally, this is HPLaserJet 3 and higher) render chart output.

The PRINT-CHART command directs SQR to output a chart according to the named chart, if any, and the overridden attributes, if any.

Copyright © 1988-2007, Oracle. All rights reserved. 

**Note.** 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 the Chart Array Field Types (fewer than four fields) table 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. Do not be alarmed if the chart does not fit exactly inside the box you have specified. It 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 might become part of your output, leaving one or more lines of meaningless data across your report.

All the attributes defined for DECLARE-CHART are valid for the PRINT-CHART command. PRINT-CHART has five additional parameters. The position of the chart is described using the first parameter. The data that supports the chart is defined in the additional attributes: DATA-ARRAY, DATA-ARRAY-ROW-COUNT, DATA-ARRAY-COLUMN-COUNT, and DATA-ARRAY-COLUMN-LABELS.

As mentioned, each chart type meets a specific organizational requirement. The Chart Array Field Types (fewer than four fields) table describes these requirements.

**Note.** 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.

**Note.** 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.

| Chart Type | Field 0 | Field 1 | Field 2 | Field 3 |
|---|---|---|---|---|
| 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*' | |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Chart Type | Field 0 | Field 1 | Field 2 | Field 3 |
|---|---|---|---|---|
| LINE BAR STACKED-BAR 100%-BAR OVERLAPPED-BAR HISTOGRAM AREA STACKED-AREA 100%-AREA | Type=char X-Axis values | Type=num Series 1 Y-Axis values | (Optional) Type=num Series 2 Y-Axis values | (Optional) Type=num Series 3… Y-Axis values |
| XY-SCATTER-PLOT | Type=num Series 1 X-Axis values | Type=num Series 1 Y-Axis values | (Optional) Type=num Series 2 X-Axis values | (Optional) Type=num Series 2 ... Y-Axis values |
| FLOATING-BAR | Type=char X-Axis values | Type=num Series 1 Y-Axis offset | Type=num Series 1 Y-Axis duration | (Optional) Type=Num Series 2 ... Y-Axis offset |

| 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 |

Copyright © 1988-2007, Oracle. All rights reserved.

## Parameters

| Parameter | Description |
|---|---|
| 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.) |
| DATA-ARRAY | Specifies the name of the array containing the data to be plotted. This must be the name of an array defined with CREATE-ARRAY. |
| DATA-ARRAY-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. |
| DATA-ARRAY-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. |
| DATA-ARRAY-COLUMN-LABELS | 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.<br>See the Chart Array Field Types (fewer than four fields) table for applicable fields for each type of chart.<br>For definitions of the other arguments, see the DECLARE-CHART Command Arguments table. |

## Example

In this example, a pie chart is printed without explicit reference to a chart declared with DECLARE-CHART. All necessary arguments must be supplied in PRINT-CHART.

```
.
.
create-array
      name=unit_sales
      size=12
      field=product:char
      field=units:number
      field=explode:char
.
.
print-chart (15, 20)
   title                    = 'Green City Store Sales'
   sub-title                = '(Second Quarter)'
   chart-size               = (50, 28)
   type                     = pie
   data-array               = unit_sales
   data-array-column-count  = 3
   data-array-row-count     = 7
   3d-effects               = yes
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
fill                    = color
```

### See Also

DECLARE-CHART

## PRINT-DIRECT

### Syntax

```
PRINT-DIRECT
[NOLF]
[PRINTER={LINEPRINTER|POSTSCRIPT|HPLASERJET|HTML|LP|PS|HP|HT}]
{txt_lit|_var|_col}...
```

### Description

Writes directly to the print output file without using the SQR page buffer.

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. Because this text is often printer-dependent and because 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.

### Parameters

| Parameter | Description |
|---|---|
| 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. |

### Example

The following example shows the PRINT-DIRECT command:

```
print-direct  printer=ps  '%%Page: '  $page-number
print-direct  nolf  printer=lp  reset
```

## PRINT-IMAGE

### Syntax

```
PRINT-IMAGE[image_name]position
[TYPE={image_type_lit|_var|_col}]
[IMAGE-SIZE=(width_num_lit|_var|_col,height_num_lit
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
|_var|_col)]
[SOURCE={file_name_txt_lit|_var|_col}]
```

**Note.** If TYPE, IMAGE-SIZE, and SOURCE are not defined in PRINT-IMAGE, they must be defined in DECLARE-IMAGE.

## Description

Prints an image.

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 may be used without referencing a named image if all required attributes for the DECLARE-IMAGE are supplied in addition to all its required parameters.

## Parameters

| Parameter | Description |
|---|---|
| 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 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, or BMP-FILE (for Windows). |
| IMAGE-SIZE | Specifies the width and height of the image. |
| SOURCE | Specifies the name of a file containing the image. |

## Example

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)
```

## See Also

DECLARE-IMAGE

 Copyright © 1988-2007, Oracle. All rights reserved.

# PUT

## Syntax

```
PUT {src_any_lit|_var|_col}...
INTO dst_array_name(element)[field[(occurs)]]...
```

## Description

Moves data into an array.

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.

## Considerations using PUT

When a date variable or column is moved into a text or char array field, the date is converted to a string according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.

  If this has not been set, SQR uses the first database-dependent format as listed the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

When a string variable, column, or literal is moved to a date array field, the string must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats as listed in the DATE Column Formats table is used, or the database-independent format ’SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]’.

```
dst_array_name(element)
```

If array fields are listed, data is placed into each field in the sequence it is listed, in the occurrence specified of that field.

If array fields are not listed, data is placed into consecutive fields in the order in which 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.

Copyright © 1988-2007, Oracle. All rights reserved. 

### Parameters

| Parameter | Description |
|---|---|
| src_any_var | 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 char, text, or date fields. Date variables can be put into date, char, or text fields. |

### Example

In the following example, the four variables *&name*, *#count*, *$date1*, and *$code* are 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

### Syntax

READ {*filenum_lit*|*_var*|*_col*} INTO {*any_var*:*length_int_lit*}...[STATUS=*status_num_var*]

### Description

Reads the next record of a file into the specified variables.

Text and binary data is parsed according to the following criteria:

- 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 the DATE Column Formats table is used, or the database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]'.
- Binary numbers, may be 1, 2, or 4 bytes in length.

  They must be read into numeric variables. Note that the bytes making up the binary number must be in the standard sequence expected by your operating system.
- When reading binary data the file must be opened with the FIXED or FIXED-NOLF qualifier.
- Only the integer portion of the number is represented with binary numbers.

  To maintain the decimal portion of the number convert the number to a string variable.
- If you use binary numbers, the file is not portable across platforms.

  Different hardware represents binary numbers differently.

 Copyright © 1988-2007, Oracle. All rights reserved.

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.

## Parameters

| Parameter | Description |
|---|---|
| filenum_lit\|_var \| _col | Specifies the number assigned in the OPEN command to the file to be read. |
| any_var :length_int_lit | Specifies one or more variables into which data from the record read are to be put. *length_int_lit* specifies the length of each field of data. |
| STATUS | Specifies an optional variable into which a read status is returned. |

## Example

The following example shows several READ commands:

```
read  1  into  $name:30  $addr:30  $city:20  $state:2  $zip:5
read  3  into  $type:2  #amount:2  #rate:1  $code:5  $date:11
read  #j into  #sequence:2  $name:20  $title:15
```

The following example shows a READ command that reads two dates. One is loaded into a date variable; the other is loaded into a string variable, which is then converted to a date using the strtodate function.

```
.
.
.
declare-variable
 date $date1 $date2
 text $text
end-declare
.
.
.
read 4 into $date1:18 $text1:18
let $date2 = strtodate($text1,'SYYYYMMDDHHMISSNNN')
         or
let $date2 = strtodate($text1)
```

The following example shows a READ command with an INSERT loop:

```
begin-sql
 begin transaction
end-sql

while  1    ! Infinite loop, exited by BREAK, below.
 read  10  into  $company:40  $parent:30  $location:50
 if #end-file
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
  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 information about files.

## ROLLBACK

### Syntax

ROLLBACK

### Description

An automatic rollback is performed whenever SQR ends 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 Oracle, DB2 , Informix, and ODBC(Microsoft SQL Server is accessible only with SQR Server for ODBC). For Sybase and Microsoft SQL Server , use BEGIN TRANSACTION and ROLLBACK TRANSACTION within SQL paragraphs as in the following example. See the COMMIT command for an example of ROLLBACK.

---

### Example

The following example shows the ROLLBACK command:

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

### See Also

The COMMIT command.

 Copyright © 1988-2007, Oracle. All rights reserved.

# SBTOMBS

## Syntax

```
SBTOMBS { txt_var }
```

## Description

Converts a single-byte character into a multibyte equivalent.

This command converts the specified string as follows: Any occurrence of a single-byte character that also has a multibyte 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 multibyte character, which combines the two elements.

## Parameters

| Parameter | Description |
|---|---|
| txt_var | Specifies the string to be converted. |

## See Also

The TO_MULTI_BYTE function of the LET command.

# SECURITY

## Syntax

```
SECURITY
[SET=(sid [,sid]...)]
[APPEND=(sid [,sid]...)]
[REMOVE=(sid [,sid]...)]
[MODE=mode]
```

## Description

Enables you to mark sections of a report for security purposes.

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 *sids* until the report ends or another SECURITY command executes.

You can use the SECURITY command wherever you use the PRINT command.

Copyright © 1988-2007, Oracle. All rights reserved. 

### Parameters

| Parameter | Description |
|---|---|
| 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. |
| *sid* | 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. |
| *mode* | Can be any string literal, column, or variable. The value is not case-sensitive and can be either ON or OFF. |

### Example

The following example shows the SECURITY command:

```
Begin-Report
   Security Set=('Directors', 'Vice-Presidents')
      .
      .         ! Only Directors and VPS can see this
      .
   Security 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 Append=('Engineers')
      .
      .         ! Only VPs, Managers, and Engineers can see this
      .
End-report
```

## SET-COLOR

### Syntax

```
SET-COLOR
   [PRINT-TEXT-FOREGROUND=({color_name_lit|_var|_col|{rgb})]
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
[PRINT-TEXT-BACKGROUND=({color_name_lit|_var|_col|{rgb})]
```

## Description

Defines default colors.

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.

## Parameters

| Parameter | Description |
|---|---|
| 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. |
| {*color_name_lit* \| *_var* \| *_col*} | A *color_name* is composed of alphanumeric characters (A-Z, 0-9), the underscore (_) character, and the hyphen (-) character. It must start with an alphabetical (A-Z) character and 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.<br>The default colors implicitly installed with SQR include:<br>black=(0,0,0)<br>white=(255,255,255)<br>gray=(128,128,128)<br>silver=(192,192,192)<br>red=(255,0,0)<br>green=(0,255,0)<br>blue=(0,0,255)<br>yellow=(255,255,0)<br>purple=(128,0,128)<br>olive=(128,128,0)<br>navy=(0,0,128)<br>aqua=(0,255,255)<br>lime=(0,128,0)<br>maroon=(128,0,0)<br>teal=(0,128,128)<br>fuchsia=(255,0,255) |

## Example

The following example shows the SET-COLOR command:

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

Copyright © 1988-2007, Oracle. All rights reserved. 

```
begin-program
   alter-color-map name = 'light_blue' value = (193, 233, 230)

   print 'Yellow Submarine' ()
      foreground = ('yellow')
      background = ('light_blue')

   get-color print-text-foreground = ($print-foreground)
   set-color print-text-foreground = ('purple')
   print 'Barney' (+1,1)
   set-color print-text-foreground = ($print-foreground)
end-program
```

### See Also

DECLARE-COLOR-MAP, ALTER-COLOR-MAP, GET-COLOR

## SET-GENERATIONS

### Syntax

`SET-GENERATIONS=(`*`dimension, hierarchy, dimension, hierarchy, dimension, hierarchy,...,...`*`)`

### Description

Specifies dimension hierarchy for the previously declared dimension.

Returns the set of members in the dimension 'product' that are at the 5th generation in the dimension's hierarchy. (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 dimension 'time' that are at the 1st generation deep into the dimension. (Returns all 'Year' members (Generation Level 1) under the time hierarchy of '2004.Q1.2'. This reduces result set to '2004'.)

### Example

The following example shows the SET-GENERATIONS command:

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

## SET-LEVELS

### Syntax

`Set-levels=(`*`dimension, level, dimension, level,...,...`*`)`

### Description

Extends the dimension hierarchy for the previously declared dimension.

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.'

 Copyright © 1988-2007, Oracle. All rights reserved.

### Example

The following example shows the SET-LEVELS command:

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

## SET-MEMBERS

### Syntax

set-members=(*dimension, hierarchy, dimension, hierarchy*,..., ...)

### Description

Returns the set of members in a dimension, level, or hierarchy whose name is specified by a string.

### Example

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 '2004', at the specific level of 'Q1', at the specific level of '2'.

```
set-members=('product','all products.drink.alcoholic beverages.beer⇒
 and wine','time','2004.Q1.2')
```

## SHOW

### 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]...

### Description

Displays one or more variables or literals on the screen. In addition, cursor control is supported for ANSI terminals.

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.

Copyright © 1988-2007, Oracle. All rights reserved. 

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 has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the Default Database Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

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 the Default Database Formats table, or the database-independent format ’SYYYYMMDD[HH24[MI[SS[NNNNNN]]]].

 Copyright © 1988-2007, Oracle. All rights reserved.

## Parameters

| Parameter | Description |
|---|---|
| cursor_position | Specifies the position on the screen to begin the display. |
| {CLEAR-SCREEN \| CS} | Clears the screen and sets the cursor position to (1,1). |
| {CLEAR-LINE \| 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 valid 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 valid 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 valid for numeric variables. If DATE-EDIT-MASK has not been specified, the date is displayed using the default format for that database (see the Default Database Formats table). |
| 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. |

## Example

The following program segments illustrate the various features of the SHOW command:

```
!
! Show a string using an edit mask
!
let $ssn = '123456789'
show $ssn edit xxx-xx-xxxx
```

Produces the following output:

```
123-45-6789
```

```
!
! Show a number using an edit mask
!
show 1234567.89 edit 999,999,999.99
```

Copyright © 1988-2007, Oracle. All rights reserved. 

Produces the following output:

```
1,234,567.89
```

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

Produces the following output:

```
123.780000
```

```
!
! 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
```

```
!
! 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
```

```
!
! 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-2004
```

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

 Copyright © 1988-2007, Oracle. All rights reserved.

Produces the following output:

```
Hello World
```

```
 !
 ! 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 about 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.
DISPLAY

Copyright © 1988-2007, Oracle. All rights reserved. 

# STOP

## Syntax

```
STOP [QUIET]
```

## Description

The STOP command halts SQR and executes a ROLLBACK command (not in Sybase, Microsoft SQL Server, 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.

## Parameters

| Parameter | Description |
|---|---|
| QUIET | Causes the report to complete with the "SQR: End Of Run" message, instead of ending with an error message. |

## Example

The following example shows the STOP command:

```
if #error-status = 1
   rollback
   stop
else
   commit
   stop quiet
end-if
```

# STRING

## Syntax

STRING {*src_any_lit*|*_var*|*_col*}...BY {*delim_txt_lit*|*_var*|*_col*}
INTO *dst_txt_var*

## Description

Concatenates a list of variables, columns, or literals into a single text variable. Each member of the list separated by the specified delimiter string.

The destination string must not be included in the list of source strings.

 Copyright © 1988-2007, Oracle. All rights reserved.

### Parameters

| Parameter | Description |
|---|---|
| {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.<br>If the source is a date variable or column, it is converted to a string according to the following rules:<br>For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.<br>If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.<br>For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.<br>If this has not been set, SQR uses the format listed in the Default Database Formats table.<br>For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.<br>If this has not been set, SQR uses the format as listed in the TIME Column Formats table. |
| {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. |

### Example

The following example shows the STRING command:

```
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-04,127
                  ! Use null delimiter.
string  &code1  &code2  &code3  by  ''  into  $codes123
                  ! Result:  AGL
```

### See Also

The UNSTRING command for additional information.
The "||" concatenation operator in the Operators table under the LET command.

## SUBTRACT

### Syntax

`SUBTRACT {src_num_lit|_var|_col} FROM dst_num_var[ROUND=nn]`

### Description

SUBTRACT subtracts the first value from the second and moves the result into the second field.

When dealing with money-related values (dollars and cents), use decimal variables rather than float variables. Float variables are stored as double precision floating point numbers, and small inaccuracies can appear when subtracting many numbers in succession. These inaccuracies can appear due to the way floating point numbers are represented by different hardware and software implementations.

Copyright © 1988-2007, Oracle. All rights reserved. 

### Parameters

| Parameter | Description |
|---|---|
| {src_num_lit \| _var \| _col} | Is subtracted from the contents of *dst_num_var*. |
| dst_num_var | Contains the result after execution. |
| ROUND | Rounds the result to the specified number of digits to the right of the decimal point. For float variables this value can be from 0 to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate. |

### Example

The following example shows the SUBTRACT command:

```
subtract  1  from  #total      ! #total - 1
subtract &discount from #price ! #price - &discount
```

### See Also

The ADD command for more information.
The LET command for information about complex arithmetic expressions.

## TOC-ENTRY

### Syntax

```
TOC-ENTRY
TEXT={src_txt_lit|_var|_col}
[LEVEL={level_num_lit|_var|_col}]
```

### Description

Enter the text in the table of contents at the desired level.

### Parameters

| Parameter | Description |
|---|---|
| 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. |

### Example

The following example shows the TOC-ENTRY command:

```
toc-entry text = &heading
toc-entry text = &caption level=2
```

### See Also

The DECLARE-TOC command.

 Copyright © 1988-2007, Oracle. All rights reserved.

# UNSTRING

## Syntax

```
UNSTRING {{src_txt_lit|_var|_col}|{src_date_var|_col}}
BY {delim_txt_lit|_var|_col}
INTO dst_txt_var...
```

## Description

Copies portions of a string into one or more text variables.

Each substring is located using the specified delimiter. The source string must not be included in the list of destination strings.

If more destination strings than substrings are found in the source strings, the extra destination strings are each set to an empty string.

If more substrings are found in the source string than in the destination strings, the extra substrings are not processed. It is up to the programmer to ensure that enough destination strings are specified.

If the source is a date variable or column, it is converted to a string according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.

  If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the Default Database Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

## Parameters

| Parameter | Description |
|---|---|
| {src_txt_lit\|_var\|_col}\|{ src_date_var\|_col} | Specifies the source field to be parsed. |
| delim_txt_lit\|_var\|_col | Specifies one or more characters to be used to delimit the fields within {*src_txt_lit*\|*_var*\|*_col*}\|{*src_date_var*\|*_col*} |
| dst_txt_var | Specifies one or more destination fields to receive the results. |

## Example

The following example shows the UNSTRING command:

```
unstring $show-info by ' - ' into $name $city $state $zip
unstring $cust-data by ',' into $cust_num $entry-date $total
```

## See Also

STRING, EXTRACT
The substr and instr functions in the Miscellaneous Functions table under the LET command.

Copyright © 1988-2007, Oracle. All rights reserved. 

# UPPERCASE

## Syntax

UPPERCASE *txt_var*

## Description

Converts a string variable to uppercase.

## Parameters

| Parameter | Description |
|---|---|
| txt_var | Specifies a text variable to be converted to uppercase. |

## Example

The following example shows the UPPERCASE command:

```
input  $state  'Enter state abbreviation'
uppercase  $state     ! Force uppercase.
```

## See Also

The upper function in the Miscellaneous Functions table under the LET command.

# USE

## Syntax

USE *database*

## Description

Uses the named database, rather than the default database associated with your username. (Sybase and Microsoft SQL Server only.)

Use USE in the SETUP section only. When used, it must appear at the top of your report, before any queries are defined.

To reference more than one database in a program, specify secondary databases explicitly. For example:

```
from sqdb.sqr.customers
```

You cannot issue the Sybase or Microsoft SQL Server USE command from within an SQL paragraph.

## Parameters

| Parameter | Description |
|---|---|
| database | Specifies the name of the database to use. |

## Example

The following example shows the USE command:

```
begin-setup
```

 Copyright © 1988-2007, Oracle. All rights reserved.

```
 use pubs
end-setup
```

### See Also

Chapter 4, "Invoking SQR Execute," page 255

## USE-COLUMN

### Syntax

`USE-COLUMN {column_number_int_lit|_var|_col}`

### Description

Sets the current column.

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.

### Parameters

| Parameter | Description |
|---|---|
| {column_number_int_lit\|_var\|_col} | Specifies the number of the defined column (not the location on the page). For example, if five columns are defined, the *column_number_int_lit\|_var\|_col* can be 1 to 5. |

### Example

The following example shows the USE-COLUMN command:

```
use-column  3     ! Print total in 3rd column.
print  #total  ()  999,999
use-column  0     ! End of column printing.
```

## USE-PRINTER-TYPE

### Syntax

`USE-PRINTER-TYPE printer-type`

### Description

Sets the printer type to be used for the current report.

The USE-PRINTER-TYPE command sets or alters the printer type to be 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.

Copyright © 1988-2007, Oracle. All rights reserved. 

### Parameters

| Parameter | Description |
|---|---|
| printer-type | Specifies the printer type to be used for the current report. See DECLARE-PRINTER for valid types. |

### Example

The following example shows the USE-PRINTER-TYPE command:

```
use-report customer_orders
use-printer-type PostScript
print (1, 1) 'Customer Name: '
print () $customer_name
```

### See Also

DECLARE-PRINTER, DECLARE-REPORT, USE-REPORT

## USE-PROCEDURE

### Syntax

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

### Description

Changes the procedure usage.

The USE-PROCEDURE must be issued in the PROGRAM or PROCEDURE sections of an SQR program. USE-PROCEDURE is a runtime 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, because that causes the BEFORE-REPORT procedure to be executed.

 Copyright © 1988-2007, Oracle. All rights reserved.

## Parameters

| Parameter | Description |
|---|---|
| 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. |
| 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. |
| 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. |
| 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. |
| 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.<br>You can also specify arguments to be passed to the procedure. Arguments can be any variable, column, or literal. |

## Example

The following example shows the USE-PROCEDURE command:

```
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

DECLARE-PROCEDURE

Copyright © 1988-2007, Oracle. All rights reserved. 

# USE-REPORT

## Syntax

```
USE-REPORT {report_name_lit|_var|_col}
```

## Description

For programs with multiple reports, enables the user to switch between reports.

The USE-REPORT command specifies to which report files 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.

## Parameters

| Parameter | Description |
|---|---|
| {report_name_lit\|_var\|_col} | Specifies the report to become the current report. All subsequent PRINT and PRINT-DIRECT statements are written to this report until the next USE- REPORT is encountered. |

## Example

The following example shows the USE-REPORT command:

```
use-report customer_orders
use-printer-type PostScript
print (1, 1) 'Customer Name: '
print () $customer_name
```

## See Also

DECLARE-REPORT, DECLARE-LAYOUT, DECLARE-PRINTER, USE-PRINTER-TYPE

# WHILE

## Syntax

```
WHILE logical_expression
```

The general format of a WHILE command is as follows:

```
WHILE logical_expression
SQR_commands...
[BREAK]
SQR_commands...
END-WHILE
```

## Description

Begins a **WHILE ... END-WHILE** loop.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

BREAK causes an immediate exit of the WHILE loop; SQR continues with the command immediately following END-WHILE.

WHILE commands can be nested to any level and can include or be included within IF and EVALUATE commands.

## Parameters

| Parameter | Description |
|---|---|
| logical_expression | A valid logical expression. See the LET command for a description of logical expressions. |

## Example

This 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
```

This example sets up an infinite loop:

```
while 1
    ...processing...
    if ...
       break     ! Exit loop
    end-if
end-while
```

Any complex expression can be used in the WHILE command, as shown in this example:

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

## See Also

The LET command for a description of expressions.

Copyright © 1988-2007, Oracle. All rights reserved. 

# WRITE

## Syntax

```
WRITE {filenum_lit|_var|_col} FROM
{{{txt_lit|_var|_col}|{date_var|_col}|num_col}
[:len_int_lit]}|{num_lit|_var:len_int_lit}}...
[STATUS=status_num_var]
```

## Description

Writes a record to a file from data stored in variables, columns, or literals.

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 because 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 has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

  If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

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.

 Copyright © 1988-2007, Oracle. All rights reserved.

## Parameters

| Parameter | Description |
|---|---|
| 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]}\| {num_lit\|_var:len_int_lit} | Specifies one or more variables to be written. *len_int_lit* specifies the length of each field of data. |
| STATUS | Specifies an optional variable into which a write status is returned. |

## Example

The following example shows the WRITE command:

```
write  5  from  $name:20  $city:15  $state:2
write  17  from  $company  ' - '  $city  ' - '  $state  ' '  $zip
write  #j2  from  #rate:2  #amount:4  #quantity:1
move  #total to $tot 99999.99          ! Convert floating point to string.
write  1  from  $tot
let $date1 = datenow()                ! Put the current date and time
                                      ! into DATE variable
write 3 from $date1:20
```

## See Also

OPEN, CLOSE, READ

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

# CHAPTER 3

# Generating HTML Output

This chapter describes the procedures that enable SQR for PeopleSoft to generate HTML output. You can publish the output on an internet, intranet, or extranet website. An SQR program without HTML procedures has limited HTML capabilities. Adding HTML procedures to the SQR program enhances the appearance of the HTML output.

This chapter discusses:

- HTML general-purpose procedures.
- HTML heading procedures.
- HTML highlighting procedures.
- HTML hypertext link procedures.
- HTML list procedures.
- HTML table procedures.

### See Also

*Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, "Generating and Publishing HTML from an SQR Program"

## HTML General Purpose Procedures

The following table lists the HTML general-purpose procedures:

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_br | Produces the specified number of line breaks in a paragraph by using the HTML<BR> tag. The paragraph continues onto the next line.<br><br>Syntax:<br><br>`html_br(number count, string attributes)`<br><br>count: The number of HTML <BR> tags that are inserted.<br><br>attributes: The HTML attributes that are incorporated inside the HTML <BR> tag.<br><br>Example:<br>`print 'Here is some text' ()`<br>`do html_br(3,'')`<br>`print 'Here is some text three lines down' ()` |
| html_center | Marks the start of text to be centered on the HTML document by using the HTML <CENTER> tag. You can also accomplish this by using the SQR print statement with CENTER specified in the code.<br><br>Syntax:<br><br>`html_center()`<br><br>attributes: The HTML attributes that are incorporated inside the HTML <CENTER> tag.<br><br>Example:<br>`do html_center('')`<br>`print 'Here is some centered text' ()`<br>`do html_center_end` |
| html_center_ end | Marks the end of text that was previously specified as centered.<br><br>Syntax:<br><br>`html_center_end` |
| html_hr | Produces a horizontal divider between sections of text by using the HTML <HR> tag.<br><br>Syntax:<br><br>`html_hr(string attributes)`<br><br>attributes: The HTML attributes that are incorporated inside the HTML <HR> tag.<br><br>Example:<br>`print 'Here is some text' ()`<br>`do html_hr('')`<br>`print 'Text after a horizonal divider' ()` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_img | Inserts an image by using the HTML <IMG> tag. You can also do this by using the PRINT-IMAGE command; however, the html_img procedure enables you to specify the full set of available HTML attributes.<br>Syntax:<br>`html_img(string attributes)`<br>attributes: The HTML attributes that are incorporated inside the HTML <IMG> tag. Some common attributes include:<br>• src: The URL of the image to be inserted.<br>Example: src=/images/abc.gif<br>• height: The height of the image in pixels.<br>Example: height=200<br>• width: The width of the image in pixels.<br>Example: width=400<br>Example:<br>`do html_img('src="/images/stop.gif"')` |
| html_nobr | Marks the start of text that cannot be wrapped by using the HTML <NOBR> tag.<br>Syntax:<br>`html_nobr`<br>Example:<br>`do html_nobr('')`<br>`print 'Long line of text that shouldn't wrap' ⇒`<br>`()`<br>`do html_nobr_end` |
| html_nobr_end | Marks the end of text that cannot be wrapped by using the HTML </NOBR> tag.<br>Syntax:<br>`html_nobr_end` |
| html_on | Activates the HTML procedures. Call this procedure at the start of an SQR program; otherwise, the HTML procedures are not activated. After the HTML procedures are activated, format the appearance of the web page by using the various HTML procedures.<br>Syntax:<br>`html_on`<br>Example:<br>`do html_on` |

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_p | Marks the start of a new paragraph by using the HTML <P> tag.<br>Syntax:<br>`html_p(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <P> tag. A common attribute is align = left\|right\|center, which specifies the alignment of the paragraph.<br>Example:<br>`do html_p('ALIGN=RIGHT')`<br>`print 'Right aligned text' (1,1)`<br>`do html_p_end`<br>`print 'Normally aligned text' (+1,1)` |
| html_p_end | Marks the end of a paragraph by using the HTML </P> tag. The end of a paragraph is typically implied and the procedure is technically not needed; however, it's good practice to specify it for completeness.<br>Syntax:<br>`html_p_end` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_set_body_attributes | Specifies the attributes that are incorporated into the HTML <BODY> tag. Call this procedure at the start of the SQR program.<br>Syntax:<br>`html_set_body_attributes(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <BODY> tag. Some common attributes include:<br>• background: Specifies the image to display in the background of the web page.<br>Example: background=/images/logo.gif<br>• bgcolor=#rrggbb: Specifies the background color of the web page.<br>Example: bgcolor=#80FFF<br>Example:<br>`do html_set_body_attributes('BACKGROUND="⇒ /images/x.gif"')` |
| html_set_head_tags | Specifies the tags that are incorporated between the HTML <HEAD> and </HEAD> tags. These are empty by default. One common tag to set is the HTML <TITLE> tag, which specifies the title to display for the web page. Call this procedure at the start of the SQR program.<br>Syntax:<br>`html_set_head_tags(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated between the HTML <HEAD> and </HEAD> tags.<br>Example:<br>`do html_set_head_tags('<TITLE>My Report<⇒ /TITLE>')` |

# HTML Heading Procedures

The following table lists the HTML heading procedures:

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_h1 | Marks the start of text for heading level one by using the HTML <H1> tag. This heading text appears more prominently than heading level two.<br>Syntax:<br>`html_h1(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <H1> tag.<br>Example:<br>`do html_h1('')`<br>`print 'This is a heading' ()`<br>`do html_h1_end` |
| html_h1_end | Marks the end of text for heading level one by using the HTML </H1> tag.<br>Syntax:<br>`html_h1_end` |
| html_h2 | Marks the start of text for heading level two by using the HTML <H2> tag. This heading text appears less prominently than heading level one and more prominently than heading level three.<br>Syntax:<br>`html_h2(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <H2> tag.<br>Example:<br>`do html_h2('')`<br>`print 'This is a heading' ()`<br>`do html_h2_end` |
| html_h2_end | Marks the end of text for heading level two by using the HTML </H2> tag.<br>Syntax:<br>`html_h2_end` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_h3 | Marks the start of text for heading level three by using the HTML <H3> tag. This heading text appears less prominently than heading level two and more prominently than heading level four.<br><br>**Note.** This heading level is the default value.<br><br>Syntax:<br><br>`html_h3(string attributes)`<br><br>attributes: Defines the HTML attributes that are incorporated inside the HTML <H3> tag. |
| html_h3_end | Marks the end of text for heading level three by using the HTML </H3> tag.<br><br>Syntax:<br><br>`html_h3_end` |
| html_h4 | Marks the start of text for heading level four by using the HTML <H4> tag. This heading text appears less prominently than heading level three and more prominently than heading level five.<br><br>Syntax:<br><br>`html_h4(string attributes)`<br><br>attributes: Defines the HTML attributes that are incorporated inside the HTML <H4> tag. |
| html_h4_end | Marks the end of text for heading level four by using the HTML </H4> tag.<br><br>Syntax:<br><br>`html_h4_end` |
| html_h5 | Marks the start of text for heading level five by using the HTML <H5> tag. This heading text appears less prominently than heading level four and more prominently than heading level six.<br><br>Syntax:<br><br>`html_h5(string attributes)`<br><br>attributes: Defines the HTML attributes that are incorporated inside the HTML <H5> tag. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_h5_end | Marks the end of text for heading level five by using the HTML </H5> tag.<br>Syntax:<br>`html_h5_end` |
| html_h6 | Marks the start of text for heading level six by using the HTML <H6> tag. This heading text appears less prominently than heading level five.<br>Syntax:<br>`html_h6(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <H6> tag. |
| html_h6_end | Marks the end of text for heading level six by using the HTML </H6> tag.<br>Syntax:<br>`html_h6_end` |

# HTML Highlighting Procedures

The following table lists the HTML highlighting procedures:

| Procedure | Description |
|---|---|
| html_blink | Marks the start of blinking style text by using the HTML <BLINK> tag.<br>Syntax:<br>`html_blink(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <BLINK> tag.<br>Example:<br>`do html_blink('')`<br>`print 'This is blinking text' ()`<br>`do html_blink_end` |
| html_blink_end | Marks the end of blinking style by using the HTML </BLINK> tag.<br>Syntax:<br>`html_blink_end` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_cite | Marks the start of text in citation style by using the HTML <CITE> tag.<br>Syntax:<br>`html_cite(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <CITE> tag.<br>Example:<br>`do html_cite('')`<br>`print 'This is a citation' ()`<br>`do html_cite_end` |
| html_cite_end | Marks the end of text in citation style by using the HTML </CITE> tag.<br>Syntax:<br>`html_cite_end` |
| html_code | Marks the start of text in code style by using the HTML <CODE> tag.<br>Syntax:<br>`html_code(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <CODE> tag.<br>Example:<br>`do html_code('')`<br>`print 'Here is code style text' ()`<br>`do html_code_end` |
| html_code_end | Marks the end of text in code style by using the HTML </CODE> tag.<br>Syntax:<br>`html_code_end` |

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_kbd | Marks the start of text in keyboard input style by using the HTML <KBD> tag.<br>Syntax:<br>`html_kbd(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <KBD> tag.<br>Example:<br>`do html_kbd('')`<br>`print 'Here is keyboard style text' ()`<br>`do html_kbd_end` |
| html_kbd_end | Marks the end of text in keyboard style by using the HTML </KBD> tag.<br>Syntax:<br>`html_kbd_end` |
| html_samp | Marks the start of text in sample style by using the HTML <SAMP> tag.<br>Syntax:<br>`html_samp(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <SAMP> tag.<br>Example:<br>`do html_samp('')`<br>`print 'Here is sample style text' ()`<br>`do html_samp_end` |
| html_samp_end | Marks the end of text in sample style by using the HTML </SAMP> tag.<br>Syntax:<br>`html_samp_end` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_strike | Marks the start of text in strikethrough style by using the HTML <STRIKE> tag.<br>Syntax:<br>`html_strike(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <STRIKE> tag.<br>Example:<br>`do html_strike('')`<br>`print 'Here is strikethrough text' ()`<br>`do html_strike_end` |
| html_strike_end | Marks the end of text in strikethrough style by using the HTML </STRIKE> tag.<br>Syntax:<br>`html_strike_end` |
| html_sub | Marks the start of text in subscript style by using the HTML <SUB> tag.<br>Syntax:<br>`html_sub(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <SUB> tag.<br>Example:<br>`print 'Here is' ()`<br>`do html_sub('')`<br>`print 'subscript text' ()`<br>`do html_sub_end` |
| html_sub_end | Marks the end of text in subscript style by using the HTML </SUB> tag.<br>Syntax:<br>`html_sub_end` |

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_sup | Marks the start of text in superscript style by using the HTML <SUP> tag.<br>Syntax:<br>`html_sup(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <SUP> tag.<br>Example:<br>`print 'Here is' ()`<br>`do html_sup('')`<br>`print 'superscript text' ()`<br>`do html_sup_end` |
| html_sup_end | Marks the end of text in superscript style by using the HTML </SUP> tag.<br>Syntax:<br>`html_sup_end` |

# HTML Hypertext Link Procedures

The following table lists the HTML hypertext link procedures:

 Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_a | Marks the start of a hypertext link by using the HTML <A> tag. When the user clicks the area with the hypertext link, the web browser switches to the specified HTML document.<br><br>Syntax:<br><br>`html_a(string attributes)`<br><br>attributes: Defines the HTML attributes that are incorporated inside the HTML <A> tag. At a minimum, you should define the href attribute, which specifies the URL of an HTML document.<br><br>Some common attributes include:<br><br>• href: Indicates where the hypertext link points.<br>Example: href=home.html<br>• name: Indicates an anchor to which a hypertext link can point.<br>Example: name=marker1<br><br>Example:<br><br>The anchor is positioned at the top of the document. The first hypertext link points to the HTML document named otherdoc.html. The second hypertext link points to the anchor named TOP.<br>`do html_a('NAME=TOP')`<br>`do html_a_end`<br>`print 'At the top of document' ()`<br>`do html_br(20, '')`<br>`do html_a('HREF=otherdoc.html')`<br>`print 'Go to other document' ()`<br>`do html_a_end`<br>`do html_p('')`<br>`do html_a('HREF=#TOP')`<br>`print 'Go to top of document' ()`<br>`do html_a_end` |
| html_a_end | Marks the end of a hypertext link by using the HTML </A> tag.<br><br>Syntax:<br><br>`html_a_end` |

# HTML List Procedures

The following table lists the HTML list procedures:

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_dd | Marks the start of a definition in a definition list by using the HTML <DD> tag.<br>Syntax:<br>`html_dd(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <DD> tag. |
| html_dd_end | Marks the end of a definition in a definition list by using the HTML </DD> tag. The end of a definition in a definition list is typically implied and not needed; however, it's good form to specify it for completeness.<br>Syntax:<br>`html_dd_end` |
| html_dir | Marks the start of a directory list by using the HTML <DIR> tag.<br>Syntax:<br>`html_dir(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <DIR> tag.<br>Example:<br>`do html_dir('')`<br>`do html_li('')`<br>`print 'First item' ()`<br>`do html_li('')`<br>`print 'Second item' ()`<br>`do html_li('')`<br>`print 'Last item' ()`<br>`do html_dir_end` |
| html_dir_end | Marks the end of a directory list by using the HTML </DIR> tag.<br>Syntax:<br>`html_dir_end` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_dl | Marks the start of a definition list by 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 html_dt procedure to display a term. Use the html_dd procedure to display a definition.<br>Syntax:<br>`html_dl(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <DL> tag.<br>Example:<br>`do html_dl('')`<br>`do html_dt('')`<br>`print 'A Daisy' ()`<br>`do html_dd('')`<br>`print 'A sweet and innocent flower.' ()`<br>`do html_dt('')`<br>`print 'A Rose' ()`<br>`do html_dd('')`<br>`print 'A very passionate flower.' ()`<br>`do html_dl_end` |
| html_dl_end | Marks the end of a definition list by using the HTML </DL> tag.<br>Syntax:<br>`html_dl_end` |
| html_dt | Marks the start of a term in a definition list by using the HTML <DT> tag.<br>Syntax:<br>`html_dt(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <DT> tag. |
| html_dt_end | Marks the end of a term in a definition list by using the HTML </DT> tag.<br>Syntax:<br>`html_dt_end` |
| html_li | Marks the start of a list item by using the HTML <LI> tag.<br>Syntax:<br>`html_li(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <LI> tag. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_li_end | Marks the end of a list item by using the HTML </LI> tag. The end of a list item is typically implied and not needed; however, you should specify it for completeness.<br>Syntax:<br>`html_li_end` |
| html_menu | Marks the start of a menu by using the HTML <MENU> tag. Use the html_li procedure to identify each item in the list.<br>Syntax:<br>`html_menu(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <MENU> tag.<br>Example:<br>`do html_menu('')`<br>`do html_li('')`<br>`print 'First item' ()`<br>`do html_li('')`<br>`print 'Second item' ()`<br>`do html_li('')`<br>`print 'Last item' ()`<br>`do html_menu_end` |
| html_menu_end | Marks the end of a menu by using the HTML </MENU> tag.<br>Syntax:<br>`html_menu_end` |
| html_ol | Marks the start of an ordered list by using the HTML <OL> tag. Each item in the list typically appears indented to the right with a number to the left. Use the html_li procedure to identify each item in the list.<br>Syntax:<br>`html_ol(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <OL> tag.<br>Example:<br>`do html_ol('')`<br>`do html_li('')`<br>`print 'First item' ()`<br>`do html_li('')`<br>`print 'Second item' ()`<br>`do html_li('')`<br>`print 'Last item' ()`<br>`do html_ol_end` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_ol_end | Marks the end of an ordered list by using the HTML </OL> tag.<br>Syntax:<br>`html_ol_end` |
| html_ul | Marks the start of an unordered list by using the HTML <UL> tag. Each item in the list typically appears indented to the right with a bullet to the left. Use the html_li procedure to identify each item in the list.<br>Syntax:<br>`html_ul(string attributes)`<br>attributes: defines the HTML attributes that are incorporated inside the HTML <UL> tag.<br>Example:<br>`do html_ul('')`<br>`do html_li('')`<br>`print 'First item' ()`<br>`do html_li('')`<br>`print 'Second item' ()`<br>`do html_li('')`<br>`print 'Last item' ()`<br>`do html_ul_end` |
| html_ul_end | Marks the end of an unordered list by using the HTML </UL> tag.<br>Syntax:<br>`html_ul_end` |

# HTML Table Procedures

The following table lists the HTML table procedures:

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
| --- | --- |
| html_caption | Marks the start of a table caption by using the HTML <CAPTION> tag.<br>Syntax:<br>`html_caption(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <CAPTION> tag. |
| html_caption_end | Marks the end of a table caption by using the HTML </CAPTION> tag. The end of a table caption is typically implied and not needed; however, you should specify it for completeness.<br>Syntax:<br>`html_caption_end` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_table | Marks the start of a table by using the HTML <TABLE> tag.<br>Syntax:<br>`html_table(string attributes)`<br>attributes: Defines the HTML attributes that are incorporated inside the HTML <TABLE> tag.<br>Some common attributes include:<br>• border: Specifies that a border appears around each cell of the table.<br>• width: Specifies the width of the entire table in pixels.<br>• cols: Specifies the number of columns in the table.<br>Example: COLS=4<br>Example: Displaying database records in a tabular format. The html_caption_end, html_tr_end, html_td_end, and html_th_end procedures are used for completeness; however, they are typically implied and not needed.<br>`! start the table & display the column headings`<br>`do html_table('border')`<br>`do html_caption('')`<br>`print 'Customer Records' (1,1)`<br>`do html_caption_end`<br>`do html_tr('')`<br>`do html_th('')`<br>`print 'Cust No' (+1,1)`<br>`do html_th_end`<br>`do html_th('')`<br>`print 'Name" (,10)`<br>`do html_th_end`<br>`do html_tr_end`<br>`! display each record`<br>`begin-select`<br>`do html_tr('')`<br>`do html_td('')`<br>`cust_num (1,1,6) edit 099999`<br>`do html_td_end`<br>`do html_td('')`<br>`name (1,10,25)`<br>`do html_td_end`<br>`do html_tr_end`<br>`next-listing skiplines=1 need=1`<br>`from customers`<br>`end-select`<br>`! end the table`<br>`do html_table_end` |
| html_table_end | Marks the end of a table by using the HTML </TABLE> tag.<br>Syntax:<br>`html_table_end` |

Copyright © 1988-2007, Oracle. All rights reserved.

| Procedure | Description |
|---|---|
| html_td | Marks the start of a new column in a table row by using the HTML <TD> tag. This specifies that the text that follows is displayed in the column.<br><br>Syntax:<br><br>`html_td(string attributes)`<br><br>attributes: Defines the HTML attributes that are incorporated inside the HTML <TD> tag. |
| html_td_end | Marks the end of a column in a table by using the HTML </TD> tag. The end of a column is typically implied and not needed; however, you should specify it for completeness.<br><br>Syntax:<br><br>`html_td_end` |
| html_th | Marks the start of a new column header in a table row by using the HTML <TH> tag. This specifies that the text that follows is displayed as the header of the column.<br><br>Syntax:<br><br>`html_th(string attributes)`<br><br>attributes: Defines the HTML attributes that are incorporated inside the HTML <TH> tag. |
| html_th_end | Marks the end of a column header in a table by using the HTML </TH> tag. The end of a column header is typically implied and not needed; however, you should specify it for completeness.<br><br>Syntax:<br><br>`html_th_end` |
| html_tr | Marks the start of a new row in a table by using the HTML <TR> tag.<br><br>Syntax:<br><br>`html_tr(string attributes)`<br><br>attributes: Defines the HTML attributes that are incorporated inside the HTML <TR> tag. |
| html_tr_end | Marks the end of a row by using the HTML </TR> tag. The end of a row in a table is typically implied and not needed; however, you should specify it for completeness.<br><br>Syntax:<br><br>`html_tr_end` |

 Copyright © 1988-2007, Oracle. All rights reserved.

# CHAPTER 4

# Invoking SQR Execute

SQR Execute is a runtime program that enables you to run a previously compiled SQR program.

This chapter discusses how to:

- Run SQR Execute
- Use SQR Execute flags

## Running SQR Execute

To run SQR Execute, enter the following command. (If you are using Microsoft Windows, invoke SQRWT rather than SQRT.)

```
SQRT [program][connectivity][flags...][args...][@file...]
```

## Using SQR Execute Flags

The following table describes the SQR Execute command-line flags. See the SQR command-line arguments section for information about program, connectivity, args, and @file.

See Chapter 1, “Understanding SQR for PeopleSoft,” SQR Command-Line Arguments, page 4.

| Flag | Description |
|---|---|
| -A | Appends the output to an existing output file of the same name. If the file does not exist, it creates a new one. This flag is useful when you want to run the same report more than once, but you want to create only one output file. |
| -C | (Microsoft Windows) Specifies that the Cancel dialog box appears while the program is running so that you can easily terminate the program. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -BURST:{xx} | Specifies the type of bursting to be performed.<br><br>-BURST:T generates the table of contents file only.<br><br>-BURST:S generates the report output according to the symbolic table of contents entries that are set in the program with the TOC- ENTRY command's LEVEL argument. In -BURST:S[ {l} ], *l* is the level at which to burst. The -BURST:S setting is equivalent to -BURST:S1.<br><br>-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 contains and *s* is the page selection: *n, n-, m, -m,* or *n-*. The -BURST:P setting is equivalent to -BURST:P0,1- when using -PRINTER:HT or -BURST:P1 when using -PRINTER:EH.<br><br>**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. |
| -CB | (Microsoft Windows) Forces the communication box. |
| -Dnn | (Non-Microsoft Windows) Causes SQR to display the report output on the terminal at the same time that it is being written to the output file. The *nn* is the maximum number of lines to display before pausing. If you don't enter a number after -D, the display scrolls continuously. The printer type must be LP; otherwise SQR does not display any output. If the program is producing more than one report, SQR displays only the first report. |
| -DBdatabase | (Sybase) Causes the SQR program to use the specified database, overriding any USE command in the SQR program. |
| -E[file] | Directs error messages to the named file or to the default file, program.err. If no errors occur, no file is created. |
| -EH_APPLETS:dir | Specifies the directory location of the enhanced HTML applets. The default directory for theses applets is IMAGES.<br><br>**Note.** The -EH flags in this table are applicable only when either the -PRINTER:EH or -PRINTER:EP flag is specified. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -EH_BQD | Generates a {report}.bqd file from the report data. Also associates a BQD icon with {report}.bqd in the navigation bar. |
| -EH_BQD:file | Associates the BQD icon with the specified file. |
| -EH_BROWSER:xx | Specifies the target browser. When set to ALL, SQR automatically determines which browser is being used, invokes a browser-specific file, and generates HTML that is designed for that browser.<br><br>When set to BASIC, SQR generates HTML that is suitable for all browsers.<br><br>When set to IE, SQR generates HTML that is designed for Microsoft Internet Explorer.<br><br>When set to NETSCAPE, SQR generates HTML that is designed for Netscape. |
| -EH_CSV | Generates a {report}.csv file from the report data. |
| -EH_CSV:file | Associates the CSV icon with the specified file. |
| -EH_CSVONLY | Creates a CSV file, but does not create an HTML file. |
| -EH_FULLHTML:xx | Switches between HTML 3.0 and HTML 3.2. When set to TRUE, SQR generates HTML 3.2. When set to FALSE, SQR generates HTML 3.0 |
| -EH_Icons:dir | Specifies the directory in for the referenced icons. |
| -EH_LANGUAGE:xx | Sets the language that is used for the HTML navigation bar. You can specify English, French, German, Portuguese, Spanish, Japanese, Simplified Chinese, and Korean. |
| -EH_PDF | Associates a PDF icon with {report}.pdf in the navigation bar. |
| -EH_Scale:{nn} | Sets the scaling factor from 50 to 200. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -F[file\|directory] | Overrides the default output file name, program.lis. The default action places the program.lis file in the same directory as the program.sqr file.<br><br>To use the current directory, specify -F without an argument.<br><br>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.<br><br>The output file is not created until data is actually printed on the page. If no data is printed, no output file is created.<br><br>Specify the file name and directory for different operating systems:<br>• UNIX\Linux<br>• MVS |
| -GPRINT=YES \| NO | (MVS) -GPRINT=YES causes ANSI control characters written to the first column of each record of the SQR output file. |
| -ID | Displays the copyright banner on the console. |
| -KEEP | In addition to .lis files, creates an .spf file for each report that the program generates. |
| -NOLIS | Prevents the creation of .lis files. Instead, .spf files are created. |
| -O[file] | Directs log messages to the specified file or to programt.log, if no file is specified. By default, the sqr.log file is used in the current working directory. |
| -P | (MVS) Suppresses printer control characters from column 1. |
| -PB | (Informix) Causes column data to retain trailing blanks. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -PRINTER:xx | Causes printer type *xx* to be used when creating output files:<br>• EH<br>• EP<br>• HP<br>• HT<br>• LP<br>• PD<br>• PS<br>• WP<br>Types LP, HP, and PS produce files with the .lis extension.<br>Types EH and HT produce .htm file output.<br>Type HT produces files in HTML version 2.0 with the report content inside the <PRE></PRE> tags.<br>Type EH produces reports in which content is fully formatted with HTML version 3.0 or 3.2 tags.<br>On Microsoft Windows systems, the WP extension sends the output to the default Microsoft Windows printer. To specify a nondefault Microsoft Windows printer, enter -PRINTER:WP:{printer name}. The {printer name} is the name that is assigned to the printer. For example, to send output to a Microsoft Windows printer named NewPrinter, use - PRINTER:WP:NewPrinter. If the printer name has spaces, enclose the entire argument in quotes. If you also want to create an .spf file, use -KEEP. |
| -S | Requests that the status of all cursors be displayed at the end of the report run. Status includes the text of each SQL statement, the number of times that each was compiled and run, and the total number of rows selected. |
| -Tnn | Specifies that you want to test the report for *nn* pages. SQR ignores all Order By clauses in Select statements to save time during testing. If the program is producing more than one report, SQR stops after the specified number of pages that are defined for the first report have been printed. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -T{B\|Z\|BZ\|ZB} | (IBM MVS and DB2):<br>• -TB prevents SQR from removing trailing blanks from database character columns.<br>• -TZ prevents SQR from removing trailing zeros from the decimal portion of numeric columns.<br>• -TBZ or -TZB prevents both.<br>(Windows/DB2, Sybase CT-Lib, and ODBC): -TB trims trailing blanks from database character columns.<br>**Note.** The -TB flag only has an effect if SQR is connecting to either a DB2, Sybase CT LIB, or ODBC (MSS) database. Confusingly, the behavior of the -TB command-line flag varies depending on your platform. If you are using one of the above mentioned databases and are running SQR on z/OS, the -TB flag will behave as follows:<br>If you do not use the -TB flag, trailing blanks are trimmed.<br>If you do use the -TB flag, trailing blanks are not trimmed.<br>If you are running SQR on any other platform, the behavior of -TB is the opposite:<br>If you do not use the -TB flag, trailing blanks are not trimmed.<br>If you do use the -TB flag, trailing blanks are trimmed. |
| -Vserver | (Sybase) Uses the named server. |
| -XB | (Non-Microsoft Windows) Suppresses the SQR banner and the *SQR.... End of Run* message. |
| -XCB | (Microsoft Windows) Does not use the communication box. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -XL | Prevents SQR from signing in to the database. Programs that are run in this mode cannot contain any SQL statements.<br><br>-XL enables you to 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: `sqr myprog / -xl`<br><br>Some database files must be available for SQR to run whether SQR signs in to the database or not.<br><br>See information about your particular operating system and database to determine which files you need. |
| -XMB | (Microsoft Windows) Disables the error message display so that a program can be run without interruption by error message boxes. Error messages are sent only to an .err file. See the -E flag for more information. |
| -XNAV | Prevents SQR from creating the navigation bar in .htm files that are generated with -PRINTER:HT. This occurs when only a single .htm file is produced. Multiple .htm files that are generated from a single report always contain the navigation bar. |
| -XP | (Sybase DBLib) Prevents SQR from creating temporary stored procedures. See BEGIN-SELECT for more information. |
| -XTB | Preserves the trailing blanks in an .lis file at the end of a line. |
| -XTOC | Prevents SQR from generating the table of contents for the report. SQR ignores this flag when -PRINTER:EH or - PRINTER:HT is also specified. |
| -ZIF{file} | Sets the full path and name of the SQR initialization file, sqr.ini. |
| -ZIV | Invokes the SPF Viewer after generating the program.spf file. This flag implicitly invokes the -KEEP flag to create program.spf. In the case of multiple output files, only the first report file is passed to the viewer. |
| -ZMF{file} | Specifies the full path and name of the SQR error message file, sqrerr.dat. |

Copyright © 1988-2007, Oracle. All rights reserved.

## See Also

Chapter 1, “Understanding SQR for PeopleSoft,” SQR Command-Line Arguments, page 4

 Copyright © 1988-2007, Oracle. All rights reserved.

# CHAPTER 5

# Using SQR Print

This chapter provides an overview of SQR Print and discusses how to:

- Generate output from the command line.
- Use SQR Print command-line flags.
- Generate output in Microsoft Windows.

## Understanding SQR Print

SQR Print enables you to create printer-specific reports for any of the file types that are supported by SQR. SQR Print converts portable printer-independent files (spfs) into printer-specific files. SQR and SQRT (or SQRWT, for Windows) create .spf files when you use the -KEEP and -NOLIS command-line flags.

## Generating Output from the Command Line

To begin running SQR Print, enter the following command. (If you are in Microsoft Windows, invoke SQRWP rather than SQRP.)

```
SQRP [spf-file] [flags...]
```

The following table describes *spf-file* and *flags*.

SQR Print writes an .lis file with the same name as the .spf file, but with the lis extension. You can override this name with the -F command-line flag.

The -PRINTER command-line flag specifies the printer type. SQR offers several options:

- Line printer
- HP LaserJet
- PostScript
- HTML
- Enhanced HTML
- Adobe PDF
- Enhanced HTML and Adobe PDF

If the report contains graphics and you select a line printer, SQR Print ignores graphic elements (such as lines, boxes, and charts) and prints only text.

Copyright © 1988-2007, Oracle. All rights reserved.

# Using SQR Print Command-Line Flags

The following table describes the SQR Print command-line flags:

| Flag | Description |
|---|---|
| -A | Appends the output to an existing output file of the same name. If the file does not exist, it creates a new one. This flag is useful when you want to run the same report more than once, but you want to create only one output file. |
| -BURST:{xx} | Specifies the type of bursting to be performed.<br><br>-BURST:T generates the table of contents file only.<br><br>-BURST:S generates the report output according to the symbolic table of contents entries that are set in the program with the TOC-ENTRY command's LEVEL argument. In -BURST:S[ {l} ], *l* is the level at which to burst. The -BURST:S setting is equivalent to -BURST:S1.<br><br>-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 contains and *s* is the page selection: *{n}, {n}-{m}, -{m},* or *n-*. The -BURST:P setting is equivalent to -BURST:P0,1- when using -PRINTER:HT or -BURST:P1 when using -PRINTER:EH.<br><br>**Note.** -BURST:P and -BURST:S require -PRINTER:EH or -PRINTER:HT. The page range selection feature of -BURST:P requires -PRINTER:HT.<br><br>BURST:T requires -PRINTER:HT. |
| -D*nn* | (Non-Microsoft Windows) Displays the report output on the terminal at the same time that it is being written to the output file. The *nn* is the maximum number of lines is displayed before pausing. If you don't enter a number after -D, the displayed output scrolls continuously. The printer type must be LP or the no output is displayed. If the program is producing more than one report, only the first report is displayed. |
| -E[file] | Directs error messages to the named file or to the default file, program.err. If no errors occur, no file is created.<br><br>**Note.** The following -EH flags are applicable only when either the −PRINTER:EH or −PRINTER:EP flag is specified. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -EH_APPLETS:dir | Specifies the directory location of the enhanced HTML applets. The default directory for theses applets is IMAGES. |
| -EH_BQD | Generates a {report}.bqd file from the report data. Also associates a BQD icon with {report}.bqd in the navigation bar. |
| -EH_BQD:file | Associates the BQD icon with the specified file. |
| -EH_BROWSER:xx | Specifies the target browser. When set to ALL, SQR automatically determines which browser is being used, invokes a browser-specific file, and generates HTML that is designed for that browser.<br>When set to BASIC, SQR generates HTML that is suitable for all browsers.<br>When set to IE, SQR generates HTML that is designed for Microsoft Internet Explorer.<br>When set to NETSCAPE, SQR generates HTML that is designed for Netscape. |
| -EH_CSV | Generates a {report}.csv file from the report data. |
| -EH_CSV:file | Associates the CSV icon with the specified file. |
| -EH_CSVONLY | Creates a CSV file, but does not create an HTML file. |
| -EH_FULLHTM L:xx | Switches between HTML 3.0 and HTML 3.2. When set to TRUE, SQR generates HTML 3.2.<br>When set to FALSE, SQR generates HTML 3.0 |
| -EH_Icons:dir | Specifies the directory in which the HTML should find the referenced icons. |
| -EH_LANGUAGE:xx | Sets the language that is used for the HTML navigation bar. You can specify English, French, German, Portuguese, Spanish, Japanese, Simplified Chinese, or Korean. |
| -EH_PDF | Associates a PDF icon with {report}.pdf in the navigation bar. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
| --- | --- |
| -EH_Scale:{nn} | Sets the scaling factor from 50 to 200. |
| -F[file \| directory] | Overrides the default output file name, program.lis. The default action places program.lis in the same directory as the program.sqr file.<br><br>To use the current directory, specify -F without an argument.<br><br>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 is actually printed on the page. If no data is printed, no output file is created.<br><br>Specify these file names and directories for different operating systems:<br><br>• UNIX\Linux<br>• MVS |
| -ID | Displays the copyright banner on the console. |
| -O[file] | Directs log messages to the specified file or to programt.log if no file is specified. By default, the sqr.log file is used in the current working directory. |
| -P | (IBM MVS) Suppresses printer control characters from column 1. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Flag | Description |
|---|---|
| -PRINTER:xx | Causes SQR to use printer type *xx* when creating output files:<br>• EH<br>• EP<br>• HP<br>• HT<br>• LP<br>• PD<br>Types LP, HP, and PS produce files with the .lis extension.<br>Types EH and HT produce .htm file output.<br>On Microsoft Windows systems, the WP extension sends the output to the default Microsoft Windows printer. To specify a nondefault Microsoft Windows printer, enter -PRINTER:WP:{printer name}. The {printer name} is the name that is assigned to the printer. For example, to send output to a Microsoft Windows printer named NewPrinter, use -PRINTER:WP:NewPrinter.If your printer name has spaces, enclose the entire argument in quotes. If you also want to create an .spf file, use -KEEP. |
| -XB | (Non-Microsoft Windows) Suppresses the SQR banner and the *SQR.... End of Run* message. |
| -XNAV | Prevents SQR from creating the navigation bar in .htm files that are generated with -PRINTER:HT. This occurs when only a single .htm file is produced. Multiple .htm files that are generated from a single report always contain the navigation bar. |
| -XTB | Preserves the trailing blanks in a .lis file at the end of a line. |
| -XTOC | Prevents SQR from generating the table of contents for the report. This flag is ignored when -PRINTER:EH or -PRINTER:HT is also specified. |
| -ZIF{file} | Sets the full path and name of the SQR initialization file, sqr.ini. |
| -ZMF{file} | Specifies the full path and name of the SQR error message file, sqrerr.dat. |

Copyright © 1988-2007, Oracle. All rights reserved.

# Generating Output in Microsoft Windows

In Microsoft Windows, SQR Print's graphical user interface enables you to generate output from the Print dialog box. In addition to the previously mentioned SQR Print output options, you can also select a Microsoft Windows printer. This spools the SQR output to your default Microsoft Windows printer or print server.

To generate output in Microsoft Windows:

1. Select File, Print.

   The Print dialog box appears.
2. Under Generate output for, select the option next to the type of output that you want.
3. Specify a file path.
4. Select the Print to file check box.
5. Click OK.

 Copyright © 1988-2007, Oracle. All rights reserved.

# CHAPTER 6

# Avoiding Older SQR Commands

This chapter provides an overview of older SQR commands and discusses how to use them.

## Understanding Older SQR Commands

Avoid incorporating the commands covered in this chapter in your SQR code. Even though they are technically supported in this release, they do not interact well with the current SQR lexicon and may cause unpredictable results. SQR may not support these commands in future releases, so you should remove these commands from your code as soon as it is feasible.

If your code still contains older SQR commands, refer to this table as you replace them with their updated alternatives:

| Old Commands | Alternative to 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 |

Copyright © 1988-2007, Oracle. All rights reserved.

| Old Commands | Alternative to Use Instead |
|---|---|
| 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.

# Using Older SQR Commands

This section discusses each of the older SQR commands.

## BEGIN-REPORT

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use BEGIN-PROGRAM.

### Syntax

Use this syntax:

```
BEGIN-REPORT
```

### Description

Begins a report.

After processing the commands in the SETUP section, SQR starts running the program at the BEGIN-REPORT section. The PROGRAM section typically contains a list of DO commands, though you can also use other commands. This is the only required section in an SQR program.

### Example

For example:

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

## DATE-TIME

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the datenow function in the LET command.

### Syntax

Use this syntax:

```
DATE-TIME  position [date_format[col_var]]
```

 Copyright © 1988-2007, Oracle. All rights reserved.

## Description

Retrieves the current date and time from the local machine (or from the database for Oracle and some IBM DB2 platforms) and places it in the output file at the specified position or into a column variable.

If *col_var* is specified, *date_format* must be supplied and the current date and time is retrieved each time this command is run. Otherwise, the date is retrieved only at the program start, and the same date and time is printed each time.

If *date_format* is not specified, the date is returned in the default format for that database. The following table provides the default date-time formats for SQR-supported databases:

| Database | Default Date-Time Format |
|---|---|
| Oracle | DD-Mon-YYYY HH:MI PM |
| Informix | YYYY-MM-DD HH:MI YYYY-MM-DD HH:MI:SS.NNN |
| IBM DB2 | YYYY-MM-DD-HH:MI YYYY-MM-DD-HH:MI:SS.NNNNNN |
| 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,:

```
date-time () '' &date1
```

For those databases with only one default format, that format is always used in either of these cases.

See the table showing miscellaneous functions under the LET command for information about the valid edit mask format codes.

## Parameters

**position** Specifies the position for printing the date.

**date_format** Represents a string literal containing the date format mask.

**col_var** Places the retrieved date-time into a column variable rather than in the output file.

## Example

For example:

```
date-time (1,50)  MM/DD/YY
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
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 the $current-date reserved and datenow functions that are described in the table showing miscellaneous functions under the LET command.

See ALTER-LOCALE

# DECLARE PRINTER

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use DECLARE-LAYOUT and DECLARE-PRINTER.

## Syntax

Use this 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]
```

## Description

Specifies the printer type and sets printer characteristics.

Use the DECLARE PRINTER command either in 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 can put several DECLARE PRINTER commands in the body of the report and run the one that 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
```

 Copyright © 1988-2007, Oracle. All rights reserved.

SQR maps its line and column positions on the page by using a grid that is 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 by means of this grid. Because the characters in proportional fonts vary in width, a word or string may be wider than the horizontal space that 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 value in the DECLARE PRINTER command.

## Arguments

The following table describes the arguments for the DECLARE PRINTER command:

| Argument | Choice or Measure | Default Value | Description |
|---|---|---|---|
| TYPE | LINEPRINTER, POSTSCRIPT, HPLASERJET | LINEPRINTER | SQR creates output that is specific to each printer.<br>Line printer files generally contain ASCII characters and can be viewed by a text editor.<br>PostScript files contain ASCII characters, but you need to know PostScript to understand what will appear on the printer.<br>HP LaserJet files are binary files and cannot be edited or viewed. |
| ORIENTATION | PORTRAIT, LANDSCAPE | PORTRAIT | Portrait pages are printed vertically.<br>Landscape pages are printed horizontally.<br>Printing in landscape mode on HP LaserJet printers requires landscape fonts. |
| LEFT-MARGIN | inches | 0.5 | This argument does not apply to line printers.<br>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 line printers.<br>This is the amount of blank space to leave at the top of the page. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Argument | Choice or Measure | Default Value | Description |
|---|---|---|---|
| LINE-SIZE | points | 12 | This argument does not apply to line printers.<br>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 line printers.<br>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 line printers.<br>This is an alternate way of indicating the, in lines per inch rather than in points (as in LINE-SIZE). |
| CHARS-INCH | characters | 10 | This argument does not apply to line printers.<br>This is an alternate way of indicating the width of each SQR character column, in characters per inch rather than in points (as in CHAR-SIZE). |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Choice or Measure | Default Value | Description |
|---|---|---|---|
| POINT-SIZE | points | 12 | This argument does not apply to line printers.<br>This is the beginning size of the selected font. |
| FONT-TYPE | PROPORTIONAL, FIXED | Depends on the font | This argument applies only to HP LaserJet printers and must be specified only for font types that are not listed as being available for HP LaserJet printers in SQR in the previous DECLARE-PRINTER section. |
| SYMBOL-SET | HP defined sets | 0U | This argument applies only to HP LaserJet printers. The default value, 0U, is for the ASCII symbol set. Additional symbol sets exist.<br>See *HP LaserJet Technical Reference Manual.* |
| STARTUP-FILE | filename | POSTSCRI.STR | This argument applies only to PostScript printers. Use this to specify an alternate startup file. Unless otherwise specified, the default startup file is located in the directory that is specified by the environment variable SQRDIR. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Choice or Measure | Default Value | Description |
|---|---|---|---|
| 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.<br><br>See *HP LaserJet Technical Reference Manual.*<br><br>For PostScript printers, SQR supplies a list of fonts and arbitrary font number assignments in the postscri.str file. 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. See the fonts that are listed as being available for HP LaserJet printers in SQR in the previous DECLARE-PRINTER section. See also the table that lists the fonts that are available in the SQR postscri.str file, also in the DECLARE-PRINTER section. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Argument | Choice or Measure | Default Value | Description |
|---|---|---|---|
| BEFORE-BOLD | any string | None | The BEFORE-BOLD and AFTER- BOLD arguments are for line printers only. They specify the character string to turn bold on and off. If the string contains blank characters, enclose it in single quotation marks. To specify nonprintable characters, such as ESC, enclose the decimal value within angle brackets as follows: BEFORE-BOLD=<27>[r ! Turn on bold AFTER-BOLD=<27>[u ! Turn it off<br><br>These arguments work with the BOLD argument of the PRINT command. |
| AFTER-BOLD | any string | None | See BEFORE-BOLD. |

The font that 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 that are not listed as being available for HP LaserJet printers in SQR in the “DECLARE-PRINTER” section in the “SQR Command Reference” chapter, you must indicate the font style by using the FONT-TYPE argument; otherwise, the correct typeface cannot be selected by the printer.

# DECLARE PROCEDURE

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use DECLARE-PROCEDURE.

## Syntax

Use this syntax:

```
DECLARE PROCEDURE
 [BEFORE-REPORT=procedure_name]
 [AFTER-REPORT=procedure_name]
 [BEFORE-PAGE=procedure_name]
 [AFTER-PAGE=procedure_name]
```

## Description

Defines specific event procedures.

Copyright © 1988-2007, Oracle. All rights reserved. 

Use the DECLARE PROCEDURE command 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 running. The referenced procedures do not take any arguments; however, they may be local. In addition, they can print only into the body of the report; that is, they cannot print into the header and footer areas.

### Parameters

**BEFORE-REPORT** Specifies a procedure to run at the time of the first PRINT command. For example, you use this to create a report heading.

**AFTER-REPORT** Specifies a procedure to run just before the report file is closed at the end of the report. Use this to print totals or other closing summary information. If no report was generated, the procedure does not run.

**BEFORE-PAGE** Specifies a procedure to run at the beginning of every page, just before the first PRINT command for the page. For example, you use this to set up page totals.

**AFTER-PAGE** Specifies a procedure to run just before each page is written to the file. For example, you use this to display page totals.

### Example

For example:

```
declare procedure
   before-page=page_setup
   after-page=page_totals
```

## DOLLAR-SYMBOL

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use ALTER-LOCALE.

### Syntax

Use this syntax:

```
DOLLAR-SYMBOL  new_symbol
```

### Description

Redefines the currency symbol within numeric edit masks .

The dollar sign ($) is the default currency symbol for coding edit masks in the program that prints on report listings. DOLLAR-SYMBOL provides a way to change that symbol for both the edit mask and for printing.

To change the symbol that prints on the report, use MONEY-SYMBOL in the PROCEDURE section. Use DOLLAR-SYMBOL and MONEY-SYMBOL together to configure SQR programs and the reports that they produce.

Use this command only in the SETUP section.

 Copyright © 1988-2007, Oracle. All rights reserved.

**Note.** The MONEY-SYMBOL command has the same effect as these options of the ALTER-LOCALE command: MONEY-SIGN and MONEY-SIGN-LOCATION=LEFT.

The following table lists the characters that DOLLAR-SYMBOL cannot take:

| Type | Character |
|---|---|
| Numbers | 0 8 9 |
| Alphabetical | b e n r v B E N R V |
| Symbols | . , - + ! * _ ‘ < > ( ) |

### Parameters

**new_symbol** Specifies a new, single character to be used in edit masks instead of the dollar sign ($).

### Example

For example:

```
begin-setup
 dollar-symbol £      ! Define £ as the currency symbol
end-setup
begin-procedure
...
print #amount () edit £££,999.99
...
end-procedure
```

In the previous example, if you used the dollar sign in the edit mask after defining the dollar symbol as £, the following error message appears:

```
Bad numeric 'edit' format: $$$,999.99
```

See the ALTER-LOCALE command for a description of other locale-specific parameters.

## GRAPHIC FONT

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use ALTER-PRINTER and DECLARE-PRINTER to set the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

### Syntax

Use this syntax:

```
GRAPHIC  ()
FONT  { font_number_int_lit|_var }
```

Copyright © 1988-2007, Oracle. All rights reserved. 

```
[point_size_int_lit|_var[{1|0}
[pitch_int_lit|_var]]]
```

### Description

Changes a font.

### Parameters

**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 *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 recognize. (See the fonts that are listed as being available for HP LaserJet printers in SQR in the DECLARE-PRINTER section in the chapter "SQR Command Reference.") A 1 indicates a proportional font, and a 0 indicates a fixed-pitch font. The default is proportional.

**pitch** If the specified font is fixed pitch, also indicate the pitch in characters per inch.

### Example

For example:

```
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 ALTER-PRINTER and DECLARE-PRINTER for information about setting and changing the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

## MONEY-SYMBOL

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use ALTER-LOCALE.

### Syntax

Use this syntax:

```
MONEY-SYMBOL new_symbol
```

### Description

Redefines the currency symbol to be printed.

To change the symbol that prints on the report, use the MONEY-SYMBOL in the PROCEDURE section. When the MONEY-SYMBOL is set, that value is used until the next MONEY-SYMBOL command runs.

Use DOLLAR-SYMBOL and MONEY-SYMBOL together to configure SQR programs and the reports that they produce.

 Copyright © 1988-2007, Oracle. All rights reserved.

To indicate a nonedit character, surround its decimal value with angle brackets (<>). See the table under the DOLLAR-SYMBOL command for characters that cannot be used with MONEY-SYMBOL.

---

**Note.** The MONEY-SYMBOL command has the same effect as the MONEY-SIGN and MONEY-SIGN-LOCATION=LEFT options of the ALTER- LOCALE command.

---

### Parameters

**new_symbol** Specifies a new, single character to replace the dollar sign ($) or DOLLAR-SYMBOL character on the printed report.

### Example

For example:

```
begin-setup
 dollar-symbol £! Define £ as the

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

See the DOLLAR-SYMBOL and ALTER-LOCALE commands.

## NO-FORMFEED

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the FORMFEED parameter of the DECLARE-LAYOUT command.

### Syntax

Use this syntax:

```
NO-FORMFEED
```

### Description

Prevents form-feed characters from being written to the output file.

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.

Copyright © 1988-2007, Oracle. All rights reserved. 

### Example

For example:

```
begin-setup
  no-formfeed
end-setup
```

### See Also

See the FORMFEED qualifier in DECLARE-LAYOUT.

## PAGE-SIZE

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the MAX-LINES and MAX-COLUMNS parameters of the DECLARE-LAYOUT command.

### Syntax

Use this syntax:

```
PAGE-SIZE  page_depth_num_lit  page_width_num_lit
```

### Description

Sets the page size.

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 that is 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 can define a page to be 1 line deep and 4,000 characters wide, which you can use for writing large flat files, perhaps for copying to magnetic tape. Each time a NEW-PAGE occurs, one record is written. Use the NO-FORMFEED command in the SETUP section to suppress form-feed characters between pages.

Use a page width that is at least one character larger than the rightmost 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 because SQR trims trailing blanks on each line before writing the report file.

Determine the size of the internal page buffer that stores a complete page in memory by multiplying the page depth by the width in the PAGE-SIZE command. For personal computers, the page buffer is limited to 64K bytes. On other computers, the page buffer is limited only by the amount of memory that is available.

 Copyright © 1988-2007, Oracle. All rights reserved.

### Example

For example:

```
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

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use PRINT … CODE-PRINTER.

If you use CODE, the sequence is assumed to be for the printer type that is specified in the DECLARE-REPORT or for the default printer, if none is specified.

### Syntax

Use this syntax:

```
PRINT . . . CODE
```

### Parameters

| | |
|---|---|
| **CODE** | CODE is a qualifier that may be discontinued in a future release. Use CODE-PRINTER instead. |

## PRINTER-DEINIT

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the RESET-STRING parameter of the DECLARE-PRINTER command.

### Syntax

Use this syntax:

```
PRINTER-DEINIT  initialization_string
```

### Description

Sends control or other characters to the printer at the end of a report.

Specify nondisplay characters by placing their decimal values inside angle 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 printers. It has limited functionality with HP LaserJet and PostScript printers.

### Example

For example:

Copyright © 1988-2007, Oracle. All rights reserved. 

```
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 the table listing miscellaneous functions under the LET command in the chapter "SQR Command Reference."

## PRINTER-INIT

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the INIT-STRING parameter of the DECLARE-PRINTER command.

### Syntax

Use this syntax:

```
PRINTER-INIT initialization_string
```

### Description

Sends control or other characters to the printer at the beginning of a report.

Specify nondisplay characters by placing their decimal values inside angle 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 printers. It has limited functionality with HP LaserJet and PostScript printers.

### Example

For example:

```
begin-setup
  printer-init<27>[7J ! Set the printer
end-setup
```

### See Also

See the ENCODE command for another method of printing nondisplay characters. See the chr function in the table listing miscellaneous functions under the LET command in the chapter "SQR Command Reference."

 Copyright © 1988-2007, Oracle. All rights reserved.

CHAPTER 7

# Using the PSSQR.INI File and the PSSQR Command Line

This chapter provides an overview of the pssqr.ini file and discusses the following topics:

- pssqr.ini installation process.
- Default settings section.
- Processing-limits section.
- Environment sections.
- Locale section.
- Fonts section.
- HTML-Images section.
- PDF Fonts section.
- PDF Fonts: Exclusion Ranges Section
- TrueType Fonts Section
- Enhanced-HTML section.
- Colors section.
- pssqr.exe command-line options.

## Installing PSSQR.INI

The pssqr.ini file is the initialization file for SQR for PeopleSoft . SQR uses the settings and parameters in this file during the compile and execution phases.

The installation process installs a default initialization file called pssqr.ini, which is located in the SQR directory under the <PS_HOME> directory. The installation process also installs files named pssqr<language_cd>.ini, which are used to create language-specific configurations.

### Microsoft Windows Platforms

On Microsoft Windows, SQR looks for the initialization file in the following locations, in order:

1. The file name specified by the -ZIF{file} command-line flag.
2. The directory in which the executable image resides.
3. The Microsoft Windows system directory.

Copyright © 1988-2007, Oracle. All rights reserved.

### z/OS

On z/OS, SQRINI is required during initialization. The dataset member &PSHLQ..SQRSRC(PSSQRINI) must be specified in the JCL to start the SQR process.

### All Other Platforms

On all other platforms, SQR looks for the initialization file in the following locations, in order:

1. The file name specified by the -ZIF{file} command-line flag.
2. The current working directory.
3. The directory specified using the SQRDIR environment variable.

   SQR automatically sets up SQRDIR.

You can make changes or additions to the pssqr.ini file.

This example shows the format of the pssqr.ini file:

```
; Comments are lines that 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

This table describes the SQR default settings:

 Copyright © 1988-2007, Oracle. All rights reserved.

| Entry | Value | Description |
|---|---|---|
| `ForceSpaceAfterComma=⇒ {TRUE\|FALSE}` | TRUE \| FALSE | The default setting is FALSE.<br>DB2 only: Forces a space after every comma not in a literal value. This is to support DECIMAL=COMMA setting on the OS390. |
| `ShowDBWarnings=⇒ {TRUE\|FALSE}` | TRUE \| FALSE | The default setting is FALSE.<br>DB2 only: If set to TRUE, SQR displays database warnings in the SQR output file. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Entry | Value | Description |
|---|---|---|
| `AllowDateAsChar=⇒ {TRUE\|FALSE}` | TRUE \| FALSE | The default setting is FALSE.<br><br>By default, SQR produces an error when a dynamic column specification does not match the database definition of the column. That is, character equals character, date equals date, and numeric equals numeric.<br><br>When this value is set to TRUE, SQR allows character to be equal to either character or date columns.<br><br>When a date column is type cast to be a character, SQR creates the string according to the following rules:<br><br>For DATETIME columns, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this has not been set, SQR uses the first database-dependent format as listed in the table showing default database formats in the Edit section under the PRINT command.<br><br>For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_ FORMAT setting. If this has not been set, the format listed in the table showing default database formats in the Edit section under the PRINT command.<br><br>For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_ FORMAT setting. If this has not been set, SQR uses the format as listed the table showing the TIME column formats in the Edit section under the PRINT command.<br><br>In the following example, AllowDateAsChar= True. This allows $Col1 to be either date or text.<br><br>`Begin-Select`<br>`  [$Col1] &col1=Text`<br>`  [$Col2] &col2=Date`<br>`  [$Col3] &col3=Number`<br>`  from MyTable`<br>`End-Select` |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Entry | Value | Description |
|---|---|---|
| `OUTPUT-FILE-MODE` | LONG \| SHORT | Specifies the file name convention used for HTML output. SHORT specifies DOS style (8.3) and LONG specifies UNIX\Linux style (non 8.3). The default is LONG. This setting is ignored on 16-bit platforms. The DECLARE-TOC and -BURST commands force Output-File-Mode = LONG.<br><br>The following represents the file formats for UNIX\Linux, DOS, and Windows.<br><br>*SQR and SQRT*: {Program} is the name of the SQR/SQRT file without the extension. |
| `OUTPUT-FILE-MODE`<br><br>(continued) | LONG \| SHORT | (Continued)<br><br>For Output-File-Mode = SHORT, SQR-generated file names are limited to a DOS 8.3 format.<br><br>• Output file = {program}.lis for first, and {program}.lnn for multi-reports<br>• SPF file = {program}.spf for first, and {program}.snn for multi-reports<br>• PDF file = {program}.pdf for first; and {program}.pnn for multi-reports<br>• HTM file = {program}.htm for “frame, and {program}.hbb for report bodies<br>• GIF file={program}.gxx for all reports<br><br>bb ranges from 00 to 99 and represents the report number.<br><br>nn ranges from 01 to 99 and represents the report number.<br><br>xx ranges from 00 to ZZ and represents the graphic number. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Entry | Value | Description |
|---|---|---|
| `OUTPUT-FILE-MODE`<br>(continued) | LONG \| SHORT | (Continued)<br>For Output-File-Mode = LONG, SQR-generated file names are not constrained to a DOS 8.3 format. {output}={program} of first report and {program}_nn for multi-reports.<br>• Output file = {output}..lis<br>• SPF file = {output}..spf<br>• PDF file = {output}..pdf<br>• GIF file = {output}_zz..spf<br>• HTM files = {output}..htm, {output}_bb..htm, {output}_frm..htm, {output}_toc./htm, {output}_nav.htm<br>bb ranges from 01 to ZHJOZI and represents the bursted page group number.<br>nn ranges from 01 to 99 and represents the report number.<br>zz ranges from 00 to ZHJOZI and represent the graphic number. |
| `OUTPUT-FILE-MODE`<br>(continued) | LONG \| SHORT | (Continued)<br>SQRP: {file name} is the name of the SPF file without the extension<br>For Output-File-Mode = SHORT, SQR-generated file names are limited to a DOS 8.3 format.<br>• Output file = {file name}.lis<br>• GIF file = {file name}.gxx<br>• PDF file = {file name}.pdf<br>• HTM file = .htm and {file name}.h00<br>xx ranges from 00 to ZZ and represents the graphic number. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Entry | Value | Description |
|---|---|---|
| `OUTPUT-FILE-MODE`<br>(continued) | LONG \| SHORT | (Continued)<br>For Output-File-Mode = LONG, SQR-generated file names are not limited to a DOS 8.3 format.<br>• Output file = {file name}.lis<br>• PDF file = {file name}.pdf<br>• GIF file = {file name}_zz.spf<br>• HTM files = {file name}.htm, {file name}_bb.htm, {file name}_frm.HTM, {file name}_toc.htm {file name}_nav.htm<br>bb ranges from 01 to ZHJOZI and represents the bursted page group number.<br>zz ranges from 00 to ZHJOZI and represents the graphic number. |
| `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. The value of SYSTEM is used to reference the default locale.<br>See Chapter 2, "SQR Command Reference," ALTER-LOCALE, page 25. |
| `DEFAULT-NUMERIC` | INTEGER \| FLOAT \| DECIMAL[(p)] \| V30 | Specifies the default numeric type for variables. The command line flag -DNT overrides this setting. Similarly, the DECLARE-VARIABLE command overrides this setting.<br>See Chapter 2, "SQR Command Reference," DECLARE-VARIABLE, page 99. |
| `OutputFormFeedWithDashD=⇒ {TRUE\|FALSE}` | TRUE \| FALSE | The default value is FALSE.<br>When set to TRUE, the -Dnn command line flag outputs the Form-Feed character that denotes a page break. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Entry | Value | Description |
|---|---|---|
| `OutputTwoDigitYearWarningMsg=⇒ {TRUE\|FALSE}` | TRUE \| FALSE | The default value is TRUE.<br>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. |
| `UseY2kCenturyAlgorithm=⇒ {TRUE\|FALSE}` | TRUE \| FALSE | The default value is FALSE.<br>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 process 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.

# Processing-Limits Section

The Processing-Limits section is used to define the sizes and limitations of some of the internal structures used by SQR, which has a direct impact on memory requirements. The entries are the same as those used in the file specified with the -MFILE command-line flag. If the -MFILE command-line flag is used, the Processing-Limits section of the file is not processed.

| Entry | Default Value | Maximum Value | Entry Size | Description |
|---|---|---|---|---|
| `BREAKS` | 100 | 64K-1 | 4 | Number of BREAK arguments allowed for each EVALUATE or IF command. |
| `DYNAMICARGS` | 70 | 32K-1 | 14 | Maximum number of dynamic SQL arguments. |
| `EXPRESSIONSP⇒ ⇒ ⇒ ACE` | 8192 | 64K-1 | 1 | Maximum length, in bytes, of temporary string storage used during LET operations. |
| `FORWARDREFS` | 200 | 32K-1 | 8 | Maximum number of column forward references. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Entry | Default Value | Maximum Value | Entry Size | Description |
|---|---|---|---|---|
| LONGSPACE | 32K-2 | 32K-2 | 1 | Maximum buffer size to transfer text and image data in bytes. |
| ONBREAKS | 30 | 64K-1 | 8 | Maximum number of ON-BREAK LEVEL=values per SET. |
| POSITIONS | 1800 | 64K-1 | 14 | Maximum number of placement parameters, for example, (10,5,30). |
| PROGLINEPARS | 18000 | 64K-1 | 2 | Maximum number of arguments for all program lines. This value is generally 3 or 4 times the value set for PROGLINES. |
| PROGLINES | 5000 | 32K-1 | 8 | Maximum number of program lines (SQR commands). |
| QUERIES | 60 | 32K-1 | 60 | Maximum number of BEGIN-SQL and BEGIN-SELECT paragraphs. This is database dependent and can vary. This size is used as a close approximation. |
| QUERYARGS | 240 | 64K-1 | 6 | Maximum number of arguments (bind variables) for all SQL or SELECT statements. The number of arguments required is one more than the number used in your report file. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Entry | Default Value | Maximum Value | Entry Size | Description |
|---|---|---|---|---|
| SQLSIZE | 4000 | 64K-1 | 1 | Maximum length of a SQL statement in characters. |
| STRINGSPACE | 15000 | 64K-1 | 1 | Maximum size of string space for program line arguments, in bytes. |
| SUBVARS | 100 | 32K-1 | 8 | Maximum number of runtime substitution variables. |
| VARIABLES | 1500 | 64K-1 | 18 | Maximum number of variables *(string, float, integer, decimal)*, literal values, and database columns. Add 4 to the entry size for Informix. |
| WHENS | 70 | 64K-1 | 4 | Maximum number of WHEN arguments allowed for each EVALUATE command. |

The maximum value refers to the number of entries allowed as shown in the previous table; however, limits are lower for 32-bit machines. In any case, SQR indicates the limit if you exceed it.

In addition to increasing the sizes, you may also lower them to decrease the amount of memory used. This might be advantageous, for example, for certain applications running in the PC environment, where memory is limited.

# Environment Sections

The Environment sections { Common | Oracle | Informix | ODBC | DB2 | RDB | Sybase ] } define the environment variables 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

The environment variables that can be set are *SQRDIR, SQRFLAGS,* and *DSQUERY.* On Microsoft 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. *SQRFLAGS* specifies the default command-line flags for all invocations of SQR. *DSQUERY* (Sybase only) identifies the default Sybase server to use.

On Microsoft 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 being 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\sqrext1.dll=
c:\sqrexts\sqrext2.dll=
c:\sqrexts\sqrext3.dll=
```

Any new extension DLLs containing new user functions must be listed in the SQR Extension section in sqr.ini.

See *Enterprise PeopleTools 8.49 PeopleBook: SQR for PeopleSoft Developers*, “Using Interoperability Features”.

For Oracle on Windows, SQR uses dynamic binding of Oracle routines. When SQR attempts 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).
- ORANT71.DLL (Oracle supplied).

| Entry | Value | Description |
|---|---|---|
| SQRParseSQLEnable= {TRUE\|FALSE} | TRUE \| FALSE | The default setting is FALSE. If SQRParseSQLEnable=TRUE is added to the PSSQR.INI file (in the Common or Oracle stanza). SQR performs SQL parsing.<br><br>**Note.** The function required to perform this parsing is not a supported function on Oracle 8.X but does appear to function. If a customer wishes to assume the risk of using a feature that might not be fixed if a problem is found, then they can turn on this parameter. Oracle 9.X customers can use this feature without any problem, although there is a performance hit of between 2-10%. |

Copyright © 1988-2007, Oracle. All rights reserved. 

# Locale Section

This section specifies the default settings for the locale identified by locale-name (which can consist of A-Z, 0-9, hyphen, and underscore). A number of locales are predefined in the pssqr.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 runtime using the ALTER-LOCALE command. The entries for a locale section are described in the following table.

**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 runtime.

See Chapter 6, "Avoiding Older SQR Commands," page 269.

| 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. |
| `MONEY-SIGN` | Specifies the characters to replace the ($) edit character. |
| `MONEY-SIGN-LOCATION` | Specifies the location of the MONEY-SIGN character. Valid values are LEFT and RIGHT. |
| `THOUSAND-SEPARATOR` | Specifies the character to replace the comma (,) edit character. |
| `DECIMAL-SEPARATOR` | Specifies the character to replace the period (.) edit character. |
| `DATE-SEPARATOR` | Specifies the character to replace the slash (/) character. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Entry | Description |
|---|---|
| TIME-SEPARATOR | Specifies the character to replace the colon (:) character. |
| EDIT-OPTION-NA | Specifies the characters to replace the na option. |
| EDIT-OPTION-AM | Specifies the characters to replace AM |
| EDIT-OPTION-PM | Specifies the characters to replace PM. |
| EDIT-OPTION-AD | Specifies the characters to replace AD. |
| EDIT-OPTION-BC | Specifies the characters 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.<br><br>Use UPPER and LOWER to force the output to either all uppercase or lowercase, ignoring the case of the format code in the edit mask.<br><br>Use EDIT to follow the case as specified with the format code in the edit mask.<br><br>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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Entry | Description |
|---|---|
| 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.<br><br>Use UPPER and LOWER to force the output to either all uppercase or lowercase, ignoring the case of the format code in the edit mask.<br><br>Use EDIT to follow the case as specified with the format code in the edit mask.<br><br>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. |

# Fonts Section

The Fonts section lists the fonts available to SQR when printing on Microsoft Windows printer devices (using the -PRINTER:WP command-line flag), viewing SPF output using SPF Viewer on Microsoft Windows (SQRWV), and creating style sheet for Enhanced HTML output. This section does not apply to PostScript or HP LaserJet printer types.

See Chapter 2, “SQR Command Reference,” DECLARE-PRINTER, page 85 and Chapter 6, “Avoiding Older SQR Commands,” page 269.

## 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.

 Copyright © 1988-2007, Oracle. All rights reserved.

Using the font number, commands such as ALTER-PRINTER and DECLARE-PRINTER can reference a particular font style.

## Specifying Character Sets in Windows

On Microsoft Windows, you can use the CharacterSet entry either to determine the Microsoft Windows default character set or to specify a character set. This enables you to print any standard character set to a Windows printer (-PRINTER:WP) or to view an SPF file that displays the appropriate character set.

The syntax is:

```
CharacterSet=DEFAULT|AUTO|character_set
```

The arguments are:

- DEFAULT reflects current SQR functionality.
- AUTO automatically determines the default character set of the Microsoft Windows installation and uses the default set when generating reports.

  CharacterSet specifies one of these keywords: ANSI, ARABIC, BALTIC, CHINESEBIG5, EASTEUROPE, GB2312, GREEK, HANGUL, HEBREW, JOHAB, MAC, OEM, RUSSIAN, SHIFTJIS, SYMBOL, THAI, TURKISH, VIETNAMESE.

# HTML-Images Section

The HTML-Images section defines the parameters that SQR uses when generating HTML report output files.

| Entry | Value | DefaultValue | 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 |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Entry | Value | DefaultValue | Description |
|---|---|---|---|
| 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. |
| Navbar Background | NAME | | Background image of navigation bar. |

**Note.** SQR does not perform any validation of the graphic image file names provided. The user is responsible for ensuring that the graphic image files are in a location that the browser can access.

# PDF Fonts Section

The PDF Fonts section lists the available fonts for SQR when printing using the -PRINTER:PD command-line flag. Fonts specified are case sensitive.

Unlike the Fonts section, the PDF Fonts section takes a list of fonts mapped to a single font number. You can specify up to 10 fonts for a single font number. Depending on the character to print, SQR determines the font to use from the list. The following is the syntax of this font list:

```
Font Number=Font1, Font2, Font3 ...
```

The list is ordered by priority from the left. If Font1 has a glyph (an image of a character) for the character you want to print, it is used, but if Font1 does not have a glyph, Font2 is checked and used if it has a glyph for that character.

Font Number is a decimal number that is used to specify fonts in the SQR program using statements like DEFINE-PRINTER or ALTER-PRINTER.

Font numbers that are multiples of 100 (300, 3200, and so on) are recognized as a bold version of the base font. For example, font 300 is the bold version of Font3.

Font numbers 30 to 39 are recognized as italic fonts, and multiples of 30 to 39 (3000, 3100, and so on) are bold italic versions of the font. If you assign nonbold fonts for a font number recognized as bold, (such as 600), the text is printed in bold.

Font1, Font2, and Font3 represent the font name. You can use Adobe Reader core fonts, Adobe Reader Asian Font Pack fonts, or TrueType fonts.

Adobe Reader core fonts are the fonts that Adobe Reader natively supports. The following is a list of the core fonts:

- Courier

 Copyright © 1988-2007, Oracle. All rights reserved.

- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Symbol
- ZapfDingbats

Adobe Reader Asian Font Pack fonts are the font packages Adobe provides as add-on to Adobe Reader for the purpose of viewing Asian text. You can download these fonts from the Adobe website, or if you use Adobe Reader 6, the Asian Font Pack automatically downloads when you open a PDF file that contains a font from the Asian Font Pack. Alternatively, you can use the Adobe Reader localized to any of the Japanese, Chinese, or Korean languages. These localized versions come with the same fonts that are included in the Adobe Asian Font Pack. The following is a list of the Asian Font Pack fonts used with SQR:

- HeiseiKakuGo-W5 (Japanese)
- HeiseiMin-W3 (Japanese)
- MHei-Medium (Traditional Chinese)
- MSung-Light (Traditional Chinese)
- STSong-Light (Simplified Chinese)
- HYGoThic-Medium (Korean)
- HYSMyeongJo-Medium (Korean)

In addition, you can use TrueType/OpenType fonts that exist on the machine where you run SQR. If you use TrueType fonts for PDF output, they will be always embedded as subset of the original font. To use TrueType fonts for PDF output, you need to specify the mapping between the font name and the font file path in the [TrueType Fonts] section, and then include the font name in the font list under the [PDF Fonts] section. See below more information about the [TrueType Fonts] section.

See Chapter 7, “Using the PSSQR.INI File and the PSSQR Command Line,” TrueType Font Section, page 302.

Copyright © 1988-2007, Oracle. All rights reserved. 

# PDF Fonts: Exclusion Ranges Section

The PDF Fonts: Exclusion Ranges section defines character ranges you want to exclude from the range a font covers. Without an exclusion range, SQR uses all of the character ranges a font covers, and then checks the font listed in the next priority only when it does not cover a character. By specifying an exclusion range, you can use a font in a lower priority for a specific character or a range of characters. This mechanism allows users to control which font to use down to a single character level.

The syntax in this section is as follows:

```
Font Number=Font1 Exclusion Range, Font2 Exclusion Range, ...
```

Exclusion ranges apply to the list of fonts defined in the PDF Fonts section. For example, if you have a PDF Fonts section like the following:

```
3=Courier,Cumberland,HeiseiKakuGo-W5,...
```

In addition, if you define the PDF Fonts: Exclusion Ranges section like this

```
3=0x20AC,0x3070-0x30FF,,
```

then you are defining that Euro currency symbol (0x20AC in Unicode) is rendered using the Cumberland font, even though Courier has it. In addition, Greek characters (from 0x3070 to 0x30FF in Unicode) are rendered using HeiseiKakuGo-W5 font, even though Cumberland has them.

Each Exclusion range is specified using Unicode (UCS-2) codepoint, in decimal or in hexadecimal, in the following ways:

0x20AC (single character)

160-255 (range)

0x20AC|0x00A0-0x00FF (multiple characters and ranges)

Hexadecimal numbers need to be preceded by 0x; otherwise, they are recognized as decimal.

# TrueType Font Section

The TrueType Font section defines the mapping of font name, which is an alias SQR uses internally to look up the font, and physical font file name. You need to specify the font in the full path, unless you have placed fonts in a directory specified in the Font Path parameter, or in a Windows font folder if you are running SQR on Windows.

The syntax is:

```
font name=fontfile path
```

For example, if you have the font courier.ttf in the c:\user\fonts directory, you set the following in this section:

```
CourierNew=c:\user\fonts\courier.ttf
```

Font name can be any string that is convenient for you to identify the font, and does not need correspond to the internal name of the font. You should not use the same font name that is used for Adobe core fonts or for Asian Font Pack fonts. If you set the same name for any of Adobe core fonts or Asian Font Pack fonts, the TrueType font is used.

 Copyright © 1988-2007, Oracle. All rights reserved.

**Note.** TrueType font embedding is now supported by SQR running on z/OS. However, the Font Path parameter under the TrueType Font section is not supported for z/OS. If you're on z/OS, you need to specify the full path to each entry.

If you have a TrueType Collection (TTC) file, you will also need to specify the font number to access the specific font included in the collection. For example, if MS P Mincho is included in the TrueType Collection file msmincho.ttc, then you will need to specify:

```
MSPMincho=c:\winnt\fonts\msmincho.ttc,1
```

The font number within the TrueType collection starts at 0. msmincho.ttc contains MS Mincho and MS P Mincho in this order; thus the number 0 represents MS Mincho and 1 represents MS P Mincho. If you do not specify font number, SQR uses font with font number 0 from the TrueType Collection file.

**Note.** You can use only Microsoft type of Unicode based TrueType/OpenType fonts with SQR. SQR requires TrueType/OpenType font to have CMAP table with Platform ID 3 (Microsoft), Encoding ID 0 (Symbol), 4 (UCS-2), or 10 (UCS-4) and table format 4 or 12. SQR does not support OpenType fonts with CFF (Postscript) outline. You can use OpenType fonts with TrueType outlines, but SQR does not make use of advanced layout features provided with OpenType font.

# Enhanced-HTML Section

The Enhanced-HTML section is used to define various default actions that SQR takes when generating HTML output using the -EH command-line flag.

| Entry | Value | Description |
|---|---|---|
| Browser={ALL\|IE\|NETS CAPE} | ALL\|IE\|NETSCAPE | When set to ALL, the generated HTML automatically determines which browser is being used and invokes the proper browser-specific file.<br><br>When set to IE, the generated HTML is designed for Internet Explorer.<br><br>When set to NETSCAPE, the generated HTML is designed for Netscape.<br><br>The default action is to generate HTML suitable for all browsers. |
| Language={English\|French\| German\| Portuguese\|Spanish\|Japanese\| Simplified Chinese\|Korean} | English\|French\| German\| Portuguese\|Spanish\|Japanese\| Simplified Chinese\|Korean | Sets the language used for the HTML navigation bar.<br><br>The default setting is English. |
| FullHTML={TRUE \| FALSE} | TRUE \| FALSE | When set to TRUE, HTML 3.2 is generated. When set to FALSE, HTML 3.0 is generated. The default setting is FALSE. |

Copyright © 1988-2007, Oracle. All rights reserved. 

# Colors Section

The Colors section defines the default colors that you can use in your SQRs. Enter the default colors in this format:

```
[Colors]
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)

# Using PSSQR.EXE Command-Line Options

PeopleSoft provides a shell called pssqr.exe that extends SQR to handle the submission of SQR programs under Microsoft Windows and UNIX/Linux operating systems. Pssqr.exe implements a process that ensures that output is sent to the appropriate destination in a way that is consistent across platforms.

**Note.** PeopleSoft does not support running pssqr.exe directly from a DOS or Unix/Linux command line. Pssqr.exe is a wrapper program used by PeopleSoft Process Scheduler to run SQR reports; it is not designed to run manually outside of Process Scheduler.

Pssqr.exe does not run on OS/390, however, all output formats are supported on all platforms.

Pssqr.exe provides the following features:

 Copyright © 1988-2007, Oracle. All rights reserved.

- Expanded output formats: SQR Viewer, HTML, PDF, CSV (Spreadsheet Standard), HP, Postscript, Line Printer.
- Expanded printer format: Microsoft Windows Default Printer (Win32 only), HP, Postscript, Line Printer.
- Enhanced delivery of reports to printers: PSSQR sends reports to the printer instead of SQR, which resolves issues that can be encountered on non-Microsoft networks.
- File output and logs have unique names. If the process instance is sent, the file names will be <SQR Program>_<Instance>.xxx.

  If not, then the log names will be <SQR Program>_<timestamp>.xxx.
- Common command line interface for both Microsoft Windows and UNIX/Linux.
- Reads Configuration Manager settings to determine the flags (SQRFLAGS) to use.
- Support for multiple report output.

  Default values come from Configuration Manager if run from the command line.

| Command-Line Parameter | Description |
|---|---|
| -CT | Database type. Valid values are ORACLE, DB2, DB2390, INFORMIX, MICROSFT, SYBASE. |
| -CS | Server name. |
| -COWN | Database owner. This applies to OS/390 only. |
| -CD | Database name. |
| -CA | Access ID. |
| -CAP | Access password. |
| -RP | Program name. |
| -I | Process instance. |
| -R | Run control ID. |
| -CO | User ID. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Command-Line Parameter | Description |
|---|---|
| -OT | Output type:<br>• 2 (File)<br>• 3 (Printer)<br>• 4 (Window) |
| -OF | Output format. Column headings refer to the -OT parameter.<br>• 2 (Adobe Acrobat)<br>• 3 (Comma delimited)<br>• 4 (HP format)<br>• 5 (HTML documents)<br>• 6 (Line printer)<br>• 10 (Postscript)<br>• 13 (SQRiBE portable)<br>• 14 (Text files)<br>• 15 (Windows default printer) |
| -OP | Output destination. |
| -MR | Multiple report output. Values: # (maximum 99) |
| -AF | Additional flags. |
| -AP | Additional information. |
| -TR | Enable trace. |
| -DL | Display log file. |
| -FILE | Enter the name of a parameter file containing the parameters you want to pass to PSSQR. The system deletes the file immediately after use. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Command-Line Parameter | Description |
|---|---|
| -LG | Language Code. Takes a three-letter language code used across PeopleSoft systems. This flag is used to determine which language version of the PSSQR.INI file to use. If you specify LG JPN, then pssqr.exe picks up the Japanese configuration file (PSSQRJPN.INI) instead of PSSQR.INI. This parameter takes effect only when the %LANGUAGE_CD% variable is used for specifying the PSSQR.INI file in PSSQRFLAGS entry of the Process Scheduler configuration or Configuration Manager. |
| -LPFLAGS "<flags>" | UNIX\Linux only : Overides the flags passed to the lp command. |
| -DEBUGLP ON | Displays command used to print the SQR report. |

Copyright © 1988-2007, Oracle. All rights reserved.

 Copyright © 1988-2007, Oracle. All rights reserved.

# APPENDIX A

# SQR Messages

This appendix describes all the messages produced by SQR for PeopleSoft. It discusses:

- Unnumbered messages
- Numbered messages

**Note.** Two digits (‘nn) appear as replacement markers in the messages. Descriptions of these replacement markers are listed in the message itself. The messages contain the proper value when they appear on the screen.

## Unnumbered Messages

This table lists all unnumbered SQR for PeopleSoft messages:

| Message | Description |
|---|---|
| Out of memory. | This error occurs when a call to the C routine ’malloc()’ fails.<br>(32-bit machines) Use the -Mfile command-line flag to reduce some of the different memory requirements. Remove unneeded TSRs.<br>(UNIX\Linux) Increase the size of the system swap file. |
| No cursors defined. | This message is issued from the -S command-line flag. The SQR program did not contain any commands that required a database cursor. |
| Not processed due to report errors. | This message is issued from the -S command-line flag. SQR cannot provide information about the cursor due to errors in the program. |
| Enter ‘01‘02 | This message prompts the user to type the value to be assigned to the specified variable.<br>‘01 = First character of the variable name.<br>‘02 = Rest of the variable name. |
| NOPROMPT used - Enter value below | (Microsoft Windows) This message appears when an INPUT command is defined with the NOPROMPT argument. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Message | Description |
|---|---|
| Enter ‘01 | This message prompts the user to enter the value to be assigned to the specified substitution variable.<br>‘01 = Name of the substitution variable |
| Enter this run’s parameters: | This message prompts the user to enter the values for the parameters defined in the program. |
| Error on line ‘01:<br>‘02 | SQR detected an error while processing the report file. Correct the error and rerun.<br>‘01 = Source line number.<br>‘02 = Source line. |
| Error in include file "‘01" on line ‘02:<br>‘03 | SQR detected an error while processing the report file. Correct the error and rerun.<br>‘01 = Name of the include file.<br>‘02 = Source line number.<br>‘03 = Source line. |
| Warning on line ‘01:<br>‘02 | SQR detected a nonfatal error while processing the report file.<br>‘01 = Source line number.<br>‘02 = Source line. |
| Warning in include file "‘01" on line ‘02:<br>‘03 | SQR detected a nonfatal error while processing the report file.<br>‘01 = Name of the include file.<br>‘02 = Source line number.<br>‘03 = Source line. |
| Type RETURN for more, C to continue w/o display, X to exit run: | This informational message is used in conjunction with the -D command-line flag. |
| Error at: ‘01<br>Loading Oracle DLL Failed!!! | (Oracle) This title for the dialog box informs the user that SQR could not load the Oracle DLL. |
| Errors were found in the program file. | This error message tells the user that there were errors in the program. The user can correct the errors and rerun. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Message | Description |
|---|---|
| Errors were found during the program run. | This error message tells the user to there were errors running the program. The user can correct the errors and rerun. |
| ‘01: End of Run. | This is an informational message.<br>‘01 = Image name (for example, SQR) |
| Enter report name: | This message prompts the user to enter the name of the report (.sqr or .sqt) to run. |
| Enter database name: | This message prompts the user to enter the name of the database. |
| Enter database[/username]:<br>Enter Username: | This message prompts the user to enter the user name to log onto the database. |
| Enter Password: | This message prompts the user to enter the password. For security reasons, the password is not echoed. |
| Customer ID: | This is a text message. |
| Press Enter to close... | This is a text message. |
| ‘01: Program Aborting. | This is an informational message. ‘01 = Image name (for example, SQR) |
| *** Internal Coding Error *** | This is an informational message. |
| SQL DataServer Message | (Microsoft Windows) This message is a title for the error message dialog box. |
| Operating-System error | (Microsoft Windows) This message is a title for the error message dialog box. |
| DB-Library error | (Microsoft Windows) This message is a title for the error message dialog box. |
| ‘01 is running.<br>Click the Cancel button to interrupt it. | (Microsoft Windows) This message is the body of the -C cancel dialog box. The user can quit the program run by clicking the Cancel button. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Message | Description |
|---|---|
| Table of Contents | This message is the text for HTML driver. |
| Previous | This message is the text for HTML driver. |
| Next | This message is the text for HTML driver. |
| First Page | This message is the text for HTML driver. |
| Last Page | This message is the text for HTML driver. |
| PAGE | This message is the text for HTML driver. |

# Numbered Messages

This table lists numbered messages:

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 000001 | Error while opening the message file: ’‘01’ (‘02): ‘03 | Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = Name of the error message file.<br>‘02 = System error code.<br>‘03 = System error message. |
| 000002 | Error while reading the message file. (‘01): ‘02 | Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = Name of the error message file.<br>‘02 = System error code.<br>‘03 = System error message. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 000003 | Error while closing the message file. (‘01): ‘02 | Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = Name of the error message file.<br>‘02 = System error code.<br>‘03 = System error message. |
| 000004 | Error while seeking the message file. (‘01): ‘02 | Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = Name of the error message file.<br>‘02 = System error code.<br>‘03 = System error message. |
| 000005 | Corrupt message file: Invalid header information. | Try reloading the sqrerr.dat 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.<br>1. Make sure SQRDIR points to the correct directory.<br>2. Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = The value read from the header.<br>‘02 = The correct value. |
| 000007 | Cannot handle message file version ‘01. | This message occurs when the release of SQR does not support the header version.<br>1. Make sure SQRDIR points to the correct directory.<br>2. Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = Unsupported version read from the header. |

Copyright © 1988-2007, Oracle. All rights reserved.

| 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.<br>‘01 = Invalid code. |
| 000011 | Unknown conversion type (‘01) for code ‘02. | Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = Invalid type.<br>‘02 = Internal error code. |
| 000012 | Message ‘01 must be either Preload or BuiltIn. | The type error code is not correct. Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = Error code. |
| 000013 | Cannot point to message ‘01. | The error handler cannot position to the desired error code. Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = Error code. |
| 000014 | The required environment variable ‘01 has not been defined. | Define the named environment variable and restart SQR.<br>‘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 sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = What was found in the header.<br>‘02 = What was expected to be found |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Name of the routine<br>‘02 = Error code |
| 000017 | Message ‘01 must be ReportParameters or CopyrightNotice. | Try reloading the sqrerr.dat file from the release media. If the error persists, contact technical support.<br>‘01 = Error code |
| 000018 | Allocation header does not point to a valid heap. | (Microsoft Windows) This message is the result of a memory overwrite. Record the steps leading up to the error and contact technical support. |
| 000019 | Allocation header has an invalid size. | (Microsoft Windows) This message 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. | (Microsoft Windows) This message is the result of a memory overwrite. Record the steps leading up to the error, and contact technical support. |
| 000021 | Cannot free GLOBAL allocation. | (Microsoft Windows) This message 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.<br>‘01 = Name of the file.<br>‘02 = System error code.<br>‘03 = System error message. |
| 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Name of the routine.<br>‘02 = Name of the command. |
| 000205 | DDO: DO arguments do not match procedure’s. | <obsolete> |
| 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. |
| 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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 putlin. | 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.<br>‘01 = Name of the routine.<br>‘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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 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.<br>‘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.<br>‘01 = Internal command format string.<br>‘02 = Bad format field found. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘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.<br>‘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.<br>‘01 = Internal command format string. |
| 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.<br>‘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.<br>‘01 = Internal command format string. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Routine name.<br>‘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.<br>‘01 = Internal variable type code. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Invalid element number. |
| 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.<br>‘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.<br>‘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.<br>‘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.<br>‘01 = Variable name. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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, 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. |
| 000256 | DBFETCH: Unknown variable dbtype encountered: ‘01 (‘02) | (DB2, 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.<br>‘01 = Variable name.<br>‘02 = Unknown database type. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘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.<br>‘01 = Name of the routine.<br>‘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.<br>‘01 = Name of the routine.<br>‘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.<br>‘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.<br>‘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.<br>‘01 = Variable type (from).<br>‘02 = Variable type (to). |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 001100 | Out of query arguments; use -Mfile to increase QUERYARGS. | This is the total number of variable references ($Var, #Var, &Col) allowed in the context of a BEGIN-SQL or BEGIN- SELECT command. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 001201 | Cannot open the argument file: ''01'. ('02): '03 | Depends on the system error message.<br>'01 = Name of the file.<br>'02 = System error code.<br>'03 = System error message. |
| 001202 | Cannot close the argument file. ('01): '02 | Depends on the system error message.<br>'01 = System error code.<br>'02 = System error message. |
| 001203 | Cannot open the - Mfile: ''01'. ('02): '03 | Depends on the system error message.<br>'01 = Name of the file.<br>'02 = System error code.<br>'03 = System error message. |
| 001204 | Minimum value for ''01' in the -Mfile is '02. | Correct the -Mfile entry.<br>'01 = Keyword in question.<br>'02 = Minimum value allowed. |
| 001205 | Maximum value for ''01' in the -Mfile is '02. | Correct the -Mfile entry.<br>'01 = Keyword in question.<br>'02 = Maximum value allowed. |
| 001206 | Invalid -Mfile entry: ''01'. | Correct the -Mfile entry.<br>'01 = The line from the -Mfile. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 001207 | Cannot close the - Mfile. (‘01): ‘02 | Depends on the system error message.<br>‘01 = System error code.<br>‘02 = System error message. |
| 001209 | The minimum value for ’‘01’ (‘02) is ‘03. | Value out of range.<br>‘01 = Entry name.<br>‘02 = Specified value.<br>‘03 = Minimum value. |
| 001210 | The maximum value for ’‘01’ (‘02) is ‘03. | Value out of range.<br>‘01 = Entry name.<br>‘02 = Specified value.<br>‘03 = Maximum value. |
| 001211 | The value for ’‘01’ (‘02) is not an integer number. | Value must be a integer value.<br>‘01 = Entry name.<br>‘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. |
| 001302 | SQL buffer too small; use -Mfile to increase SQLSIZE. | The SQL statement exceeds the size of the internal SQL buffer. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |

Copyright © 1988-2007, Oracle. All rights reserved.

| 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.<br>'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. | (Oracle, ODBC, 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. | (Oracle, ODBC, 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.<br>'01 = Name of the SQR routine.<br>'02 = Name of the column. |
| 001309 | The type for '&'01' ('02) does not match the type from the database ('03). | Correct the source code.<br>'01 = Name of the column /expression pseudonym.<br>'02 = User specified type.<br>'03 = Database type. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Maximum value for ROUND. |
| 001402 | Optional qualifiers for DIVIDE are ON- ERROR={HIGH\|ZER O} and ROUND=n. | Correct the syntax. |
| 001403 | Attempting division by zero. | Use the ON-ERROR = HIGH \| ZERO option to prevent this error from halting the program. |
| 001404 | Bad number of digits to ROUND or TRUNC (0-15). | Correct the syntax. |
| 001405 | WARNING: The ROUND or TRUNC qualifier is greater than the number’s precision. | Correct the syntax. |
| 001500 | Array element out of range (‘01) for array ’‘02’ on line ‘03. | Correct the source logic.<br>‘01 = Element number passed.<br>‘02 = Name of the array.<br>‘03 = Program line number. |
| 001501 | Field element out of range (‘01) for array ’‘02’, field ’‘03’, on line ‘04. | Correct the source logic.<br>‘01 = Element number passed.<br>‘02 = Name of the array.<br>‘03 = Name of the field.<br>‘04 = Program line number. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Program line number.<br>‘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.<br>‘01 = Name of the current report.<br>‘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.<br>‘01 = Name of the current report.<br>‘02 = Goto-Top value.<br>‘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.<br>‘01 = Name of the current report.<br>‘02 = Erase-Page value.<br>‘03 = Page width. |
| 001704 | Report ’‘01’: The NEXT-COLUMN command is not legal in the ‘02 section with the qualifier AT-END=NEWPAGE. | Correct the source line.<br>‘01 = Name of the current report<br>‘02 = Name of the section |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 001705 | Report ''01': Column number '02 is not defined. | The column number specified with the USE- COLUMN command is greater than the highest column defined in the COLUMNS command. Correct the source line. '01 = Name of the current report '02 = Column number |
| 001800 | Format for CONNECT: username/password [ON-ERROR= procedure[(arg1[,argi]...)] ] | Correct the syntax. |
| 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 ends the program run since it cannot continue. |
| 001803 | CONNECT failed. Perhaps user name/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 ends 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. |
| 001806 | Out of query space. Use -Mfile to increase QUERIES. | The number of SQL statements has been exceeded. Use the -Mfile flag on the command line specify a file that contains an entry that increases a greater value than is currently defined. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘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. | (Oracle, DB2, ODBC) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. |
| 001810 | Database rollback failed. | (Oracle, DB2, ODBC) 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. | ( Oracle, ODBC) 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.<br>‘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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 001917 | Ambiguous date-time. | Correct the date. |
| 001918 | ’‘01’ is not a valid date part. | Correct the date part.<br>‘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.<br>‘01 = Improper format characters. |
| 001921 | Bad date mask starting at: ’‘01’. | Correct the edit mask.<br>‘01 = Improper format characters. |
| 001922 | Seconds past midnight must be between 0 and 86399. | Correct the date. |
| 001923 | Seconds must be between 0 and 59. | Correct the date. |
| 001924 | Minutes must be between 0 and 59. | Correct the date. |
| 001925 | Month must be between 1 and 12. | Correct the date. |
| 001926 | Day must be between 1 and ‘01. | Correct the date. |
| 001927 | Hour must be between 1 and 12. | Correct the date. |
| 001928 | Hour must be between 0 to 23. | Correct the date. |
| 001929 | HH24 precludes the use of meridian indicator. | Correct the edit mask. |
| 001930 | HH12 requires meridian indicator. | Correct the edit mask. |
| 001931 | Day of year must be between 1 and 365 (366 for leap year). | Correct the date. |
| 001932 | Date string too long. | Correct the date. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 001933 | The month (‘01) is not valid for the current locale or database. | Correct the date.<br>‘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) The format mask needs to be corrected or there is a problem with the database server. |
| 001937 | Bad SQL for default date-time. (Table DUAL required for syntax.) | (DB2) The format mask needs to be corrected or 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 001942 | The date ’‘01’ is not in the format SYYYYMMDD[HH24[MI[SS[N NNNNN]]]]. | When specifying an SQR date, at a minimum, the date must be specified. The time is optional.<br>‘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[.NNNNN N]] [AM \| PM]]<br>MM-DD-YYYY [BC \| AD] [HH:MI[:SS[.NNNNN N]] [AM \| PM]] MM.DD.YYYY [BC \| AD] [HH:MI[:SS[.NNNNN N]] [AM \| PM]] SYYYYMMDD[HH24[MI [SS[N NNNNN]]]] | The date specified with the INPUT command was not in one the default formats. Reenter the date in a valid format.<br>‘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:<br>DD-MON-YY SYYYYMMDD [HH24[MI[SS[N NNNNN]]]] | (Oracle) The date was not in one of the expected formats for this database.<br>‘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:<br>Mon DD YYYY [HH:MI[:SS [.NNN]][ AM \| PM]]<br>Mon DD YYYY [HH:MI[:SS[: NNN]][ AM \| PM]]<br>YYYYMMDD [HH:MI[:SS [.NNN]][ AM \| PM]]<br>YYYYMMDD [HH:MI[:SS[: NNN]][ AM \| PM]]<br>SYYYYMMDD[HH24[MI[SS[N NNNNN]]]] | (Sybase) The date was not in one of the expected formats for this database.<br>‘01 = The invalid date. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| 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:<br>Mon DD YYYY [HH:MI[:SS[.NNN]][ AM \| PM]]<br>Mon DD YYYY [HH:MI[:SS[:NNN]][ AM \| PM]]<br>YYYYMMDD [HH:MI[:SS[.NNN]][ AM \| PM]]<br>YYYYMMDD [HH:MI[:SS[:NNN]][ AM \| PM]]<br>SYYYYMMDD[HH24[MI[SS[NNNNNN]]]] | (ODBC) The date was not in one of the expected formats for this database.<br>'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:<br>YYYY-MM-DD HH:MI:SS.NNN<br>SYYYYMMDD[HH24[MI[SS[NNNNNN]]]] | (Informix) The date was not in one of the expected formats for this database.<br>'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:<br>YYYY-MM-DD[-HH.MI.SS[.NNNNNN ]]<br>MM/DD/YYYY DD.MM.YYYY<br>SYYYYMMDD[HH24[MI[SS[NNNNNN]]]] | (DB2) The date was not in one of the expected formats for this database.<br>'01 = The invalid date. |
| 001945 | SQR does not support dates before ''01'. | SQR does not support dates before the one specified in the message.<br>'01 = Smallest date |
| 001946 | The date variables are incompatible with each other. | The SQR function references two date variables that cannot logically be used together, for example, DateDiff of 'date- only' and 'time-only' dates. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 002000 | Procedure name used more than once: ’‘01’. | Specify a unique name for the procedure.<br>‘01 = Procedure name |
| 002001 | Could not find procedure: ’‘01’. | Verify the spelling of the procedure name.<br>‘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.<br>‘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.<br>‘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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 002200 | ENCODE string too large; maximum is ‘01. | Break up the ENCODE command.<br>‘01 = Maximum length of an ENCODE string supported by this version of SQR for PeopleSoft. |
| 002300 | EXIT-SELECT failed. | The database command to cancel the query returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator. |
| 002301 | EXIT-SELECT valid only within SELECT paragraph. | Remove the EXIT-SELECT command. |
| 002400 | Duplicate label’s - do not know which one to GOTO. | Labels must be unique within the section or paragraph where they are defined. Give each label a unique name. |
| 002401 | (Labels must be in same section or paragraph as GOTO.) Cannot find a matching label for GOTO command. | Check the source code. |
| 002500 | Error getting INPUT. | The C routine "fgets()" returned an error and SQR ends 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.<br>‘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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 002508 | Incorrect. Format for integer: [+\|-]999999 | Invalid integer was entered for an INPUT request. Re-enter the data. |
| 002510 | A format mask can only be specified when TYPE=DATE is used. | Correct the syntax. |
| 002511 | The format mask cannot be stored in a date variable. | Correct the syntax. |
| 002512 | The input variable type does not match the TYPE qualifier. | Correct the syntax. |
| 002513 | Number too large for INTEGER. Valid range is -2147483648 to 2147483647. | The number was too large to be stored as an integer. Values are from -2147483648 to 2147483647. Re-enter the data. |
| 002514 | Enter a date in one of the following formats:<br>MM/DD/YYYY [HH:MI[:SS [.NNNNN N]] [AM \| PM]]<br>MM-DD-YYYY [HH:MI[:SS [.NNNNN N]] [AM \| PM]]<br>MM.DD.YYYY [HH:MI[:SS [.NNNNN N]] [AM \| PM]]<br>SYYYYMMDD[HH24[MI[SS[N NNNNN]]]] | 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.<br>‘01 = Load lookup table name |
| 002601 | Missing value for ‘01= in LOAD-LOOKUP. | Correct the syntax.<br>‘01 = Name of missing required parameter |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 002602 | Bad value for ‘01= in LOAD-LOOKUP. | Correct the syntax.<br>‘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.<br>‘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.<br>‘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. Verify the column and table names. Review the WHERE= clause for errors.<br>‘01 = Load lookup table name |
| 002607 | Problem executing the cursor for LOAD- LOOKUP table ’‘01’. | The database server returned an error while trying to execute the SQL statement needed to process the LOAD-LOOKUP command.<br>‘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.<br>‘01 = maximum length supported |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 002611 | Bad return fetching row from database in LOAD-LOOKUP table ’‘01’. | The database server returned an error while fetching the data.<br>‘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.<br>‘01 = Name of the load lookup table<br>‘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.<br>‘01 = Number of duplicate keys<br>‘02 = Name of the load lookup table |
| 002616 | LOAD-LOOKUP ‘01= must reference a numeric variable or literal. | Correct the source line.<br>‘01 = Name of the parameter |
| 002617 | LOAD-LOOKUP ‘01= must reference a string variable or literal. | Correct the source line.<br>‘01 = Name of the parameter |
| 002618 | LOAD-LOOKUP ‘01= variable ’‘02’ has not been defined. | Correct the source line.<br>‘01 = Name of the parameter<br>‘02 = Name of the undefined variable |
| 002619 | LOAD-LOOKUP cannot support ‘01 rows; maximum is ‘02. | Reduce the ROWS= value.<br>‘01 = ROWS= value<br>‘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.<br>‘01 = SQR command |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 002700 | Line to stop erasing for ’NEW-PAGE’ is larger than the page depth. | Correct the source line. |
| 002800 | ’ON-BREAK’ not appropriate for numeric data. | The ON-BREAK argument to the PRINT command may be used only for text fields. Move the #numeric variable to a $string variable, and then print the $string variable. |
| 002801 | SET= and LEVEL= must be >= zero when indicated. | Correct the source line. |
| 002802 | Cannot use old style PROCEDURE= with BEFORE= or AFTER=. | Correct the syntax. |
| 002803 | Out of ON-BREAKS;<br>use -Mfile to increase ONBREAKS. | Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 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.<br>‘01 = SQR command affected |
| 002902 | OPEN missing required qualifiers: RECORD={rec_len} FOR-READING\|FOR-WRITING\|FOR- APPEND | Correct the syntax. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 002903 | Too many external files opened; maximum is ‘01. | Reduce the number of open external files needed by the program.<br>‘01 = Maximum number of open external files supported by this version of SQR |
| 002904 | File number already opened. | Verify the program logic. |
| 002905 | Cannot open file ’‘01’ AS ‘02.(‘03): ‘04 | SQR stops.<br>‘01 = Filename<br>‘02 = File number<br>‘03 = System error code<br>‘04 = System error message |
| 002906 | Cannot close file ‘01. (‘02): ‘03 | SQR stops.<br>‘01 = File number<br>‘02 = System error code<br>‘03 = System error message |
| 002907 | Problem closing user file(s) at the end of run. | This message may indicate system problems. |
| 002908 | Warning: Cannot CLOSE file ‘01 -- file not opened. | While not an error, this message indicates a problem with your SQR code.<br>‘01 = File number |
| 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. Verify that the @ marker names are spelled correctly. |
| 003101 | Only ’COLUMNS nn...’ allowed after document marker in POSITION command. | Correct the syntax. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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<br>‘02 = System error code<br>‘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<br>‘02 = System error code<br>‘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. Search for a typographical error or recalculate the RECORD= value. |
| 003205 | Numeric binary transfer allowed with FIXED or FIXED_NOLF records only. | By default, all files are opened in VARY (variable length) mode, thus prohibiting the transfer of numeric binary data. Add the:FIXED or FIXED_NOLF option to the RECORD= argument on the appropriate OPEN command. |
| 003206 | Command not complete. | Correct the syntax. |
| 003207 | File number must be a numeric literal, variable, or column. | Correct the syntax. |
| 003208 | Missing required length in READ command. | Correct the syntax. |
| 003209 | Bad length for READ or WRITE command. | Correct the syntax. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 003300 | Unknown qualifier for STOP. | Correct the syntax. |
| 003301 | Program stopped by user request. | This is an 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.<br>‘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.<br>‘01 = Maximum number of characters supported by this version of SQR |
| 003403 | Bad <number> in WRAP qualifier. | The number inside the angled brackets must be a valid 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 003405 | The value for ’‘01’ (‘02) must be ‘03 0. | The value specified for the specified qualifier is invalid. Correct the program logic.<br>‘01 = Qualifier name<br>‘02 = Value encountered<br>‘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. Verify that the array names are spelled correctly. |
| 003507 | Bad element reference for array (#variable\|123). | The element number is larger than the number of rows defined in the CREATE-ARRAY command. Review the program logic to make sure that the element number was not inadvertently changed. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 003508 | Did not find ending ')' for field. | The "occurs" number for an array field is missing a right parenthesis. Correct the source line. |
| 003509 | Field not defined in array: '01 | Verify that there are no misspelled field names in the CREATE-ARRAY command.<br>'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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Maximum number of arrays supported by this version of SQR |
| 003517 | Missing ’=specifier’ in qualifier: ‘01 | Correct the syntax.<br>01 = Name of missing required parameter |
| 003518 | Duplicate array name: ‘01 | Change the name of the array.<br>‘01 = Array name in question |
| 003519 | Too many fields defined; maximum is ‘01. | Reduce the number of fields.<br>‘01 = Maximum number of fields allowed per array |
| 003520 | Missing ’:type’ in CREATE-ARRAY FIELD= ‘01 | Correct the syntax.<br>‘01 = The name of the field |
| 003521 | Duplicate FIELD name: ‘01 | Change the name of one of the fields.<br>‘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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 003527 | Missing FIELD= statements in CREATE-ARRAY. | Correct the syntax. |
| 003528 | Array dimensioned too large for PC in CREATE-ARRAY. | On 32–bit 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.<br>01 = Name of the field |
| 003600 | Missing ’ask’ variable name. | Correct the syntax. |
| 003601 | Out of substitution or #DEFINE variables; use -Mfile to increase SUBVARS. | Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 003603 | WARNING: Substitution variables do not vary when saved with run-time. | This is an informational message. |
| 003605 | No substitution variable entered. | The C routine "fgets()" returned an error and SQR ends the program run. |
| 003700 | Did not find end of paragraph: ‘01 | The END-paragraph command to match the specified paragraph is missing. Correct the source file. ‘01 = BEGIN-paragraph in question |
| 003701 | Invalid command. | Verify the spelling of the command. |
| 003702 | Command not allowed in this section: ‘01 | Correct the syntax.<br>‘01 = Offending command name |
| 003703 | Paragraph not allowed inside procedure. | The BEGIN-paragraph command is not allowed here. Review your SQR code for a misplaced paragraph. |
| 003704 | Missing procedure name. | Correct the syntax. |
| 003705 | Extra argument found. | Correct the syntax. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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 $string and #number variables allowed for BEGIN-PROCEDURE parameters. | Correct the syntax. |
| 003710 | Unknown argument type. | An argument in a DO or BEGIN-PROCEDURE command is incorrect. Verify the spelling of the variable types. |
| 003711 | Indicate :$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.<br>‘01 = BEGIN-command in question |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 003716 | Error in SQL statement. | The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement. |
| 003717 | Extra characters after expression continuation. | Remove the extra characters after the dash. |
| 003718 | Did not find end of expression. | An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line. |
| 003719 | Columns names and expressions must be unique or be given unique pseudonyms (&name). | You are trying to select the same &column name more than once. Change the assigned &column name by using an alias after the name. Columns retrieved from the database are assigned names by prepending an "&" to the beginning of 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. |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 003722 | Could not set up cursor. | An error occurred while trying to compile the SQL statement. Review 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. Review the $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. This is commonly due to the buffer not being large enough. If selecting expressions, make sure that the length of the first expression is 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 003727 | SQL expression not ended, perhaps parentheses not balanced. | An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line. |
| 003728 | SQL expression not ended, perhaps missing &name. | An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line. |
| 003729 | SQL expression is missing &name or has unbalanced parentheses. | An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line. |
| 003730 | Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]] | (DB2) Correct the syntax. |
| 003730 | Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]] | (Informix) Correct the syntax. |
| 003730 | Incorrect arguments for BEGIN-SQL: [-Cnn] [-NR] [ON-ERROR=procedure(arg1[,argi]...)]] [-DB=] | (ODBC) Correct the syntax. |
| 003730 | Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]] | (Oracle) Correct the syntax. |
| 003730 | Incorrect arguments for BEGIN-SQL: [-Cnn] [-XP] [ON-ERROR=procedure[(arg1[,argi]...)]] | (Sybase) 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. | (Oracle, Informix, ODBC) Correct the source code. |
| 003737 | Bad fetch buffer count. | (Oracle, Sybase) The -B flag specifies an illegal value. Correct the source code. |
| 003738 | Report interrupted by request. | This is an informational message. |
| 003741 | Dynamic column must be $string variable. | Correct the syntax. |
| 003742 | Dynamic column missing ’‘01’. | Correct the syntax.<br>‘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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 003745 | Only a variable name may be between the ’‘01’ and ’‘02’ characters. | Correct the syntax.<br>‘01 = Leading character<br>‘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. |
| 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.<br>‘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.<br>‘01 = Maximum number supported by this version of SQR |
| 003802 | Duplicate document marker. | Specify a unique name for the the document marker. |
| 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 #). | Correct the source line. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 003903 | Missing '=' after '01. | Correct the source line.<br>'01 = The parameter in question |
| 003904 | Unknown variable type. | Variable names must begin with $, #, or &. Correct the source line. |
| 003905 | OUT[PUT] variables for EXECUTE may only be $variable or #variable. | Correct the syntax. |
| 003906 | The only EXECUTE option is WITH RECOMPILE. | Correct the syntax. |
| 003907 | You must EXECUTE ... INTO &columns. | Correct the syntax. |
| 003908 | Unknown datatype for EXECUTE...INTO &columns. | Verify the spelling of the 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. Specify a unique name for the column. |
| 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. | Record the database error message displayed with this message.<br>If needed, contact your system administrator. |
| 003914 | Bad return fetching row from database. | Record the database error message displayed with this message. If necessary, contact your system administrator. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 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.<br>'01 = Maximum number supported by this version of SQR |
| 004004 | Empty expression. | The expression is invalid. Correct the source line. |
| 004005 | Extra comma in expression. | An argument is missing after a comma in the expression. Correct the source line. |
| 004006 | Unknown operator ''01'. Do you mean '02 ? | The concatenation operator is \|\|. Correct the source line. |
| 004007 | Too many &column forward references in expression; maximum is '01. | The expression contains too many forward references. Break the expression into smaller expressions.<br>'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. Verify that the function names are spelled correctly.<br>'01 = Function name |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>'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.<br>'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.<br>'01 = Operator |
| 004015 | Function ''01' missing argument. | Correct the syntax of the function. Functions are described under the LET command.<br>'01 = SQR function name |
| 004016 | Function or operator ''01' missing arguments. | Correct the syntax of the function. Functions are described under the LET command.<br>'01 = SQR function name |
| 004017 | User function ''01' requires character argument. | Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function.<br>'01 = User function name |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>'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.<br>'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.<br>'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.<br>'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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 004026 | Function or operator ’‘01’ requires character argument. | Correct the source line.<br>‘01 = Function or operator |
| 004027 | Function or operator ’‘01’ requires numeric argument. | Correct the source line.<br>‘01 = Function or operator |
| 004028 | IF or WHILE expression must return logical result. | The expression used must evaluate a statement that will be TRUE or FALSE. Correct the source line. |
| 004029 | Attempting division by zero in expression. | The expression tried to divide a number by zero. Use the COND() function to determine whether 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 determine whether 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 only with integers. Correct the program logic.<br>‘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, 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 004034 | Could not find array ’‘01’ in ARRAY function. | Verify the spelling of the array name.<br>‘01 = Array name |
| 004035 | Could not find array field ’‘01’ in ARRAY function. | Verify the spelling of the array field name.<br>‘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;<br>Use -Mfile to increase EXPRESSIONSPACE. | The expression requires more temporary string storage than is currently allocated. Use the -Mfile flag on the command line to specify a file that contains an entry that increases by a greater value than is currently defined. |
| 004039 | ’‘01’ assumed to be a variable name, not an expression. | Warning message.<br>‘01 = Expression in question |
| 004040 | The array ’‘01’ has not been defined. | Define the array using the CREATE-ARRAY command.<br>‘01 = Array name |
| 004041 | The field ’‘01’ is not valid for array ’‘02’. | Correct the source code.<br>‘01 = Field name<br>‘02 = Array name |
| 004042 | The array reference ’‘01’ has an incorrect number of parameters specified. | Correct the source code.<br>‘01 = Array name |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 004043 | The array reference ’‘01’ requires numeric parameters for the element and occurs arguments. | Correct the source code.<br>‘01 = Array name. |
| 004045 | Function or operator ’‘01’ requires date argument. | Correct the source code.<br>‘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.<br>‘01 = Field name |
| 004048 | Function or operator ’‘01’ must be a string or date argument. | Correct the source line.<br>‘01 = Function or operator |
| 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. Search for a missing parentheses. |
| 004105 | Placement parameter incorrect. | The "Row", "Column" or "Length" fields are invalid or ill-formed. Correct the source line. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 004106 | Invalid second function on line. | An SQR command used as a qualifier for a primary command (for example, PRINT) is incorrect. Correct the source line. |
| 004107 | Second function must be FORMAT type. | The PRINT command may have format command qualifiers such as WRAP, CENTER, or FILL. Other qualifier commands are not permitted. |
| 004108 | Missing operator =, <, >, ... | Correct the source line. |
| 004109 | Invalid operator. | Correct the source line. |
| 004110 | Missing variable. | Correct the syntax. |
| 004111 | Please give this expression a &pseudonym. | Expressions selected in 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.<br>‘01 = What was expected |
| 004114 | Expecting ’‘01’, found ’‘02’. | Correct the syntax.<br>‘01 = What was expected<br>‘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.<br>‘01 = Duplicated parameter |
| 004117 | Unexpected equal sign found with ’‘01’. | Correct the syntax.<br>‘01 = Parameter name |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 004118 | Qualifier ''01' cannot be used with the following qualifiers: | Correct the syntax.<br>'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.<br>'01 = Parameter name |
| 004200 | Page width and depth must be > 0 and < 32767. | The values specified with the PAGE-SIZE command are out of specified range. Specify legal values. |
| 004201 | Page buffer must be < 65536 on PC SQR. | The maximum page buffer 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, specify PRINTER:{HP,EH,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:{HP,EH,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. | A 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 004400 | Program too large; use -Mfile to increase PROGLINES. | The SQR program contains too many SQR command lines. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 004401 | Out of param storage; use -Mfile to increase PROGLINEPARS. | The SQR program contains too many SQR command line parameters. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 004402 | Out of string storage; use -Mfile to increase STRINGSPACE. | The space allocated to hold the static string variables ('...') has been used. Use the - Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 004403 | Out of variables; use - Mfile to increase VARIABLES. | There are too many variables (string, numeric), literals and database columns. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 004405 | Out of forward &column or $variable references; use -Mfile to increase FORWARDREFS. | A forward referenced variable is a variable that is referenced before it is defined. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 004406 | Number '01 not allowed. | Use a different value.<br>'01 = Internal number |
| 004407 | Referenced variables not defined: | References were made to column variables (&var) that are not defined in the program. The list of variable names follows this message. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 004500 | Out of Print positions; use -Mfile to increase POSITIONS. | A print position is the "(Row,Col,Len)" parameter. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 004501 | Use '+' and negate variable for reverse relative placement. | The use of "-#variable" is not legal here. Negate the #variable value and use "+#variable". |
| 004503 | Fixed line placement #variable must be > 0. Use relative positioning, (+#line,10,0). | Correct the source line as indicated. |
| 004504 | Fixed column placement #variable must be > 0. Use relative positioning, (5,+#col,0). | Correct the source line as indicated. |
| 004505 | Length placement #variable must be >= 0. | The length field cannot be a negative value. Correct the source line. |
| 004600 | CODE not appropriate for numeric data. | The CODE 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 ends the program run.<br>'01 = Row<br>'02 = Column<br>'03 = Width<br>'04 = Depth |
| 004603 | GRAPHIC VERT- LINE out of bounds. Row: '01, Column: '02, Length: '03 | SQR ends the program run.<br>'01 = Row<br>'02 = Column<br>'03 = Length |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 004604 | GRAPHIC HORZ- LINE out of bounds. Row: ‘01, Column: ‘02, Length: ‘03 | SQR ends the program run.<br>‘01 = Row<br>‘02 = Column<br>‘03 = Length |
| 004700 | Cannot open the program file: ’‘01’ (‘02): ‘03 | Depends on the system error message.<br>‘01 = Name of the program file<br>‘02 = System error code<br>‘03 = System error message |
| 004701 | Cannot logon to the database. | The connectivity information is either incorrect or the database server is unavailable. Verify 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.<br>‘01 = System error code<br>‘02 = System error message |
| 004704 | #ENDIF not found for #IF. | Missing an #ENDIF to complete conditional compilation. Correct the source code. |
| 004705 | Program line too long; maximum is ‘01. | Break the program line into smaller lines.<br>‘01 = Maximum line length supported by this version of SQR |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Name of the substitution variable<br>‘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. Verify the spelling of the substitution variable.<br>‘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.<br>‘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.<br>‘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 path name. Verify the spelling of both the -I command flag and the #INCLUDE filename. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 004713 | Cannot open the #INCLUDE file: ’‘01’ (‘02): ‘03 | ‘01 = Include file name<br>‘02 = System error code<br>‘03 = System error message |
| 004714 | Cannot close the #INCLUDE file: ’‘01’ (‘02): ‘03 | ‘01 = Include file name<br>‘02 = System error code<br>‘03 = System error message |
| 004716 | ’BEGIN-REPORT’ command not found in program. | This section is required for all reports. Correct the source code. |
| 004717 | Cannot open the report output file: ’‘01’ (‘02): ‘03 | ‘01 = Output file name<br>‘02 = System error code<br>‘03 = System error message |
| 004719 | Cannot logoff the database. | The database server returned an error while trying to log off from the database. SQR ends the program run. |
| 004720 | Cannot open the run-time file: ’‘01’. (‘02): ‘03 | SQR ends the program run.<br>‘01 = Run-Time file name<br>‘02 = System error code<br>‘03 = System error message |
| 004721 | Cannot close the run-time file. (‘01): ‘02 | SQR ends the program run.<br>‘01 = System error code<br>‘02 = System error message |
| 004722 | Error reading the run-time file. (‘01): ‘02 | SQR ends the program run.<br>‘01 = System error code<br>‘02 = System error message |
| 004723 | Run time file must be recreated for this version of SQR. | The runtime file was created by a earlier version of SQR and is incompatible with the current version. Recreate the .sqt (runtime) file. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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 runtime file. (‘01): ‘02 | ‘01 = System error code<br>‘02 = System error message |
| 004728 | You must specify a Partitioned Data Set name and member to build a .sqt(member) run-time file. Could not create the run-time file. | (MVS) Use the proper format to specify the name of the .sqt file. |
| 004729 | Cannot find inactive database cursor. Program too large. | (Oracle, DB2) The program has too many concurrent database cursors. Reduce the complexity of the program. |
| 004730 | Run-time saved in file: ‘01 | This is an informational message.<br>‘01 = Name of the .sqt file created |
| 004735 | Unknown variable type encountered in run-time file: ‘01 | SQR ends loading the runtime file.<br>‘01 = Variable type |
| 004736 | Unexpected End-Of-File while processing the run-time file. | SQR ends loading the runtime file. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 004737 | Cannot load the run- time file because it was built for the ‘01database and ‘02 is built for the ‘03 database. | SQR ends loading the runtime file.<br>‘01 = Database name from runtime file<br>‘02 = SQR image name<br>‘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.<br>‘01 = Array name<br>‘02 = Field name |
| 004746 | Layout ’‘01’ specifications exceeds the PC 64K limit. | The layout is too large for the 32–bit version of SQR to handle.<br>‘01 = Layout name |
| 004747 | The SQT file is corrupted and cannot be processed. | SQR ends loading the runtime 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.<br>‘01 = User function name<br>‘02 = Entry in the user function table<br>‘03 = Return type<br>‘04 = Argument count<br>‘05 = Argument types |

Copyright © 1988-2007, Oracle. All rights reserved.

| 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.<br>‘01 = Number of characters to be moved<br>‘02 = Variable name<br>‘03 = Maximum characters allowed |
| 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. |
| 004900 | Out of dynamic SQL arguments [$...];<br>use -Mfile to increase DYNAMICARGS. | Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 004901 | Date variables (‘01) cannot be used in BEGIN-SQL or BEGIN-SELECT paragraphs. | Correct the source code.<br>‘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. Review any procedure invoked by the BEGIN-HEADING section to ensure that the commands do not overflow the page or cause a page eject. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. Review 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.<br>‘01 = Command name<br>‘02 = Section name |
| 005005 | Report ’‘01’ already has been assigned a ‘02 section. | Correct the source code.<br>‘01 = Report name<br>‘02 = Duplicated section name |
| 005006 | You cannot define more than one default ’‘01’ section. | Correct the source code.<br>‘01 = Duplicated section name |
| 005007 | Report ’‘01’ has overlapping heading and footing sections. | Correct the source code.<br>‘01 = Report name |
| 005008 | TOC ’‘01’ already has been assigned a ‘02 section. | Correct the source code.<br>‘01 = Table of Contents name<br>‘02 = Duplicated section name |
| 005100 | ’IF’, ’WHILE’, ’EVALUATE’ commands nested too deeply; maximum is ‘01. | Reduce the nested commands.<br>‘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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005102 | Out of Break commands; Use -Mfile to increase BREAKS. | This is the number of BREAK commands allowed per EVALUATE command. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 005103 | END-WHILE found without matching ’WHILE’. | Correct the source code. |
| 005104 | ’IF’ or ’EVALUATE’ command not completed before ’END-WHILE’. | Correct the syntax. |
| 005105 | ’ELSE’ found without matching ’IF’. | ELSE can be used only within the context of an IF command. Correct the source code. |
| 005106 | Single ’ELSE’ found inside ’WHILE’ or ’EVALUATE’ statement. | ELSE can be used only within the context of an IF command. Correct the source code. |
| 005107 | Only one ’ELSE’ allowed per ’IF’. | Rewrite the source code to use nested IF statements. |
| 005108 | Found ’END-IF’ without matching ’IF’. | Each IF command must have a matching END-IF command. Correct the source code. |
| 005109 | ’WHILE’ or ’EVALUATE’ command not completed before ’END-IF’. | You are missing a closing END-WHILE or END-EVALUATE command before END- IF. IF, WHILE, and EVALUATE statements can be nested, but they cannot cross each other’s boundaries. Each inner statement must be complete before a closing statement is ended. Correct the source code. |
| 005110 | EVALUATE statements nested too deep; maximum is ‘01. | Reduce the number of nested statements.<br>‘01 = Maximum depth supported by this version of SQR |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 005113 | Out of When commands; Use -Mfile to increase WHENS. | Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value. |
| 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005200 | Did not find ’>’ after <.... | A leading left angled bracket (<) indicates that you are beginning an decimal value representing a character, which must be ended by a right angled bracket (>). Correct the source line. |
| 005201 | Bad character in <...>. | Numbers in angled brackets (<>) must be between 1 and 255. Correct the source line. |
| 005202 | Bad 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.<br>‘01 = Qualifier name |
| 005301 | Qualifier ’‘01’ requires a numeric value. | Correct the syntax.<br>‘01 = Qualifier name |
| 005302 | Incorrect value for qualifier ’‘01’. Valid values are: | Correct the source line.<br>‘01 = Qualifier name |
| 005303 | Invalid qualifier ’‘01’. Valid qualifiers are: | Correct the source line.<br>‘01 = Qualifier name |
| 005304 | Qualifier ’‘01’ requires a numeric literal, variable, or column. | Correct the source line.<br>‘01 = Qualifier name |
| 005305 | Qualifier ’‘01’ references a numeric variable that has not been defined. | Correct the source line.<br>‘01 = Qualifier name |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005306 | Qualifier ’‘01’ requires a string literal, variable, or column. | Correct the source line.<br>‘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.<br>‘01 = Qualifier name |
| 005310 | Qualifier ’‘01’ has already been specified. | Correct the source line.<br>‘01 = Qualifier name |
| 005311 | Qualifier ’‘01’ requires a string literal. | Correct the source line.<br>‘01 = Qualifier name |
| 005312 | Qualifier ’‘01’ requires a list of values: (val [,val]...). | Correct the source line.<br>‘01 = Qualifier name |
| 005313 | Qualifier ’‘01’ requires a integer value. | Correct the source line.<br>‘01 = Qualifier name |
| 005314 | Invalid character in variable name ’‘01’. | Correct the source line.<br>‘01 = Invalid character |
| 005315 | Qualifier ’‘01’ references a string variable that has not been defined. | Correct the source line.<br>‘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.<br>‘01 = Qualifier name |
| 005400 | Second page write attempted while writing current page. Check BEFORE- PAGE, AFTER-PAGE procedures. | Review 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Text value<br>‘02 = Row<br>‘03 = Column<br>‘04 = Length |
| 005403 | Error writing the output file. (‘01): ‘02 | ‘01 = System error code<br>‘02 = System error message |
| 005404 | Cannot open the Postscript startup file: ‘01 (‘02): ‘03 | ‘01 = Name of the file<br>‘02 = System error code<br>‘03 = System error message |
| 005405 | SQR trial copy exiting after ‘01 pages. | ‘01 = Number of pages. |
| 005406 | Exiting after requested number of test pages (‘01). | ‘01 = Number of pages. |
| 005408 | Program stopped by user request. | This is an informational message. |
| 005500 | Cannot set parse_only option. | (Sybase) The DB-Library routine dbsetopt() returned an error. This error should never occur. Contact technical support. |
| 005501 | Cannot reset parse_only option. | (Sybase) The DB-Library routine dbclropt() returned an error. This error should never occur. Contact technical support. |
| 005502 | Cannot drop SQR generated stored procedure: ‘01. | (Sybase) See the database server error message that was also output. This error should never occur. Contact technical support.<br>‘01 = Stored procedure name |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005503 | Cannot use ‘01 datatype as bind variable. | (Sybase) Use another database column.<br>‘01 = The database datatype. |
| 005504 | Unknown datatype for bind variable: ‘01 Cannot create stored procedure. | (Sybase) Contact technical support.<br>‘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 process. 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) Verify the definition for the stored procedure you are referencing. |
| 005509 | Incorrect number of INTO &columns defined in EXECUTE. | (Sybase) Verify 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) Verify the definition for the stored procedure you are referencing. |
| 005512 | Missing default database name for USE. | (Sybase) Correct the syntax. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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) Verify that there are no misspellings. '01 = Variable name |
| 005523 | Database commit failed. | The database command to perform a commit returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator. |
| 005524 | Cannot close database cursor. | The database command to close the database cursor returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator. |
| 005528 | DB2 SQL '01 error '02 in cursor '03: | (DB2) '01 = Routine name<br>'02 = Error code<br>'03 = SQR cursor number |
| 005528 | INFORMIX SQL '01 error '02 (ISAM: '03) in cursor '04: '05 | (Informix)<br>'01 = Routine name<br>'02 = Error code<br>'03 = ISAM code<br>'04 = SQR cursor number<br>'05 = Error message from database |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005528 | ODBC SQL ‘01 error ‘02 in cursor ‘03: ‘04 | (ODBC)<br>‘01 = Routine name<br>‘02 = Error code<br>‘03 = SQR cursor number<br>‘04 = Error message from database |
| 005528 | ORACLE ‘01 error ‘02 in cursor ‘03: ‘04 | (Oracle)<br>‘01 = Routine name<br>‘02 = Error code<br>‘03 = SQR cursor number<br>‘04 = Error message from database |
| 005528 | Sybase ‘01 error in cursor ‘02: ‘03 | (Sybase)<br>‘01 = Routine name<br>‘02 = SQR cursor number<br>‘03 = Error message from database |
| 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 version earlier than System 10 server. |
| 005534 | SQL too long for PREPARE /DECLARE; maximum ‘01 characters. | (DB2) The SQL statement is too large.<br>‘01 = Maximum number of characters supported by this version of SQR |
| 005536 | Unknown error message number: ‘01. | (DB2)<br>‘01 = Error message number |
| 005537 | Empty error message returned from system for error number: ‘01. | (DB2)<br>‘01 = Error message number |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005538 | Invalid SELECT statement; COMPUTE clauses are not supported. | (Sybase) The select statement contains a COMPUTE clause that is not supported. |
| 005539 | Could not connect to datasource specified in -db variable: ’‘01’. | (ODBC) Could not connect to the specified datasource. |
| 005540 | Not connected to a database, database access is not allowed. | The SQR program is no longer connected to a database. Commands that access the database can no longer be used. This situation can occur if the CONNECT fails and the ON-ERROR option was used. |
| 005543 | Specify the Oracle DLL name in the sqr.ini file in [Environment:Oracle] section for ORACLE_DLL entry, such as ORACLE_DLL=orant 71.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".<br>‘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.<br>‘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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005703 | Error opening the SQR log file: ’‘01’ (‘02): ‘03 | ‘01 = Name of the file<br>‘02 = System error code<br>‘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<br>‘02 = System error code<br>‘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.<br>‘01 = File name |
| 005709 | ‘01 environment variable is not defined. | As of version 2.5, the environment variable SQRDIR must be defined.<br>‘01 = Name of the environment variable |
| 005710 | ‘01 path too long. | The length of the directory path plus the length of the file name to be opened is too long for SQR to handle. Reduce the length of the directory path.<br>‘01 = Environment variable name |
| 005711 | Bad number in -T test flag. | The number specified must be > zero. Correct the value. |
| 005716 | Unknown flag on command line: ‘01 | Correct the syntax.<br>‘01 = Unknown command line flag |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005720 | Error opening tty. (‘01): ‘02 | (Tru64, UNIX/Linux) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.<br>‘01 = System error code<br>‘02 = System error message |
| 005721 | Error with ’ioctl()’. (‘01): ‘02 | (Tru64, UNIX/Linux) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.<br>‘01 = System error code<br>‘02 = System error message |
| 005722 | Error reading tty. (‘01): ‘02 | (Tru64, UNIX/Linux) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.<br>‘01 = System error code<br>‘02 = System error message |
| 005723 | Error closing tty. (‘01): ‘02 | (Tru64, UNIX/Linux) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.<br>‘01 = System error code<br>‘02 = System error message |
| 005724 | Bad number in -B flag. | (Oracle, Sybase) The number specified must be greater than 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Maximum number allowed |
| 005740 | -F and outfile name are required with DDN or DD style SQR {program} name. | (MVS) Correct the JCL stream. |
| 005741 | Attempting to use SQR {program} file for outfile. | (MVS) Correct the JCL stream. |
| 005742 | Attempt to invoke viewer (using WinExec) failed; error code = ‘01. | (Windows)<br>‘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.<br>‘01 = Specified precision<br>‘02 = Minimum allowed<br>‘03 = Maximum allowed |
| 005745 | The specified default numeric type ’‘01 = ‘02’ is invalid. | Correct the sqr.ini file entry.<br>‘01 = Entry<br>‘02 = Value |
| 005746 | The decimal precision ’‘01 = ‘02’ is out of range (‘03 - ‘04). | Correct the sqr.ini file entry.<br>‘01 = Entry<br>‘02 = Value<br>‘03 = Minimum allowed<br>‘04 = Maximum allowed |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005747 | The following error(s) occurred while processing the [‘01] section from the sqr.ini file. | See the error message(s) that follow.<br>‘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. |
| 005900 | Bad number in -‘01 | (Windows) Specify a valid number.<br>‘01 = Command line option |
| 005901 | Bad filename in -‘01 | (Windows) Specify a valid file name.<br>‘01 = Command line option |
| 005902 | Bad directory in -‘01 | (Windows) Specify a valid directory path.<br>‘01 = Command line option |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 005903 | Cannot access the @ parameter file (‘01): ‘02 | (Windows) Depends on the system error message.<br>‘01 = System error code<br>‘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.<br>‘01 = Maximum number supported by this version of SQR. |
| 005905 | Cannot open the report file (‘01): ‘02 | (Windows) Depends on the system error message.<br>‘01 = System error code<br>‘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 and permissions). Record the steps leading up to the error and contact your system administrator.<br>‘01 = System error code<br>‘02 = System error message |
| 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 and permissions). Record the steps leading up to the error and contact your system administrator.<br>‘01 = System error code<br>‘02 = System error message |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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 and permissions). Record the steps leading up to the error and contact your system administrator.<br>‘01 = Name of the file<br>‘02 = System error code<br>‘03 = System error message |
| 006003 | Unexpected End-Of- File while processing the printer file. | The file might be 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. | The file might be corrupted. Try to recreate the .spf file. If the error persists, contact technical support.<br>‘01 = Unknown SPF code |
| 006100 | Duplicate chart (‘01). | Each chart must be given a unique name.<br>‘01 = Chart name |
| 006101 | Unknown chart (‘01). | Chart could not be found.<br>‘01 = Chart name |
| 006102 | Number of chart data- array columns specified (‘01) exceeds the number of array columns (‘02). | Correct the source code.<br>‘01 = Number of data-array columns<br>‘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.<br>‘01 = Number of data-array rows<br>‘02 = Number of array rows |
| 006104 | Too many pie segments (‘01). Max is ‘02. | Correct the source code.<br>‘01 = Number of segments<br>‘02 = Maximum allowed segments |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 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.<br>‘01 = Chart index |
| 006121 | INTERNAL: Unknown SQR BG Interface 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.<br>‘01 = Message code |
| 006122 | INTERNAL: Unsupported Grafsman 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.<br>‘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.<br>‘01 = Setting value |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Placement value |
| 006125 | Grafsman 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.<br>‘01 = Message code |
| 006126 | Unrecognized return code (‘01) from Grafsman 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.<br>‘01 = Return code<br>‘02 = Message code |
| 006127 | Cannot fit Chart/Image into the current page. Position: (‘01, ‘02) Size: (‘03, ‘04) | Correct the source code. SQR ends the program run.<br>‘01 = Row<br>‘02 = Column<br>‘03 = Width<br>‘04 = Depth |
| 006128 | Check coordinate values. | Correct the syntax. |
| 006140 | Duplicate image (‘01). | Images must be given unique names.<br>‘01 = Image name |
| 006141 | Unknown image (‘01). | Image name could not be found.<br>‘01 = Image name |
| 006142 | Cannot open image file (‘01). (‘02): ‘03 | ‘01 = Name of the file<br>‘02 = System error code<br>‘03 = System error message |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘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.<br>‘01 = Qualifier name |
| 006204 | Qualifiers ’01’ and ’02’ are mutually exclusive. | Correct the syntax.<br>‘01 = Qualifier name<br>‘02 = Qualifier name |
| 006205 | Qualifier ’01’ is not applicable with a ’default’ printer. | Correct the syntax.<br>‘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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Report name<br>‘02 = Printer type |
| 006214 | Qualifier ’01’ is not allowed with a ’02’ printer. | Correct the syntax.<br>‘01 = Qualifier name<br>‘02 = Printer type |
| 006215 | The value for ’01’ must be ‘02 0. | Correct the syntax.<br>‘01 = Qualifier name<br>‘02 = Relation to zero (<,<=,=,>=,>) |
| 006216 | Report ’01’ does not exist. | Correct the syntax.<br>‘01 = Report name |
| 006217 | The report name can be a string literal, variable, or column. | Correct the syntax.<br>‘01 = Report name |
| 006218 | Referenced layouts not defined: | A list of undefined layouts follows this message. |
| 006219 | Referenced reports not defined: | A list of undefined reports follows this message. |
| 006220 | Referenced printers not defined: | A list of undefined printers follows this message. |
| 006221 | The following SQR commands (listed below) cannot be used when any of the following NEW SQR commands are also used in the same report: | Correct the syntax. |
| 006224 | No printer type was specified. | Correct the syntax. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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 ends the program run.<br>‘01 = SQR command<br>‘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.<br>‘01 = Specified LEVEL= value<br>‘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.<br>‘01 = SQR command |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 006233 | The TOC (Table of Contents) entry "A" cannot be processed because the existing entry "B" is positioned below it.<br>A: Line = ‘01, Level = ‘02, Text = ’‘03’<br>B: Line = ‘04, Level = ‘05, Text = ’‘06’ | Correct the program logic to eliminate the conflict between the two TOC (Table of Contents) entries.<br>‘01 = A: Line number<br>‘02 = A: Level value<br>‘03 = A: Text value<br>‘04 = B: Line number<br>‘05 = B: Level value<br>‘06 = B: Text value |
| 006300 | Unknown parameter (‘01). | Correct the syntax.<br>‘01 = Parameter name |
| 006301 | Value not valid for parameter (‘01). | Correct the syntax.<br>‘01 = Parameter name |
| 006302 | Invalid option (‘02) for parameter (‘01). | Correct the syntax.<br>‘01 = Parameter name<br>‘02 = Option |
| 006303 | Parameter (‘01) is required, but has not been specified. | Correct the syntax.<br>‘01 = Parameter name |
| 006304 | Parameter (‘01) already specified. | Correct the syntax.<br>‘01 = Parameter name |
| 006305 | Parameter (‘01) does not support &columns. | Correct the syntax.<br>‘01 = Parameter name |
| 006306 | Parameter (‘01) requires equal sign. | Correct the syntax.<br>‘01 = Parameter name |
| 006307 | Parameter (‘01) has an unquoted string. | Correct the syntax.<br>‘01 = Parameter name |


Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 006308 | Missing part of specification for parameter (‘01). | Correct the syntax.<br>‘01 = Parameter name |
| 006309 | Parameter (‘01) requires literal. | Correct the syntax.<br>‘01 = Parameter name |
| 006310 | Parameter (‘01) requires valid numeric value. | Correct the syntax.<br>‘01 = Parameter name |
| 006311 | Parameter (‘01) requires integer value. | Correct the syntax.<br>‘01 = Parameter name |
| 006312 | Parameter (‘01) does not support type supplied. | Correct the syntax.<br>‘01 = Parameter name |
| 006313 | Parameter (‘01) requires valid string. Perhaps quote or $ is missing. | Correct the syntax.<br>‘01 = Parameter name |
| 006314 | Parameter (‘01) does not accept ’NONE’ in this context. | Correct the syntax.<br>‘01 = Parameter name |
| 006315 | Parameter (‘01) requires proper object name. | Correct the syntax.<br>‘01 = Parameter name |
| 006316 | Parameter (‘01) requires array name. | Correct the syntax.<br>‘01 = Parameter name |
| 006317 | Parameter (‘01) does not accept ’AUTOSCALE’ in this context. | Correct the syntax.<br>‘01 = Parameter name |
| 006318 | Parameter (‘01) has improper value list. | Correct the syntax.<br>‘01 = Parameter name |
| 006320 | Parameter (‘01) does not support relative values. | Correct the syntax.<br>‘01 = Parameter name |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = From type<br>‘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.<br>‘01 = Option/request code<br>‘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.<br>‘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.<br>‘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.<br>‘01 = Parameter name<br>‘02 = Data type |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Data location<br>‘02 = Data type |
| 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.<br>‘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.<br>‘01 = Parameter name<br>‘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.<br>‘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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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, 006450 | 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. |
| 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. |
| 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Row<br>‘02 = Column<br>‘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.<br>‘01 = BCL error code |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 006502 | Invalid bar code type (‘01): Valid values are from 1 to 15. | Correct the source code.<br>‘01 = Bar code type. |
| 006503 | The length of the bar code text ‘01’ must be between 1 and 30 characters. | Correct the source code.<br>‘01 = Bar code text |
| 006504 | The length of the caption text ’01’ must be between 1 and 30 characters. | Correct the source code.<br>‘01 = Caption text |
| 006505 | Invalid printer type (‘01): Valid values are from 0 to 13. | Correct the source code.<br>‘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.<br>‘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.<br>‘01 = Bar code text<br>‘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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘01 = Page number |
| 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.<br>‘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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘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.<br>‘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 |
| 006614 | The file (‘01) does not contain a valid bitmap. | (Windows) Specify a valid bitmap file. ‘01 = Name of the file |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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.<br>‘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.<br>‘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.<br>‘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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |
| 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 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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 (for example, file locking and permissions). Record the steps leading up to the error and contact your system administrator.<br>‘01 = Name of the file<br>‘02 = System error code<br>‘03 = System error message |
| 006705 | File (‘01) is not in SPF packet format. | (Windows) The file was not produced by SQR or has been corrupted.<br>‘01 = Name of the file |
| 006706 | Failed to identify the start of the report (‘01). | (Windows) The file was not produced by SQR or has been corrupted.<br>‘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. Contact technical support.<br>‘01 = Page number |
| 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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 (for example, file locking and permissions). Record the steps leading up to the error and contact your system administrator.<br>‘01 = Name of the file<br>‘02 = System error code<br>‘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.<br>‘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.<br>‘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. Review the documentation for the current upper and lower bounds of a decimal object.<br>‘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. Review 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 |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 006804 | ‘01: Decimal to Float Conversion Under/Overflow. | Floating Point Under/Overflow: Cannot convert input decimal object into a valid floating point number. The decimal object exceeds the established floating point boundaries for this machine architecture. Review 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 |
| 006805 | ‘01: Decimal Precision Under/Overflow. | Decimal Precision Under/Overflow: Attempt made to initialize decimal object with an invalid precision. Verify 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: The length of input string is greater than the 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. 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. Ensure that the divisor 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 007000 | The locale ’‘01’ is not defined in the sqr.ini file. | Verify the spelling of the locale name or the sqr.ini file.<br>‘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.<br>‘01 = Qualifier<br>‘02 = Number of entities in list |
| 007003 | The values for ’01’ and ’02’ cannot be the same. | Correct the source code.<br>‘01 = Qualifier<br>‘02 = Qualifier |
| 007004 | The value for ’01’ (‘02) must be a single character which is not in the list: "03". | Correct the source code.<br>‘01 = Qualifier<br>‘02 = Value<br>‘03 = List of invalid characters |
| 007005 | The value for ’‘01’ (‘02) is invalid. Valid values are: | Correct the source code.<br>‘01 = Qualifier<br>‘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.<br>‘01 = Qualifier<br>‘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.<br>‘01 = Qualifier<br>‘02 = Invalid character |
| 00700 | 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.<br>‘01 = Locale name |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 007009 | The value for ’‘01’ cannot be ’DEFAULT’ or ’SYSTEM’. | Correct the syntax.<br>‘01 = Qualifier |
| 007010 | The value for ’‘01’ (‘02) is not properly formatted: Did not find the ’>’ for the ’<nnn>’ construct. | Correct the syntax.<br>‘01 = Qualifier<br>‘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.<br>‘01 = Qualifier<br>‘02 = Value |
| 007012 | The default locale (‘01) specified in the [‘02] section of the sqr.ini file has not been defined. | Correct the syntax.<br>‘01 = Locale name<br>‘02 = Section name |
| 007013 | The value for ’‘01’ (‘02) must be a list of ‘03 quoted string literals. | Correct the syntax.<br>‘01 = Qualifier<br>‘02 = Value<br>‘03 = Number of entities in list |
| 007014 | The entry (‘01 = ‘02) is not valid. | Correct the sqr.ini entry.<br>‘01 = Qualifier from the sqr.ini file<br>‘02 = Qualifier’s value |
| 007100 | The use of an edit mask or the keywords NUMBER, MONEY or DATE is not legal when storing numeric variables. | Correct the source code. |
| 007101 | The last keyword is not ’‘01’. | Correct the source code.<br>‘01 = Keyword |
| 007102 | Incompatible source and destination variable types. | Correct the source code. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 007103 | The keyword (‘01) is not compatible with the variable (‘02). | Correct the source code.<br>‘01 = Keyword<br>‘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.<br>‘01 = Specified precision<br>‘02 = Minimum precision<br>‘03 = Maximum precision |
| 007201 | The precision is specified by a value from ‘01 to ‘02 surrounded by parentheses. | Correct the source code.<br>‘01 = Minimum precision<br>‘02 = Maximum precision |
| 007202 | Variable (‘01) is not a decimal variable and cannot have a precision associated with it. | Correct the source code.<br>‘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.<br>‘01 = Variable name<br>‘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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 007209 | Only string ($) and numeric (#) variables may be declared. | Correct the source code. |
| 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.<br>‘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.<br>‘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.<br>‘01 = SQR command name |
| 007404 | Double-Byte .sqt files are not supported by this version of SQR. | The runtime 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.<br>‘01 = Bar code text |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 007501 | Using ‘01 edit mask from (‘02) against (‘03) | A date edit mask element, which could cause date data to be incorrectly interpreted, was detected. This warning message can be turned off by setting the “OutputTwoDigitYearWarningMsg” entry in the [Default- Settings] section of the sqr.ini file to FALSE.<br>‘01 = Edit mask element<br>‘02 = Edit mask being used<br>‘03 = Value being applied to the edit mask |
| 007601 | Cannot access the Java file (‘01) (‘02): ‘03 | SQR cannot access the required file.<br>‘01 = Name of the file<br>‘02 = System error code<br>‘03 = System error message |
| 007602 | -EH_Scale: value (‘01) is out of range (‘02 - ‘03). | Correct the command line.<br>‘01 = Specified scale<br>‘02 = Minimum allowed<br>‘03 = Maximum allowed |
| 007603 | -Printer:EH functionality is not available on this platform. | Enhanced HTML functionality is not available on this platform. |
| 007604 | -Printer:PD functionality is not available on this platform. | PDF functionality is not available on this platform. |
| 007701 | Did not find end of paragraph: ‘01 (No ’end-execute’ clause found.) | Correct the source code.<br>‘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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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 row sets. |
| 007713 | Missing or invalid Registry.properties file. Verify that the CLASSPATH includes SQRDIR, that SQRDIR contains the folder with the Registry.properties file, and that the Registry.properties file is valid. | Incorrect environment setup. |
| 007714 | The datasource (’‘01’) does not support the requested capability (’‘02’). Check the capabilities list for the datasource, located in the Properties folder. | Invalid query for datasource. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| 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. | <obsolete> |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. | <obsolete> |
| 007736 | Unable to open Connection (’01’) to datasource (’02’). Possible causes: (a) the Declare-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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 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. | <obsolete> |
| 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. | <obsolete> |
| 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 row sets. |
| 007747 | The query raised a DDO exception. DETAILS: '01. | <obsolete> |
| 007748 | A BEGIN-SELECT paragraph was coded, but the query returned no Rows. | No data warning. |
| 007749 | Invalid syntax for PARAMETERS= (...) statement. Use: PARAMETERS=( %v \| $v \| #v \| &v \| NULL \| SKIP \| numlit \| datelit \| textlit [IN \| INOUT], ...) All parameters must be specified. Optional parameters which are to be ignored may be specified by the keyword 'NULL' or 'SKIP'. Correct the syntax. | Incorrect syntax. |
| 007750 | FATAL: Failure creating Java object. | General failure. |
| 007751 | Attempt to create a List variable of size greater than the maximum size of '01 items. | General failure. |
| 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. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 007754 | Attempt to assign/modify a List row is not compatible with the List definition. | Bad list assignment/setup. |
| 007755 | Attempt to assign a row to a non-existent List variable. Define the List first, using the syntax: let %lname[ size ] = list( NUMBER \| DATE \| TEXT #var \| $var [, ...] ) | Bad list assignment/setup. |
| 007756 | Incorrect syntax for List-variable reference. Use: let [$ \| #]var = %listname[nlit \| #var].colname | Bad list assignment/setup. |
| 007757 | Alter-connection statement missing ’DSN=...’. | Improper alter-conn. |
| 007758 | List-definition size specifier must be literal. | Improper alter-conn. |
| 007759 | Attempt to access a non-existent List- column (’01’). | No such list column name. |
| 007760 | Must specify one of the keywords, FROM- ROWSETS or FROM_PARAMETER . | Incorrect syntax for LOAD_LOOKUP. |
| 007761 | Incorrect syntax to Load-lookup ’PARAMETERS=’ keyword. Use: PARAMETERS=(slit \| nlit \| $var \| #var \| %var \| &var, ...) No line wrapping is allowed for this usage. | Incorrect syntax for LOAD_LOOKUP. |
| 007762 | Too many parameters ( ‘02) entered to Load- Lookup command. Max parameters is ‘01. | Incorrect syntax for LOAD_LOOKUP. |
| 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.<br>‘01 = Load lookup table name |

Copyright © 1988-2007, Oracle. All rights reserved. 

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 007764 | Bad return fetching row from database in LOAD-LOOKUP table ''01'. DETAILS: '02 | The database server returned an error while fetching the data.<br>'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. <obsolete> |
| 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. |
| 009000 | Error reading the font information from the ['01] section in sqr.ini. Font name '02 is too long. The maximum length allowed for a font name is '03. | Use a shorter font name in sqr.ini.<br>'01 = Printing device specific font configuration section.<br>'02 = Font name.<br>'03 = Internal limit for font name. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 009001 | Error reading the font information from the [‘01] section in sqr.ini. Font file path ‘02 is too long. The maximum length allowed for a font file path is ‘03. | Use a shorter font file path in sqr.ini. Relocate font file if necessary.<br>‘01 = Font configuration section name.<br>‘02 = Font file path.<br>‘03 = Internal limit for file path. |
| 009002 | Font ‘01 is not valid for ‘02 output. Please correct the font configuration in the [‘02 Fonts] section of the sqr.ini file. | Use a valid font. Refer to the documentation for supported fonts. If you are using disk based fonts like TrueType, also ensure that the font type specific configuration section (like [TrueType Fonts]) is correctly configured.<br>‘01 = Font name.<br>‘02 = Selected output type. |
| 009003 | File name for ‘01 font ‘02 is not specified correctly. Please correct the configuration in the [‘01 Fonts] section of sqr.ini file. | The font file path is not specified correctly. Correct the sqr.ini file.<br>‘01 = Font type.<br>‘02 = Font name. |
| 009004 | ‘01 font file ‘02 cannot be opened. Please check if the file exists, or correct the font configuration in the [‘01 Fonts] section of the sqr.ini file. | Either the font file does not exist in the location specified in sqr.ini file, or the path specified in sqr.ini file is not correct. Correct the error and try again.<br>‘01 = Font type.<br>‘02 = Font file path. |
| 009005 | The directory ID specified for ‘01 in the [TrueType Fonts] section of the sqr.ini file exceeds the actual number of fonts included in the TrueType collection font file. ‘02 includes ‘03 fonts. Please specify a valid directory ID. | Correct the configuration in sqr.ini so that the directory ID is set correctly for the font. Note that the directory ID starts from 0.<br>‘01 = Font name.<br>‘02 = Font file path.<br>‘03 = Actual number of fonts included in TrueType collection. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Error Number | Error Message | Suggestion/Interpretation |
|---|---|---|
| 009006 | Font ‘01 is not a supported type of TrueType/OpenType font. | Currently, SQR does not support CFF based OpenType font. Remove the font from the configuration and try again. |
| 009007 | Font ‘01 is not a supported type of TrueType/OpenType font. The ’‘02’ table is missing from the font file. | This TrueType/OpenType font does not have the table SQR needs for processing. Remove the font from the configuration and try again.<br>‘01 = Font name.<br>‘02 = Table name. |
| 009008 | Font ‘01 is not a supported type of TrueType/OpenType font. This font does not allow embedding in a document. | SQR requires TrueType/OpenType font with embedding feature allowed to include the subset of the font in the print output. This TrueType/OpenType font does not allow embedding. Remove the font from the configuration and try again. |
| 009009 | Font ‘01 is not a supported type of TrueType/OpenType font. This font does not have a supported type of character to glyph mapping (CMAP) table. | SQR requires a TrueType/OpenType font with CMAP table with Platform ID 3 (Microsoft), Encoding ID 0 (Symbol), 4 (UCS-2), or 10 (UCS-4) and table format 4 or 12. Other CMAP table types and encodings are currently not supported. Remove the font from the configuration and try again.<br>‘01 = Font name |
| 009010 | Error in the [‘01:Exclusion Ranges] section in theINI file. Start character code must be greater than the end character code. | In the exclusion range section, the start character code must be smaller than the end character code. Review the section in INI file and correct error. You can specify the range in a decimal or hexadecimal number. If you use a hexadecimal number, you must prefix the number with ’0x’ or the number is recognized as a decimal. |

 Copyright © 1988-2007, Oracle. All rights reserved.

# APPENDIX B

# Using SQR Sample Programs

This appendix discusses SQR samples.

## Understanding SQR Samples

SQR Samples is a library of SQR programs that you can use to adapt and experiment with programs. These programs are stored in the <PS_HOME>\bin\sqr\<database_platform>\SAMPLE (SAMPLEW for Microsoft Windows) directory. You can modify these programs to create configured SQRs.

### SQR Sample Programs

Each program comprises a report specification and a sample of the output. This table describes all of the sample SQR programs.

| Name | Description |
|---|---|
| _____.DAT | Data files that are used by the loadall.sqr programs. |
| _____.MEM | SQR startup files to run small, medium, and large SQR programs. |
| APPEND.SQR | Demonstrates the APPEND and FIXED-NOLF (no line feed) commands. |
| APTDIARY.SQR | Demonstrates columns and 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. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Name | Description |
|---|---|
| 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 a 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 a user, enter the data in the database, and write a form letter by using a DOCUMENT paragraph. |
| CUST.SQR | Prints a list of all of the customers, bursted by page. |
| CUSTLBLS.SQR | Demonstrates printing mailing labels within columns. |
| CUSTOMER.SQR | Demonstrates multiple detail lines and the 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 the report output. |
| CUSTOMR4.SQR | Demonstrates the use of arrays. |
| CUSTOMR5.SQR | Demonstrates dynamic queries to enable the user to qualify a report as it runs. |
| CUST_SUM.SQR | Prints a group of information about each customer in the customer table. |
| CUSTTAPE.SQR | Demonstrates the flat file output for magnetic tape or other postprocessing. |
| DATAA.DAT | Needed for append.sqr. |

 Copyright © 1988-2007, Oracle. All rights reserved.

| Name | Description |
|---|---|
| DATAB.DAT | Needed for append.sqr. |
| DROPALL.SQR | Drops all of the SQR sample tables that are created by the LOADALL program. |
| DROPPROC.SQR | (Sybase) Deletes leftover, temporary stored procedures belonging to the user. |
| DYNAMCOL.SQR | Demonstrates the use of dynamic columns, dynamic tables, and variables that are passed to the ON-ERROR procedure. |
| EMP.SQR | Prints a list of all of the employees, bursted by page. |
| EMP_COMM.SQR | Calculates each employee's commission, based on sales. |
| EMP_M_Q.SQR | Lists all employee quotas for a given month or quarter. |
| ENVELOPE.SQR | Demonstrates the use of printing envelope with proper bar codes. |
| EXPORT.SQR | Creates two SQR reports: one to export a database table and one 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 in a flat file. |
| FLOATBAR.SQR | Demonstrates a floating bar chart. |
| FOR_CUST.SQR | Creates a sales forecast for a given customer, grouped by month or quarter. |
| FOR_EMP.SQR | Creates a sales forecast for a given employee, grouped by month or quarter. |
| FOR_PROD.SQR | Creates a sales forecast for a given product, grouped by month or quarter. |
| FOR_REG.SQR | Creates a sales forecast for a given region, grouped by month or quarter. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Name | Description |
|---|---|
| FOR_SUM.SQR | Creates a table of projected product sales with links to more information. |
| FORMLETR.SQR | Demonstrates form letters by 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 that are selected from a database table that you specify. The resulting SQR program prompts you to qualify the rows to be selected, display those rows, then repeat. |
| INVOICE.SQR | Demonstrates creating multiple reports, printing invoices, and printing envelopes. |
| LOADALL.SQR | Creates and loads the sample tables that are used in the preceding SQR programs. |
| MAKEDATA.SQR | Creates a data file with a fixed length and NOLF attributes. |
| MAKEREPT.SQR | Helps you create SQR reports more quickly. |
| MULTIPLE.SQR | Demonstrates creating multiple reports. |
| NESTREPT.SQR | Demonstrates nesting procedures. |
| ORDERS.SQR | Lists all of the orders and the order lines that are associated with them. |
| ORD_MONG.SQR | Lists all orders for a given month and groups them by employee number. |
| ORD_M_Q.SQR | Lists all orders for a given month or quarter. |
| ORD_PROD.SQR | Lists all orders for a given product. |
| ORD_REGG.SQR | Creates a report of all orders from a given region, grouped by month or quarter. |


Copyright © 1988-2007, Oracle. All rights reserved.

| Name | Description |
|---|---|
| ORD_SUM.SQR | Displays an summary of orders, grouped by month. |
| ORD_S_Q.SQR | Prints a graph of the percent of orders for each region (in a year) and four graphs of the percent of orders for each region (one for each quarter of that year). |
| OVERBAR.SQR | Demonstrates an overlapped bar chart. |
| PHONELST.SQR | Demonstrates printing within columns, page headings, and page footings. |
| PLN_EMP.SQR | Creates a sales plan for a given employee, grouped by month or quarter. |
| PLN_GEN.SQR | Creates a sales plan, grouped by month or quarter. |
| PLN_REG.SQR | Creates a sales plan for given region, grouped by month or quarter. |
| PRODUCT.SQR | Lists products and their prices and graphs 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) Displays 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 input/output commands. |
| SQRLINE.SQR | Demonstrates a line chart. |
| SQRLOGO.SQR | Demonstrates printing images. |

Copyright © 1988-2007, Oracle. All rights reserved.

| Name | Description |
|---|---|
| SQRPIE.SQR | Demonstrates a pie chart. |
| TABREP.SQR | Creates a tabular SQR report for a table that you select. |
| UPDATE.SQR | Generates an SQR program that enables you to query and update database tables. The created program uses the SHOW command to simulate a menu interface. |


Copyright © 1988-2007, Oracle. All rights reserved.

# Glossary of PeopleSoft Enterprise Terms

**absence entitlement**
This element defines rules for granting paid time off for valid absences, such as sick time, vacation, and maternity leave. An absence entitlement element defines the entitlement amount, frequency, and entitlement period.

**absence take**
This element defines the conditions that must be met before a payee is entitled to take paid time off.

**academic career**
In PeopleSoft Enterprise Campus Solutions, all course work that a student undertakes at an academic institution and that is grouped in a single student record. For example, a university that has an undergraduate school, a graduate school, and various professional schools might define several academic careers—an undergraduate career, a graduate career, and separate careers for each professional school (law school, medical school, dental school, and so on).

**academic institution**
In PeopleSoft Enterprise Campus Solutions, an entity (such as a university or college) that is independent of other similar entities and that has its own set of rules and business processes.

**academic organization**
In PeopleSoft Enterprise Campus Solutions, an entity that is part of the administrative structure within an academic institution. At the lowest level, an academic organization might be an academic department. At the highest level, an academic organization can represent a division.

**academic plan**
In PeopleSoft Enterprise Campus Solutions, an area of study—such as a major, minor, or specialization—that exists within an academic program or academic career.

**academic program**
In PeopleSoft Enterprise Campus Solutions, the entity to which a student applies and is admitted and from which the student graduates.

**accounting class**
In PeopleSoft Enterprise Performance Management, the accounting class defines how a resource is treated for generally accepted accounting practices. The Inventory class indicates whether a resource becomes part of a balance sheet account, such as inventory or fixed assets, while the Non-inventory class indicates that the resource is treated as an expense of the period during which it occurs.

**accounting date**
The accounting date indicates when a transaction is recognized, as opposed to the date the transaction actually occurred. The accounting date and transaction date can be the same. The accounting date determines the period in the general ledger to which the transaction is to be posted. You can only select an accounting date that falls within an open period in the ledger to which you are posting. The accounting date for an item is normally the invoice date.

**accounting split**
The accounting split method indicates how expenses are allocated or divided among one or more sets of accounting ChartFields.

**accumulator**
You use an accumulator to store cumulative values of defined items as they are processed. You can accumulate a single value over time or multiple values over time. For example, an accumulator could consist of all voluntary deductions, or all company deductions, enabling you to accumulate amounts. It allows total flexibility for time periods and values accumulated.

**action reason**
The reason an employee's job or employment information is updated. The action reason is entered in two parts: a personnel action, such as a promotion, termination, or change from one pay group to another—and a reason for that action. Action reasons are used by PeopleSoft Enterprise Human Resources, PeopleSoft Enterprise Benefits

Copyright © 1988-2007, Oracle. All rights reserved.

Administration, PeopleSoft Enterprise Stock Administration, and the COBRA Administration feature of the Base Benefits business process.

**action template**

In PeopleSoft Enterprise Receivables, outlines a set of escalating actions that the system or user performs based on the period of time that a customer or item has been in an action plan for a specific condition.

**activity**

In PeopleSoft Enterprise Learning Management, an instance of a catalog item (sometimes called a class) that is available for enrollment. The activity defines such things as the costs that are associated with the offering, enrollment limits and deadlines, and waitlisting capacities.

In PeopleSoft Enterprise Performance Management, the work of an organization and the aggregation of actions that are used for activity-based costing.

In PeopleSoft Enterprise Project Costing, the unit of work that provides a further breakdown of projects—usually into specific tasks.

In PeopleSoft Workflow, a specific transaction that you might need to perform in a business process. Because it consists of the steps that are used to perform a transaction, it is also known as a step map.

**address usage**

In PeopleSoft Enterprise Campus Solutions, a grouping of address types defining the order in which the address types are used. For example, you might define an address usage code to process addresses in the following order: billing address, dormitory address, home address, and then work address.

**adjustment calendar**

In PeopleSoft Enterprise Campus Solutions, the adjustment calendar controls how a particular charge is adjusted on a student's account when the student drops classes or withdraws from a term. The charge adjustment is based on how much time has elapsed from a predetermined date, and it is determined as a percentage of the original charge amount.

**administrative function**

In PeopleSoft Enterprise Campus Solutions, a particular functional area that processes checklists, communication, and comments. The administrative function identifies which variable data is added to a person's checklist or communication record when a specific checklist code, communication category, or comment is assigned to the student. This key data enables you to trace that checklist, communication, or comment back to a specific processing event in a functional area.

**admit type**

In PeopleSoft Enterprise Campus Solutions, a designation used to distinguish first-year applications from transfer applications.

**agreement**

In PeopleSoft Enterprise eSettlements, provides a way to group and specify processing options, such as payment terms, pay from a bank, and notifications by a buyer and supplier location combination.

**allocation rule**

In PeopleSoft Enterprise Incentive Management, an expression within compensation plans that enables the system to assign transactions to nodes and participants. During transaction allocation, the allocation engine traverses the compensation structure from the current node to the root node, checking each node for plans that contain allocation rules.

**alternate account**

A feature in PeopleSoft Enterprise General Ledger that enables you to create a statutory chart of accounts and enter statutory account transactions at the detail transaction level, as required for recording and reporting by some national governments.

**analysis database**

In PeopleSoft Enterprise Campus Solutions, database tables that store large amounts of student information that may not appear in standard report formats. The analysis database tables contain keys for all objects in a report that an application program can use to reference other student-record objects that are not contained in the printed report. For instance, the analysis database contains data on courses that are considered


Copyright © 1988-2007, Oracle. All rights reserved.

for satisfying a requirement but that are rejected. It also contains information on courses captured by global limits. An analysis database is used in PeopleSoft Enterprise Academic Advisement.

**Application Messaging** PeopleSoft Application Messaging enables applications within the PeopleSoft Enterprise product family to communicate synchronously or asynchronously with other PeopleSoft Enterprise and third-party applications. An application message defines the records and fields to be published or subscribed to.

**AR specialist** Abbreviation for *receivables specialist.* In PeopleSoft Enterprise Receivables, an individual in who tracks and resolves deductions and disputed items.

**arbitration plan** The arbiter when multiple price rules match the transaction. This plan determines the order in which the price rules are applied to the transaction base price.

**assessment rule** In PeopleSoft Enterprise Receivables, a user-defined rule that the system uses to evaluate the condition of a customer's account or of individual items to determine whether to generate a follow-up action.

**asset class** An asset group used for reporting purposes. It can be used in conjunction with the asset category to refine asset classification.

**attribute/value pair** In PeopleSoft Enterprise Directory Interface, relates the data that makes up an entry in the directory information tree.

**auction event** In PeopleSoft Strategic Sourcing, a sourcing event where bidders actively compete against one another to achieve the best price or score.

**audience** In PeopleSoft Enterprise Campus Solutions, a segment of the database that relates to an initiative, or a membership organization that is based on constituent attributes rather than a dues-paying structure. Examples of audiences include the Class of '65 and Undergraduate Arts & Sciences.

**authentication server** A server that is set up to verify users of the system.

**base time period** In PeopleSoft Enterprise Business Planning, the lowest level time period in a calendar.

**benchmark job** In PeopleSoft Enterprise Workforce Analytics Solution, a benchmark job is a job code for which there is corresponding salary survey data from published, third-party sources.

**bid response** In PeopleSoft Strategic Sourcing, the response by a bidder to an event.

**billing career** In PeopleSoft Enterprise Campus Solutions, the one career under which other careers are grouped for billing purposes if a student is active simultaneously in multiple careers.

**bio bit or bio brief** In PeopleSoft Enterprise Campus Solutions, a report that summarizes information stored in the system about a particular constituent. You can generate standard or specialized reports.

**book** In PeopleSoft Enterprise Asset Management, used for storing financial and tax information, such as costs, depreciation attributes, and retirement information on assets.

**branch** A tree node that rolls up to nodes above it in the hierarchy, as defined in PeopleSoft Tree Manager.

**budgetary account only** An account used by the system only and not by users; this type of account does not accept transactions. You can only budget with this account. Formerly called "system-maintained account."

Copyright © 1988-2007, Oracle. All rights reserved.

**budget check** In commitment control, the processing of source transactions against control budget ledgers, to see if they pass, fail, or pass with a warning.

**budget control** In commitment control, budget control ensures that commitments and expenditures don't exceed budgets. It enables you to track transactions against corresponding budgets and terminate a document's cycle if the defined budget conditions are not met. For example, you can prevent a purchase order from being dispatched to a vendor if there are insufficient funds in the related budget to support it.

**budget period** The interval of time (such as 12 months or 4 quarters) into which a period is divided for budgetary and reporting purposes. The ChartField allows maximum flexibility to define operational accounting time periods without restriction to only one calendar.

**business activity** The name of a subset of a detailed business process. This might be a specific transaction, task, or action that you perform in a business process.

**business event** In PeopleSoft Enterprise Receivables, defines the processing characteristics for the Receivable Update process for a draft activity.

In PeopleSoft Enterprise Sales Incentive Management, an original business transaction or activity that may justify the creation of a PeopleSoft Enterprise Incentive Management event (a sale, for example).

**business process** A standard set of 17 business processes are defined and maintained by the PeopleSoft Enterprise product families and are supported by the Business Process Engineering group. An example of a business process is Order Fulfillment, which is a business process that manages sales orders and contracts, inventory, billing, and so forth.

See also *detailed business process.*

**business unit constraints** In PeopleSoft Strategic Sourcing, these constraints apply to a selected Strategic Sourcing business unit. Spend is tracked across all of the events within the selected Strategic Sourcing business unit.

**business task** The name of the specific function depicted in one of the business processes.

**business unit** A corporation or a subset of a corporation that is independent with regard to one or more operational or accounting functions.

**buyer** In PeopleSoft Enterprise eSettlements, an organization (or business unit, as opposed to an individual) that transacts with suppliers (vendors) within the system. A buyer creates payments for purchases that are made in the system.

**buy event** In PeopleSoft Strategic Sourcing, for event creators, the purchase of goods or services, most typically associated with a request for quote, proposal, or reverse auction.For bidders, the sale of goods or services.

**campus** In PeopleSoft Enterprise Campus Solutions, an entity that is usually associated with a distinct physical administrative unit, that belongs to a single academic institution, that uses a unique course catalog, and that produces a common transcript for students within the same academic career.

**cash drawer** A repository for monies and payments taken locally.

**catalog item** In PeopleSoft Enterprise Learning Management, a specific topic that a learner can study and have tracked. For example, "Introduction to Microsoft Word." A catalog item contains general information about the topic and includes a course code, description, categorization, keywords, and delivery methods. A catalog item can have one or more learning activities.

**catalog map** In PeopleSoft Enterprise Catalog Management, translates values from the catalog source data to the format of the company's catalog.

 Copyright © 1988-2007, Oracle. All rights reserved.

**catalog partner** In PeopleSoft Enterprise Catalog Management, shares responsibility with the enterprise catalog manager for maintaining catalog content.

**categorization** Associates partner offerings with catalog offerings and groups them into enterprise catalog categories.

**category** In PeopleSoft Enterprise Campus Solutions, a broad grouping to which specific comments or communications (contexts) are assigned. Category codes are also linked to 3C access groups so that you can assign data-entry or view-only privileges across functions.

**channel** In PeopleSoft MultiChannel Framework, email, chat, voice (computer telephone integration [CTI]), or a generic event.

**ChartField** A field that stores a chart of accounts, resources, and so on, depending on the PeopleSoft Enterprise application. ChartField values represent individual account numbers, department codes, and so forth.

**ChartField balancing** You can require specific ChartFields to match up (balance) on the debit and the credit side of a transaction.

**ChartField combination edit** The process of editing journal lines for valid ChartField combinations based on user-defined rules.

**ChartKey** One or more fields that uniquely identify each row in a table. Some tables contain only one field as the key, while others require a combination.

**checkbook** In PeopleSoft Enterprise Promotions Management, enables you to view financial data (such as planned, incurred, and actual amounts) that is related to funds and trade promotions.

**checklist code** In PeopleSoft Enterprise Campus Solutions, a code that represents a list of planned or completed action items that can be assigned to a staff member, volunteer, or unit. Checklists enable you to view all action assignments on one page.

**claimback** In the wholesale distribution industry, a contract between supplier and distributor, in which monies are paid to the distributor on the sale of specified products or product groups to targeted customers or customer groups.

**class** In PeopleSoft Enterprise Campus Solutions, a specific offering of a course component within an academic term.

See also *course.*

**Class ChartField** A ChartField value that identifies a unique appropriation budget key when you combine it with a fund, department ID, and program code, as well as a budget period. Formerly called *sub-classification.*

**clearance** In PeopleSoft Enterprise Campus Solutions, the period of time during which a constituent in PeopleSoft Enterprise Contributor Relations is approved for involvement in an initiative or an action. Clearances are used to prevent development officers from making multiple requests to a constituent during the same time period.

**clone** In PeopleCode, to make a unique copy. In contrast, to *copy* may mean making a new reference to an object, so if the underlying object is changed, both the copy and the original change.

**cohort** In PeopleSoft Enterprise Campus Solutions, the highest level of the three-level classification structure that you define for enrollment management. You can define a cohort level, link it to other levels, and set enrollment target numbers for it.

See also *population* and *division*.

Copyright © 1988-2007, Oracle. All rights reserved.

**collection**

To make a set of documents available for searching in Verity, you must first create at least one collection. A collection is set of directories and files that allow search application users to use the Verity search engine to quickly find and display source documents that match search criteria. A collection is a set of statistics and pointers to the source documents, stored in a proprietary format on a file server. Because a collection can only store information for a single location, PeopleTools maintains a set of collections (one per language code) for each search index object.

**collection rule**

In PeopleSoft Enterprise Receivables, a user-defined rule that defines actions to take for a customer based on both the amount and the number of days past due for outstanding balances.

**comm key**

See *communication key.*

**communication key**

In PeopleSoft Enterprise Campus Solutions, a single code for entering a combination of communication category, communication context, communication method, communication direction, and standard letter code. Communication keys (also called *comm keys* or *speed keys*) can be created for background processes as well as for specific users.

**compensation object**

In PeopleSoft Enterprise Incentive Management, a node within a compensation structure. Compensation objects are the building blocks that make up a compensation structure's hierarchical representation.

**compensation structure**

In PeopleSoft Enterprise Incentive Management, a hierarchical relationship of compensation objects that represents the compensation-related relationship between the objects.

**component interface**

A component interface is a set of application programming interfaces (APIs) that you can use to access and modify PeopleSoft Enterprise database information using a program instead of the PeopleSoft client.

**condition**

In PeopleSoft Enterprise Receivables, occurs when there is a change of status for a customer's account, such as reaching a credit limit or exceeding a user-defined balance due.

**configuration parameter catalog**

Used to configure an external system with PeopleSoft Enterprise. For example, a configuration parameter catalog might set up configuration and communication parameters for an external server.

**configuration plan**

In PeopleSoft Enterprise Incentive Management, configuration plans hold allocation information for common variables (not incentive rules) and are attached to a node without a participant. Configuration plans are not processed by transactions.

**constituents**

In PeopleSoft Enterprise Campus Solutions, friends, alumni, organizations, foundations, or other entities affiliated with the institution, and about which the institution maintains information. The constituent types delivered with PeopleSoft Enterprise Contributor Relations Solutions are based on those defined by the Council for the Advancement and Support of Education (CASE).

**constraint**

A business policy or rule that affects how a sourcing event is awarded. There are three types of constraints: business, global, and event.

**content reference**

Content references are pointers to content registered in the portal registry. These are typically either URLs or iScripts. Content references fall into three categories: target content, templates, and template pagelets.

**context**

In PeopleCode, determines which buffer fields can be contextually referenced and which is the current row of data on each scroll level when a PeopleCode program is running.

In PeopleSoft Enterprise Campus Solutions, a specific instance of a comment or communication. One or more contexts are assigned to a category, which you link to


Copyright © 1988-2007, Oracle. All rights reserved.

3C access groups so that you can assign data-entry or view-only privileges across functions.

In PeopleSoft Enterprise Incentive Management, a mechanism that is used to determine the scope of a processing run. PeopleSoft Enterprise Incentive Management uses three types of context: plan, period, and run-level.

**control table** Stores information that controls the processing of an application. This type of processing might be consistent throughout an organization, or it might be used only by portions of the organization for more limited sharing of data.

**cost plus contract line** A rate-based contract line associated with a fee component of Award, Fixed, Incentive, or Other. Rate-based contract lines associated with a fee type of None are not considered cost-plus contract lines.

**cost plus pricing** In PeopleSoft Enterprise Pricer, a pricing method that begins with cost of goods as the basis.

**cost profile** A combination of a receipt cost method, a cost flow, and a deplete cost method. A profile is associated with a cost book and determines how items in that book are valued, as well as how the material movement of the item is valued for the book.

**cost row** A cost transaction and amount for a set of ChartFields.

**counter sale** A face-to-face customer transaction where the customer typically selects items from the storefront or picks up products that they ordered ahead of time. Customers pay for the goods at the counter and take the goods with them instead of having the goods shipped from a warehouse.

**course** In PeopleSoft Enterprise Campus Solutions, a course that is offered by a school and that is typically described in a course catalog. A course has a standard syllabus and credit level; however, these may be modified at the class level. Courses can contain multiple components such as lecture, discussion, and lab.

See also *class.*

**course share set** In PeopleSoft Enterprise Campus Solutions, a tag that defines a set of requirement groups that can share courses. Course share sets are used in PeopleSoft Enterprise Academic Advisement.

**current learning** In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's in-progress learning activities and programs.

**data acquisition** In PeopleSoft Enterprise Incentive Management, the process during which raw business transactions are acquired from external source systems and fed into the operational data store (ODS).

**data cube** In PeopleSoft Analytic Calculation Engine, a data cube is a container for one kind of data (such as Sales data) and works with in tandem with one or more dimensions. Dimensions and data cubes in PeopleSoft Analytic Calculation Engine are unrelated to dimensions and online analytical processing (OLAP) cubes in PeopleSoft Cube Manager.

**data elements** Data elements, at their simplest level, define a subset of data and the rules by which to group them.

For Workforce Analytics, data elements are rules that tell the system what measures to retrieve about your workforce groups.

**dataset** A data grouping that enables role-based filtering and distribution of data. You can limit the range and quantity of data that is displayed for a user by associating dataset rules with user roles. The result of dataset rules is a set of data that is appropriate for the user's roles.

Copyright © 1988-2007, Oracle. All rights reserved.

**delivery method**

In PeopleSoft Enterprise Learning Management, identifies the primary type of delivery method in which a particular learning activity is offered. Also provides default values for the learning activity, such as cost and language. This is primarily used to help learners search the catalog for the type of delivery from which they learn best. Because PeopleSoft Enterprise Learning Management is a blended learning system, it does not enforce the delivery method.

In PeopleSoft Enterprise Supply Chain Management, identifies the method by which goods are shipped to their destinations (such as truck, air, and rail). The delivery method is specified when creating shipment schedules.

**delivery method type**

In PeopleSoft Enterprise Learning Management, identifies how learning activities can be delivered—for example, through online learning, classroom instruction, seminars, books, and so forth—in an organization. The type determines whether the delivery method includes scheduled components.

**detailed business process**

A subset of the business process. For example, the detailed business process named Determine Cash Position is a subset of the business process called Cash Management.

**dimension**

In PeopleSoft Analytic Calculation Engine, a dimension contains a list of one kind of data that can span various contexts, and it is a basic component of an analytic model. Within the analytic model, a dimension is attached to one or more data cubes. In PeopleSoft Cube Manager, a dimension is the most basic component of an OLAP cube and specifies the PeopleSoft metadata to be used to create the dimension's rollup structure. Dimensions and data cubes in PeopleSoft Analytic Calculation Engine are unrelated to dimensions and OLAP cubes in PeopleSoft Cube Manager.

**direct receipt**

Items shipped from a warehouse or vendor to another warehouse.

**direct ship**

Items shipped from the vendor or warehouse directly to the customer (formerly referred to as *drop ship*).

**directory information tree**

In PeopleSoft Enterprise Directory Interface, the representation of a directory's hierarchical structure.

**division**

In PeopleSoft Enterprise Campus Solutions, the lowest level of the three-level classification structure that you define in PeopleSoft Enterprise Recruiting and Admissions for enrollment management. You can define a division level, link it to other levels, and set enrollment target numbers for it.

See also *population* and *cohort.*

**document sequencing**

A flexible method that sequentially numbers the financial transactions (for example, bills, purchase orders, invoices, and payments) in the system for statutory reporting and for tracking commercial transaction activity.

**dynamic detail tree**

A tree that takes its detail values—dynamic details—directly from a table in the database, rather than from a range of values that are entered by the user.

**edit table**

A table in the database that has its own record definition, such as the Department table. As fields are entered into a PeopleSoft Enterprise application, they can be validated against an edit table to ensure data integrity throughout the system.

**effective date**

A method of dating information in PeopleSoft Enterprise applications. You can predate information to add historical data to your system, or postdate information in order to enter it before it actually goes into effect. By using effective dates, you don't delete values; you enter a new value with a current effective date.

**EIM ledger**

Abbreviation for *Enterprise Incentive Management ledger.* In PeopleSoft Enterprise Incentive Management, an object to handle incremental result gathering within the scope of a participant. The ledger captures a result set with all of the appropriate traces to the data origin and to the processing steps of which it is a result.


Copyright © 1988-2007, Oracle. All rights reserved.

**elimination set** In PeopleSoft Enterprise General Ledger, a related group of intercompany accounts that is processed during consolidations.

**entry event** In PeopleSoft Enterprise General Ledger, Receivables, Payables, Purchasing, and Billing, a business process that generates multiple debits and credits resulting from single transactions to produce standard, supplemental accounting entries.

**equitization** In PeopleSoft Enterprise General Ledger, a business process that enables parent companies to calculate the net income of subsidiaries on a monthly basis and adjust that amount to increase the investment amount and equity income amount before performing consolidations.

**equity item limit** In PeopleSoft Enterprise Campus Solutions, the amounts of funds set by the institution to be awarded with discretionary or gift funds. The limit could be reduced by amounts equal to such things as expected family contribution (EFC) or parent contribution. Students are packaged by Equity Item Type Groups and Related Equity Item Types. This limit can be used to assure that similar student populations are packaged equally.

**event** A predefined point either in the Component Processor flow or in the program flow. As each point is encountered, the event activates each component, triggering any PeopleCode program that is associated with that component and that event. Examples of events are FieldChange, SavePreChange, and RowDelete.

In PeopleSoft Enterprise Human Resources, also refers to an incident that affects benefits eligibility.

**event constraints** In PeopleSoft Strategic Sourcing, these constraints are associated with a specific sourcing event. Spend is tracked within the selected event.

**event propagation process** In PeopleSoft Enterprise Sales Incentive Management, a process that determines, through logic, the propagation of an original PeopleSoft Enterprise Incentive Management event and creates a derivative (duplicate) of the original event to be processed by other objects. PeopleSoft Enterprise Enterprise Sales Incentive Management uses this mechanism to implement splits, roll-ups, and so on. Event propagation determines who receives the credit.

**exception** In PeopleSoft Enterprise Receivables, an item that either is a deduction or is in dispute.

**exclusive pricing** In PeopleSoft Enterprise Order Management, a type of arbitration plan that is associated with a price rule. Exclusive pricing is used to price sales order transactions.

**fact** In PeopleSoft Enterprise applications, facts are numeric data values from fields from a source database as well as an analytic application. A fact can be anything you want to measure your business by, for example, revenue, actual, budget data, or sales numbers. A fact is stored on a fact table.

**financial aid term** In PeopleSoft Enterprise Campus Solutions, a combination of a period of time that the school determines as an instructional accounting period and an academic career. It is created and defined during the setup process. Only terms eligible for financial aid are set up for each financial aid career.

**financial sanctions** For U.S. based companies and their foreign subsidiaries, a federal regulation from the Office of Foreign Assets Control (OFAC) requires that vendors be validated against a Specially Designated Nationals (SDN) list prior to payment.

For PeopleSoft Payables, eSettlements, Cash Management, and Order to Cash, you can validate your vendors against any financial sanctions list (for example, the SDN list, a European Union list, and so on).

**forecast item** A logical entity with a unique set of descriptive demand and forecast data that is used as the basis to forecast demand. You create forecast items for a wide range of uses, but they ultimately represent things that you buy, sell, or use in your organization and for which you require a predictable usage.

Copyright © 1988-2007, Oracle. All rights reserved. 

**fund**

In PeopleSoft Enterprise Promotions Management, a budget that can be used to fund promotional activity. There are four funding methods: top down, fixed accrual, rolling accrual, and zero-based accrual.

**gap**

In PeopleSoft Enterprise Campus Solutions, an artificial figure that sets aside an amount of unmet financial aid need that is not funded with Title IV funds. A gap can be used to prevent fully funding any student to conserve funds, or it can be used to preserve unmet financial aid need so that institutional funds can be awarded.

**generic process type**

In PeopleSoft Process Scheduler, process types are identified by a generic process type. For example, the generic process type SQR includes all SQR process types, such as SQR process and SQR report.

**gift table**

In PeopleSoft Enterprise Campus Solutions, a table or so-called *donor pyramid* describing the number and size of gifts that you expect will be needed to successfully complete the campaign in PeopleSoft Enterprise Contributor Relations. The gift table enables you to estimate the number of donors and prospects that you need at each gift level to reach the campaign goal.

**GDS**

Abbreviation for *Global Distribution System.* Broad-based term to describe all computer reservation systems for making travel plans.

**GL business unit**

Abbreviation for *general ledger business unit.* A unit in an organization that is an independent entity for accounting purposes. It maintains its own set of accounting books.

See also *business unit.*

**GL entry template**

Abbreviation for *general ledger entry template.* In PeopleSoft Enterprise Campus Solutions, a template that defines how a particular item is sent to the general ledger. An item-type maps to the general ledger, and the GL entry template can involve multiple general ledger accounts. The entry to the general ledger is further controlled by high-level flags that control the summarization and the type of accounting—that is, accrual or cash.

**GL Interface process**

Abbreviation for *General Ledger Interface process.* In PeopleSoft Enterprise Campus Solutions, a process that is used to send transactions from PeopleSoft Enterprise Student Financials to the general ledger. Item types are mapped to specific general ledger accounts, enabling transactions to move to the general ledger when the GL Interface process is run.

**global constraints**

In PeopleSoft Strategic Sourcing, these constraints apply across multiple Strategic Sourcing business units. Spend is tracked across all of the events from the multiple Strategic Sourcing business units.

**group**

In PeopleSoft Enterprise Billing and Receivables, a posting entity that comprises one or more transactions (items, deposits, payments, transfers, matches, or write-offs).

In PeopleSoft Enterprise Human Resources Management and Supply Chain Management, any set of records that are associated under a single name or variable to run calculations in PeopleSoft business processes. In PeopleSoft Enterprise Time and Labor, for example, employees are placed in groups for time reporting purposes.

**ideal response**

In PeopleSoft Strategic Sourcing, a question that requires the response to match the ideal value for the bid to be considered eligible for award. If the response does not match the ideal value, you can still submit the bid, but it will be disqualified and ineligible for award.

**incentive object**

In PeopleSoft Enterprise Incentive Management, the incentive-related objects that define and support the PeopleSoft Enterprise Incentive Management calculation process and results, such as plan templates, plans, results data, and user interaction objects.


Copyright © 1988-2007, Oracle. All rights reserved.

**incentive rule**

In PeopleSoft Enterprise Sales Incentive Management, the commands that act on transactions and turn them into compensation. A rule is one part in the process of turning a transaction into compensation.

**incur**

In PeopleSoft Enterprise Promotions Management, to become liable for a promotional payment. In other words, you owe that amount to a customer for promotional activities.

**initiative**

In PeopleSoft Enterprise Campus Solutions, the basis from which all advancement plans are executed. It is an organized effort targeting a specific constituency, and it can occur over a specified period of time with specific purposes and goals. An initiative can be a campaign, an event, an organized volunteer effort, a membership drive, or any other type of effort defined by the institution. Initiatives can be multipart, and they can be related to other initiatives. This enables you to track individual parts of an initiative, as well as entire initiatives.

**inquiry access**

In PeopleSoft Enterprise Campus Solutions, a type of security access that permits the user only to view data.

See also *update access.*

**institution**

In PeopleSoft Enterprise Campus Solutions, an entity (such as a university or college) that is independent of other similar entities and that has its own set of rules and business processes.

**integration**

A relationship between two compatible integration points that enables communication to take place between systems. Integrations enable PeopleSoft Enterprise applications to work seamlessly with other PeopleSoft Enterprise applications or with third-party systems or software.

**integration point**

An interface that a system uses to communicate with another PeopleSoft Enterprise application or an external application.

**integration set**

A logical grouping of integrations that applications use for the same business purpose. For example, the integration set ADVANCED_SHIPPING_ORDER contains all of the integrations that notify a customer that an order has shipped.

**item**

In PeopleSoft Enterprise Inventory, a tangible commodity that is stored in a business unit (shipped from a warehouse).

In PeopleSoft Enterprise Demand Planning, Inventory Policy Planning, and Supply Planning, a noninventory item that is designated as being used for planning purposes only. It can represent a family or group of inventory items. It can have a planning bill of material (BOM) or planning routing, and it can exist as a component on a planning BOM. A planning item cannot be specified on a production or engineering BOM or routing, and it cannot be used as a component in a production. The quantity on hand will never be maintained.

In PeopleSoft Enterprise Receivables, an individual receivable. An item can be an invoice, a credit memo, a debit memo, a write-off, or an adjustment.

**item shuffle**

In PeopleSoft Enterprise Campus Solutions, a process that enables you to change a payment allocation without having to reverse the payment.

**itinerary**

In PeopleSoft Expenses, a collection of travel reservations. Itineraries can have reservations that are selected and reserved with the travel vendor. These itineraries are not yet paid for and can be referred to as *pending reservations.* Reservations that have been paid for are referred to as *confirmed reservations.*

**joint communication**

In PeopleSoft Enterprise Campus Solutions, one letter that is addressed jointly to two people. For example, a letter might be addressed to both Mr. Sudhir Awat and Ms. Samantha Mortelli. A relationship must be established between the two individuals in the database, and at least one of the individuals must have an ID in the database.

Copyright © 1988-2007, Oracle. All rights reserved.

**keyword** In PeopleSoft Enterprise Campus Solutions, a term that you link to particular elements within PeopleSoft Enterprise Student Financials, Financial Aid, and Contributor Relations. You can use keywords as search criteria that enable you to locate specific records in a search dialog box.

**KPI** An abbreviation for *key performance indicator.* A high-level measurement of how well an organization is doing in achieving critical success factors. This defines the data value or calculation upon which an assessment is determined.

**KVI** Abbreviation for *Known Value Item.* Term used for products or groups of products where the selling price cannot be reduced or increased.

**landlord** In PeopleSoft Real Estate Management, an entity that owns real estate and leases the real estate to tenants.

**LDIF file** Abbreviation for *Lightweight Directory Access Protocol (LDAP) Data Interchange Format file.* Contains discrepancies between PeopleSoft Enterprise data and directory data.

**learner group** In PeopleSoft Enterprise Learning Management, a group of learners who are linked to the same learning environment. Members of the learner group can share the same attributes, such as the same department or job code. Learner groups are used to control access to and enrollment in learning activities and programs. They are also used to perform group enrollments and mass enrollments in the back office.

**learning components** In PeopleSoft Enterprise Learning Management, the foundational building blocks of learning activities. PeopleSoft Enterprise Learning Management supports six basic types of learning components: web-based, session, webcast, test, survey, and assignment. One or more of these learning component types compose a single learning activity.

**learning environment** In PeopleSoft Enterprise Learning Management, identifies a set of categories and catalog items that can be made available to learner groups. Also defines the default values that are assigned to the learning activities and programs that are created within a particular learning environment. Learning environments provide a way to partition the catalog so that learners see only those items that are relevant to them.

**learning history** In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's completed learning activities and programs.

**lease** In PeopleSoft Real Estate Management, a legally binding agreement between a landlord and a tenant, where the tenant rents all or part of a physical property from the landlord.

**lease abstract** In PeopleSoft Real Estate Management, a summarized version of the complete lease contract with only the important terms. The lease abstract usually fits on one page and does not include legal terminology.

**ledger mapping** You use ledger mapping to relate expense data from general ledger accounts to resource objects. Multiple ledger line items can be mapped to one or more resource IDs. You can also use ledger mapping to map dollar amounts (referred to as *rates*) to business units. You can map the amounts in two different ways: an actual amount that represents actual costs of the accounting period, or a budgeted amount that can be used to calculate the capacity rates as well as budgeted model results. In PeopleSoft Enterprise Warehouse, you can map general ledger accounts to the EW Ledger table.

**library section** In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan (or template) and that is available for other plans to share. Changes to a library section are reflected in all plans that use it.

**line** In PeopleSoft Strategic Sourcing, an individual item or service upon which there can be a bid.

 Copyright © 1988-2007, Oracle. All rights reserved.

**linked section**

In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan template but appears in a plan. Changes to linked sections propagate to plans using that section.

**linked variable**

In PeopleSoft Enterprise Incentive Management, a variable that is defined and maintained in a plan template and that also appears in a plan. Changes to linked variables propagate to plans using that variable.

**LMS**

Abbreviation for *learning management system.* In PeopleSoft Enterprise Campus Solutions, LMS is a PeopleSoft Enterprise Student Records feature that provides a common set of interoperability standards that enable the sharing of instructional content and data between learning and administrative environments.

**load**

In PeopleSoft Enterprise Inventory, identifies a group of goods that are shipped together. Load management is a feature of PeopleSoft Enterprise Inventory that is used to track the weight, the volume, and the destination of a shipment.

**local functionality**

In PeopleSoft Enterprise HRMS, the set of information that is available for a specific country. You can access this information when you click the appropriate country flag in the global window, or when you access it by a local country menu.

**location**

Locations enable you to indicate the different types of addresses—for a company, for example, one address to receive bills, another for shipping, a third for postal deliveries, and a separate street address. Each address has a different location number. The primary location—indicated by a *1*—is the address you use most often and may be different from the main address.

**logistical task**

In PeopleSoft Enterprise Services Procurement, an administrative task that is related to hiring a service provider. Logistical tasks are linked to the service type on the work order so that different types of services can have different logistical tasks. Logistical tasks include both preapproval tasks (such as assigning a new badge or ordering a new laptop) and postapproval tasks (such as scheduling orientation or setting up the service provider email). The logistical tasks can be mandatory or optional. Mandatory preapproval tasks must be completed before the work order is approved. Mandatory postapproval tasks, on the other hand, must be completed before a work order is released to a service provider.

**market template**

In PeopleSoft Enterprise Incentive Management, additional functionality that is specific to a given market or industry and is built on top of a product category.

**mass change**

In PeopleSoft Enterprise Campus Solutions, mass change is a SQL generator that can be used to create specialized functionality. Using mass change, you can set up a series of Insert, Update, or Delete SQL statements to perform business functions that are specific to the institution.

See also *3C engine.*

**match group**

In PeopleSoft Enterprise Receivables, a group of receivables items and matching offset items. The system creates match groups by using user-defined matching criteria for selected field values.

**MCF server**

Abbreviation for *PeopleSoft MultiChannel Framework server.* Comprises the universal queue server and the MCF log server. Both processes are started when *MCF Servers* is selected in an application server domain configuration.

**merchandising activity**

In PeopleSoft Enterprise Promotions Management, a specific discount type that is associated with a trade promotion (such as off-invoice, billback or rebate, or lump-sum payment) that defines the performance that is required to receive the discount. In the industry, you may know this as an offer, a discount, a merchandising event, an event, or a tactic.

Copyright © 1988-2007, Oracle. All rights reserved.

**meta-SQL** Meta-SQL constructs expand into platform-specific SQL substrings. They are used in functions that pass SQL strings, such as in SQL objects, the SQLExec function, and PeopleSoft Application Engine programs.

**metastring** Metastrings are special expressions included in SQL string literals. The metastrings, prefixed with a percent (%) symbol, are included directly in the string literals. They expand at run time into an appropriate substring for the current database platform.

**multibook** In PeopleSoft Enterprise General Ledger, multiple ledgers having multiple-base currencies that are defined for a business unit, with the option to post a single transaction to all base currencies (all ledgers) or to only one of those base currencies (ledgers).

**multicurrency** The ability to process transactions in a currency other than the business unit's base currency.

**national allowance** In PeopleSoft Enterprise Promotions Management, a promotion at the corporate level that is funded by nondiscretionary dollars. In the industry, you may know this as a national promotion, a corporate promotion, or a corporate discount.

**NDP** Abbreviation for *Non-Discountable Products.* Term used for products or groups of products where the selling price cannot be decreased.

**need** In PeopleSoft Enterprise Campus Solutions, the difference between the cost of attendance (COA) and the expected family contribution (EFC). It is the gap between the cost of attending the school and the student's resources. The financial aid package is based on the amount of financial need. The process of determining a student's need is called *need analysis.*

**node-oriented tree** A tree that is based on a detail structure, but the detail values are not used.

**Optimization Engine** A PeopleTools component that Strategic Sourcing leverages to evaluate bids and determine an ideal award allocation. The award recommendation is based on maximizing the value while adhering to purchasing and company objectives and constraints.

**pagelet** Each block of content on the home page is called a pagelet. These pagelets display summary information within a small rectangular area on the page. The pagelet provide users with a snapshot of their most relevant PeopleSoft Enterprise and non-PeopleSoft Enterprise content.

**participant** In PeopleSoft Enterprise Incentive Management, participants are recipients of the incentive compensation calculation process.

**participant object** Each participant object may be related to one or more compensation objects.

See also *compensation object.*

**partner** A company that supplies products or services that are resold or purchased by the enterprise.

**pay cycle** In PeopleSoft Enterprise Payables, a set of rules that define the criteria by which it should select scheduled payments for payment creation.

**payment shuffle** In PeopleSoft Enterprise Campus Solutions, a process allowing payments that have been previously posted to a student's account to be automatically reapplied when a higher priority payment is posted or the payment allocation definition is changed.

**pending item** In PeopleSoft Enterprise Receivables, an individual receivable (such as an invoice, a credit memo, or a write-off) that has been entered in or created by the system, but hasn't been posted.

 Copyright © 1988-2007, Oracle. All rights reserved.

**PeopleCode** PeopleCode is a proprietary language, executed by the PeopleSoft Enterprise component processor. PeopleCode generates results based on existing data or user actions. By using various tools provided with PeopleTools, external services are available to all PeopleSoft Enterprise applications wherever PeopleCode can be executed.

**PeopleCode event** See *event.*

**PeopleSoft Pure Internet Architecture** The fundamental architecture on which PeopleSoft 8 applications are constructed, consisting of a relational database management system (RDBMS), an application server, a web server, and a browser.

**performance measurement** In PeopleSoft Enterprise Incentive Management, a variable used to store data (similar to an aggregator, but without a predefined formula) within the scope of an incentive plan. Performance measures are associated with a plan calendar, territory, and participant. Performance measurements are used for quota calculation and reporting.

**period context** In PeopleSoft Enterprise Incentive Management, because a participant typically uses the same compensation plan for multiple periods, the period context associates a plan context with a specific calendar period and fiscal year. The period context references the associated plan context, thus forming a chain. Each plan context has a corresponding set of period contexts.

**person of interest** A person about whom the organization maintains information but who is not part of the workforce.

**personal portfolio** In PeopleSoft Enterprise Campus Solutions, the user-accessible menu item that contains an individual's name, address, telephone number, and other personal information.

**phase** A level 1 task, meaning that if a task had subtasks, the level 1 task would be considered the phase.

**pickup quantity** The product quantity that the customer is taking with them from the counter sales environment.

**plan** In PeopleSoft Enterprise Sales Incentive Management, a collection of allocation rules, variables, steps, sections, and incentive rules that instruct the PeopleSoft Enterprise Incentive Management engine in how to process transactions.

**plan context** In PeopleSoft Enterprise Incentive Management, correlates a participant with the compensation plan and node to which the participant is assigned, enabling the PeopleSoft Enterprise Incentive Management system to find anything that is associated with the node and that is required to perform compensation processing. Each participant, node, and plan combination represents a unique plan context—if three participants are on a compensation structure, each has a different plan context. Configuration plans are identified by plan contexts and are associated with the participants that refer to them.

**plan template** In PeopleSoft Enterprise Incentive Management, the base from which a plan is created. A plan template contains common sections and variables that are inherited by all plans that are created from the template. A template may contain steps and sections that are not visible in the plan definition.

**planned learning** In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's planned learning activities and programs.

**planning instance** In PeopleSoft Enterprise Supply Planning, a set of data (business units, items, supplies, and demands) constituting the inputs and outputs of a supply plan.

**population** In PeopleSoft Enterprise Campus Solutions, the middle level of the three-level classification structure that you define in PeopleSoft Enterprise Recruiting and

Copyright © 1988-2007, Oracle. All rights reserved. 

Admissions for enrollment management. You can define a population level, link it to other levels, and set enrollment target numbers for it.

See also *division* and *cohort.*

**portal registry** In PeopleSoft Enterprise applications, the portal registry is a tree-like structure in which content references are organized, classified, and registered. It is a central repository that defines both the structure and content of a portal through a hierarchical, tree-like structure of folders useful for organizing and securing content references.

**predecessor task** A task that you must complete before you start another task.

**price breaks** In PeopleSoft Strategic Sourcing, a price discount or surcharge that a bidder may apply based on the quantity awarded.

**price components** In PeopleSoft Strategic Sourcing, the various components, such as material costs, labor costs, shipping costs, and so on that make up the overall bid price.

**price list** Enables you to select products and conditions for which the price list applies to a transaction. During a transaction, the system either determines the product price based on the predefined search hierarchy for the transaction or uses the product's lowest price on any associated, active price lists. This price is used as the basis for any further discounts and surcharges.

**price rule** The conditions that must be met for adjustments to be applied to the base price. Multiple rules can apply when conditions of each rule are met.

**price rule conditions** Conditions that select the price-by fields, the values for the price-by fields, and the operator that determines how the price-by fields relate to the transaction.

**price rule key** The fields that are available to define price rule conditions (which are used to match a transaction) on the price rule.

**primacy number** In PeopleSoft Enterprise Campus Solutions, a number that the system uses to prioritize financial aid applications when students are enrolled in multiple academic careers and academic programs at the same time. The Consolidate Academic Statistics process uses the primacy number indicated for both the career and program at the institutional level to determine a student's primary career and program. The system also uses the number to determine the primary student attribute value that is used when you extract data to report on cohorts. The lowest number takes precedence.

**primary name type** In PeopleSoft Enterprise Campus Solutions, the name type that is used to link the name stored at the highest level within the system to the lower-level set of names that an individual provides.

**process category** In PeopleSoft Process Scheduler, processes that are grouped for server load balancing and prioritization.

**process group** In PeopleSoft Enterprise Financials, a group of application processes (performed in a defined order) that users can initiate in real time, directly from a transaction entry page.

**process definition** Process definitions define each run request.

**process instance** A unique number that identifies each process request. This value is automatically incremented and assigned to each requested process when the process is submitted to run.

**process job** You can link process definitions into a job request and process each request serially or in parallel. You can also initiate subsequent processes based on the return code from each prior request.

 Copyright © 1988-2007, Oracle. All rights reserved.

**process request** A single run request, such as a Structured Query Report (SQR), a COBOL or Application Engine program, or a Crystal report that you run through PeopleSoft Process Scheduler.

**process run control** A PeopleTools variable used to retain PeopleSoft Process Scheduler values needed at runtime for all requests that reference a run control ID. Do not confuse these with application run controls, which may be defined with the same run control ID, but only contain information specific to a given application process request.

**product** A PeopleSoft Enterprise or third-party product. PeopleSoft organizes its software products into product families and product lines. Interactive Services Repository contains information about every release of every product that PeopleSoft sells, as well as products from certified third-party companies. These products appear with the product name and release number.

**product adds** The pricing functionality where buying product A gets product B for free or at a price (formerly referred to as *giveaways*).

**product bidding** In PeopleSoft Strategic Sourcing, the placing of a bid on behalf of the bidder, up or down to the bidder's specified amount, so that the bidder can be the leading bidder.

**product category** In PeopleSoft Enterprise Incentive Management, indicates an application in the PeopleSoft Enterprise Incentive Management suite of products. Each transaction in the PeopleSoft Enterprise Incentive Management system is associated with a product category.

**product family** A group of products that are related by common functionality. The family names that can be searched using Interactive Service Repository are Oracle's PeopleSoft Enterprise, PeopleSoft EnterpriseOne, PeopleSoft World, and third-party, certified partners.

**product line** The name of a PeopleSoft Enterprise product line or the company name of a third-party certified partner. Integration Services Repository enables you to search for integration points by product line.

**programs** In PeopleSoft Enterprise Learning Management, a high-level grouping that guides the learner along a specific learning path through sections of catalog items. PeopleSoft Enterprise Learning Systems provides two types of programs—curricula and certifications.

**progress log** In PeopleSoft Enterprise Services Procurement, tracks deliverable-based projects. This is similar to the time sheet in function and process. The service provider contact uses the progress log to record and submit progress on deliverables. The progress can be logged by the activity that is performed, by the percentage of work that is completed, or by the completion of milestone activities that are defined for the project.

**project transaction** In PeopleSoft Enterprise Project Costing, an individual transaction line that represents a cost, time, budget, or other transaction row.

**promotion** In PeopleSoft Enterprise Promotions Management, a trade promotion, which is typically funded from trade dollars and used by consumer products manufacturers to increase sales volume.

**prospects** In PeopleSoft Enterprise Campus Solutions, students who are interested in applying to the institution.

In PeopleSoft Enterprise Contributor Relations, individuals and organizations that are most likely to make substantial financial commitments or other types of commitments to the institution.

**proxy bidding** In PeopleSoft Strategic Sourcing, the placing of a bid on behalf of the bidder, up or down to the bidder's specified amount, so that the bidder can be the leading bidder.

Copyright © 1988-2007, Oracle. All rights reserved.

**publishing** In PeopleSoft Enterprise Incentive Management, a stage in processing that makes incentive-related results available to participants.

**rating components** In PeopleSoft Enterprise Campus Solutions, variables used with the Equation Editor to retrieve specified populations.

**record group** A set of logically and functionally related control tables and views. Record groups help enable TableSet sharing, which eliminates redundant data entry. Record groups ensure that TableSet sharing is applied consistently across all related tables and views.

**record input VAT flag** Abbreviation for *record input value-added tax flag.* Within PeopleSoft Enterprise Purchasing, Payables, and General Ledger, this flag indicates that you are recording input VAT on the transaction. This flag, in conjunction with the record output VAT flag, is used to determine the accounting entries created for a transaction and to determine how a transaction is reported on the VAT return. For all cases within Purchasing and Payables where VAT information is tracked on a transaction, this flag is set to Yes. This flag is not used in PeopleSoft Enterprise Order Management, Billing, or Receivables, where it is assumed that you are always recording only output VAT, or in PeopleSoft Enterprise Expenses, where it is assumed that you are always recording only input VAT.

**record output VAT flag** Abbreviation for *record output value-added tax flag.*

See *record input VAT flag.*

**recname** The name of a record that is used to determine the associated field to match a value or set of values.

**recognition** In PeopleSoft Enterprise Campus Solutions, the recognition type indicates whether the PeopleSoft Enterprise Contributor Relations donor is the primary donor of a commitment or shares the credit for a donation. Primary donors receive hard credit that must total 100 percent. Donors that share the credit are given soft credit. Institutions can also define other share recognition-type values such as memo credit or vehicle credit.

**reference data** In PeopleSoft Enterprise Sales Incentive Management, system objects that represent the sales organization, such as territories, participants, products, customers, and channels.

**reference object** In PeopleSoft Enterprise Incentive Management, this dimension-type object further defines the business. Reference objects can have their own hierarchy (for example, product tree, customer tree, industry tree, and geography tree).

**reference transaction** In commitment control, a reference transaction is a source transaction that is referenced by a higher-level (and usually later) source transaction, in order to automatically reverse all or part of the referenced transaction's budget-checked amount. This avoids duplicate postings during the sequential entry of the transaction at different commitment levels. For example, the amount of an encumbrance transaction (such as a purchase order) will, when checked and recorded against a budget, cause the system to concurrently reference and relieve all or part of the amount of a corresponding pre-encumbrance transaction, such as a purchase requisition.

**regional sourcing** In PeopleSoft Enterprise Purchasing, provides the infrastructure to maintain, display, and select an appropriate vendor and vendor pricing structure that is based on a regional sourcing model where the multiple ship to locations are grouped. Sourcing may occur at a level higher than the ship to location.

**relationship object** In PeopleSoft Enterprise Incentive Management, these objects further define a compensation structure to resolve transactions by establishing associations between compensation objects and business objects.

**remote data source data** Data that is extracted from a separate database and migrated into the local database.

 Copyright © 1988-2007, Oracle. All rights reserved.

**REN server** Abbreviation for *real-time event notification server* in PeopleSoft MultiChannel Framework.

**requester** In PeopleSoft Enterprise eSettlements, an individual who requests goods or services and whose ID appears on the various procurement pages that reference purchase orders.

**reservations** In PeopleSoft Expenses, travel reservations that have been placed with the travel vendor.

**reversal indicator** In PeopleSoft Enterprise Campus Solutions, an indicator that denotes when a particular payment has been reversed, usually because of insufficient funds.

**RFI event** In PeopleSoft Strategic Sourcing, a request for information.

**RFx event** In PeopleSoft Strategic Sourcing, a request for proposal or request for a quote event when bidders submit their overall best bids and during which bidders do not actively compete against one another.

**role** Describes how people fit into PeopleSoft Workflow. A role is a class of users who perform the same type of work, such as clerks or managers. Your business rules typically specify what user role needs to do an activity.

**role user** A PeopleSoft Workflow user. A person's role user ID serves much the same purpose as a user ID does in other parts of the system. PeopleSoft Workflow uses role user IDs to determine how to route worklist items to users (through an email address, for example) and to track the roles that users play in the workflow. Role users do not need PeopleSoft user IDs.

**roll up** In a tree, to roll up is to total sums based on the information hierarchy.

**run control** A run control is a type of online page that is used to begin a process, such as the batch processing of a payroll run. Run control pages generally start a program that manipulates data.

**run control ID** A unique ID to associate each user with his or her own run control table entries.

**run-level context** In PeopleSoft Enterprise Incentive Management, associates a particular run (and batch ID) with a period context and plan context. Every plan context that participates in a run has a separate run-level context. Because a run cannot span periods, only one run-level context is associated with each plan context.

**saved bid** In PeopleSoft Strategic Sourcing, a bid that has been created but not submitted. Only submitted bids are eligible for award.

**score** In PeopleSoft Strategic Sourcing, the numerical sum of answers (percentages) to bid factors on an event. Scores appear only to bidders on auction events.

**SCP SCBM XML message** Abbreviation for *Supply Chain Planning Supply Chain Business Modeler Extensible Markup Language message.* Supply Chain Business Modeler uses XML as the format for all data that it imports and exports.

**search query** You use this set of objects to pass a query string and operators to the search engine. The search index returns a set of matching results with keys to the source documents.

**search/match** In PeopleSoft Enterprise Campus Solutions and PeopleSoft Enterprise Human Resources Management Solutions, a feature that enables you to search for and identify duplicate records in the database.

**seasonal address** In PeopleSoft Enterprise Campus Solutions, an address that recurs for the same length of time at the same time of year each year until adjusted or deleted.

Copyright © 1988-2007, Oracle. All rights reserved.

**section** In PeopleSoft Enterprise Incentive Management, a collection of incentive rules that operate on transactions of a specific type. Sections enable plans to be segmented to process logical events in different sections.

**security event** In commitment control, security events trigger security authorization checking, such as budget entries, transfers, and adjustments; exception overrides and notifications; and inquiries.

**sell event** In PeopleSoft Strategic Sourcing, for event creators, the sale of goods or services most typically associated with forward auctions. For bidders, the purchase of goods or services.

**serial genealogy** In PeopleSoft Enterprise Manufacturing, the ability to track the composition of a specific, serial-controlled item.

**serial in production** In PeopleSoft Enterprise Manufacturing, enables the tracing of serial information for manufactured items. This is maintained in the Item Master record.

**service impact** In PeopleSoft Enterprise Campus Solutions, the resulting action triggered by a service indicator. For example, a service indicator that reflects nonpayment of account balances by a student might result in a service impact that prohibits registration for classes.

**service indicator** In PeopleSoft Enterprise Campus Solutions, indicates services that may be either withheld or provided to an individual. Negative service indicators indicate holds that prevent the individual from receiving specified services, such as check-cashing privileges or registration for classes. Positive service indicators designate special services that are provided to the individual, such as front-of-line service or special services for disabled students.

**session** In PeopleSoft Enterprise Campus Solutions, time elements that subdivide a term into multiple time periods during which classes are offered. In PeopleSoft Enterprise Contributor Relations, a session is the means of validating gift, pledge, membership, or adjustment data entry . It controls access to the data entered by a specific user ID. Sessions are balanced, queued, and then posted to the institution's financial system. Sessions must be posted to enter a matching gift or pledge payment, to make an adjustment, or to process giving clubs or acknowledgements.

In PeopleSoft Enterprise Learning Management, a single meeting day of an activity (that is, the period of time between start and finish times within a day). The session stores the specific date, location, meeting time, and instructor. Sessions are used for scheduled training.

**session template** In PeopleSoft Enterprise Learning Management, enables you to set up common activity characteristics that may be reused while scheduling a PeopleSoft Enterprise Learning Management activity—characteristics such as days of the week, start and end times, facility and room assignments, instructors, and equipment. A session pattern template can be attached to an activity that is being scheduled. Attaching a template to an activity causes all of the default template information to populate the activity session pattern.

**setup relationship** In PeopleSoft Enterprise Incentive Management, a relationship object type that associates a configuration plan with any structure node.

**share driver expression** In PeopleSoft Enterprise Business Planning, a named planning method similar to a driver expression, but which you can set up globally for shared use within a single planning application or to be shared between multiple planning applications through PeopleSoft Enterprise Warehouse.

**short-term customer** A customer not in the system who is entered during sales order entry using a template.

 Copyright © 1988-2007, Oracle. All rights reserved.

**single signon** With single signon, users can, after being authenticated by a PeopleSoft Enterprise application server, access a second PeopleSoft Enterprise application server without entering a user ID or password.

**source key process** In PeopleSoft Enterprise Campus Solutions, a process that relates a particular transaction to the source of the charge or financial aid. On selected pages, you can drill down into particular charges.

**source transaction** In commitment control, any transaction generated in a PeopleSoft Enterprise or third-party application that is integrated with commitment control and which can be checked against commitment control budgets. For example, a pre-encumbrance, encumbrance, expenditure, recognized revenue, or collected revenue transaction.

**sourcing objective** For constraints, the option to designate whether a business rule is required (mandatory) or is only recommended (target).

**speed key** See *communication key.*

**SpeedChart** A user-defined shorthand key that designates several ChartKeys to be used for voucher entry. Percentages can optionally be related to each ChartKey in a SpeedChart definition.

**SpeedType** A code representing a combination of ChartField values. SpeedTypes simplify the entry of ChartFields commonly used together.

**staging** A method of consolidating selected partner offerings with the offerings from the enterprise's other partners.

**standard letter code** In PeopleSoft Enterprise Campus Solutions, a standard letter code used to identify each letter template available for use in mail merge functions. Every letter generated in the system must have a standard letter code identification.

**statutory account** Account required by a regulatory authority for recording and reporting financial results. In PeopleSoft Enterprise, this is equivalent to the Alternate Account (ALTACCT) ChartField.

**step** In PeopleSoft Enterprise Sales Incentive Management, a collection of sections in a plan. Each step corresponds to a step in the job run.

**storage level** In PeopleSoft Enterprise Inventory, identifies the level of a material storage location. Material storage locations are made up of a business unit, a storage area, and a storage level. You can set up to four storage levels.

**subcustomer qualifier** A value that groups customers into a division for which you can generate detailed history, aging, events, and profiles.

**Summary ChartField** You use summary ChartFields to create summary ledgers that roll up detail amounts based on specific detail values or on selected tree nodes. When detail values are summarized using tree nodes, summary ChartFields must be used in the summary ledger data record to accommodate the maximum length of a node name (20 characters).

**summary ledger** An accounting feature used primarily in allocations, inquiries, and PS/nVision reporting to store combined account balances from detail ledgers. Summary ledgers increase speed and efficiency of reporting by eliminating the need to summarize detail ledger balances each time a report is requested. Instead, detail balances are summarized in a background process according to user-specified criteria and stored on summary ledgers. The summary ledgers are then accessed directly for reporting.

**summary time period** In PeopleSoft Enterprise Business Planning, any time period (other than a base time period) that is an aggregate of other time periods, including other summary time periods and base time periods, such as quarter and year total.

Copyright © 1988-2007, Oracle. All rights reserved.

**summary tree** A tree used to roll up accounts for each type of report in summary ledgers. Summary trees enable you to define trees on trees. In a summary tree, the detail values are really nodes on a detail tree or another summary tree (known as the *basis* tree). A summary tree structure specifies the details on which the summary trees are to be built.

**syndicate** To distribute a production version of the enterprise catalog to partners.

**system function** In PeopleSoft Enterprise Receivables, an activity that defines how the system generates accounting entries for the general ledger.

**system source** The system source identifies the source of a transaction row in the database. For example, a transaction that originates in PeopleSoft Enterprise Expenses contains a system source code of BEX (Expenses Batch).

When PeopleSoft Enterprise Project Costing prices the source transaction row for billing, the system creates a new row with a system source code of PRP (Project Costing pricing), which represents the system source of the new row. System source codes can identify sources that are internal or external to the PeopleSoft Enterprise system. For example, processes that import data from Microsoft Project into PeopleSoft Enterprise applications create transaction rows with a source code of MSP (Microsoft Project).

**TableSet** A means of sharing similar sets of values in control tables, where the actual data values are different but the structure of the tables is the same.

**TableSet sharing** Shared data that is stored in many tables that are based on the same TableSets. Tables that use TableSet sharing contain the SETID field as an additional key or unique identifier.

**target currency** The value of the entry currency or currencies converted to a single currency for budget viewing and inquiry purposes.

**task** A deliverable item on the detailed sourcing plan.

**tax authority** In PeopleSoft Enterprise Campus Solutions, a user-defined element that combines a description and percentage of a tax with an account type, an item type, and a service impact.

**template** A template is HTML code associated with a web page. It defines the layout of the page and also where to get HTML for each part of the page. In PeopleSoft Enterprise, you use templates to build a page by combining HTML from a number of sources. For a PeopleSoft Enterprise portal, all templates must be registered in the portal registry, and each content reference must be assigned a template.

**tenant** In PeopleSoft Real Estate Management, an entity that leases real estate from a landlord.

**territory** In PeopleSoft Enterprise Sales Incentive Management, hierarchical relationships of business objects, including regions, products, customers, industries, and participants.

**third party** A company or vendor that has extensive PeopleSoft Enterprise product knowledge and whose products and integrations have been certified and are compatible with PeopleSoft Enterprise applications.

**tiered pricing** Enables different portions of a schedule to be priced differently from one another.

**time span** A relative period, such as year-to-date or current period, that various PeopleSoft General Ledger functions and reports can use when a rolling time frame, rather than a specific date, is required.

**total cost** In PeopleSoft Strategic Sourcing, the estimated dollar cost (sum of real price dollars and potential “soft” or non-price dollars) of a particular award approach.

 Copyright © 1988-2007, Oracle. All rights reserved.

**travel group**
In PeopleSoft Expenses, the organization's travel rules and polices that are associated with specific business units, departments, or employees. You must define at least one travel group when setting up the PeopleSoft Expenses travel feature. You must define and associate at least one travel group with a travel vendor.

**travel partner**
In PeopleSoft Expenses, the travel vendor with which the organization has a contractual relationship.

**3C engine**
Abbreviation for *Communications, Checklists, and Comments engine.* In PeopleSoft Enterprise Campus Solutions, the 3C engine enables you to automate business processes that involve additions, deletions, and updates to communications, checklists, and comments. You define events and triggers to engage the engine, which runs the mass change and processes the 3C records (for individuals or organizations) immediately and automatically from within business processes.

**3C group**
Abbreviation for *Communications, Checklists, and Comments group.* In PeopleSoft Enterprise Campus Solutions, a method of assigning or restricting access privileges. A 3C group enables you to group specific communication categories, checklist codes, and comment categories. You can then assign the group inquiry-only access or update access, as appropriate.

**trace usage**
In PeopleSoft Enterprise Manufacturing, enables the control of which components will be traced during the manufacturing process. Serial- and lot-controlled components can be traced. This is maintained in the Item Master record.

**transaction allocation**
In PeopleSoft Enterprise Incentive Management, the process of identifying the owner of a transaction. When a raw transaction from a batch is allocated to a plan context, the transaction is duplicated in the PeopleSoft Enterprise Incentive Management transaction tables.

**transaction state**
In PeopleSoft Enterprise Incentive Management, a value assigned by an incentive rule to a transaction. Transaction states enable sections to process only transactions that are at a specific stage in system processing. After being successfully processed, transactions may be promoted to the next transaction state and "picked up" by a different section for further processing.

**Translate table**
A system edit table that stores codes and translate values for the miscellaneous fields in the database that do not warrant individual edit tables of their own.

**tree**
The graphical hierarchy in PeopleSoft Enterprise systems that displays the relationship between all accounting units (for example, corporate divisions, projects, reporting groups, account numbers) and determines roll-up hierarchies.

**tuition lock**
In PeopleSoft Enterprise Campus Solutions, a feature in the Tuition Calculation process that enables you to specify a point in a term after which students are charged a minimum (or *locked*) fee amount. Students are charged the locked fee amount even if they later drop classes and take less than the normal load level for that tuition charge.

**unclaimed transaction**
In PeopleSoft Enterprise Incentive Management, a transaction that is not claimed by a node or participant after the allocation process has completed, usually due to missing or incomplete data. Unclaimed transactions may be manually assigned to the appropriate node or participant by a compensation administrator.

**universal navigation header**
Every PeopleSoft Enterprise portal includes the universal navigation header, intended to appear at the top of every page as long as the user is signed on to the portal. In addition to providing access to the standard navigation buttons (like Home, Favorites, and signoff) the universal navigation header can also display a welcome message for each user.

**update access**
In PeopleSoft Enterprise Campus Solutions, a type of security access that permits the user to edit and update data.

Copyright © 1988-2007, Oracle. All rights reserved.

See also *inquiry access.*

**user interaction object** In PeopleSoft Enterprise Sales Incentive Management, used to define the reporting components and reports that a participant can access in his or her context. All PeopleSoft Enterprise Sales Incentive Management user interface objects and reports are registered as user interaction objects. User interaction objects can be linked to a compensation structure node through a compensation relationship object (individually or as groups).

**variable** In PeopleSoft Enterprise Sales Incentive Management, the intermediate results of calculations. Variables hold the calculation results and are then inputs to other calculations. Variables can be plan variables that persist beyond the run of an engine or local variables that exist only during the processing of a section.

**VAT exception** Abbreviation for *value-added tax exception.* A temporary or permanent exemption from paying VAT that is granted to an organization. This terms refers to both VAT exoneration and VAT suspension.

**VAT exempt** Abbreviation for *value-added tax exempt.* Describes goods and services that are not subject to VAT. Organizations that supply exempt goods or services are unable to recover the related input VAT. This is also referred to as exempt without recovery.

**VAT exoneration** Abbreviation for *value-added tax exoneration.* An organization that has been granted a permanent exemption from paying VAT due to the nature of that organization.

**VAT suspension** Abbreviation for *value-added tax suspension.* An organization that has been granted a temporary exemption from paying VAT.

**warehouse** A PeopleSoft Enterprise data warehouse that consists of predefined ETL maps, data warehouse tools, and DataMart definitions.

**weight or weighting** In PeopleSoft Strategic Sourcing, how important the line or question is to the overall event. Weighting is used to score and analyze bids. For RFx and RFI events, weightings may or may not appear to bidders.

**work order** In PeopleSoft Enterprise Services Procurement, enables an enterprise to create resource-based and deliverable-based transactions that specify the basic terms and conditions for hiring a specific service provider. When a service provider is hired, the service provider logs time or progress against the work order.

**worker** A person who is part of the workforce; an employee or a contingent worker.

**workset** A group of people and organizations that are linked together as a set. You can use worksets to simultaneously retrieve the data for a group of people and organizations and work with the information on a single page.

**worksheet** A way of presenting data through a PeopleSoft Enterprise Business Analysis Modeler interface that enables users to do in-depth analysis using pivoting tables, charts, notes, and history information.

**worklist** The automated to-do list that PeopleSoft Workflow creates. From the worklist, you can directly access the pages you need to perform the next action, and then return to the worklist for another item.

**XML link** The XML Linking language enables you to insert elements into XML documents to create a links between resources.

**XML schema** An XML definition that standardizes the representation of application messages, component interfaces, or business interlinks.

**XPI** Abbreviation for *eXtended Process Integrator.* PeopleSoft XPI is the integration infrastructure that enables both real-time and batch communication with JD Edwards EnterpriseOne applications.

 Copyright © 1988-2007, Oracle. All rights reserved.

**yield by operation** In PeopleSoft Enterprise Manufacturing, the ability to plan the loss of a manufactured item on an operation-by-operation basis.

**zero-rated VAT** Abbreviation for *zero-rated value-added tax.* A VAT transaction with a VAT code that has a tax percent of zero. Used to track taxable VAT activity where no actual VAT amount is charged. Organizations that supply zero-rated goods and services can still recover the related input VAT. This is also referred to as exempt with recovery.

Copyright © 1988-2007, Oracle. All rights reserved.

Copyright © 1988-2007, Oracle. All rights reserved.

# Index

## Numerics/Symbols


# character 16
$ character 16
% character 16
& character 15
@ character 16
’ character, in text literals 20
: character, in variable names 16
_ character, in variable names 16
3D-EFFECTS argument, DECLARE-CHART command 69


## A


-A flag 6, 255, 264
absolute value (abs) function 135
acos function 135
ADD command 22
  example 23
additional documentation xii
-AF command-line parameter 306
AFTER argument, BEGIN-EXECUTE command 39
AFTER-BOLD argument
  DECLARE PRINTER command 277
  DECLARE-PRINTER command 90
  PRINT command 184
AFTER-PAGE argument
  DECLARE PROCEDURE command 278
  DECLARE-PROCEDURE command 95
  DECLARE-TOC command 98
  USE-PROCEDURE command 229
AFTER-REPORT argument
  DECLARE PROCEDURE command 278
  DECLARE-PROCEDURE command 95
  USE-PROCEDURE command 229
AFTER-TOC argument, DECLARE-TOC command 98
aggregate functions 15
ALTER-COLOR-MAP command 23
  example 24
ALTER-LOCALE command 25, 269, 278, 280, 296
  example 28
ALTER-PRINTER command 32, 85, 93, 269, 279
  example 33
ALTER-REPORT command 33
  example 34
any_col abbreviation 21
any_lit abbreviation 21
any_var abbreviation 21
-AP command-line parameter 306
APPEND argument, SECURITY command 214
applets, HTML 7
application fundamentals xi
arccosine (acos) function 135
arcsine (asin) function 135
arctangent (atan) function 136
args... command-line argument 5
arguments
  separating from commands 4
  using with procedures 107
array field operands 133
array function 145
ARRAY-ADD command 34
  example 35, 36
ARRAY-DIVIDE command 34
  example 36
ARRAY-MULTIPLY command 34
  example 35
ARRAY-SUBTRACT command 34
  example 35
arrays
  arithmetic 35
  ARRAY-ADD command 34
  ARRAY-DIVIDE command 34
  ARRAY-MULTIPLY command 34
  ARRAY-SUBTRACT command 34
  CLEAR-ARRAY command 57
  CREATE-ARRAY command 62
  elements 63
  maximums 62
  moving data into 209
  resetting fields 58


Copyright © 1988-2007, Oracle. All rights reserved.

retrieving data from 118
searching 167
ascii function 145
asciic function 146
asin function 135
ASK command 5, 49, 101, 127, 128
example 37
preventing user input 14
understanding 36
AT-END argument, NEXT-COLUMN command 175
atan function 136

**B**

bar codes
printing 200
types 200
UPC-A 201
BATCH-MODE argument, INPUT command 130, 131
BEEP argument, SHOW command 219
BEFORE argument, BEGIN-EXECUTE command 39
BEFORE-BOLD argument
DECLARE PRINTER command 277
DECLARE-PRINTER command 90
PRINT command 184
BEFORE-PAGE argument
DECLARE PROCEDURE command 278
DECLARE-PROCEDURE command 95
DECLARE-TOC command 98
USE-PROCEDURE command 229
BEFORE-REPORT argument
DECLARE PROCEDURE command 278
DECLARE-PROCEDURE command 95
USE-PROCEDURE command 229
BEFORE-TOC argument, DECLARE-TOC command 98
BEGIN TRANSACTION command, Sybase and Microsoft SQL Server 212
BEGIN-DOCUMENT command 37, 109
example 38
BEGIN-DOCUMENT paragraph
marker locations 16
BEGIN-EXECUTE command 38
example 40
BEGIN-FOOTING command 41, 109
example 42, 109
BEGIN-HEADING command 42, 109
example 43
BEGIN-PROCEDURE command 43, 111
example 45
BEGIN-PROGRAM command 46, 111, 269, 270
example 46
BEGIN-REPORT command 269, 270
BEGIN-SELECT command 39, 46, 111
example 48
BEGIN-SETUP command 2, 49, 111
example 50
BEGIN-SQL command 49, 50, 111
example 52
bind variables 48
blanks, trailing 11, 13, 15
BLINK argument, SHOW command 219
-Bnn argument, BEGIN-SELECT command 48
-Bnn flag 6
BOLD argument
PRINT command 184
SHOW command 219
BORDER argument, DECLARE-CHART command 69
BOTTOM-MARGIN argument, DECLARE-LAYOUT command 81
BOX argument
GRAPHIC command 123
PRINT command 184
boxes, drawing 122
braces, use of in syntax 3
branching, EVALUATE command 111
BREAK argument, EVALUATE command 112
BREAK command 53
browser, determining 8
buffer 6
-BURST:{xx} flag 6, 256, 264
buttons, HTML 299

**C**

-C flag 6, 255
C, writing functions in 161
-CA command-line parameter 305
CALL command 53
example 55
CALL SYSTEM command 53



Copyright © 1988-2007, Oracle. All rights reserved.

Cancel dialog box 6
-CAP command-line parameter 305
CAPTION argument, PRINT-BAR-CODE command 201
case/switch 111
-CB flag 6, 256
-CD command-line parameter 305
ceil function 136
CENTER argument, PRINT command 184
CHAR, TEXT data type 129
CHAR-SIZE argument, DECLARE PRINTER command 274
CHAR-WIDTH argument
  DECLARE-LAYOUT command 78, 82
character columns 15
character sequences, finding 117
character sets, specifying in Windows 299
characters
  control 10, 283, 284
  converting single-byte to multibyte 213
  display 108
  multibyte 213
  nondisplay 108, 283, 284
  printer control 11
  single-byte 213
CHARS-INCH argument, DECLARE PRINTER command 274
CHART.FILL option 67
CHART-SIZE argument, DECLARE-CHART command 68
ChartAreaBackground 68
ChartAreaForeground 68
ChartBackground 68
ChartForeground 68
charts 202
  business 68
  colors 68
  defining attributes 67
  printing 203
  types 204
CHECKSUM argument, PRINT-BAR-CODE command 201
chr function 146
CLEAR-ARRAY command 57, 63
  example 58
CLEAR-LINE argument, SHOW command 219
CLEAR-SCREEN argument, SHOW command 219
CLOSE command 58, 177
  example 58
-Cnn argument
  BEGIN-SELECT command 48
  BEGIN-SQL command 52
-CO command-line parameter 305
CODE-PRINTER argument, PRINT command 184
COLOR argument, DECLARE-PRINTER command 87
color palettes 67
COLOR_1 argument, CREATE-COLOR-PALETTE command 64
COLOR_2 argument, CREATE-COLOR-PALETTE command 64
COLOR_n argument, CREATE-COLOR-PALETTE command 64
COLOR-PALETTE argument, DECLARE-CHART command 69
colors
  altering defined 23
  aqua 24
  black 24
  blue 24
  defaults 24, 75, 120, 215
  defining in SQR reports 75
  defining new 23
  fuchsia 24
  gray 24
  green 24
  lime 24
  maroon 24
  naming 23, 120
  navy 24
  olive 24
  palettes 64, 67
  purple 24
  red 24
  references 64
  retrieving 119
  RGB values 23
  silver 24
  teal 24
  white 24
  yellow 24
Colors section, pssqr.ini 304
columns 15
  character 15

Copyright © 1988-2007, Oracle. All rights reserved.

date 15
determining value 111
displaying 103
dynamic 15
global 16
local 16
not supported in DECLARE-CHART or PRINT-CHART command 67
number 15
reserved variable for current 16
reserved variable for maximum 18
setting current 227
COLUMNS argument, POSITION command 179
COLUMNS command 59, 174, 227
COMMAND argument, BEGIN-EXECUTE command 39
command line, SQR 4
command-line arguments 4
args... 5
connectivity 5
@file... 5
flags 5
program 4
command-line flags 5
SQR Print 264
commands
abbreviations in 21
avoiding older 269
breaking 4
case insensitivity 4
hyphens in 4
length of 16
listed alphabetically 21
overview 21
rules for entering 4
separating from arguments 4
comments 4
comments, submitting xvi
COMMIT command
example 60
suppressing 14
understanding 59
COMMIT TRANSACTION command 60
common elements xvi
comparison operators 125, 132
compiler directives 108, 110, 127, 128
nesting 124
table of 124
CONCAT command
edit masks 192
example 61
understanding 60
concatenation, with STRING command 222
cond function 146
conditional processing 111, 126, 230
Configuration Manager 305
CONNECT command 59, 61
example 62
CONNECTION argument, BEGIN-EXECUTE command 39
connection_name argument 75
connectivity command-line argument 5
constants
specifying 101
understanding 20
contact information xvi
control characters 10, 283, 284
copyright banner 10
cos function 136
cosh function 137
cosine (cos) function 136
-COWN command-line parameter 305
CREATE-ARRAY command 35, 49, 62, 209
example 63
CREATE-COLOR-PALETTE command 64
example 64
cross-references xv
-CS command-line parameter 305
-CT command-line parameter 305
currency symbols
redefining 280
specifying 27, 278
#current-column reserved variable 16
$current-date reserved variable 16
#current-line reserved variable 17, 173
cursors
controlling 217
database 61
displaying 13
Customer Connection website xii


## D


data
elements 15
input 129



Copyright © 1988-2007, Oracle. All rights reserved.

retrieving from array 118
types supported by INPUT command 129
data elements
columns 15
defining 15
literals 20
variables 16
DATA-ARRAY argument, PRINT-CHART command 202, 206
DATA-ARRAY-COLUMN-COUNT argument, PRINT-CHART command 202, 206
DATA-ARRAY-COLUMN-LABELS argument, PRINT-CHART command 202, 206
DATA-ARRAY-ROW-COUNT argument, PRINT-CHART command 202, 206
DATA-LABELS argument, DECLARE-CHART command 69
database
commit 59
cursors 61
joins 165
preventing signin 14
suppressing commit 14
types 18
username 19
using designated 226
database functions 15
datasource, logon parameters 76
date
columns 15
default formats 27, 193
displaying 103
edit format characters 189
edit masks 27
reserved variable for current 16
variables 16
DATE argument
DECLARE-VARIABLE command 100
DISPLAY command 104
MOVE command 169
PRINT command 185
SHOW command 219
DATE column type 185
DATE data type 129
date functions
dateadd 147
datediff 147
datenow 147
datetostr 148
strtodate 156
date_col abbreviation 21
DATE-EDIT-MASK keyword, ALTER-LOCALE command 26, 27
DATE-SEPARATOR keyword, ALTER-LOCALE command 26, 28
DATE-TIME command 269, 270
date_var abbreviation 21
dateadd function 147
datediff function 147
datenow function 147, 269, 270
DATETIME column type 185
datetostr function 148
DAY-OF-WEEK-CASE keyword, ALTER-LOCALE command 26, 28
DAY-OF-WEEK-FULL keyword, ALTER-LOCALE command 27, 28
DAY-OF-WEEK-SHORT keyword, ALTER-LOCALE command 27, 28
DB2 212
COMMIT command 59
ROLLBACK command 212
-DBconnectionstring argument, BEGIN-SELECT command 48, 52
-DBdatabase flag 7, 256
#DEBUG command 65
example 65
-DEBUG flag 7, 65, 124, 127, 128
-DEBUGLP ON command-line parameter 307
DECIMAL argument, DECLARE-VARIABLE command 100
decimal variables 16, 22
DECIMAL-SEPARATOR keyword, ALTER-LOCALE command 26, 28
DECLARE LAYOUT command 272
DECLARE PRINTER command 269, 272
DECLARE PROCEDURE command 269, 277
DECLARE-CHART command 49, 66, 109, 202
example 73
DECLARE-COLOR-MAP command 74


Copyright © 1988-2007, Oracle. All rights reserved.

example 75
DECLARE-CONNECTION command 75
example 76
DECLARE-IMAGE command 49, 77, 109
example 77
DECLARE-LAYOUT command 49, 78, 109, 173, 269
arguments of 80
example 82
FORMFEED argument 281
MAX-COLUMNS argument 282
MAX-LINES argument 282
PAPER-SIZE argument 282
DECLARE-PRINTER command 49, 85, 109, 184, 269, 270, 272, 279
example 94
FILL specification 67
INIT-STRING argument 284
RESET-STRING argument 283
DECLARE-PROCEDURE command 50, 94, 109, 207, 228, 269, 277
example 95
DECLARE-REPORT command 50, 78, 96, 109
example 96
DECLARE-TOC command 50, 97
example 98
DECLARE-VARIABLE command 50, 99, 109
example 100
default colors 215
DEFAULT layout 78
default settings section
pssqr.ini 286
DEFAULT value 27
DEFAULT-NUMERIC argument, DECLARE-VARIABLE command 100
#DEFINE command 101, 127, 128
example 102
deg function 137
degrees 137
delete function 144
destination field 168
dimension hierarchy
extending 216
specifying 216
display characters 108
DISPLAY command
edit masks 192
example 104
understanding 103
DISTINCT argument, BEGIN-SELECT command 48
DIVIDE command 106
example 106
-DL command-line parameter 306
-Dnn flag 7, 256, 264
-DNT flag 7, 99
DO argument, EXECUTE command 114
DO command 45, 46, 107
example 107
document marker 38, 179
document paragraphs 4, 37, 179, 428
documentation
printed xii
related xii
updates xii
DOLLAR-SYMBOL command 269, 278, 280
DOT-LEADER argument, DECLARE-TOC command 98
double precision floating point numbers 22
double-byte character strings 18
double-byte strings 168
dst_any_var argument, MOVE command 169
dynamic columns 15
dynamic query variables 48


## E


-E[file] flag 7, 256, 264
e10 function 137
EDIT argument
PRINT command 185
SHOW command 219
edit function 148
edit masks 61
case sensitivity 191
changing dynamically 192
currency symbols 27, 278
date 27
date samples 191
numeric 27
numeric samples 187
uses 189
edit types 185



Copyright © 1988-2007, Oracle. All rights reserved.

EDIT-OPTION-AD keyword, ALTER-LOCALE command 26, 28
EDIT-OPTION-AM keyword, ALTER-LOCALE command 26, 28
EDIT-OPTION-BC keyword, ALTER-LOCALE command 26, 28
EDIT-OPTION-NA keyword, ALTER-LOCALE command 26, 28
EDIT-OPTION-PM keyword, ALTER-LOCALE command 26, 28
-EH_APPLETS:dir flag 7, 256, 265
-EH_BQD flag 7, 257, 265
-EH_BQD:file flag 7, 257, 265
-EH_BROWSER:xx flag 8, 257, 265
-EH_CSV flag 8, 257, 265
-EH_CSV:file flag 8, 257, 265
-EH_CSVONLY flag 8, 257, 265
-EH_FULLHTML:xx flag 8, 257, 265
-EH_Icons:dir flag 8, 257, 265
-EH_IMAGES:dir flag 8
-EH_KEEP flag 9
-EH_LANGUAGE:xx flag 9, 257
-EH_PDF flag 9, 257, 265
-EH_Scale:{nn} flag 9, 257, 266
-EH_XML:file flag 9
-EH_ZIP[:file] flag 9
ellipses, use of in syntax 3
ELSE command 108
#ELSE compiler directive 108, 124
ENCODE command 108
  example 109
ENCODING environment variable 18
END-DECLARE command 109
END-DOCUMENT command 37, 109
END-EVALUATE command 109
END-EXECUTE command 38
#end-file reserved variable 17, 211
END-FOOTING command 41, 109
  example 109
END-HEADING command 42, 109
END-IF command 110, 126
#END-IF compiler directive 110
  example 110
END-PROCEDURE command 43, 110
END-PROGRAM command 46, 110, 269
  example 111
END-REPORT command 269
END-SELECT command 46, 110
END-SETUP command 2, 49, 110
END-SQL command 50, 110
END-WHILE command 110, 231
#ENDIF compiler directive 110, 124
ending a query 115
Enhanced-HTML section, pssqr.ini 303
Environment sections, pssqr.ini 294
ERASE-PAGE argument, NEXT-COLUMN command 175
error messages, *See* messages
errors
  testing conditions 212
  trapping 48
escape sequences 108
EVALUATE command 109, 111, 231
  branching 111
  example 112
  exiting from with BREAK 53
  nesting 111
exclamation points
  displaying 4
  for comment lines 4
  use of in syntax 3, 4
exclusion ranges 302
execute
  begin 38
  end 38
EXECUTE command
  example 114
  Sybase and Microsoft SQL Server 113
exists function 144
EXIT-SELECT command 115
  example 115
exp function 138
exponents 137
expressions 132
  operands 133
  operators 133
EXTENT argument, LOAD-LOOKUP command 165
EXTRACT command 115
  example 116

## F

-F[file|directory] flag 10, 258, 266
FIELD argument, CREATE-ARRAY command 63

Copyright © 1988-2007, Oracle. All rights reserved.

-FILE command-line parameter 306
file number 177
file types
FIXED 177
FIXED_NOLF 177
VARY 177
@file... command-line argument 5
file-related functions 144
delete 144
exists 144
rename 144
files
closing 58
include 128
including external source 128
reading 210
FILL argument
DECLARE-CHART command 69
PRINT command 194
FIND command 117
example 118
FIXED argument, OPEN command 177
FIXED file type 177
FIXED_NOLF argument, OPEN command 177
FIXED_NOLF file types 177
flagging SQR 5
flags command-line argument 5
FLOAT argument, DECLARE-VARIABLE command 100
float variables 16, 22
floating point numbers 22
floor function 138
FONT argument
ALTER-PRINTER command 33
DECLARE PRINTER command 276
DECLARE-PRINTER command 89
FONT-TYPE argument
ALTER-PRINTER command 33
DECLARE PRINTER command 275
DECLARE-PRINTER command 87
fonts
adding entries 298
changing 280
HP LaserJet 90
PostScript 91
Windows 93
Fonts section, pssqr.ini 93, 298
specifying character sets 299
FooterBackground 68
FooterForeground 68
footers 2
FOOTING argument, ALTER-REPORT command 33, 34
FOOTING section 2, 41
FOOTING-SIZE argument, ALTER-REPORT command 34
footings 33
FOR-APPEND argument, OPEN command 177
FOR-READING argument, OPEN command 177
FOR-REPORTS argument
BEGIN-FOOTING command 41
BEGIN-HEADING command 43
DECLARE-PRINTER command 86
DECLARE-PROCEDURE command 95
DECLARE-TOC command 98
USE-PROCEDURE command 229
FOR-TOCS argument
BEGIN-FOOTING command 41
BEGIN-HEADING command 43
FOR-WRITING argument, OPEN command 177
FOREGROUND/BACKGROUND argument, PRINT command 194
form feeds, trailing 14
form letters 4
form-letter reports 4
FORMAT argument, INPUT command 130
format_mask argument, MOVE command 169
FORMFEED argument, DECLARE-LAYOUT command 80, 281
frame.html 8
FROM PARAMETER, BEGIN-EXECUTE command 39
FROM ROWSET argument, BEGIN-EXECUTE command 39
functions
custom 161
file-related 134, 144
miscellaneous 134, 144
numeric 134, 162
random 162
string 161
transcendental 143
functions, miscellaneous


Copyright © 1988-2007, Oracle. All rights reserved.

array 145
ascii 145
asciic 146
chr (character) 146
cond 146
edit 148
getenv (get environment variable) 148
isblank 149
isnull 150
to_char 158
to_multi_byte 159
to_number 159
to_single_byte 159
wrapdepth 161

## G

GET command 118
example 118
GET-COLOR command 119
example 120
getenv function 148
glossary 431
GOTO command 121
example 121
GOTO-TOP argument, NEXT-COLUMN command 175
-GPRINT=YES | NO flag 10, 258
GRAPHIC BOX command 121
example 123
GRAPHIC command 121
GRAPHIC FONT command 269, 279
GRAPHIC HORZ-LINE command 121
example 123
GRAPHIC VERT-LINE command 121
example 123
grid, SQR 78

## H

HeaderBackground 68
HeaderForeground 68
headers 2
HEADING argument, ALTER-REPORT command 33, 34
HEADING section 2, 42
HEADING-SIZE argument, ALTER-REPORT command 34
headings 33, 42
HEIGHT argument, PRINT-BAR CODE command 201
HORZ-LINE argument, GRAPHIC command 123
hostname, reserved variable for current 18
HPLASERJET 86
HTML
activating procedures 237
adding horizontal divider 236
applets 7
blinking text 242
<BODY> tags 239
centering 236
citations 243
code listings 243
column headers 254
columns in tables 254
definition lists 249
definitions in definition lists 248
directory lists 248
displaying 8
ending paragraphs 238
enhanced code 8
general-purpose procedures 235
generating 8
generating output 235
<HEAD> tags 239
heading procedures 239
highlighting procedures 242
hypertext link procedures 246
hypertext links 247
inserting images 237
keyboard input 244
line breaks 236
list items 249
list procedures 247
marking text that cannot wrap 237
menus 250
ordered lists 250
preventing wrapping 237
publishing 235
rows in tables 254
sample style text 244
starting new paragraph 238
strikethrough text 245
subscript text 245
superscript text 246
table captions 252
table procedures 251
tables 253
terms in definition lists 249
unordered lists 251


Copyright © 1988-2007, Oracle. All rights reserved.

HTML buttons 299
html_a procedure 247
html_a_end procedure 247
html_blink procedure 242
html_blink_end procedure 242
html_br procedure 236
html_caption procedure 252
html_caption_end procedure 252
html_center procedure 236
html_center_end procedure 236
html_cite procedure 243
html_cite_end procedure 243
html_code procedure 243
html_code_end procedure 243
html_dd procedure 248
html_dd_end procedure 248
html_dir procedure 248
html_dir_end procedure 248
html_dl procedure 249
html_dl_end procedure 249
html_dt procedure 249
html_dt_end procedure 249
html_h1 procedure 240
html_h1_end procedure 240
html_h2 procedure 240
html_h2_end procedure 240
html_h3 procedure 241
html_h3_end procedure 241
html_h4 procedure 241
html_h4_end procedure 241
html_h5 procedure 241
html_h5_end procedure 242
html_h6 procedure 242
html_h6_end procedure 242
html_hr procedure 236
HTML-Images section, pssqr.ini 299
html_img procedure 237
html_kbd procedure 244
html_kbd_end procedure 244
html_li procedure 249
html_li_end procedure 250
html_menu procedure 250
html_menu_end procedure 250
html_nobr procedure 237
html_nobr_end procedure 237
html_ol procedure 250
html_ol_end procedure 251
html_on procedure 237
html_p procedure 238
html_p_end procedure 238
html_samp procedure 244
html_samp_end procedure 244
html_set_body_attributes procedure 239
html_set_head_tags procedure 239
html_strike procedure 245
html_strike_end procedure 245
html_sub procedure 245
html_sub_end procedure 245
html_sup procedure 246
html_sup_end procedure 246
html_table procedure 253
html_table_end procedure 253
html_td procedure 254
html_td_end procedure 254
html_th procedure 254
html_th_end procedure 254
html_tr procedure 254
html_tr_end procedure 254
html_ul procedure 251
html_ul_end procedure 251
hyperbolic cosine (cosh) function 137
hyperbolic sine 141
hyphens
  in command names 3
  in command-line flags 5
  in commands 4

## I

-I command-line parameter 305
-ID flag 10, 258, 266
-Idir_list flag 10
IF command 110, 126
  and WHILE commands 231
  example 127
  nested within a WHILE loop 231
  nesting 126
#IF compiler directive 110, 124
  example 125
#IFDEF compiler directive 110, 124, 125, 127
#IFNDEF compiler directive 110, 124, 125, 127
IMAGE-SIZE argument
  DECLARE-IMAGE command 77
  PRINT-IMAGE command 208
images 77
  declaring 77
  printing 208
#INCLUDE command 10, 128
  example 128


Copyright © 1988-2007, Oracle. All rights reserved.

include files
  nesting 128
  understanding 128
INDENTATION argument, DECLARE-TOC command 98
indenting text 38
Informix
  COMMIT command 59
  ROLLBACK command 212
  STOP command 222
INIT-STRING argument, DECLARE-PRINTER command 86, 284
initcap function 56
initialization file 285
INPUT command
  example 130
  preventing user input 14
  prompts 129
  understanding 129
INPUT commands 5
INPUT-DATE-EDIT-MASK keyword, ALTER-LOCALE command 26, 27
INSERT command 129
instr function 149
instrb function 149
int_lit abbreviation 22
INTEGER argument, DECLARE-VARIABLE command 100
INTEGER data type 129
integer variables 16
INTO argument, EXECUTE command 114
isblank function 149
isnull function 150
ITEM-COLOR argument, DECLARE-CHART command 68, 69

## J

joins 165

## K

-KEEP flag 10, 12, 96, 258
KEY argument, LOAD-LOOKUP command 165

## L

labels 121
LAST-PAGE command 131
  example 132
layout
  DEFAULT 78
  reports 78
LAYOUT argument, DECLARE-REPORT command 96
LEFT-MARGIN argument
  DECLARE PRINTER command 273
  DECLARE-LAYOUT command 80
LEGEND argument, DECLARE-CHART command 70
LEGEND-PLACEMENT argument, DECLARE-CHART command 70
LEGEND-PRESENTATION argument, DECLARE-CHART command 70
LEGEND-TITLE argument, DECLARE-CHART command 70
LegendBackground 68
LegendForeground 68
length function 150
lengthb function 150
lengthp function 151
lengtht function 151
LET command 132
  datenow function 270
  edit masks 192
  example 163
LEVEL argument, TOC-ENTRY command 224
-LG command-line parameter 307
LINE-HEIGHT argument, DECLARE-LAYOUT command 78, 82
LINE-SIZE argument, DECLARE PRINTER command 274
LINE-WIDTH argument, DECLARE-LAYOUT command 81
LINEPRINTER 86
lines
  breaking in HTML 236
  drawing 122
  horizontal 123
  reserved variable for current 17
  reserved variable for maximum 18
  use of in syntax 4
  vertical 123
LINES-INCH argument, DECLARE PRINTER command 274
lis files 11
list variables 16, 19
  accessing 19

Copyright © 1988-2007, Oracle. All rights reserved.

arguments 19
assigning 19
creating 19
defining 19
literals 20
determining value 111
displaying 103, 217
numeric 20
text 20
-LL{s | d}{c | i} flag 11
LOAD-LOOKUP command 11, 50, 163, 167
example 166
loading an internal table 164
LOCAL argument, BEGIN-PROCEDURE command 45
local procedures 45
LOCALE keyword, ALTER-LOCALE command 27
Locale section, pssqr.ini 296
locales
changing parameters 25
defined 25
reserved variable for current 18
selecting 25
SYSTEM 25
log base 10 139
log base e 138
log function 138
log messages 11
log10 function 139
logical comparison operators 132
logical expressions 126
logon, defining parameters 76
LOOKUP command 164, 167
example 167
lookup tables, searching 167
LOOPS argument, BEGIN-SELECT command 48
lower function 151
LOWER value, DAY-OF-WEEK-CASE keyword 28
LOWERCASE command 167
example 168
lowercase, use of in syntax 3, 4
lpad function 152
-LPFLAGS "<flags>" command-line parameter 307
ltrim function 152


## M


marker locations 16
MATCH argument, PRINT command 194
mathematical operators 132
MAX-COLUMNS argument, DECLARE-LAYOUT command 81
MAX-LINES argument, DECLARE-LAYOUT command 81
MAXLEN argument, INPUT command 129, 130
MBTOSBS command 168
members, returning set 217
messages
database errors 17
directing to a file 7
disabling display 14
numbered 312
setting path and name of error message file 15
unnumbered 309
-Mfile flag 11
Microsoft SQL Server
BEGIN TRANSACTION command 212
ROLLBACK TRANSACTION command 212
STOP command 222
miscellaneous functions
ltrim 153
range 154
replace 155
roman 155
rpad 155
rtrim 156
mod function 139
MODE argument, SECURITY command 214
MONEY argument
DISPLAY command 104
MOVE command 169
PRINT command 195
SHOW command 219
MONEY-EDIT-MASK keyword, ALTER-LOCALE command 26, 27
MONEY-SIGN keyword, ALTER-LOCALE command 26, 27
MONEY-SIGN-LOCATION keyword, ALTER-LOCALE command 26, 27



Copyright © 1988-2007, Oracle. All rights reserved.

MONEY-SYMBOL command 269, 278, 280
MONTHS-CASE keyword, ALTER-LOCALE command 27, 28
MONTHS-FULL keyword, ALTER-LOCALE command 27, 28
MONTHS-SHORT keyword, ALTER-LOCALE command 27, 28
MOVE command
  edit masks 192
  example 169
  understanding 168
-MR command-line parameter 306
multibyte characters 213
multiple reports 32, 41
MULTIPLY command 172
  example 172
multirow lists 19

## N

NAME argument
  ALTER-COLOR-MAP command 23
  BEGIN-FOOTING command 41
  BEGIN-HEADING command 43
  CLEAR-ARRAY command 58
  CREATE-ARRAY command 63
  CREATE-COLOR-PALETTE command 64
  LOAD-LOOKUP command 165
natural log base e raised to x power 138
navigation bar 14
NEED argument, NEXT-LISTING command 176
nesting
  arguments 134
  IF command 126
NEW-PAGE command 173
  example 173
NEW-REPORT command 173
  example 174
NEXT-COLUMN command 59, 174
  example 175
NEXT-LISTING command 175
  example 176
nn abbreviation 22
NO-ADVANCE argument, NEXT-LISTING command 176
NO-CHANGE value, DAY-OF-WEEK-CASE keyword 28
NO-DUPLICATE argument, DECLARE-CONNECTION command 75
NO-FORMFEED command 269, 281
NOLF argument, PRINT-DIRECT command 207
NOLINE argument
  DISPLAY command 103, 104
  SHOW command 219
-NOLIS flag 11, 96, 258
nondisplay characters 108, 283, 284
nonrecursive variables 19
NOP argument, PRINT command 195
NOPROMPT argument, INPUT command 130
NORMAL argument, SHOW command 219
notes xv
NOWAIT argument, CALL command 55
num_col abbreviation 21
num_lit abbreviation 22
num_var abbreviation 21
NUMBER argument
  DISPLAY command 104
  MOVE command 169
  PRINT command 195
  SHOW command 219
number columns 15
NUMBER data type 129
NUMBER-EDIT-MASK keyword, ALTER-LOCALE command 26, 27
numerals
  numeric literals 20
numeric edit format characters 186
numeric edit masks 27
numeric functions 134, 162
  absolute value 135
  arccosine (acos) 135
  arcsine (asin) 135
  arctangent (atan) 136
  ceil 136
  cosine (cos) 136
  degrees (deg) 137
  e10 (10 raised to x power) 137
  exp (natural log base e raised to x⇒ power) 138
  floor (largest integer) 138
  hyperbolic cosine (cosh) 137

Copyright © 1988-2007, Oracle. All rights reserved.

hyperbolic sine (sinh) 141
hyperbolic tangent (tanh) 143
log base e 138
log10 (log base 10) 139
mod 139
power 139
radians (rad) 140
round 140
sign 141
sin 141
sine 141
square root (sqrt) 142
tangent (tan) 142
truncate (trunc) 143
numeric literals 20
numeric operands 133
numeric variables 16
nvl function 153


## O


-O[file] flag 11, 258, 266
ODBC
COMMIT command 59
ROLLBACK command 212
-OF command-line parameter 306
-olim flag 11
ON-BREAK argument, PRINT command 195
ON-ERROR argument
BEGIN-EXECUTE command 39
BEGIN-SELECT command 48
BEGIN-SQL command 52
CONNECT command 62
DIVIDE command 106
EXECUTE command 114
-OP command-line parameter 306
OPEN command 176
example 177
operands
converting 133
understanding 133
operating system
issuing commands 53
opening files 177
reserved variable for returning 18
operators, precedence 133
Oracle 48
BEGIN-SQL command, -Cnn 52
COMMIT command 59
PL/SQL 51
ROLLBACK command 212
ORIENTATION argument
DECLARE PRINTER command 273
DECLARE-LAYOUT command 78, 80
-OT command-line parameter 306
output
available types 2
choosing printer type 12
HTML 235
layout of files 78
overriding default file 10
reserved variable for returning file name 19
SQR Print 263, 268
using the command line 263
writing directly to files 207
OUTPUT argument, EXECUTE command 114


## P


-P flag 11, 258, 266
page buffer 207
page footers 2
page headers 2
page numbers
last 131
reserved variable for current 17
page overflow 173
#page-count reserved variable 17
PAGE-DEPTH argument, DECLARE-LAYOUT command 78, 81
PAGE-NUMBER command 178
example 178
PAGE-SIZE command 269, 282
pages
adding numbers 178
new 173
setting position 179
writing 173
writing data at specified position 180
palettes 67
PAPER-SIZE argument, DECLARE-LAYOUT command 78, 80, 282
paragraphs
completing 109, 111
document 4, 37, 179, 428
SELECT 47, 115
SQL 50



Copyright © 1988-2007, Oracle. All rights reserved.

PARAMETER_LIST argument, BEGIN-EXECUTE command 39
PARAMETERS argument, DECLARE-CONNECTION command 75
parentheses, use of in syntax 3
PASSWORD argument, DECLARE-CONNECTION command 75
-PB flag 11, 258
PDF Fonts section, pssqr.ini 300
PDF Fonts: Exclusion Ranges section, pssqr.ini 302
PeopleBooks
  ordering xii
PeopleCode, typographical conventions xiv
pie chart, printing 206
PIE-SEGMENT-EXPLODE argument, DECLARE-CHART command 71
PIE-SEGMENT-PERCENT-DISPLAY argument, DECLARE-CHART command 70
PIE-SEGMENT-QUANTITY-DISPLAY argument, DECLARE-CHART command 70
PITCH argument
  ALTER-PRINTER command 33
  DECLARE-PRINTER command 88
PL/SQL 51
PlotAreaBackground 68
PlotAreaForeground 68
POINT-MARKERS argument, DECLARE-CHART command 70
POINT-SIZE argument
  ALTER-PRINTER command 33
  DECLARE PRINTER command 275
  DECLARE-PRINTER command 87
position abbreviation 22
position argument, PRINT-IMAGE command 208
POSITION command 179
  example 179
postscri.str file 184
POSTSCRIPT 86
power function 139
precedence, operators 133
precision 133
predefined variables, *See* reserved variables
prerequisites xi
PRINT ... CODE command 270, 283
PRINT ... CODE-PRINTER command 270, 283
PRINT command 119, 180
  encode characters 108
  format commands 180
Print dialog box, Windows 268
PRINT-BAR-CODE command 200
  example 201
PRINT-CHART command 67, 202
  example 206
PRINT-DIRECT command
  encode characters 108
  example 207
  understanding 207
PRINT-IMAGE command
  and DECLARE-IMAGE 77
  example 208
  understanding 207
PRINT-TEXT-BACKGROUND argument
  GET-COLOR command 119
  SET-COLOR command 215
PRINT-TEXT-FOREGROUND argument
  GET-COLOR command 119
  SET-COLOR command 215
printed documentation xii
PRINTER argument, PRINT-DIRECT command 207
printer control characters 11
printer type
  choosing 12
  setting 227
PRINTER.COLOR option 67
PRINTER-DEINIT command 269, 283
-PRINTER:EH flag 12
-PRINTER:EP flag 12
-PRINTER:HP flag 12
-PRINTER:HT flag 12
PRINTER-INIT command 270, 284
-PRINTER:LP flag 12
-PRINTER:PD flag 12
-PRINTER:PS flag 12
PRINTER-TYPE argument, DECLARE-REPORT command 96
-PRINTER:WP flag 12, 93
-PRINTER:xx flag 12, 259, 267
printers
  altering parameters at runtime 32
  overriding defaults 85
printing
  Hebrew 186

Copyright © 1988-2007, Oracle. All rights reserved.

on Windows 93
PROCEDURE argument, BEGIN-EXECUTE command 39
PROCEDURE section 2
procedures
beginning 44
calling recursively 44
declaring 94
global 44
invoking 107
local 44
passing date arguments to 44
passing numeric variables to 44
passing string arguments to 44
running 2, 107
stored 51
USE-PROCEDURE command 228
process file, reserved variable for returning 19
process ID, reserved variable for returning 18
Processing-Limits section, pssqr.ini 292
program command-line argument 4
PROGRAM section 2, 46
program structure 2
program.lis file 10
program.log file 11
program.spf file 15
program.sqr file 10
program.sqt file 12
programs, samples 425
prompts, INPUT command 129
PROPERTIES argument, BEGIN-EXECUTE command 39
pssqr.exe, command-line options 304
pssqr.ini
for other platforms 285
for Windows 285
pssqr.ini sections
Colors 304
default settings 286
Enhanced-HTML 303
Environment 294
Fonts 298
HTML-Images 299
Locale 296
PDF Fonts 300
PDF Fonts: Exclusion Ranges 302
Processing-Limits 292
TrueType font 302
publishing HTML 235
PUT command
example 210
rules 209
understanding 209


## Q


queries, ending 115
QUIET argument
LOAD-LOOKUP command 165
STOP command 222
quotes
use of in commands 21


## R


-R command-line parameter 305
radians (rad) function 140
random function 162
range function 154
READ command 210
example 211
RECORD argument, OPEN command 177
records
reading 210
writing to a file 232
related documentation xii
relinking SQR 161
REMOVE argument, SECURITY command 214
rename function 144
replace function 155
report footer 2
report headers 2
report procedures
running 2
{report}.bqd file 7
{report}.csv file 8
reports
BEGIN-HEADING command 42
closing 173
defining 96
designing 2
displaying output on screen 7
layout 2, 78
marking sections for security 213
multiple 32, 41, 174
opening 173
output types 2



Copyright © 1988-2007, Oracle. All rights reserved.

referencing external source files 128
reserved variable for returning name of output file 19
samples 20
switching between 230
testing 13
reserved variables 16
global versus local 44
$username 61
RESET-STRING argument, DECLARE-PRINTER command 87
#return-status reserved variable 17
RETURN_VALUE argument, LOAD-LOOKUP command 165
REVERSE argument, SHOW command 219
rgb argument
CREATE-COLOR-PALETTE command 64
DECLARE-COLOR-MAP command 75
RGB values 23
RIGHT-MARGIN argument, DECLARE-LAYOUT command 81
ROLLBACK command 212, 222
and COMMIT command 59
example 212
ROLLBACK TRANSACTION command 60
ROLLBACK TRANSACTION command, Sybase and Microsoft SQL Server 212
roman function 155
ROUND argument
ADD command 23, 106
MULTIPLY command 172
SUBTRACT command 224
round function 140
ROWCOUNT command, Oracle and Sybase 17
rows
affected by SELECT paragraph 17
buffering 6
eliminating duplicates from query 48
reserved variable for counting 17
specifying number to retrieve 48
specifying number to retrieve simultaneously 48
ROWS argument, LOAD-LOOKUP command 165
-RP command-line parameter 305
rpad function 155
-RS flag 12, 13
RSV argument, BEGIN-EXECUTE command 39
rtrim function 156
runtime file
saving 12
using 13


## S


-S flag 13, 259
sample programs 425
sample reports 20
SBTOMBS command 213
scaling factor 9
SCHEMA argument, BEGIN-EXECUTE command 39
screen I/O 217
searches, using LOAD-LOOKUP 164
sections, completing 109, 111
SECURITY command 213
example 214
security IDs 214
SELECT paragraphs
beginning 47
exiting 115
sequential processing, FOR-READING 177
SET argument, SECURITY command 214
SET-COLOR command 214
example 215
SET-GENERATIONS command 216
example 216
SET-LEVELS command 216
example 217
SET-MEMBERS command 217
example 217
SETUP section 2, 49
SHADE argument, PRINT command 184, 197
SHOW command
edit masks 192
example 219
understanding 217
versus DISPLAY command 103
sid argument, SECURITY command 214
sign function 141
sin function 141
sine function 141


Copyright © 1988-2007, Oracle. All rights reserved.

single quotes, displaying 4
single quotes, use of in syntax 3
single-byte characters 213
single-byte strings 168
sinh function 141
SIZE argument, CREATE-ARRAY command 63
SKIPLINES argument, NEXT-LISTING command 176
SORT argument, LOAD-LOOKUP command 165
SOURCE argument
  DECLARE-IMAGE command 77
  PRINT-IMAGE command 208
source code 2
source field 168
source files, including external 128
spaces, use of in syntax 4
spf files 11, 12
SPF Viewer, invoking 15
SQL paragraphs 50
SQL, syntax errors 48
#sql-count reserved variable 17
$sql-error reserved variable 17
#sql-status reserved variable 17
SQR
  avoiding older commands 269
  banner 14
  benefits of 2
  command-line arguments 4
  command-line interface 4
  compiling 124
  data elements 15
  defined 1
  executables 4
  flags 5
  grid 78
  high-level programming capabilities 2
  initialization file 15, 285
  language 1, 2
  language overview 1
  object-oriented program logic 2
  page buffer 207
  preventing database signin 14
  procedures 2
  process file 19
  process ID 18
  program structure 1, 2
  relinking 161
  relinking to CALL 54
  reserved variable for returning process file 19
  reserved variable for returning process ID 18
  reserved variables 16
  rules for entering commands 1, 4
  sample programs 425
  stopping 222
  syntax conventions 1, 3
  tools 1
  understanding 1, 2
  writing applications with 2
SQR commands
  rules for entering 1
SQR executable 4
SQR Execute 1, 4
  flags 255
  running 255
SQR Print 1, 4, 263
  command-line flags 264
  generating output 268
  Windows Print dialog box 268
SQR reports 2
SQR Samples 1
$sqr-database reserved variable 18
{sqr-database} reserved variable 18
SQR_DB_DATE_FORMAT setting 44, 60
SQR_DB_DATE_ONLY_FORMAT setting 44, 61
SQR_DB_TIME_ONLY_FORMAT setting 44, 61
$sqr-dbcs reserved variable 18
{sqr-dbcs} reserved variable 18
sqr-encoding reserved variable 18
{sqr-encoding- console} reserved variable 17
$sqr-encoding- database reserved variable 17
{sqr-encoding- database} reserved variable 17
$sqr-encoding- file-input reserved variable 17
{sqr-encoding} reserved variable 18
$sqr-encoding-console reserved variable 17
{sqr-encoding-file-input} reserved variable 17
$sqr-encoding-file-output reserved variable 18


Copyright © 1988-2007, Oracle. All rights reserved.

{sqr-encoding-file-output} reserved variable 18
$sqr-encoding-report-output reserved variable 18
{sqr-encoding-report-output} reserved variable 18
$sqr-encoding-source reserved variable 18
{sqr-encoding-source} reserved variable 18
sqr-hostname, reserved variable 18
{sqr-hostname} reserved variable 18
sqr.ini file 15
  Default-Settings section of 27
$sqr-locale reserved variable 18
sqr.log file 11
#sqr-max-columns reserved variable 18
#sqr-max-lines reserved variable 18
#sqr-pid reserved variable 18
$sqr-platform reserved variable 18
{sqr-platform} reserved variable 18
$sqr-program reserved variable 19
$sqr-report reserved variable 19
$sqr-ver reserved variable 19
sqrerr.dat file 15
SQREXT.MAK 54
SQRMAKE 54
SQRP executable 4, 263
SQRT executable 4, 255
sqrt function 142
SQRW executable 4
SQRWP executable 4, 263
SQRWT executable 4, 255
square brackets, use of in syntax 3
square root 142
STARTUP-FILE argument
  DECLARE PRINTER command 275
  DECLARE-PRINTER command 88
STATUS argument
  BEGIN-EXECUTE command 39
  INPUT command 130
  INPUT command, values for 130
  OPEN command 177
  READ command 211
  WRITE command 233
STOP command 222
  example 222
stored procedures
  executing 113
  preventing creation 15, 48, 52
  understanding 51
STRING command 222
  example 223
string functions 161
  extract 115
  instr 149
  instrb 149
  length 150
  lengthb 150
  lengthp 151
  lengtht 151
  lower 151
  lowercase 167
  lpad 152
  ltrim 152
  substr 157
  substrb 157
  substrp 158
  substrt 158
  transform 160
  translate 160
  unicode 160
  upper 161
string operands 133
string values 144
string variables, converting to uppercase 226
strings
  converting double-byte to single-byte 168
  copying 116
  copying into text variables 225
  double-byte 168
  finding character sequences within 117
  single-byte 168
strtodate function 156, 191
Structured Query Report, *See* SQR
structuring a program 2
SUB-TITLE argument, DECLARE-CHART command 69
subroutines
  calling 53, 55
  writing 53
substitution variables
  ASK command 36
  #DEFINE command 101
  #IF compiler directive 124
  #IFDEF compiler directive 127
  #IFNDEF compiler directive 128
substr function 157

Copyright © 1988-2007, Oracle. All rights reserved.

substrb function 157
substrings 115, 225
substrp function 158
substrt function 158
SUBTRACT command 223
  example 224
suggestions, submitting xvi
Sybase
  BEGIN TRANSACTION command 212
  ROLLBACK TRANSACTION command 212
  STOP command 222
  USE command 226
SYBASE CT-Lib 48
SYMBOL-SET argument
  ALTER-PRINTER command 33
  DECLARE PRINTER command 275
  DECLARE-PRINTER command 87
syntax conventions 3
SYSTEM argument, CALL command 55, 56
SYSTEM locale 25
SYSTEM value 27

## T

-T{B} flag 13
-T{B|Z|BZ|ZB} flag 260
-T{Z} flag 14
TABLE argument, LOAD-LOOKUP command 165
table of contents 97
  *See Also* DECLARE-TOC command
  creating entries 224
  defining 97
  footing for 41
  heading for 43
  sharing 98
  suppressing 15
tables
  loading internal 164
tabs
  use of in syntax 4
tabular reports, NEXT-LISTING command 176
tan (tangent) function 142
tanh function 143
temporary stored procedures 15
terms 431
testing
  error conditions 212
  STOP command 222
TEXT argument
  DECLARE-VARIABLE command 100
  PRINT-BAR-CODE command 201
  TOC-ENTRY command 224
text edit format characters 185
text literals 20
text variables 16
  concatenating 222
  copying strings into 225
THOUSAND-SEPARATOR keyword, ALTER-LOCALE command 26, 28
TIME column type 185
TIME-SEPARATOR keyword, ALTER-LOCALE command 26, 28
times
  displaying 103
  reserved variable for current 16
TITLE argument, DECLARE-CHART command 69
-Tnn flag 13, 259
to_char function 158
to_multi_byte function 159
to_number function 159
to_single_byte function 159
TOC argument, DECLARE-REPORT command 96
TOC-ENTRY command 224
  example 224
TOP-MARGIN argument
  DECLARE PRINTER command 273
  DECLARE-LAYOUT command 80
-TR command-line parameter 306
trailing blanks
  preserving 11, 15
  trimming 13
trailing form feeds 14
trailing zeros 14
transcendental functions 143
transform function 160
translate function 160
TrueType Font section, pssqr.ini 302
trunc (truncate) function 143
txt_col abbreviation 21
txt_lit abbreviation 22
txt_var abbreviation 21
TYPE argument


Copyright © 1988-2007, Oracle. All rights reserved.

DECLARE PRINTER command 273
DECLARE-CHART command 70
DECLARE-IMAGE command 77
DECLARE-PRINTER command 86
INPUT command 129, 130
PRINT-BAR-CODE command 201
PRINT-IMAGE command 208
typographical conventions xiv


## U


UCALL function
arguments 54
avoiding database calls 53
UCALL.C source file 54
UFUNC.C 161
UNDERLINE argument
PRINT command 198
SHOW command 219
underscores
in procedure names 3
in variable names 3
unicode function 160
UNSTRING command 225
example 225
UPC-A bar code 201
UPDATE command 129
upper function 161
UPPER value, DAY-OF-WEEK-CASE keyword 28
UPPERCASE command 226
example 226
uppercase, use of in syntax 3, 4
USE command 50, 226
example 226
USE-COLUMN command 59, 227
example 227
USE-PRINTER-TYPE command 227
example 228
USE-PROCEDURE command 228
example 229
USE-REPORT command 230
example 230
USER argument, DECLARE-CONNECTION command 75
user input
INPUT command 129
preventing 14
$username reserved variable 19, 61
username, database 19


## V


VALUE argument, ALTER-COLOR-MAP command 23
value determination 111
variables 16
assigning 20
assigning values of expressions to 132
bind 48
case sensitivity 16
converting to uppercase 226
date 16
decimal 16, 22
declaring 99
determining value 111
displaying 103, 217
dynamic query 48
float 16, 22
global 16
in syntax 3
initialization 16
integer 16
length of 16
list 16, 19
list variable arguments 19
local 16
names 16
naming 16
nonrecursive 19
numeric 16
reference 37
reserved 16
rules 16
string 116
substitution 37, 101, 127
text 16
VARY argument, OPEN command 177
VARY file type 177
VERT-LINE argument, GRAPHIC command 123
vertical bars, use of in syntax 3
visual cues xv
-Vserver flag 14, 260


## W


WAIT argument, CALL command 55
warnings xv
WHEN argument, EVALUATE command 112


Copyright © 1988-2007, Oracle. All rights reserved.

WHEN-OTHER argument, EVALUATE command 112
WHERE argument, LOAD-LOOKUP command 165
WHILE ... END-WHILE loop 53
WHILE command 111
  example 231
  exiting from with BREAK 53
  understanding 230
WHILE loop 230
wildcard characters, on UNIX/Linux-based systems 4
Windows
  generating output 268
  selecting a printer 268
WITH RECOMPILE argument, EXECUTE command 114
WRAP argument, PRINT command 198
wrapdepth function 161
WRITE command 232
  example 233

## X

X-AXIS-GRID or Y-AXIS-GRID argument, DECLARE-CHART command 73
X-AXIS-LABEL or Y-AXIS-LABEL argument, DECLARE-CHART command 71
X-AXIS-MAJOR INCREMENT or Y-AXIS-MAJOR INCREMENT argument, DECLARE-CHART command 72
X-AXIS-MAJOR-TICK- MARKS argument, DECLARE-CHART command 72
X-AXIS-MAX-VALUE argument, DECLARE-CHART command 71
X-AXIS-MIN-VALUE argument, DECLARE-CHART command 71
X-AXIS-MINOR INCREMENT or Y-AXIS-MINOR INCREMENT argument, DECLARE-CHART command 72
X-AXIS-MINOR-TICK-MARKS argument, DECLARE-CHART command 72
X-AXIS-SCALE or Y-AXIS-SCALE argument, DECLARE-CHART command 71
X-AXIS-TICK-MARK-PLACEMENT or Y-AXIS-TICK-MARK-PLACEMENT argument, DECLARE-CHART command 73
-XB flag 14, 260, 267
-XC argument, EXECUTE command 114
-XC flag 14
-XCB flag 14, 260
-XI flag 14
-XL flag 14, 261
-XLFF flag 14
-XMB flag 14, 261
-XNAV flag 14, 261, 267
-XP argument
  BEGIN-SELECT command 48
  BEGIN-SQL command 52
-XP flag 15, 261
-XTB flag 15, 261, 267
-XTOC flag 15, 261, 267

## Y

Y-AXIS-MAJOR-TICK- MARKS argument, DECLARE-CHART command 72
Y-AXIS-MAX-VALUE argument, DECLARE-CHART command 71
Y-AXIS-MIN-VALUE argument, DECLARE-CHART command 71
Y-AXIS-MINOR-TICK-MARKS argument, DECLARE-CHART command 72

## Z

zeros, trailing 14
-ZIF{file} flag 15, 261, 267
ZIP+4 Postnet 201
-ZIV flag 15, 261
-ZMF{file} flag 15, 261, 267


Copyright © 1988-2007, Oracle. All rights reserved.