PeopleSoft®

Enterprise PeopleTools 8.45 PeopleBook: Data Management

June 2004

Enterprise PeopleTools 8.45 PeopleBook: Data Management
SKU PT845ADM-B 0604
Copyright © 1988-2004 PeopleSoft, Inc. All rights reserved.

All material contained in this documentation is proprietary and confidential to PeopleSoft, Inc. ("PeopleSoft"), protected by copyright laws and subject to the nondisclosure provisions of the applicable PeopleSoft agreement. No part of this documentation may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, including, but not limited to, electronic, graphic, mechanical, photocopying, recording, or otherwise without the prior written permission of PeopleSoft.

This documentation is subject to change without notice, and PeopleSoft does not warrant that the material contained in this documentation is free of errors. Any errors found in this document should be reported to PeopleSoft in writing.

The copyrighted software that accompanies this document is licensed for use only in strict accordance with the applicable license agreement which should be read carefully as it governs the terms of use of the software and this document, including the disclosure thereof.

PeopleSoft, PeopleTools, PS/nVision, PeopleCode, PeopleBooks, PeopleTalk, and Vantive are registered trademarks, and Pure Internet Architecture, Intelligent Context Manager, and The Real-Time Enterprise are trademarks of PeopleSoft, Inc. All other company and product names may be trademarks of their respective owners. The information contained herein is subject to change without notice.

Open Source Disclosure

PeopleSoft 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 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 (c) 1999-2000 The Apache Software Foundation. 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 APACHE SOFTWARE FOUNDATION 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.

OpenSSL

Copyright (c) 1998-2003 The OpenSSL Project. All rights reserved.

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.

SSLeay

Copyright (c) 1995-1998 Eric Young. All rights reserved.

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "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 AUTHOR 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.

Loki Library

Copyright (c) 2001 by Andrei Alexandrescu. This code accompanies the book:

Alexandrescu, Andrei. "Modern C++ Design: Generic Programming and Design Patterns Applied". Copyright (c) 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.

Contents

About This PeopleBook

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

This preface discusses:

- PeopleSoft application prerequisites.
- PeopleSoft application fundamentals.
- Related documentation.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common elements in PeopleBooks.

Note. PeopleBooks document only page elements that require additional explanation. If a page 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 applications are defined in this preface.

PeopleSoft 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 applications.

See *Enterprise PeopleTools 8.45 PeopleBook: Using PeopleSoft Applications*.

You might also want to complete at least one PeopleSoft introductory training course.

You should be familiar with navigating the system and adding, updating, and deleting information by using PeopleSoft windows, menus, and pages. 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 applications most effectively.

PeopleSoft Application Fundamentals

Each application PeopleBook provides implementation and processing information for your PeopleSoft database. However, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals PeopleBook. Each PeopleSoft product line has its own version of this documentation.

The application fundamentals PeopleBook consists of important topics that apply to many or all PeopleSoft applications across a product line. 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 this central PeopleBook. It is the starting point for fundamentals, such as setting up control tables and administering security.

Related Documentation

This section discusses how to:

- Obtain documentation updates.
- Order printed documentation.

Obtaining Documentation Updates

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

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

See Also

PeopleSoft Customer Connection, https://www.peoplesoft.com/corp/en/login.jsp

Ordering Printed Documentation

You can order printed, bound volumes of the complete PeopleSoft documentation that is delivered on your PeopleBooks CD-ROM. PeopleSoft makes printed documentation available for each major release shortly after the software is shipped. Customers and partners can order printed PeopleSoft documentation by using any of these methods:

- Web
- Telephone
- Email

Web

From the Documentation section of the PeopleSoft Customer Connection website, access the PeopleBooks Press website under the Ordering PeopleBooks topic. The PeopleBooks Press website is a joint venture between PeopleSoft and MMA Partners, the book print vendor. Use a credit card, money order, cashier's check, or purchase order to place your order.

Telephone

Contact MMA Partners at 877 588 2525.

Email

Send email to MMA Partners at peoplesoftpress@mmapartner.com.

See Also

PeopleSoft Customer Connection, https://www.peoplesoft.com/corp/en/login.jsp

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, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.
Italics	Indicates field values, emphasis, and PeopleSoft or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply. 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 (|).

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. 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 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.

See *About These PeopleBooks*, "ISO Country and Currency Codes," ISO Country Codes.

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.

See Appendix I, “ISO Country and Currency Codes,” ISO Currency Codes.

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 PeopleSoft reference and training materials. Please send your suggestions to:

PeopleSoft Product Documentation Manager PeopleSoft, Inc. 4460 Hacienda Drive Pleasanton, CA 94588

Or send email comments to doc@peoplesoft.com.

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

Common Elements in These 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.
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. Select Always to run the request every time the batch process runs. Select Don't Run to ignore the request when the batch process runs.
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).
Process Monitor	Click to access the Process List page, where you can view the status of submitted process requests.
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.
Request ID	An ID that represents a set of selection criteria for a report or process.
User ID	An ID that represents the person who generates a transaction.
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.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler

Enterprise PeopleTools 8.45 PeopleBook: Using PeopleSoft Applications

Data Management Preface

This preface provides a general overview of the contents discussed in the Enterprise PeopleTools 8.45 Data Management PeopleBook.

Data Management

This PeopleBook covers a wide range of different applications, tools, and techniques for managing data and databases on your PeopleSoft system, including:

- PeopleSoft Data Mover.
- PeopleSoft Data Archive Manager.
- Data integrity and auditing.
- Diagnostic Framework.
- Administration for supported database platforms.
- Remote data access.

Note. PeopleSoft supports a number of versions of UNIX and Linux in addition to Microsoft Windows. Throughout this book, there are references to operating system configuration requirements. Where necessary, this book refers to specific operating systems by name (for example, Solaris, HP/UX, Linux, etc.); however, for simplicity the word UNIX is used to refer to all UNIX-like operating systems, including Linux.

CHAPTER 1

Getting Started with Data Management

This chapter provides an overview of data management and discusses data management implementation.

Data Management Overview

This section discusses:

- PeopleSoft Data Mover.
- PeopleSoft Data Archive Manager.
- Data integrity and auditing.
- Diagnostic framework.
- Database platform considerations.

PeopleSoft Data Mover

PeopleSoft Data Mover is a stand-alone two-tier program, which you can run through a graphical interface on Microsoft Windows, or a with a command line interface on either Microsoft Windows or UNIX systems.

PeopleSoft Data Mover enables you to perform the following tasks:

- Transfer application data between PeopleSoft databases.
- Move PeopleSoft databases across operating systems and database platforms.
- Execute Structured Query Language (SQL) statements against any PeopleSoft database, regardless of the underlying operating system or database platform.
- Export data in a platform independent manner.
- Control database security and access.
- Create, edit, and run scripts which combine SQL commands and PeopleSoft Data Mover commands for exporting and importing data.

See Chapter 2, “Using PeopleSoft Data Mover,” page 5.

PeopleSoft Data Archive Manager

In any enterprise application, the ability to purge and archive transactional data is critical to data management. You need to have consistent methods to archive transactional data before your database increases to unmanageable sizes. PeopleSoft Data Archive Manager provides an integrated and consistent framework for archiving data from PeopleSoft applications.

Using a predefined template, you can select any queries and multiple objects that meet your archiving requirements. Leveraging the Archive Query in PeopleSoft Query, you can easily define your archive template. To better manage the archive process, you don't have to make any commits to the database until the entire batch has completed.

Note. This PeopleSoft Data Archive Manager replaces a deprecated feature used in PeopleSoft 8.40 through 8.43, which is discussed in an appendix of this PeopleBook.

See Also

Chapter 3, "Using PeopleSoft Data Archive Manager," page 49

Appendix H, "Archiving Data (Deprecated)," page 251

Data Integrity and Auditing

PeopleSoft provides several features to ensure the integrity of the data that is stored in your PeopleSoft system.

Data Integrity Tools

You might want to use the provided data integrity tools during upgrades and system customizations, to verify the PeopleSoft system and check how it compares to the actual SQL objects. The data integrity tools are:

- SQL Alter.

  The primary purpose of the PeopleSoft Application Designer SQL Alter function is to bring SQL tables into accordance with PeopleTools record definitions. You can run SQL Alter in an audit-only mode that alerts you to discrepancies between record definitions and SQL tables, but that doesn't actually perform an alter.

- DDDAUDIT.

  The Database Audit Report (DDDAUDIT) finds inconsistencies between PeopleTools record and index definitions and the database objects. This audit consists of nine queries: four on tables, two on views, and three on indexes.

- SYSAUDIT.

  The System Audit (SYSAUDIT) identifies orphaned PeopleSoft objects and other inconsistencies within the system. An example of an orphaned object is a module of PeopleCode that exists, but which does not relate to any other objects in the system. You can use SYSAUDIT to audit a variety of different aspects of your PeopleSoft system.

See Chapter 4, "Ensuring Data Integrity," page 63.

Trigger-Based Database Level Auditing

PeopleSoft provides trigger-based auditing functionality as an alternative to the record-based auditing that PeopleSoft Application Designer provides. Some countries require that you audit changes to certain data, while some companies audit who is making changes to sensitive data. This level of auditing is not only for maintaining the integrity of the data, but it is also a heightened security measure. PeopleSoft takes advantage of database triggers (offered by most database vendors), and when a user makes a change to a specified field that you are monitoring, the changed data triggers the audit.

The information that a trigger records could include the user that made a change, the type of change that is made, when the change is made, and so on.

See Chapter 5, "Employing Database Level Auditing," page 105.

Diagnostic Framework

PeopleSoft provides a framework for defining and retrieving application data diagnostics within the PeopleSoft Internet Architecture (PIA) environment. Diagnostic Framework retrieves diagnostic information from a PeopleSoft database. With this diagnostic information, you can:

- Discover problematic application-related data.
- Explore setup details.
- Present information to PeopleSoft support in a common format.

Using Diagnostic Framework, you can perform diagnostic tests on your system with minimal instructions from the PeopleSoft Global Support Center. These tests answer application-specific questions to help development and user support teams diagnose and troubleshoot any problems that you may be experiencing.

The tests can request additional parameters to tailor the diagnostics to your situation. They output HTML pages that you can open using any PeopleSoft-supported browser, and XML documents containing the same information in a form suitable for programmatic processing. You can email the HTML or XML documents to an application expert.

See Chapter 6, "Running Diagnostics with Diagnostic Framework," Importing Post-Release Plug-Ins, page 149.

See Chapter 7, "Diagnostic PeopleCode," page 151.

Database Platform Considerations

PeopleSoft supports a wide range of database platforms. Because each relational database management system (RDBMS) implements certain capabilities in a unique manner, it must be administered differently from the other systems with respect to those functionalities. This PeopleBook includes appendices that provide guidelines for administering the following supported platforms:

- Microsoft SQL Server.
- DB2 UDB for OS/390 and z/OS.
- DB2 UDB for Linux, Unix, and Windows.
- Informix.
- Oracle.
- Sybase.

Note. *DB2 UDB for OS/390 and z/OS* is the official IBM name for the DBMS. In the current PeopleTools release, PeopleSoft no longer supports the OS/390 operating system, only z/OS, its replacement. For the sake of brevity, this PeopleBook sometimes refers to DB2 UDB for OS/390 and z/OS as *DB2 z/OS*, and it sometimes refers to DB2 UDB for Linux, Unix, and Windows as *DB2 LUW*.

See Also

Appendix A, “Administering PeopleSoft Databases on Microsoft SQL Server,” page 171

Appendix B, “Administering PeopleSoft Databases on DB2 UDB for OS/390 and z/OS,” page 179

Appendix C, “Administering PeopleSoft Databases on DB2 UDB for Linux, Unix, and Windows,” page 203

Appendix D, “Administering PeopleSoft Databases on Informix,” page 219

Appendix E, “Administering PeopleSoft Databases on Oracle,” page 221

Appendix F, “Administering PeopleSoft Databases on Sybase,” page 239

Appendix G, “Configuring Remote Data Access,” page 245

Data Management Implementation

The functionality of data management for your PeopleSoft applications is delivered as part of the standard installation of PeopleTools, which is provided with all PeopleSoft products.

Several activities must be completed before you manage the data for your implementation:

1. Install your PeopleSoft application according to the installation guide for your database platform.

   See *PeopleSoft Installation Guide for your platform and product line.*

2. Establish a user profile that gives you access to PeopleSoft Application Designer and any other tools and processes that you’ll use.

   See *Enterprise PeopleTools 8.45 PeopleBook: Security Administration*.

Other Sources of Information

This section provides information to consider before you begin to manage your data. In addition to implementation considerations presented in this section, take advantage of all PeopleSoft sources of information, including the installation guides, release notes, and PeopleBooks.

See Also

“Data Management Preface,” page xxi

Enterprise PeopleTools 8.45 PeopleBook: Getting Started with Enterprise PeopleTools

CHAPTER 2

Using PeopleSoft Data Mover

This chapter provides overviews of PeopleSoft Data Mover, the PeopleSoft Data Mover interface, and PeopleSoft Data Mover commands, and discusses how to:

- Use the development environment.
- Create and run PeopleSoft Data Mover scripts.
- Use the Database Setup utility.
- Use the PeopleSoft Data Mover command-line interface.
- Use PeopleSoft Data Mover commands.
- Use PeopleSoft Data Mover command modifiers.
- Use SET parameters.
- Use script examples.

Note. PeopleSoft supports a number of versions of UNIX and Linux in addition to Microsoft Windows. Throughout this chapter, we make reference to operating system configuration requirements. Where necessary, this chapter refers to specific operating systems by name (eg Solaris, HP/UX, Linux, and so forth). However, for simplicity the word UNIX refers to all UNIX-like operating systems, including Linux.

Understanding PeopleSoft Data Mover

PeopleSoft Data Mover enables you to perform the following tasks:

- Transfer application data between PeopleSoft databases.
- Move PeopleSoft databases across operating systems and database platforms.
- Execute Structured Query Language (SQL) statements against any PeopleSoft database, regardless of the underlying operating system or database platform.
- Control database security and access.
- Create, edit, and run scripts.

  These scripts may include any combination of SQL commands and PeopleSoft Data Mover commands for exporting and importing data.

The PeopleSoft Data Mover development interface, which is a graphical user interface (GUI), runs only on Microsoft Windows. However, the Data Mover command-line interface, which is mainly intended for UNIX servers, runs on Windows and UNIX operating systems.

Note. PeopleSoft Data Mover runs in two-tier mode only. You must sign in to the database directly, not through an application server.

Understanding the PeopleSoft Data Mover Interface

This section discusses:

- PeopleSoft Data Mover startup.
- Operating modes.

Before you begin using PeopleSoft Data Mover to create and run database scripts, familiarize yourself with the PeopleSoft Data Mover interface.

PeopleSoft Data Mover Startup

There are two ways to start PeopleSoft Data Mover:

- Use the Data Mover shortcut in the PeopleSoft program group.

  Select Start, Programs, *PeopleSoft Group*, Data Mover. This access method only applies to the Microsoft Windows development environment.

- Use the command-line interface.

  You run PeopleSoft Data Mover in a console for Microsoft Windows and a telnet session for UNIX.

Operating Modes

Operating modes determine how you are connected to the database. PeopleSoft Data Mover modes are:

- Regular mode.

  Most of the time, you use regular mode. To sign in to regular mode, enter your PeopleSoft user ID and password during sign-in. In regular mode, all commands are valid.

- Bootstrap mode.

  In bootstrap mode, you use a database access ID and password when signing in. Typically, you use bootstrap mode for database loading, because no PeopleSoft security tables are established yet. You also use bootstrap mode for running some security commands, such as ENCRYPT_PASSWORD.

Note. In bootstrap mode, the following commands are not valid: EXPORT, RENAME, and REPLACE_VIEW.

Understanding PeopleSoft Data Mover Commands

This section discusses:

- PeopleSoft Data Mover commands.
- The Data Mover command matrix.
- PeopleSoft Data Mover COMMIT statements.

PeopleSoft Data Mover Commands

This section discusses the valid PeopleSoft Data Mover commands and SQL commands that you can include in PeopleSoft Data Mover scripts.

Data Mover Commands

PeopleSoft Data Mover commands are platform-independent and are unique to PeopleSoft Data Mover. You can use PeopleSoft Data Mover commands for importing, exporting, and other tasks, such as controlling the run environment, renaming fields and records, administering database security, and denoting comments

The following table describes the PeopleSoft Data Mover commands and the ways that you can indicate comments:

Command	Description
ENCRYPT_PASSWORD	Encrypt one or all user passwords (operator and access) defined in PSOPRDEFN for users.
EXPORT	Select record information and data from records and store the result set in a file. You can use the generated export file as input for migrating to another platform. This file is portable between ASCII and EBCDIC character sets, and also supports double-byte characters.
IMPORT	Insert data into tables using the information in an export file. If a tablespace or table does not exist, this command creates tablespace, table, and indexes for the record, using the information in the export file, and inserts the data.
REM, REMARK, and --	Indicate comment statements.
RENAME	Rename a PeopleSoft record, a field in one record, or a field in all records.
REPLACE_ALL	This is a variation of the IMPORT command. If a table already exists, use this command to drop the table and its indexes from the database and create the table and indexes using the information in the export file. Then, the command inserts data into the table using the information in the export file.
REPLACE_DATA	This is a variation of the IMPORT command. Delete data in existing tables and insert the corresponding data from the export file.
REPLACE_VIEW	Recreate specified views found in the database.
RUN	Run a specified .DMS file from within a PeopleSoft Data Mover script. The file cannot contain nested RUN commands.
SET	When a command is followed by valid SET parameters, it forms a statement that establishes the conditions under which PeopleSoft Data Mover runs the PeopleSoft Data Mover and SQL commands that follow.
SWAP_BASE_LANGUAGE	Swap all the language tables from PSRECDEFN.

Command	Description
SET IGNORE_ERRORS	(Optional) If this command is set, then all errors during the swap base language are ignored. Otherwise, the system stops on errors.
SET BASE_LANGUAGE	Swap individual language tables. You should swap individual table only when there is an error with any of the table after the SWAP_BASE_LANGUAGE command.

SQL Commands

With PeopleSoft Data Mover, you can use supported SQL commands in scripts on any supported database platform. Except where noted in the following discussion regarding standard SQL commands, you can use all the supported SQL commands with the following PeopleSoft Data Mover SET statements:

- SET LOG
- SET NO COMMIT
- SET NO TRACE

Standard SQL Commands with DMS Scripts

PeopleSoft Data Mover supports the following standard SQL commands:

- ALTER
- COMMIT
- CREATE
- DELETE
- DROP

Note. With DROP commands, any drop errors are ignored. The script continues, but the errors are reported in the log.

- GRANT
- INSERT

Important! INSERT cannot be used with SET NO COMMIT or SET NO TRACE.

- ROLLBACK
- UPDATE

Warning! PeopleSoft Data Mover does not support SELECT statements, because they require a SQL FETCH function.

Standard SQL Commands with SQL Files

PeopleSoft Data Mover supports all SQL commands and other database-specific function calls that are supported by the database engine.

Note. PeopleSoft Data Mover runs only files with the extension *.SQL*.

Nonstandard SQL Commands

With PeopleSoft Data Mover, you can also use the following nonstandard SQL commands created by PeopleSoft: STORE and ERASE. Use the commands to change COBOL SQL statements in PS_SQLSTMT_TBL.

Note. In previous versions, PeopleSoft Data Mover supported a command called PSCOPY, which was an Oracle-specific command for altering records that have a long field in an INSERT statement subquery. Now, you should run Oracle scripts generated by the Build feature in PeopleSoft Application Designer through SQL*Plus.

The STORE command first deletes the existing stored statement from PS_SQLSTMT_TBL, and then inserts the new statement using the following syntax:

```
STORE progname_type_stmtname
```

For example:

```
STORE PTPEMAIN_S_MSGSEQ
SELECT MAX (MESSAGE_SEQ), PROCESS_INSTANCE
  FROM PS_MESSAGE_LOG
  WHERE PROCESS_INSTANCE = :1
  GROUP BY PROCESS_INSTANCE
;
```

The ERASE command deletes one or all stored statements from PS_SQLSTMT_TBL. When deleting a single statement, you use the *progname_type_stmtname* format as shown for STORE. For example:

```
ERASE PTPEMAIN_S_MSGSEQ;
```

When deleting all SQL statements for a particular program, you include only the program name in the command line format. For example:

```
ERASE PTPEMAIN;
```

See Also

Chapter 2, “Using PeopleSoft Data Mover,” Using PeopleSoft Data Mover Commands, page 23

Chapter 2, “Using PeopleSoft Data Mover,” Using PeopleSoft Data Mover Command Modifiers, page 33

Chapter 2, “Using PeopleSoft Data Mover,” Using SET Parameters, page 36

The Data Mover Command Matrix

The following table shows the relationship between SQL and PeopleSoft Data Mover commands. *DDL* refers to data definition commands, which define the structure of a database. *DML* refers to data manipulation commands which define the contents of a database.

Function	Command Type	Supported SQL Commands	Data Mover Commands
Create tables, tablespaces, and indexes.	DDL	CREATE	IMPORT and REPLACE_ALL
Create views.	DDL	CREATE	REPLACE_VIEW
Drop tables.	DDL	DROP	REPLACE_ALL
Modify tables.	DDL	ALTER	None
Modify PeopleSoft records.	DDL	None	RENAME
Insert rows.	DML	INSERT and STORE	IMPORT, REPLACE_ ALL, and REPLACE_ DATA
Delete rows.	DML	DELETE and ERASE	REPLACE_DATA
Update rows.	DML	UPDATE	None
Encrypt rows.	DML	None	ENCRYPT_PASSWORD
Select rows.	Query	None	EXPORT
Save or don't save changes.	Transaction	COMMIT and ROLLBACK	SET (when used with COMMIT or NO COMMIT)
Control or run other PeopleSoft Data Mover commands.	Environment	None	SET and RUN
Denote an explanatory statement.	Comment	None	REM, REMARK, and --

PeopleSoft Data Mover COMMIT Statements

PeopleSoft Data Mover issues COMMIT statements after most successful SQL commands, except for EXPORT and IMPORT. For EXPORT and IMPORT, PeopleSoft Data Mover issues a COMMIT after each record. With IMPORT, a SET COMMIT *n* command performs a COMMIT after the system inserts every *n* rows.

If you are executing native SQL in PeopleSoft Data Mover, and no COMMIT statements exist in the SQL script, PeopleSoft Data Mover issues a COMMIT after each successful SQL statement. For example, if you run a PeopleSoft Data Mover script that contains three update commands, and the third command fails, the first and second update commands are committed, but the third command is not.

Using the Development Environment

This section discusses how to:

- Sign in to the development environment.
- Navigate the Data Mover window.

Signing In to the Development Environment

To start PeopleSoft Data Mover in the Windows development environment:

1. Select Start, Programs, PeopleSoft Group, Data Mover.

   If you don't have a PeopleSoft Data Mover shortcut, you can add one to the desktop. The executable to launch is: %PS_HOME%\bin\client\winx86\psdmt.exe

2. Sign in using the appropriate ID and password.

   In regular mode, use a user ID and password. In bootstrap mode, use a system access ID and access password, such as SYSADM.

Navigating the Data Mover Window

The PeopleSoft Data Mover interface consists of two horizontal panes: an input pane and an output pane. The input pane is on top.

The status bar at the bottom of the window provides the following information:

- Database name (for example, QEDMO, PT840HR, and so on).
- Database type (for example, Oracle, Sybase, and so on).
- Operating mode (regular or bootstrap).
- Trace status (on or off).

The input pane displays the script that you open. In this pane, you view and edit PeopleSoft Data Mover scripts.

The output window displays the results after running a script. If you encounter any errors, the output window shows where the script failed. In a multidatabase environment, always check the information at the top of the output to ensure that you run the script against the appropriate database. Specifically, verify the information on the Database line.

Note. By default, the results in the output window are saved to the file DATAMOVE.LOG, which is written to the default log directory as specified in PeopleSoft Configuration Manager (on the Profile, Common tab). You can specify a different file name.

The status of the SQL Trace utility appears on the right-hand end of the status bar. Use PeopleSoft Data Mover with tracing turned off. There are several ways to disable the SQL Trace utility (for the Microsoft Windows environment) before starting PeopleSoft Data Mover. You can use:

- PeopleSoft Configuration Manager.
- PeopleTools options.
- A Data Mover command (NO TRACE).

The operating mode on the status bar indicates either regular mode or bootstrap mode. If you connect to the database in regular mode, the operating mode status is blank. The operating mode is bootstrap if you sign in using the access ID and password.

Note. Verify the mode that you are using. Most commands require regular mode to run successfully.

See Also

Chapter 2, "Using PeopleSoft Data Mover," Using SET Parameters, page 36

Chapter 2, "Using PeopleSoft Data Mover," NO TRACE, page 42

Chapter 2, "Using PeopleSoft Data Mover," Operating Modes, page 6

Creating and Running PeopleSoft Data Mover Scripts

This section provides overviews of command types and syntax rules and discusses how to:

- Create and edit scripts.
- Prepare to run export scripts.
- Run scripts.

Understanding Command Types

A PeopleSoft Data Mover script can contain two types of commands:

- Data Mover commands.

  Use these commands to export and import database information and to otherwise modify the database. PeopleSoft Data Mover commands also control script execution, call other PeopleSoft Data Mover files, and indicate comments.

- SQL commands.

  You can use both standard and nonstandard SQL commands that modify the database.

Understanding Syntax Rules

To create or edit PeopleSoft Data Mover scripts, follow these syntax rules to ensure that the commands run successfully.

Delimiters

With the exception of double-hyphen (--) comment statements, every command statement must be followed by a delimiter.

Valid delimiters are:

- Semicolon (;)

  A semicolon can appear on the same line as the command itself, or by itself on the line immediately following a command statement. For example, the following two examples of the semicolon delimiter are valid:

```
SET OUTPUT c:\temp\abc.dat;
 SET LOG c:\temp\new.log
 ;
```

- Forward slash (/)

  This delimiter can be used only on a line by itself, in column 1, on a line immediately following a command statement. For example:

```
IMPORT *
 /
```

Multiline Statements

With the exception of double-hyphen (--) comment statements, statements can span multiple lines. For example:

```
EXPORT absence_hist
   WHERE absence_type = 'A';
```

Multiline Comments

A double-hyphen (--) comment statement does not require a delimiter termination. However, each statement can't span more than one line. Be sure to add a space after the double hyphen before you start the comment. For example:

Correct:

```
-- This script imports the information stored in
-- the ABC.DAT file.
```

Incorrect:

```
--This script imports the information stored in
the ABC.DAT file.
```

White Space

Command statements can contain any amount of white space between items.

Case Sensitivity

Statement text is not case-sensitive. For example,

```
IMPORT *
```

is equivalent to

```
import *
```

String Constants

String constants are case-sensitive and must be surrounded by single quotation marks. For example, `'ABC'` is treated differently than `'Abc'` or `'abc'`.

Record Names and Table Names

In PeopleSoft Data Mover, when a record name needs to be specified as one of the elements in the command statement syntax, as in an IMPORT statement, you can specify either the record name or the corresponding table name. For example, the following IMPORT statements are equivalent:

Correct:

```
IMPORT job;
```

Correct:

```
IMPORT ps_job;
```

However, when a table name is required for one of the elements in the command statement syntax, you must use the table name, not the record name. For example:

Correct:

```
IMPORT job  AS  ps_process;
```

Incorrect:

```
IMPORT job  AS  process;
```

Creating and Editing Scripts

When you use PeopleSoft Data Mover to manipulate the information in a database, you can either write a new script or open and edit an existing script that is similar to the one that you want to create.

The default file extension for scripts is *.DMS*, which stands for *Data Mover script.*

Creating a New Script

To create a new script:

1. Select File, New.

   When you first launch PeopleSoft Data Mover, a new file appears automatically. .

2. Enter the script text (that is, the code) in the input pane, which appears on top.

   Using proper Data Mover syntax, enter the command statements that you want the script to run.

3. Save the script.

   Select File, Save. In the Save As dialog box, select the Save as Unicode check box (if appropriate) and click Save.

Editing an Existing Script

To edit an existing PeopleSoft Data Mover script:

1. Select File, Open.
2. Select the file and click OK.

   By default, you view only .DMS files. You can select *All Files* from the Files of type drop-down list box to view all file types. After you open a script, it appears in the PeopleSoft Data Mover input pane.

3. Modify the script.

   If the file that you opened is not a .DMS file, verify that it conforms to the required syntax rules and that it doesn't contain unsupported SQL commands.

4. Save the script with a new name.

   Select File, Save As.

   In the Save As dialog box, enter a file name, select the Save as Unicode check box (if appropriate) and click Save.

   If the script is edited in Unicode format, then the default save is Unicode. However, if the file is opened in ASCII format, then the default is ASCII.

See Also

Chapter 2, "Using PeopleSoft Data Mover," Understanding PeopleSoft Data Mover Commands, page 6

Chapter 2, "Using PeopleSoft Data Mover," Understanding Syntax Rules, page 12

Preparing to Run Export Scripts

Before running a PeopleSoft Data Mover export script, you must first prepare the database.

To prepare for an export:

1. Load DDL model information by running all DDL*.DMS files through PeopleSoft Data Mover.
2. To change the DDL model information, use the DDL Model Defaults utility in PeopleTools Utilities.
3. Run DDDAUDIT.SQR and fix any errors that it finds.
4. Run SYSAUDIT.SQR and fix any errors that it finds.
5. Use the SQL Alter function in PeopleSoft Application Designer to alter all tables.

   Either let the files alter in place or run the script that it generates to alter any tables that it marks as out of synchronization.

6. Use the SQL Create function in PeopleSoft Application Designer to create all records, using the *If table exists ... Never recreate* option.

See Also

Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration, "Using PeopleTools Utilities"

Chapter 4, "Ensuring Data Integrity," page 63

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Application Designer

Running Scripts

Through PeopleSoft Data Mover, you can run DDL, DML, and SQL scripts created with the following tools:

- PeopleSoft Data Mover (DMS scripts).
- Build SQL functionality in PeopleSoft Application Designer (SQL scripts).
- Platform-specific SQL utilities (SQL scripts).

Note. You can also schedule PeopleSoft Data Mover scripts using PeopleSoft Process Scheduler. This can be useful in scheduling audit routines or extracting data from the PeopleSoft database. Additionally, logs and data files generated by PeopleSoft Data Mover can be posted to the report repository in PeopleSoft Process Scheduler so that they can be viewed either through Process Monitor or Report Manager.

When running scripts through PeopleSoft Data Mover, keep the following items in mind:

- Turn off the SQL Trace utility to run PeopleSoft Data Mover scripts.

  If SQL Trace is enabled, disable it on the Trace tab in PeopleSoft Configuration Manager before you run the script. You can also enter the SET NO TRACE statement within scripts. This disables SQL Trace for the DMS script even if it is enabled in PeopleSoft Configuration Manager.

- Records defined using the PeopleSoft Data Mover EXPORT and IMPORT commands can have a maximum of 500 columns, and they can have multiple long columns within the limitations for long columns set by the database platform.

  Check with the database vendor for restrictions on the number of long columns allowed for the platform.

- On DB2 UDB platforms, locks can occur on system catalogs.

  Do not run unattended PeopleSoft Data Mover sessions. Close the session as soon as all scripts terminate.

- To run a SQL script, you must open it by selecting File, Open so that the SQL runs properly.

  Do not copy and paste SQL from another source into PeopleSoft Data Mover.

  Note. If you plan to import or export files greater than 2 gigabytes (GB) on UNIX, you must enable large file support at the operating system level.

To run a script:

1. Select File, Open.
2. Select one of the following types of script to run.
   - PeopleSoft Data Mover files (.DMS).

     These are the files created using PeopleSoft Data Mover.

   - Query files (.SQL).

     These are the files created using the Build SQL functionality in PeopleSoft Application Designer or using a query tool specific to a relational database management system (RDBMS), such as PL/SQL on Oracle.

   - All files.

     Select to view all available files in a directory. Only .DMS and .SQL files are valid file types for PeopleSoft Data Mover.

     Note. SELECT commands are not supported. When performing upgrades, use the SQL utility for the platforms to run .SQL scripts, not PeopleSoft Data Mover.

3. Select File, Run.

   You can monitor the script's progress in the output pane, which reveals any error messages and displays the message *Script Completed* when processing has ended.

Using the Database Setup Utility

Typically, you use the Database Setup utility during PeopleSoft installations and upgrades, not on a daily basis. You use this utility to create PeopleSoft Data Mover import scripts that load data into a PeopleSoft database.

This section discusses how to:

- Access the Database Setup utility.
- Use the Database Setup utility.
- Check the generated script.

Note. If you are performing an installation, use the documentation included in your PeopleSoft installation guide, which provide specific details regarding your applications, languages, and RDBMS. This section provides a general overview and is not specific to the installation procedure.

Accessing the Database Setup Utility

To access the Database Setup utility:

1. Sign in to PeopleSoft Data Mover in bootstrap mode.

   Use the access ID and password rather than your PeopleSoft user ID and password.

2. Select File, Database Setup.

   Note. If you sign in to PeopleSoft Data Mover using regular mode, not bootstrap mode, the Database Setup menu item is not available.

Using the Database Setup Utility

This section discusses the screens that make up the utility.

Database Setup

Select Target Database Select the RDBMS against which to run the database setup script. For instance, if the database that you are creating will run on an Oracle server, select *Oracle.*

Database Type PeopleSoft supports non-Unicode (ANSI) and Unicode database types. Select the appropriate type for the system. For some RDBMS types, Unicode is not available.

Select Character Set Select a character sets. Your choices vary depending on the database type that you selected.

Select PeopleSoft Application

PeopleSoft Application Only the applications that you have licensed appear. Select the applications for which you want to create PeopleSoft Data Mover scripts. To add applications selectively, use the Add button. To add all applications available, use the Add All button.

Data Mover Scripts to Create Use theRemove button to remove a single application, or use the Remove All button to clear the list box.

Database Type Specify what the result of running the script should be. There are two database codes: *PT* for PeopleTools and *EP* for PeopleSoft applications. Options are:

- *Demo:* Select to create a demonstration database.
- *System:* Select to create a system database.
- *Add New Language:* Select to add support of new languages to an existing database.
- *Add New Product:* Select to add a new PeopleSoft product to the current system. With this option selected, only non-PT database codes appear.

Database Parameters

Database Name Enter the name of the database against which to run the script. The database name that appears is the database to which you are currently signed on. If the script that you are creating will be run against another database, specify the appropriate name here. If you generate a script for a database other than the current database, the system uses a default database using the following convention: *XX*DMO for demonstration databases and *XX*SYS for system databases. The *XX* represents the product code, such as HR.

Symbolic ID Enter the ID used as the key to retrieve the access ID and access password. For initial installation, set it equal to the database name.

Access ID This ID is the RDBMS ID with which PeopleSoft applications are ultimately connected to the database once the PeopleSoft system validates the user or connect ID. It typically has all the RDBMS privileges necessary to access and manipulate data for an entire PeopleSoft application.

Access Password Enter the password associated with the access ID.

Connect ID This ID is used for the initial connection to the database. Any two-tier connection requires a connect ID. A connect ID is a valid user ID, that when used during logon, sign-in takes the place of PeopleSoft user IDs for the sign-in process.

Table Owner (DB2 UDB for OS/390 and z/OS) This field populates the CREATOR field in the system catalog table SYSIBM.SYSTABLES. You determine the name of the table owner ID during the initial installation.

Index Storage Group (DB2 UDB for OS/390 and z/OS) Enter the storage group where the index spaces are created.

Table Space Storage Group (DB2 UDB for OS/390 and z/OS) Enter the storage group for tablespaces. This value must be the same as that used in the XXDDL.SQL script when you create tablespaces during the installation.

Checking the Generated Script

After running the Database Setup utility, check the output directory for the generated script. Some commands are added that call other scripts and perform various functions. These commands are added to reduce the number of scripts and commands that you must run manually. For example, note that the following commands appear at the end of the script:

- REPLACE_VIEW

This command creates views for the new database.

- CREATE_TEMP_TABLE

  This command creates any necessary temporary table images. The number of temporary tables is determined by the value for the Temp Table Instances setting in PeopleTools options (Utilities, Administration, PeopleTools Options) plus the number of PeopleSoft Application Engine temporary tables.

- SWAP_BASE_LANGUAGE

  If you selected a base language other than English, this command modifies the system to recognize that language as the base language. The default PeopleTools language is English if the PSSTATUS table is not available.

- RUN

  This command runs the CURR*XXX*.DMS script to load the system with the appropriate currency information, and it runs MSGTL*XXX*.DMS to load the system with the appropriate PeopleTools messages (error and informational messages). The *XXX* represents the language code, such as FRA for French. The system runs these scripts only if you have selected a base language other than English.

Note. After each DDL create table, import data, and DDL create indexes command, PeopleSoft Data Mover issues an UPDATE STATISTICS command (except on z/OS), which improves the performance of subsequent commands, such as the REPLACE_VIEW command.

See Also

Chapter 2, “Using PeopleSoft Data Mover,” CREATE_TEMP_TABLE, page 24

Chapter 2, “Using PeopleSoft Data Mover,” SWAP_BASE_LANGUAGE, page 31

Using the PeopleSoft Data Mover Command-Line Interface

This section provides an overview of the PeopleSoft Data Mover command-line interface and discusses how to:

- Set up UNIX to run PeopleSoft Data Mover.
- Set up tracing.
- Run Data Mover scripts from the command line.

Understanding the PeopleSoft Data Mover Command-Line Interface

The PeopleSoft Data Mover command-line interface enables you to run PeopleSoft Data Mover scripts from the command line in UNIX and Microsoft Windows environments. The command-line interface is designed only for running scripts, not creating and editing scripts. In Microsoft Windows, you create and edit scripts using the PeopleSoft Data Mover development environment. In UNIX, you can use any supported text editor.

When using the command-line interface, the results of the script run appear in the command-line window, much like the contents of the output pane in the PeopleSoft Data Mover GUI. The system also writes this information to the log file.

The PeopleSoft Data Mover command line supports $PS_HOME on UNIX and %PS_HOME% on Windows.

Note. Although the command-line interface also runs on Windows machines, this documentation primarily discusses UNIX.

Important! The PeopleSoft Data Mover command line on UNIX is intended to increase performance with large database loads during installation. Use the PeopleSoft Data Mover Windows interface for other types of scripts.

Setting Up UNIX to Run PeopleSoft Data Mover

Before running the PeopleSoft Data Mover command-line interface on UNIX, verify that BEA Tuxedo is installed. Tuxedo is required for PeopleSoft Data Mover to run on UNIX.

Next, configure the psconfig.sh shell script to set the UNIX and PeopleTools environment variables properly for Data Mover, then run the script. You must run it from the *PS_HOME* directory.

Note. The UNIX environment requires certain platform-specific environment variables. These variables are set automatically, but you can reconfigure the psconfig.sh script file to change their values.

UNIX Environment Variables

When you run psconfig.sh, several environment variables are automatically set to default values that reflect a standard PeopleSoft Data Mover install.

To modify them, you must edit psconfig.sh or manually change the environment. For example, to change the PeopleSoft Data Mover environment variables, modify the following statements:

- export PS_DM_DATA=*data_path*

  $PS_DM_DATA specifies the directory where PeopleSoft Data Mover searches for input data (.dat) files. The default setting is $PS_HOME/data.

- export PS_DM_SCRIPT=*script_path*

  $PS_DM_SCRIPT specifies the location of the PeopleSoft Data Mover script files. The default setting is $PS_HOME/scripts.

- export PS_DM_LOG=*log_path*

  $PS_DM_LOG specifies the location of PeopleSoft Data Mover log files. The default is $PS_HOME/log.

Note. If you want to perform tracing under UNIX, you must set additional environment variables.

See Chapter 2, “Using PeopleSoft Data Mover,” Setting Up Tracing, page 20.

Setting Up Tracing

Two environment variables are required to enable tracing for Data Mover. Edit psconfig.sh to include the following statements:

- export PS_SERVER_CFG=$PS_HOME/setup/psdmtx.cfg

  $PS_SERVER_CFG specifies the location and name of the psdmtx.cfg file, which contains parameters for tracing and character set.

- export PS_SERVDIR=$PS_HOME/log

  $PS_SERVDIR points to the output log directory.

To set up tracing in PeopleSoft Data Mover, you must edit the psdmtx.cfg file to specify the appropriate tracing behavior.

To set a specific trace, edit the psdmtx.cfg file to set the trace bitfield in the file. After running PeopleSoft Data Mover, the trace file is located in *PS_HOME*/log/LOGS/AE__0 directory.

Use the *TraceSq1* bitfield parameter to set the level of SQL trace by adding together the numerical values that represent each degree of tracing required. The values are defined as follows:

Bit Value	Type of Tracing
1	SQL statements.
2	SQL statement variables.
4	SQL connect, disconnect, commit and rollback.
8	Row Fetch (indicates that it occurred, not data).
16	All other API calls except ssb.
32	Set Select Buffers (identifies the attributes of columns to be selected).
64	Database API specific calls.
128	COBOL statement timings.
256	Sybase Bind information.
512	Sybase Fetch information.
4096	Manager information.
8192	Mapcore information.

For example, if you want to trace Sybase bind and fetch information, enter:

```
TraceSql=768
```

Running Data Mover Scripts from the Command Line

The PeopleSoft Data Mover command line program is:

- On Microsoft Windows systems — *PS_HOME*\bin\client\winx86\psdmtx.exe
- On UNIX systems — *PS_HOME*/bin/psdmtx

At the command line, navigate to the location of the program. Use the following syntax to run PeopleSoft Data Mover scripts:

```
psdmtx -CT dbtype [-CS server] -CD database_name -CO user_ID -CP user_password⇒
 [-CI connect_ID -CW connect_password] -FP dms_filepath
```

Or, if your command line parameters are stored in a text file, use the following syntax:

```
psdmtx parm_filepath
```

Or, to display a listing of all the command-line arguments and their descriptions at the command prompt enter:

```
psdmtx /help
```

The value of each parameter follows the parameter name, separated by zero or more spaces. It doesn't need to have quotation marks around it, even if it has internal spaces — the system treats all text following the parameter name as part of the value, up to the next parameter or the end of the command line.

Note. You must enclose a value in quotation marks only when it includes a hyphen or forward slash, or to include leading or trailing spaces. If the value itself includes a quotation mark character, precede the double quote with a backslash (\).

Command Line Parameters

The following table lists the command-line parameters available to pass to psdmtx for running Data Mover scripts:

Parameter	Value	Example
-CT	Specify the database type. Valid values are ORACLE, INFORMIX, SYBASE, MICROSFT, DB2ODBC, and DB2UNIX. **Note.** Notice the spelling of MICROSFT. DB2ODBC is the dtaabase type for DB2 z/OS.	`-CT ORACLE`
-CS	Specify the name of the database server for the database to which you're connecting. **Note.** This parameter is required only if you specify INFORMIX or SYBASE as the database type.	`-CS pt-sun05`
-CD	Specify the name of the database to connect to, as you would when signing in to PeopleSoft.	`-CD HR844DMO`
-CO	Specify the PeopleSoft user ID you're using to sign in.	`-CO TOM2`
-CP	Specify the user password for the PeopleSoft user ID you specified.	`-CP SAWYER2`
-CI	Specify the connect ID used to connect to the database server. **Note.** This parameter is required only if you're running PeopleSoft Data Mover in regular mode.	`-CI people`
-CW	Specify the password for the Connect ID you specified. **Note.** This parameter is required only if you're running PeopleSoft Data Mover in regular mode.	`-CW peop1e`
-FP	Specify the file name and path of the PeopleSoft Data Mover script to run.	`-FP $PS_HOME/scripts/test.dms`
/help	No value required.	

Following is an example of a psdmtx command line on a UNIX system:

```
psdmtx -CT DB2UNIX -CD FS845A1 -CO PSOFT -CP PSOFT⇒
 -CI people -CW peop1e -FP fs845a1dbo.dms
```

Using a Parameter File

Rather than submitting parameters manually on the command line, you can have PeopleSoft Data Mover read a file that contains appropriate parameters. Create a text file that contains a complete set of parameters as you would enter them on the command line.

PeopleSoft provides a sample parameter file (or parm file) to use as a template. You can find it in the *PS_HOME*/setup directory.

If you submit a parameter file name to PeopleSoft Data Mover as the first parameter in the command line, PeopleSoft Data Mover reads the contents of the file and interprets the contents as parameters entered on the command line. For example:

```
psdmtx $PS_HOME/setup/myparmfile.txt
```

Note. You must enter the full path to the parameter file.

Warning! For security reasons, after PeopleSoft Data Mover interprets the contents, it immediately deletes the parameter file.

Using PeopleSoft Data Mover Commands

This section provides the details of syntax and use for each of the PeopleSoft Data Mover commands. This section also discusses PeopleSoft Data Mover command modifiers, such as AS, WHERE, and IGNORE_DUPS, which can be used to modify certain commands.

CHANGE_ACCESS_PASSWORD

Syntax

```
CHANGE_ACCESS_PASSWORD <SymbolicID> <newAccessPswd>
```

Description

Use this command to reset the access password and make it transparent to users.

The CHANGE_ACCESS_PASSWORD command performs the following operations:

- Selects the ACCESSPSWD field from PSACCESSPRFL for the specified symbolic ID.
- Changes the access ID's database password to the new access password that you specify (for Oracle, Sybase and Microsoft SQL Server only).
- Updates PSACCESSPRFL for the specified symbolic ID with the new access password.

Parameters

LOG and NO TRACE

CREATE_TEMP_TABLE

Syntax

```
CREATE_TEMP_TABLE {record | *}
```

Description

Creates temporary table images for use with PeopleSoft Application Engine programs. To customize the number of temporary tables, you need to modify the PeopleTools Options page or updated the PSOPTIONS table using the following SQL:

```
UPDATE PSOPTIONS SET TEMPTBLINSTANCES = <#>
```

You also need to review the number of temporary tables allotted for PeopleSoft Application Engine programs

---

Note. For security reasons, this command is disabled for z/OS.DMS scripts generated by the Database Setup utility.

CREATE_TRIGGER

Syntax

```
CREATE_TRIGGER {* | <RECNAME>
```

Description

Creates database triggers on the specified table.

Note. If you use CREATE_TRIGGER in bootstrap mode, the system automatically activates SET IGNORE ERROR. This enables PeopleSoft Data Mover to continue processing until all of the view definitions have been processed, and all errors have been written to the current .LOG file (or an error log file). This is similar to the REPLACE_VIEW behavior.

ENCRYPT_PASSWORD

Syntax

```
ENCRYPT_PASSWORD {userID | *};
```

Description

Encrypts one or all user passwords (user passwords and access passwords). When encrypting a single user's password, the user ID must be present in PSOPRDEFN. You can use an asterisk instead of a name to encrypt all passwords in PSOPRDEFN.

Parameters

LOG, NO COMMIT, and NO TRACE.

Example

Here's an example of how to encrypt a single user password (FS) already listed in PSOPRDEFN:

```
ENCRYPT_PASSWORD FS;
```

To encrypt all user passwords in PSOPRDEFN, enter:

```
ENCRYPT_PASSWORD *;
```

EXPORT

Syntax

```
EXPORT {record | *} [WHERE condition(s)];
```

Description

Creates a single export file containing the specified database contents. The result set can contain any of the following: a single PeopleSoft record, a group of records, or the entire database. You can use the export file as input for the PeopleSoft Data Mover IMPORT command to migrate the data within the platform or to another platform.

Note. This command is not available in bootstrap mode.

Records exported using EXPORT can have a maximum of 500 total columns and multiple long columns within the limitations for long columns set by the database platform. Check with the database vendor for restrictions on the number of long columns allowed for the platform.

When you export all records using EXPORT*, PeopleSoft Data Mover orders the records alphabetically (with the exception of PSLOCK, which is the last record exported). After each record, PeopleSoft Data Mover indicates how many records remain. After all the tables are exported, then the views are exported.

Warning! The WHERE clause, when used in this command, supports only US-ASCII (seven-bit ASCII) values. Characters beyond this range can produce errors in the export file.

Parameters

LOG, NO COMMIT, NO DATA, NO TRACE, NO VIEW, and OUTPUT.

Note. SET NO VIEW is only valid with EXPORT *.

Note. If SET OUTPUT is not used, PeopleSoft Data Mover writes to the default file name, DATAMOVE.DAT.

Example

To export a single record, use an EXPORT command for the specific record. For example:

```
EXPORT PS_JOB;
```

Note. When specifying a particular record in the EXPORT command (as shown in the previous example), the specified record must be a table, not a view.

To export all PeopleSoft records, including views, enter

```
EXPORT *;
```

See Also

Using PeopleSoft Data Mover Command Modifiers, WHERE

IMPORT

Syntax

```
IMPORT {record | *} [IGNORE_DUPS]
  [AS new_table_name];
```

Description

Creates database spaces, create nonexisting records and indexes, and appends non-duplicate rows to records. In addition, creates views if the export file was created using EXPORT * and imported using IMPORT *.

Warning! All duplicate row-checking depends on the existence of a unique index. If no unique indexes are created before loading the data, there is a potential for duplicate data.

In the IMPORT statement, the AS clause is only valid if you specify a particular record; it is not valid and should not be used with IMPORT*. Also, the table name that you specify immediately after the AS command modifier must not exceed 18 characters (including the *ps_* prefix). If you do specify a table_name that exceeds 18 characters, the following error appears: *Error: Unable to process create statement.*

Records defined using IMPORT can have a maximum of 500 columns and multiple long columns within the limitations for long columns set by the database platform. Check with the database vendor for restrictions on the number of long columns allowed for the platform. There are also two variations of IMPORT that you can use: REPLACE_ALL and REPLACE_DATA.

Parameters

All except OUTPUT. INPUT is a required parameter.

Note. IGNORE_DUPS is only valid in bootstrap mode.

Example

To import a single record from an export file, use an IMPORT command for that record. For example:

```
SET INPUT file_name;
IMPORT PS_JOB;
```

To import all PeopleSoft records from an export file, including views, enter:

```
SET INPUT file_name;
IMPORT *;
```

Globalization Considerations

In previous releases, PeopleSoft Data Mover required multiple DAT files for base and nonbase languages. …PeopleSoft Data Mover now offers a base-language-independent method for moving application data between databases. PeopleSoft Data Mover loads a single DAT file, detects the target database base language, and inserts the data into the correct base or related language table.

If PeopleSoft provides a software fix, you don’t need to swap the base language before importing it into a database with a different base language. For example, suppose that a fix is sent with the base language English (ENG) and the related language Japanese (JAP). In this case, you can import this file directly into a database where the base language is JAP and the related language is ENG.

Upon EXPORT, the system adds the LANGUAGE_CD (language code) to the generated DAT file. For example:

```
SET BASE_LANGUAGE ENG
```

Then, when you use the IMPORT command to import the generated DAT file, the system detects the LANGUAGE_CD in the DAT file and compares it with the LANGUAGE_CD in the target database to determine how to swap the base language and related language tables.

Note. Base language is the database base language. It can be any PeopleSoft-supported language.

Consider the following points when running the IMPORT command:

- This feature is enabled whenever you import a DAT file.
- Running the IMPORT command may have an unavoidable adverse affect on performance.

See Also

Chapter 2, "Using PeopleSoft Data Mover," REPLACE_ALL, page 29
Chapter 2, "Using PeopleSoft Data Mover," REPLACE_DATA, page 30

REM, REMARK, and – –

Syntax

```
REM comments;
REMARK comments;
-- comments
```

Description

Each of these three command variations indicates explanatory text in a PeopleSoft Data Mover script.

Example

Here are three examples demonstrating the use of each:

```
REM  This example demonstrates the use of the REM command to set off script⇒
 comments.  These statements can span multiple lines and must be terminated with a⇒
 valid delimiter;
REMARK  The REMARK command variation has the same restrictions as REM
/
--  This example demonstrates the use of two dashes to denote script
--  comments.  No delimiters are required, but statements can not
--  exceed one line without using another double-dash.
```

When using a double hyphen (--), as in the third example, you need at least one space after the double hyphen, before the start of the actual text of the comment. Otherwise, you receive a syntax error.

When used in conjunction with a comment prefixed by REM or REMARK, the forward-slash delimiter (/) should be *by itself* on the last line of that comment. In such cases, instead of using a forward-slash (/), you can also use a semicolon (;) by itself on this last line. The forward slash (/) can also be used by itself without a REM or REMARK statement, in lieu of blank lines, which are also allowed in a script.

RENAME

Syntax

```
RENAME {RECORD record | FIELD {field | record.field}} AS new_name;
```

Description

Renames a PeopleSoft record, a field in one record, or a field in all records.

Note. This command is not available in bootstrap mode.

Warning! Using RENAME only modifies an object in the PeopleSoft tables. To write the change to the system tables, you must either use PeopleSoft Application Designer to alter the affected tables (for record and field renames), or you must run TLSCOPY.SQR (for recfield renames.)

To rename a recfield, you must qualify the original name of the field with the record name. If you don't qualify the record name, PeopleSoft Data Mover attempts to globally change the field name in all records.

Renaming a record field is only possible through PeopleSoft Data Mover.

To rename a record field:

1. Perform the rename in PeopleSoft Data Mover.

   For example:

   ```
   RENAME FIELD RECORD.FIELD AS NEWFIELD; COMMIT;
   ```

2. In PeopleSoft Application Designer, create a project that includes the record that contains the field that you renamed, and save the project.

   In the case of a subrecord field rename, the subrecord along with all tables that contain that subrecord must be inserted into the project.

3. Select Build, Settings.
   - Select the Alter tab.
   - Select Adds and Renames.
   - Clear Changes and Deletes.

   Note. Drop column and change column length do not apply.

   - Select the Scripts tab and select output settings.
   - Specify an output file and click OK.
4. Select Build, Project.
   - Select Alter Tables (Create Indexes is automatically selected).
   - Click Build.
   - Click Yes to continue the build process.
5. Run the generated SQL script using the query tool.

   This adds the new field to the tables within the project.

To remove the old field from the tables:

1. In PeopleSoft Application Designer, open the project that you created using the preceding steps.
   - Select Build, Settings.
   - Select the Alter tab.

- Select Drop column if data present.
- Select Deletes.
- Clear Adds and Renames.
- Select the Scripts tab.
- Give the output file a different name and click OK.

2. Select Build, Project.
   - Select Alter Tables (Create Indexes is automatically selected).
   - Click Build.
   - Click Yes to continue the build process.
3. Run the generated SQL script using the query tool.

   The old field should no longer appear on the tables included in the project.

Parameters

LOG, NO COMMIT, and NO TRACE.

Example

Here's an example of how to rename a record:

```
RENAME RECORD absence_hist AS absent_hist;
```

Here's an example of how to globally rename a field:

```
RENAME FIELD effdt AS currdate;
```

Here's an example of how to rename a recfield:

```
RENAME FIELD course_tbl.duration_days AS duration_d;
```

REPLACE_ALL

Syntax

```
REPLACE_ALL {record | *}
  [AS new_table_name];
```

Description

This is a variation of the IMPORT command. If a table already exists, use this command to drop the table and its indexes from the database and create the tables and indexes using the information in the export file. Then, the command inserts data into the table using the information in the export file.

In the REPLACE_ALL statement, the AS clause is only valid if you specify a particular record. It is not valid and should not be used with REPLACE_ALL *.

The table name that you specify after the AS command modifier should not have more than 18 characters (including the *ps_* prefix). Specifying a table name that is greater than 18 characters invokes the following error message: *Error: Unable to process create statement.*

Note. Records defined using REPLACE_ALL can have a maximum of 500 total columns and multiple long columns within the limitations for long columns set by the database platform. Check with the database vendor for restrictions on the number of long columns allowed for the platform.

Parameters

All except IGNORE_DUPS and OUTPUT. INPUT is a required parameter.

REPLACE_DATA

Syntax

```
REPLACE_DATA {record | *};
```

Description

This command is a variation of the IMPORT command. Use it to delete data in existing tables and insert the corresponding data from the export file.

Parameters

COMMIT, EXECUTE_SQL, EXTRACT, INPUT, INSERT_DATA_ONCE, LOG, NO COMMIT, NO TRACE, NO VIEW, SIZING_SET, SPACE, START, and VERSION. INPUT is a required parameter.

REPLACE_VIEW

Syntax

```
REPLACE_VIEW {view | *};
```

Description

Recreates one or all specified views in the database.

Parameters

LOG, NO COMMIT, NO TRACE, and START.

Note. If you use REPLACE_VIEW in bootstrap mode, the system automatically activates SET IGNORE ERROR. This enables PeopleSoft Data Mover to continue processing until all of the view definitions have been processed, and all errors have been written to the current .LOG file.

RUN

Syntax

```
RUN dms_file_name;
```

Description

Runs a DMS file from within a script. The specified file can contain any supported SQL commands, PeopleSoft Data Mover commands, or SET statements, but it cannot contain any RUN commands.

The RUN command cannot contain a directory path. The RUN command uses the same directory as the current PeopleSoft Data Mover script in which RUN is used.

SET

Syntax

```
SET parameter_1;
SET parameter_2; ...
SET parameter_n;
```

Description

The SET command, when combined with valid SET parameters, creates statements that establish the conditions under which PeopleSoft Data Mover runs a script.

A SET statement controls the processing environment for the commands in a script until another SET statement intervenes between commands. At that point, all SET parameters are reset to their default values.

Example

```
SET LOG c:\temp\new.log
SET OUTPUT c:\temp\new.dat;
/
EXPORT absence_hist;
EXPORT employee_tbl
/
SET NO DATA
/
REMARK  All other SET parameters will be reset to their default values at this⇒
 point
;
EXPORT bank_branch_tbl;
```

In the previous script, the specified log and output files (NEW.LOG and NEW.DAT) are used for the first two EXPORT commands. Then, because SET NO DATA interrupts the script commands, all other SET parameters are reset to their default values. So, for the third EXPORT and any subsequent PeopleSoft Data Mover or SQL commands, the log file used is the default log file, DATAMOVE.LOG, and the output file used is the default output file, DATAMOVE.DAT.

See Chapter 2, “Using PeopleSoft Data Mover,” Using SET Parameters, page 36.

SWAP_BASE_LANGUAGE

Syntax

```
SWAP_BASE_LANGUAGE <NEW LANGUAGE_CD>;
```

or

```
SET BASE_LANGUAGE <CURRENT LANGUAGE_CD>;
SWAP_BASE_LANGUAGE <RECNAME>;
```

Description

Installs any language other than English.

The command swaps all the language tables from PSRECDEFN. It gets all table names that contain related tables, and it swaps one table at a time. It copies the base table into the related table, updates the related record into the base table, and then deletes the related record from the related table.

If successful, the command updates PSOPTIONS SET LANGUAGE_CD to the new base language.

Swapping an individual table (SET BASE_LANGUAGE <CURRENT LANGUAGE_CD> and SWAP_BASE_LANGUAGE <RECNAME>) is used only when there is an error with any of the tables after the SWAP_BASE_LANGUAGE <NEW LANGUAGE_CD> command has been run.

Note. Never run a combination of SET BASE_LANGUAGE <CURRENT LANGUAGE_CD> and SWAP_BASE_LANGUAGE <RECNAME> command before SWAP_BASE_ LANGUAGE <NEW LANGUAGE_CD>.

Example

To swap English for Canadian French, enter the following:

```
SWAP_BASE_LANGUAGE CFR
```

CFR is the new language code (LANGUAGE_CD).

Note. During the initial installation, the Database Setup utility generates a script that automatically swaps the base language if, while in the Database Setup interface, you select a base language other than English.

SET IGNORE_ERRORS

Syntax

```
SET IGNORE_ERRORS;
SWAP_BASE_LANGUAGE <LANGUAGE_CD>;
```

Description

Use this command in conjunction with the SWAP_BASE_LANGUAGE command.

Example

Here's an example of how to swap one table (without the SET IGNORE_ERRORS command, it stops on error):

```
SWAP_BASE_LANGUAGE <LANGUAGE_CD>;
```

Here's an example of how to ignore all errors and swap all tables:

```
SET IGNORE_ERRORS;
SWAP_BASE_LANGUAGE <LANGUAGE_CD>;
```

When the SWAP_BASE_LANGUAGE command is run after SET IGNORE_ERRORS, the PSOPTIONS SET LANGUAGE_CD is automatically updated with new base language, even if errors were recorded.

When the command has run, you should then examine the log and swap the individual record names that failed using SWAP_BASE_LANGUAGE <RECNAME> command

SET BASE_LANGUAGE

Syntax

```
SET BASE_LANGUAGE <CURRENT LANGUAGE_CD>;
SWAP_BASE_LANGUAGE <RECNAME>;
```

Description

Use only when there is an error with any of the tables after the SWAP_BASE_LANGUAGE <NEW LANGUAGE_CD> command.

Note. Never run SET BASE_LANGUAGE <CURRENT LANGUAGE_CD>, SWAP_BASE_LANGUAGE <RECNAME> commands before SWAP_BASE_LANGUAGE <NEW LANGUAGE_CD>.

SET COMMIT

Syntax

```
Set COMMIT #of_rows;
```

Description

Sets the commit level for inserting rows and not for DDL statements. If the level is set to 0, commits are only done when all rows for a record are inserted. Due to the expense of recompiling and rebinding after a commit, the default is 0.

Note. There are performance implications associated with the SET COMMIT command. For a large database with millions of rows, there is significant degradation in performance. However, for a small database, performance slows down somewhat. Run the SET COMMIT command only as necessary.

Parameters

The default is to commit at the end of the record.

Example

The following examples demonstrate how to use SET COMMIT in conjunction with SWAP BASE_LANGUAGE:

```
Set COMMIT #of_rows;
SWAP BASE_LANGUAGE FRA;
```

or

```
Set COMMIT #of_rows;
SET BASE_LANGUAGE ENG;
SWAP_BASE_LANGUAGE <RECNAME>;
```

Using PeopleSoft Data Mover Command Modifiers

The following commands enable you to modify a PeopleSoft Data Mover command to limit its scope, rename the item being processed, or control error messaging.

AS

Syntax

```
{IMPORT | REPLACE_ALL} record
  AS table_name;
```

Description

Changes the name of a record and then imports it. When using this modifier, keep the following points in mind:

- If used with an IMPORT, the record is not imported if the table name specified in the IMPORT command already exists in the database.
- When using the AS command modifier, you can specify either the record or table name for the record or table specified preceding the AS.

  However, you must always specify the table name (not the record name) for the record or table specified following the AS. The name specified following the AS is the actual name that is used for the table to be created.
- This modifier is not supported for records containing trigger definitions.

Parameters

IMPORT and REPLACE_ALL

Example

The following example imports a new record or table originally named PS_JOB and creates it as PS_PROCESS:

```
IMPORT job
  AS ps_process;
```

Also correct:

```
IMPORT ps_job
  AS ps_process;
```

Incorrect:

```
IMPORT ps_job
  AS process;
```

Incorrect:

```
IMPORT job
  AS process;
```

The last two examples are incorrect because process is specified, instead of ps_process. This means that the table created is named PROCESS, but it should be named PS_PROCESS to comply with the convention that all non-PeopleTools tables have the prefix PS_.

The table name that you specify following the AS command modifier should not have more than 18 characters (including the ps_ prefix). Specifying a table name that is greater than 18 characters invokes the following error message: *Error: Unable to process create statement.*

When you import a record in this way, it is only created in the system tables, not in the PeopleSoft tables. You must also create the record in the PeopleSoft tables, such as PSRECDEFN.

To create a table after running the IMPORT command:

1. Launch PeopleSoft Application Designer.
2. Create or clone the new record.

   Using the job and process example from the previous discussion, you would open JOB and then select File, Save As and rename the record to PROCESS.

Note. The PS_ prefix does not appear in PeopleSoft Application Designer.

3. Select Build, Current Object.
4. In the Build dialog box, select Create Tables under Build Options.

   You may also want to make sure that all the appropriate options are set on the Build Settings tabs.

IGNORE_DUPS

Syntax

```
SET IGNORE_DUPS;
IMPORT {record | *};
```

Description

Ignores duplicate-row error messages from the database. The IMPORT process continues despite any duplicate-rows errors in the output window and log file. When IGNORE_DUPS is set, bulk loading, the ability to load more than one row at a time, is turned off. By default, bulk loading is on and inserts up to 100 rows into a table at a time. Because turning off bulk loading slows performance, you should use this feature only when required.

Note. SET IGNORE_DUPS is only valid in bootstrap mode.

Parameters

IMPORT.

UPDATE_DUPS

Syntax

```
SET UPDATE_DUPS;
IMPORT {record | *};
```

Description

On command, PeopleSoft Data Mover imports a new row and updates an existing row.

Note. This command is valid for both bootstrap mode and regular mode. In regular mode, if the table is identified as a language table, the system automatically resolves and swaps the base and related language tables.

See Chapter 2, “Using PeopleSoft Data Mover,” IMPORT, page 26.

Parameters

IMPORT.

WHERE

Syntax

```
EXPORT {record | *} WHERE
  condition(s)[;var#1_type,_var#1_value,var#2_type,var#2_value,...
```

```
var#n_type,var#n_value];
```

Note. In an EXPORT statement, the WHERE modifier must be on the same line as the EXPORT command.

Description

Exports a partial set of rows from a record. The syntax and conditions of a PeopleSoft Data Mover WHERE clause in an EXPORT are similar to a WHERE clause in SQL. You can write the WHERE clause with comparison operands inline or as bind variables. You can also use nested SELECT statements.

Warning! When comparing string or character values, use only US-ASCII (seven-bit ASCII) values. Characters beyond this range can produce errors in the export file.

Parameters

EXPORT.

Example

Here's an example of a WHERE clause using both an inline operand and bind variables in an EXPORT script:

```
EXPORT JOB WHERE
  EFFDT > :1 AND
  HOURLY_RT > :2
  AND GRADE = 'ADV';DATE,1994-01-01,NUMBER,100;
```

There are no single or double quotation marks around the bind data, as they are not necessary, and dates are formatted as YYYY-MM-DD. The valid data types for binding are CHAR, NUMBER, DATE, TIME, DATETIME, LONG, and IMAGE. Not all database platforms support LONG or IMAGE data types in the WHERE clause, so you should not use WHERE clauses with these data types.

The following operators are supported in an import WHERE clause: =, < >, <, >,< =, > =, and simple uses of AND and OR. For example, in the following formula, if A, B, and C are true, or if D is true, or if E is true, then the whole statement is true

```
WHERE
  A = :1 AND B = :2 AND C = :3
  OR D = :4
  OR E = :5;NUMBER,10,NUMBER,20,NUMBER,30,NUMBER,0,NUMBER,1;
```

Using SET Parameters

The following parameters can be appended to a SET command to create a valid SET statement.

COMMIT

Syntax

```
SET COMMIT #of_rows;
```

Description

Sets the commit level only for inserting rows and not for DDL statements. If the level is set to 0, commits are only done when all rows for a record are inserted. Due to the expense of recompiling and rebinding after a commit, the default is 0.

Parameters

IMPORT, REPLACE_ALL, and REPLACE_DATA.

CREATE_INDEX_BEFORE_DATA

Syntax

```
SET CREATE_INDEX_BEFORE_DATA;
```

Description

Creates the index before inserting rows into a record. The default method is to insert rows into a record and then create the index.

Parameters

IMPORT and REPLACE_ALL.

DBSPACE

Syntax

```
SET DBSPACE {<old dbname>.<old spcname>} AS {<new_dbname>.<new spcname>};
```

Description

The DBSPACE command is similar to the SPACE command, but it is designed to handle the combination of DBNAME.DDLSPACENAME. On DB2 UDB, the DBNAME or DDLSPACENAME alone is not necessarily unique. However, the combination of the two (DBNAME.DDLSPACENAME) provides a unique relationship. For example, DBSPACE would be needed in the following scenario:

```
PSFSDMO.HRAPP
PSHRDMO.HRAPP
PSPTDMO.HRAPP
```

Note. This command is supported only on DB2 UDB for OS/390 and z/OS. You use this command in place of the SPACE command used on other platforms.

Parameters

IMPORTand REPLACE_ALL.

Example

The wildcard (*) character is permitted for the database name and space name parameters to apply to all values being processed for the specific parameter in which the wildcard character is used. The following are examples of using this command to achieve one of the following:

To change a specific DBNAME/DDLSPACENAME combination to a single new combination:

```
SET DBSPACE <old dbname>.<old spcname> AS <new dbname>.<new spcname>
```

To keep the current database name the same but change the specific space name to a new name:

```
SET DBSPACE <*>.<old spcname> AS <*>.<new spcname>
```

To keep the current space name the same, but change the specific database name to a new name:

```
SET DBSPACE <old dbname>.<*> AS <new dbname>.<*>
```

Warning! Because of the large number of objects delivered in the PeopleSoft logical databases, do not override all old database name or space name values to a single new database name or space name value when building a SYS or DMO database. However, this feature may be useful in working with smaller data files that contain a smaller number of objects.

For large databases, do not use the following commands:

```
SET DBSPACE <*>.<*> AS <new dbname>.<new spcname>
SET DBSPACE <*>. <*> AS <*>.<new spcname>
SET DBSPACE <*>.<*> AS <new dbname>.<*>
```

You can use multiple SET DBSPACE statements to override the space name in the .DAT file. This enables you to override multiple databases in the same section of the script. For example:

```
SET DBSPACE PSFSDMO.* AS MYFSDMO1.*;
SET DBSPACE PSFSDMOF.* AS MYFSDMO2.*;
SET DBSPACE PSFSDMOD.* AS MYFSDMO3.*;
SET DBSPACE PSFSDMOM.* AS MYFSDMO4;
```

DDL

Syntax

```
SET DDL {RECORD | INDEX | UNIQUE INDEX | SPACE} {object_name | *}
  INPUT parm AS value;
```

Note. The object_name is only available for the SPACE option, not the RECORD, INDEX, and UNIQUE INDEX. The RECORD, INDEX, and UNIQUE INDEX are available for the *, not the object_name.

Description

Substitutes values for the parameters specified in the DDL template commands. Substitute the *parm* and *value* placeholders for an actual parameter and its value. If an asterisk is used instead of an object name, a SQL update on PSDDLDEFPARMS is performed on the parameter and value upon successful completion of the IMPORT or REPLACE_ALL command that corresponds to the SET DDL statement.

Parameters

IMPORTand REPLACE_ALL.

Example

Below are some examples of DDL template SET commands from a DB2 UDB import script:

```
SET DDL RECORD   *  INPUT dbname   AS pt750dg0;
SET DDL INDEX   *  INPUT stogroup   AS wps04sg;
SET DDL SPACE   *  INPUT stogroup   AS wps04sg;
```

EXECUTE_SQL

Syntax

```
SET EXECUTE_SQL [AFTER] sql_statement;
```

Description

Performs the SQL statement specified at the beginning of a transaction. Typically, this command is used to set up a specific cursor environment before PeopleSoft Data Mover begins processing. For example, in DB2 UDB, use this command to set the current setID, or for Oracle, use this command to designate a specific rollback segment.

This command doesn't run for DDL SQL statements. For example, in DB2 UDB, you cannot set the current setID before creating spaces, tables, indexes, or views.

Parameters

IMPORT, REPLACE_ALL, and REPLACE_DATA.

EXTRACT

Syntax

```
SET EXTRACT {COMMAND | DDL | INPUT | SPACE | OUTPUT file_name};
```

Description

Extracts various types of information from an export file (the DAT file specified in the corresponding SET INPUT command that precedes the IMPORT or REPLACE ALL command) and writes this information to the user-defined output file specified in the SET EXTRACT OUTPUT *file_name* statement.

Note. You must use SET EXTRACT OUPUT before issuing any other SET EXTRACT statements.

EXTRACT INPUT writes out any statements from the DAT file that are associated with the tables being imported. EXTRACT DDL writes out any CREATE TABLE, CREATE INDEX, or CREATE UNIQUE INDEX statements from the DAT file. EXTRACT COMMAND writes out the EXPORT statements from the DAT file.

When EXTRACT statements are issued, no SQL CREATE or INSERT statements are executed. The associated IMPORT or REPLACE_ALL command is not actually executed, so no import is performed.

Parameters

IMPORTand REPLACE_ALL.

IGNORE_DUPS

Syntax

```
SET IGNORE_DUPS;
```

Description

Ignores duplicate-row error messages from the database; the IMPORT process continues despite any duplicate-row errors displayed in the output window and log file. You can set this command for the entire import script or by record, using IGNORE_DUPS as a command modifier.

When IGNORE_DUPS is set, bulk loading, the ability to load more than one row at a time, is turned off (to allow checking for duplicates, so that duplicate rows can be ignored or bypassed). By default, bulk loading is on and inserts many (100) rows into a table at a time. Because turning off bulk loading slows performance, use this feature only when required or by record.

See Chapter 2, "Using PeopleSoft Data Mover," IMPORT, page 26.

See Chapter 2, "Using PeopleSoft Data Mover," IGNORE_DUPS, page 35.

Parameters

IMPORT.

Note. The command SET IGNORE_DUPS is only valid in bootstrap mode. This prevents the loss of data during a PeopleSoft Data Mover import of a language table in regular mode.

INPUT

Syntax

```
SET INPUT file;
```

Description

Specifies the name of the exported file to import; typically this file has a .DAT extension, though this is not a requirement. Because this statement is required to do an import, there is no default file.

If you don't specify a path for this file, PeopleSoft Data Mover searches for the file in the following locations in the order presented:

- It searches the Data Mover input directory as defined in PeopleSoft Configuration Manager on the Common tab.
- If the input directory setting is blank (not set) on the Common tab, PeopleSoft Data Mover searches the C:\TEMP directory.

Parameters

IMPORT, REPLACE_ALL, and REPLACE_DATA.

INSERT_DATA_ONCE

Syntax

```
SET INSERT_DATA_ONCE record;
```

Description

Skips (that is, bypasses importing) the specified record if there is already one or more rows in the table corresponding to that record. If the table is empty, only a single row is inserted.

Parameters

IMPORT, REPLACE_ALL, and REPLACE_DATA.

LOG

Syntax

```
SET LOG file;
```

Note. You must specify a file name for the SET LOG statement or else a log file is not created. If you do not want to specify a log file name, omit the SET LOG statement completely.

Description

Specifies a user-defined file name for the log file that is created when running a PeopleSoft Data Mover script or command. If the SET LOG statement is omitted completely, a default log file is created with the name DATAMOVE.LOG. PeopleSoft Data Mover writes this DATAMOVE.LOG file to the default log directory, which is determined as follows:

- The system uses the PeopleSoft Data Mover log directory specified on the Common tab in PeopleSoft Configuration Manager.
- If the preceding setting is blank, the log file is written to C:\TEMP.

Note. If you use the SET LOG statement but do not specify a file name and path, PeopleSoft Data Mover writes the user-defined log file to the default log directory according to the same rule.

When checking the DATAMOVE.LOG file in a multidatabase environment, make sure you are examining the correct log file. At the top of the output file, verify the date and the database name.

```
Logging status in C:\TEMP\datamove.log
Started: Fri Mar 17 13:47:15 2001
Data Mover Release: 8.4
Database: HR702U40
...
Ended: Fri Mar 17 13:47:20 2001
Successful completion
```

Parameters

All.

NO DATA

Syntax

```
SET NO DATA;
```

Description

During an export, the NO DATA command prevents data from being exported. In an import, this command prevents data from being inserted.

Parameters

EXPORT, IMPORT, and REPLACE_ALL.

NO INDEX

Syntax

```
SET NO INDEX;
```

Description

Prevents indexes from being created during an IMPORT or a REPLACE_ALL command.

Parameters

IMPORT and REPLACE_ALL.

NO RECORD

Syntax

```
SET NO RECORD;
```

Description

Prevents records from being created during an import

Parameters

IMPORT and REPLACE_ALL.

NO SPACE

Syntax

```
SET NO SPACE;
```

Description

Prevents tablespaces from being created. This is the default setting. You can use this statement to reset the default after executing a SET SPACE statement.

Parameters

IMPORT and REPLACE_ALL.

NO TRACE

Syntax

```
SET NO TRACE;
```

Description

Sets the PeopleSoft trace flag (TraceSQL) in PeopleSoft Configuration Manager to *Off* for the commands that follow, until the next SET statement. This is the recommended method of executing commands. If SET NO TRACE is specified, then no trace file is created, even if you specify a trace file in PeopleSoft Configuration Manager on the Trace tab. Commands that you run *without* specifying SET NO TRACE do trace SQL, if SQL tracing is enabled in PeopleSoft Configuration Manager.

Note. This statement cannot be used with an INSERT command.

Parameters

All.

NO VIEW

Syntax

```
SET NO VIEW;
```

Description

Prevents views from being created.

Parameters

EXPORT * only, IMPORT * only, REPLACE_ALL * only, and REPLACE_DATA * only.

OUTPUT

Syntax

```
SET OUTPUT file;
```

Note. You must specify a file name for the SET OUTPUT statement or else a log file is not created. If you do not want to specify a log file name, omit the SET OUTPUT statement completely.

Description

Specifies a user-defined file name for the output file that is created by the corresponding EXPORT statement. If the SET OUTPUT statement is omitted completely, a default output file with the name DATAMOVE.DAT is created. The location that the output file is created is determined by the following:

- The system uses the PeopleSoft Data Mover output directory specified on the Common tab in PeopleSoft Configuration Manager.
- If the previous setting is blank, the output file is created in the C:\TEMP directory.

Note. If you use the SET OUTPUT statement but do not specify a file name and path, PeopleSoft Data Mover writes the user-defined log file to the default log directory according to the same rule.

Parameters

EXPORT.

SIZING SET

Syntax

```
SET SIZING_SET n;
```

Description

Specifies the sizing set number as defined on the DDL Model Defaults page. The default is 0. To use this parameter, the specified sizing set must be defined in the export file.

See *Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration*, "Using PeopleTools Utilities".

Parameters

IMPORT and REPLACE_ALL.

SPACE

Syntax

```
SET SPACE old spcname AS new_spcname;
```

Description

Use for all operating systems other than z/OS.

Renames the default space names to customized space names. To name all record default space names to a single space name, substitute * for a space name.

Parameters

IMPORT and REPLACE_ALL.

Example

```
SET SPACE * AS PS;
```

START

Syntax

```
SET START [AFTER] record;
```

Description

Designates where in the export file to start the import process. The default is to start at the beginning of the file. To start immediately after a particular PeopleSoft record in the file, use SET START AFTER. This SET statement is useful for restarting a script after an error.

If the AFTER parameter is omitted, the import process starts at the record that is specified in the SET START statement. If the AFTER parameter is specified, the import process starts after the record specified in the SET START statement.

Note. If the same record name appears multiple times in the same DAT file, the SET START AFTER command begins after the last occurrence of the record name in the DAT file.

When you use the SET START command with REPLACE_VIEW and no DAT file specified, you designate at which (or after which) view in the database to start. Views are created in alphabetical order.

Parameters

IMPORT, REPLACE_ALL, REPLACE_DATA and REPLACE_VIEW.

STATISTICS

Syntax

```
SET STATISTICS { ON | OFF };
```

Description

Sets UPDATE STATISTICS to on or off. The default value is on. Set the value to off if you do not want to update statistics after an IMPORT. This command works only in bootstrap mode.

Parameters

IMPORT and REPLACE_ALL.

UNICODE

Syntax

```
SET UNICODE { ON | OFF }
```

Description

This command is recommended for use in bootstrap mode for an initial database load. It specifies whether the database is Unicode or non-Unicode.

Warning! If the database is already fully loaded, DO NOT use this command because it could result in the wrong value ENABLE_UNICODE flag being set on the PSSTATUS table.

Parameters

IMPORT and REPLACE_ALL.

VERSION

Syntax

```
SET VERSION sql_table.column condition;
```

Description

Verifies the version of the database for importing.

Parameters

IMPORT, REPLACE_ALL and REPLACE_DATA.

Example

Suppose that you state the following:

```
SET VERSION PSLOCK.TOOLSREL="8.4"
```

PeopleSoft Data Mover verifies that the TOOLSREL column in PSLOCK equals 8.4. This avoids importing an export file into the wrong database. Use the SQL table name to indicate which PeopleSoft record to check.

Using Script Examples

This section provides several example script files. Review these scripts to see how you can use PeopleSoft Data Mover to accomplish various tasks.

Exporting Databases

Description

This example shows how to export a database.

Example

```
SET OUTPUT c:\temp\pt.dat;
SET LOG c:\temp\pt.log;
EXPORT *;
```

Building Microsoft SQL Server Databases

Description

This example shows how to build a Microsoft SQL Server database.

Example

```
set log c:\temp\hcengd.log;
set input c:\HRDMO\data\hcengd.db;
set no view;
set no space;
set no trace;
import *;
update PSLOCK set OWNERID = 'ownerid';
update PSOPRDEFN set ACCESSID = 'accessid', ACCESSPSWD = 'accesspw',⇒
 OPERPSWD = '0000000000000000' where OPRTYPE = 0;
update PSACCESSPRFL set ACCESSID = 'accessid', ACCESSPSWD = 'accesspw',⇒
 VERSION = 0, ENCRYPTED = 0;
set log c:\temp\grant.log;
encrypt_password *;
```

Recreating All Views

Description

This example shows how to recreate all views.

Example

```
SET LOG c:\temp\view.log;
REPLACE_VIEW *;
```

Importing with REPLACE_ALL with a Commit Level

Description

This example shows how to import with REPLACE_ALL with a commit level.

Example

```
SET INPUT c:\ptdvl\bin\exp2.dat;
SET LOG c:\ptdvl\bin\exp2.log;
SET COMMIT 2;
REPLACE_ALL employee_review;
REPLACE_ALL course_tbl
   WHERE days_duration = :1 AND course_type > :2;number,1,char,C;
REPLACE_ALL absence_hist
   WHERE return_dt > :1;date,1988-01-01;
```

Combining SQL Commands and IMPORT

Description

This example shows how to combine SQL commands and IMPORT.

Example

```
SET INPUT c:\ptdvl\bin\exp2.dat;
SET COMMIT 10;
SET START AFTER course_tbl;
SET IGNORE_DUPS;
DELETE FROM ps_absence_hist WHERE emplid = ’8001’;
IMPORT *;
```

CHAPTER 3

Using PeopleSoft Data Archive Manager

This chapter provides overviews of PeopleSoft Data Archive Manager, archiving strategy, and archiving techniques, and discusses how to:

- Access the Data Archive Manager homepage.
- Manage archive objects.
- Define archive queries.
- Manage archive templates.
- Archive data to history.
- Audit archive processes.

Note. The Archive Data tool delivered with previous releases of PeopleTools is a deprecated feature, and has been replaced by this Data Archive Manager.

See Also

Appendix H, "Archiving Data (Deprecated)," page 251

Understanding PeopleSoft Data Archive Manager

In any enterprise application, the ability to purge and archive transactional data is critical to data management. You need to have consistent methods to archive transactional data before your database increases to unmanageable sizes. PeopleSoft Data Archive Manager provides an integrated and consistent framework for archiving data from PeopleSoft applications.

Using a predefined template, you can select any queries and multiple objects that meet your archiving requirements. Leveraging the Archive Query in PeopleSoft Query, you can easily define your archive template.

To better manage the archive process, you don't have to make any commits to the database until the entire batch has completed.

PeopleSoft Data Archive Manager includes the following main components:

- Archive object definition.

  An archive object is a collection of tables that you archive. The object definition determines how you archive data from a table. For base tables within an archive object, PeopleSoft Data Archive Manager archives data based on a user specified query. For non-base tables within an archive object, PeopleSoft Data Archive Manager archives data based on the archived data of the base table. This implementation eliminates the requirement of having query definitions for non-base tables.

- Archive query definition.

PeopleSoft Data Archive Manager uses PeopleSoft Query to define selection criteria from the base table of the base archive object (for example, archive all rows in JRNL_HEADER where BUSINESS_UNIT = 'ABC01').

- Archive template definition.

  An archive template can contain multiple objects and multiple queries. One of the archive objects in the archive template must be a base object. You can simply define the selection criteria to archive from the base table without specifying criteria for all records in the archive template. Within the archive template, you must specify what AE processes to run before and after the data has been archived, for each of the archiving processes.

- Archive job definition.

  You define archive jobs to archive data to history. Before you submit an archive job, you must first define the archive job information including the Archive Template, Archive Process, and Commit Processing. You can submit archive jobs in a batch using the process scheduler. As part of the process, PeopleSoft Data Archive Manager prompts you for run time parameters such as bind variables and what query to use.

- Archive auditing.

  To facilitate auditing, PeopleSoft Data Archive Manager retains a record of the following:

  - What process was executed.
  - Who ran the batch process.
  - When the process was executed.
  - Which Archive ID and record was affected.
  - What SQL statement was executed.

Understanding Archiving Strategy

This section discusses:

- Archiving strategy.
- History tables.

Archiving Strategy

Determining an archiving strategy is essential for using PeopleSoft Data Archive Manager efficiently. This strategy depends on how the archived data will be used. The following describes the strategy for archiving to history table:

- Use history tables for storing archived data.
- Enable reporting and queries from history tables.
- Must have a secondary step to delete archived data from online tables.
- Must have additional database space.

The system is designed to provide as much flexibility as possible. By reviewing your business requirements, you will be able to determine which strategic step best fits your business needs.

Here is a high-level-overview of the steps:

1. You move data into the history tables.

This is known as the selection process. This enables you to query the selected data for information and copy data from the online tables into the history tables.

2. If you accidentally delete the data from the online tables, there is a process to restore the data back from the history tables.

   This rollback process is the optional second step.

3. When you no longer need to reference the data from the history tables, you can delete them completely from the system.

History Tables

Archiving to history tables involves using tables that you create for the sole purpose of storing archived data. You must determine whether the archived data should be stored in the history tables temporarily or on a long-term basis.

By definition, history tables are identical copies of the online tables. However, history records must include PSARCHIVE_SBR sub-record that contains the archive ID and batch number. Some PeopleSoft applications deliver history tables prebuilt for use in common archiving processes. If you design a custom archiving scheme, you need to create the history tables using PeopleSoft Application Designer.

History Table Considerations

After the archive process moves the data into the history table, the data resides in both the online tables and in the history table; you then have two options:

- Deleting the archived data from the online tables.
- Leaving the archived rows in the online tables such that the data exists in parallel.

Procedure to Build History Tables

Before you run the archiving process, you must first create (or build) the history tables.

You must build one history table for each table to be archived. The history table must be identical to the archive table. PeopleSoft Data Archive Manager uses the PSARCHIVE_SBR sub-record that contains PSARCH_ID and PSARCH_BATCHNUM to denote when a piece of data was archived and to uniquely identify it.

The following example uses the record JRNL_HEADER.

To build a history table:

1. Open PeopleSoft Application Designer.
2. Open the JRNL_HEADER table.
3. Select File, Save As and name the history table with an appropriate name, such as JRNL_HEADER_HST.
4. When prompted to copy the PeopleCode associated with the table, click No.
5. Select Insert, Sub-Record then insert the PSARCHIVE_SBR sub-record.
6. Save the record.
7. Build the table by selectingBuild, Current Object .
   - Select the following build options: Create Tables and Create Indexes.
   - Select the following build execute options: Execute and Build script.
   - Click Build.

Understanding Archiving Techniques

This section discusses:

- Business requirements analysis.
- Commits.
- Performance enhancement.
- Index limitations.
- Data limitations.

Business Requirements Analysis

It is important to devise a business strategy before archiving the data. First, you must identify the tables that you want to archive. This includes identifying all of the parent and child tables associated with the tables. Failing to identify all of the related tables can cause corruption to the database. Next, you must know exactly which data to archive. It is important to recognize which rows are safe to remove from the online tables. Remember to remove only the data that is not required to maintain the day-to-day business and reporting.

Consider PeopleSoft General Ledger as an example. General Ledger contains the greatest amount of data to be archived because it is the module where the majority of reporting is required. There are two sets of data types that need to be maintained: balance information and transactional information. Balance information is retained in the ledger records. You might require balance information for online and reporting purposes to be available for a three-year period. On the other hand, transactional data is maintained in the journal header and line tables. Suppose that you require only one year of transactional data to be retained in the system for online purposes, but three years to be retained for reporting purposes.

Any data beyond the above time frames for balances and transactions can be archived and is only be accessed through reports. The data can be archived to history tables. If data were to be archived into history tables, the data would still be available online for reporting purposes. However, you could not view it through standard PeopleSoft Internet Architecture pages without special configuration. In addition, reports would need to be modified to access the data in history tables. Moving archived data to secondary storage devices is generally used for long-term data retention. This option is preferred for data that is rarely retrieved, and secondary storage devices are usually used to satisfy legal requirements.

Commits

By default, the Archive Selection, Remove from History, Rollback, and Delete processes issue commits after each record has been processed unless Row-based processing or Unit-of-Work processing have been specified.

Performance Enhancement

For better performance and increased speed during archiving processes, try dropping the indexes before inserting data from online tables into history tables.

Index Limitations

The database platform may have a limitation on the number of columns that an index can contain. Some have a restriction of 16 columns for an index. If the table that you want to archive already has 16 keys, then you can’t add other keys (PSARCH_ID and PSARCH_BATCHNUM from PSARCHIVE_SBR sub-record) to the corresponding history table.

To solve this problem, you can create the history table with the PSARCH_ID and PSARCH_BATCHNUM as non-key fields.

Data Limitations

Due to platform and meta-SQL restrictions, Data Archive Manager for PeopleTools 8.4x does not support archiving of records with LONG, IMAGE, or ATTACHMENT columns. The selection process (inserting data from the online records to the history records) will result in the loss of the long, image, or attachment columns in the history record.

However, this restriction applies only to templates archived using set-based processing. Long, image, and attachment data are archived to history records (and back to the transactional records) if the template is archived using row-based processing.

Accessing the Data Archive Manager Homepage

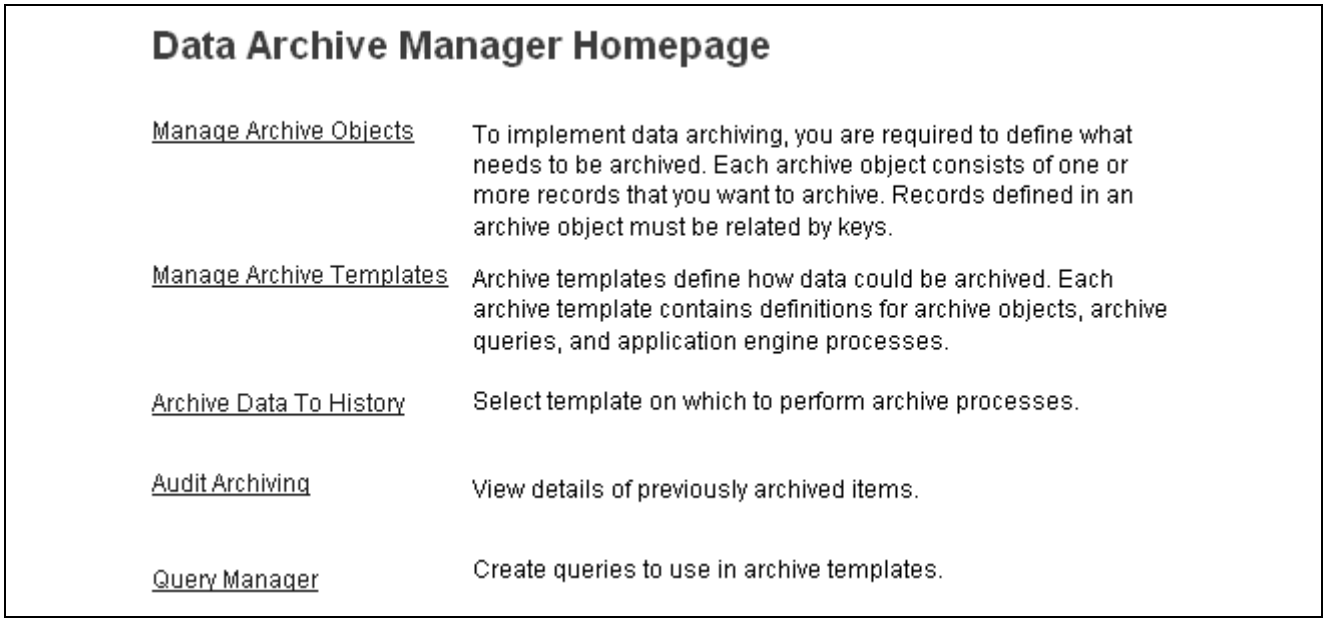
The Data Archive Manager homepage provides you with access to all of the functionality in PeopleSoft Data Archive Manager, including the Query Manager. Alternatively, you can select each menu item directly without accessing the homepage, with the exception of the Query Manager.

Page Used to Access the Data Archive Manager Homepage

Page Name	Object Name	Navigation	Usage
Data Archive Manager Homepage	PSARCHHOME	PeopleTools, Data Archive Manager, Homepage	Use the homepage to access all of the functionality in PeopleSoft Data Archive Manager, including the PeopleSoft Query Manager.

Using the Data Archive Manager Homepage

Access the PeopleSoft Data Archive Manager homepage.



Data Archive Manager Homepage

Manage Archive Objects Click to access the Manage Archive Objects page, where you can define the objects to be archived. Each object is a logical grouping of records. The records specified in an archive object must be related by keys

See Chapter 3, "Using PeopleSoft Data Archive Manager," Managing Archive Objects, page 55.

Manage Archive Templates Click to access the Manage Archive Templates page, where you can define an archive template. Archive templates define how data should be archived. Each archive template enables you to specify archive objects, archive queries, and application engine processes.

See Chapter 3, "Using PeopleSoft Data Archive Manager," Managing Archive Templates, page 56.

Archive Data to History Click this link to access the Archive Data To History page where you can define a job to move data between transactional tables and history tables.

See Chapter 3, "Using PeopleSoft Data Archive Manager," Managing Archive Templates, page 56.

Audit Archiving Click this link to access the Audit Archiving page where you can view the details of previous archive processes.

See Chapter 3, "Using PeopleSoft Data Archive Manager," Auditing Archive Processes, page 61.

Query Manager Click this link to access the Query Manager page in PeopleSoft Query, where you can create a query for your archive process.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Query

Managing Archive Objects

This section provides an overview of the base table and non-base tables, and discusses how to manage archive objects.

Understanding the Base Table and Non-base Tables

A base table is a table that contains all the keys by which all other tables in the archive object is archived from. Each archive object can have one and only one base table. You can define the selection criteria to archive from the base table.

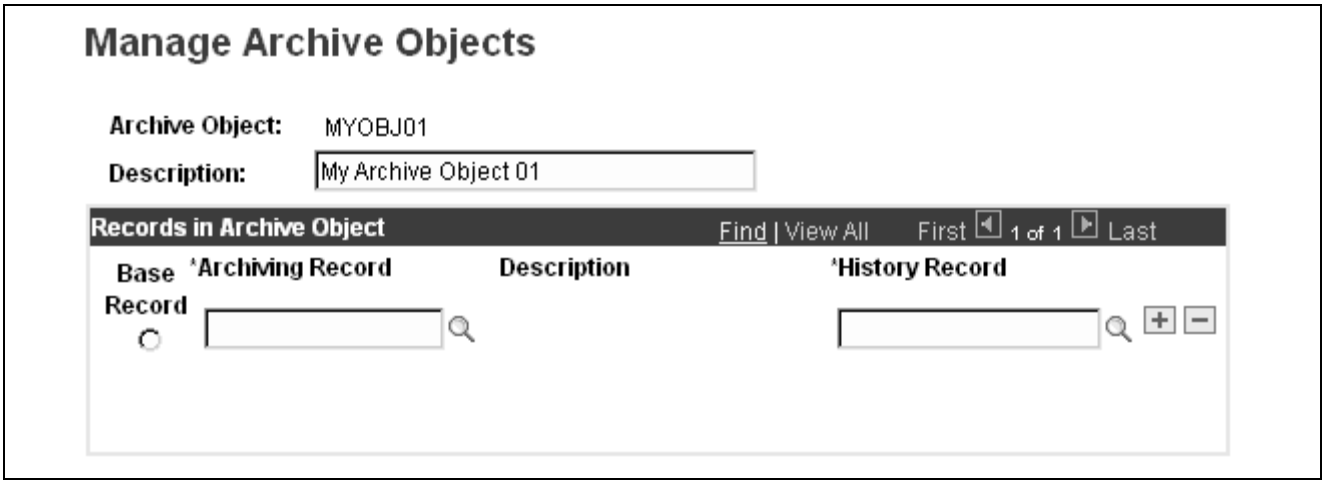
Non-base tables are joined together by common keys. In each archive object, non-base tables are archived based on the archived data of the base tables. You don't need to define archive criteria for non-base tables.

Page Used to Manage Archive Objects

Page Name	Object Name	Navigation	Usage
Manage Archive Objects	PSARCHOBJDEFN	PeopleTools, Data Archive Manager, Manage Archive Objects	Use this page to group archive records into archive objects.

Managing Archive Objects

Access the Manage Archive Objects page.



Manage Archive Objects page

Archiving Record Select the name of the record with the transactional data that you want to archive.

Base Record Select this check box if the record that you select is the base record of this archive object. By definition, there can only be one base record per archive object.

History Record Select the history record to which you want to archive the transactional data. You must first create the history record manually using Application Designer. An error message will appear if the history table has been defined incorrectly.

Defining Archive Queries

You can use PeopleSoft Query to define selection criteria to archive data from transactional tables to history tables. Each of the queries to be used by the Data Archive Manager must be defined as an *Archive* type.

When you select Archive query, you must also select *Public* as owner. The first record of the Archive Query must be the same as the base table of the base record of the archive template. Otherwise, an error message appears.

See *PeopleTools 8.45 PeopleBook: PeopleSoft Query.*

Managing Archive Templates

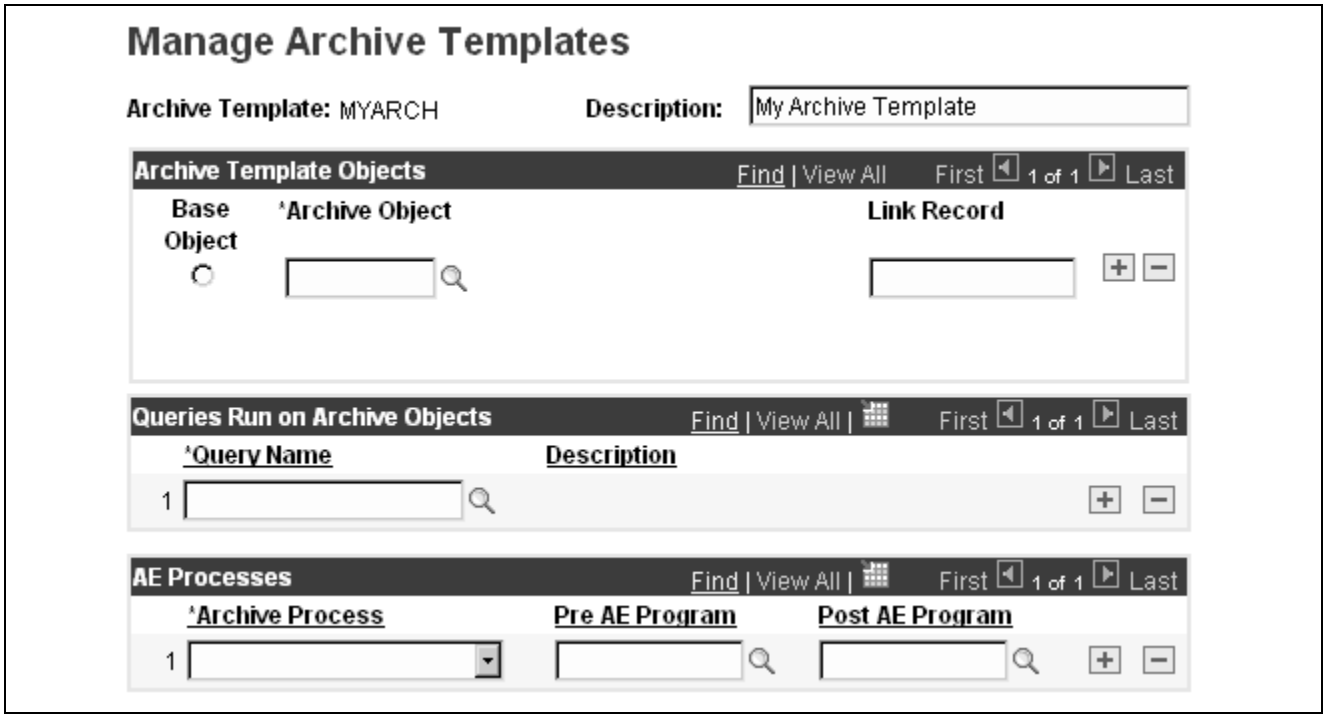
This section discusses how to manage archive templates.

Page Used to Manage Archive Templates

Page Name	Object Name	Navigation	Usage
Manage Archive Templates	PSARCHTEMPDEFN	PeopleTools, Data Archive Manager, Manage Archive Templates	Use this page to define the archive template.

Managing Archive Templates

Access the Manage Archive Templates page.



Manage Archive Templates page

Archive Template Objects

Archive Object Insert from the list of archive objects previously defined in the database.

Base Object Select this radio button if the archive object that you select is the base object of this archive template. Be definition, there can only be one base object per archive template. Data from tables in non-base objects are archived based on archived data from the link table in the base object.

Link Record If the archive object is not a base object, a link record must be defined. Similar to the concept of a foreign key constraint, the link table is used to "link" data between the base record of the non-base objects to archived data of any record in the base object. By this definition, only records that are defined in the base object of the archive template can be used as link records.

Queries Run on Archive Objects

Query Name Select from a list of queries defined in the template. The selection determines how PeopleSoft Data Archive Manager will generate the where clause for the base table of the base object at runtime. Only queries of the type *Archive* can be defined in the Archive Template. You can insert multiple archive queries into the template.

Description Displays the description of the query.

AE Processes

Archive Process Specify a PeopleSoft Application Engine archive process. Valid options are:

- *Archive Selection*
- *Archive Delete*
- *Archive Rollback*
- *Remove from History*

You can define different AE programs to run for each of the archiving processes. For example, you can define an AE program called SEL_PRE that creates summary data in a work table before the Archive Selection process (Pre-AE) is executed. If you perform a rollback, you might want to create an AE program called RBK_POST that executes after the Archive Rollback process (Post-AE) to remove the summary data in the work table.

Pre AE Program Select the custom Application Engine program that you want to run against your data before archiving.

Post AE Program Select the custom Application Engine programs that you want to run against your data after archiving.

Archiving Data to History

This section discusses how to:

- Define archive jobs.

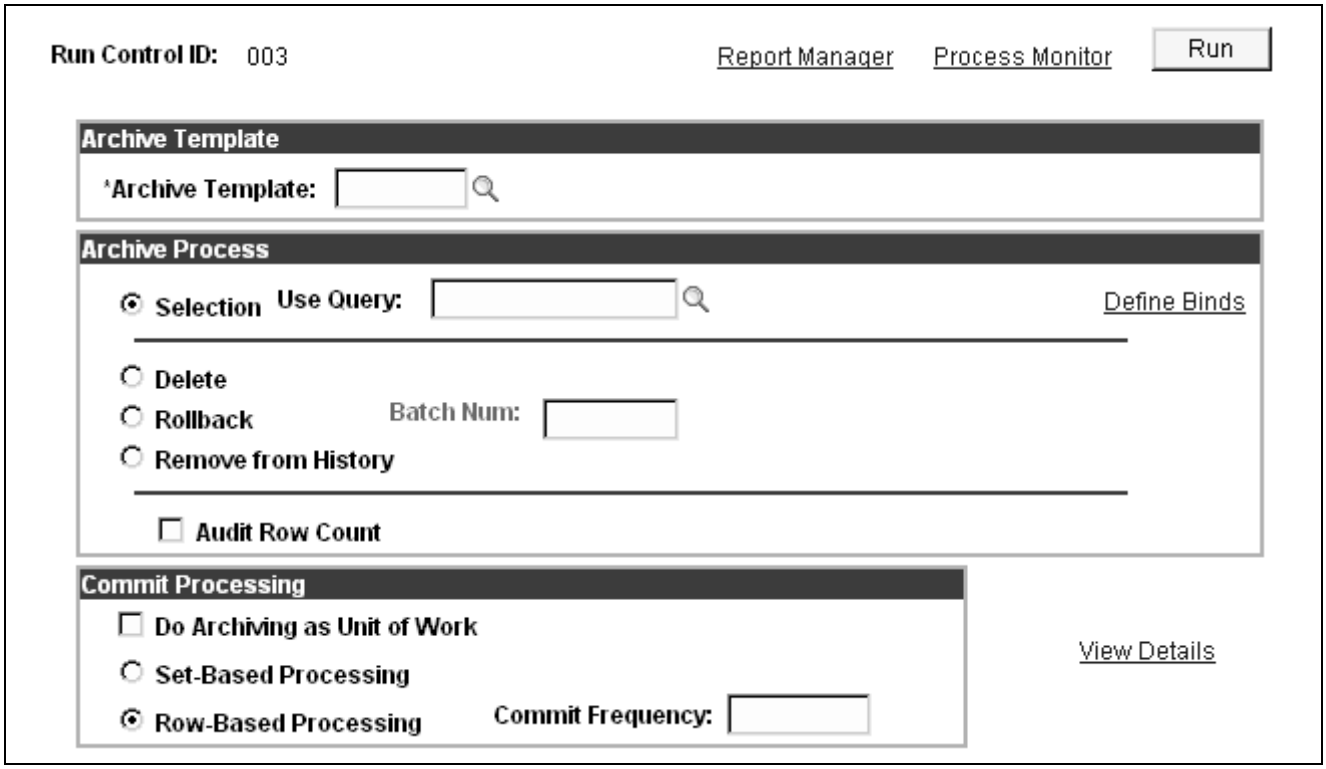
- View details.
- Define archive query binds.

Pages Used to Archive Data to History

Page Name	Object Name	Navigation	Usage
Archive Data To History	PSARCHRUNCTRL	PeopleTools, Data Archive Manager, Archive Data to History	Use this page to submit batch Application Engine jobs through Process Scheduler.
Archive Run Cntl Details	PSARCHEXAMRUNCNTL	PeopleTools, Data Archive Manager, Archive Data to History, View Details	Use this page to view the details of the data
Define Query Bind Variables	PSARCHRUNQRYBND	PeopleTools, Data Archive Manager, Archive Data to History, Define Binds	If you've defined prompts for the selection query that you use for the job, use this page to define the archive query bind variables for your archiving process.

Defining Archive Jobs

Access the Archive Data To History page.



Archive Data to History page

Archive Template Select the archive template to use for this batch job.

Run Click to run this batch job after defining the archive processs and commit processing.

View Details Click to access the Archive Run Cntl Details page to view the SQL and row counts of this batch job.

Note. If you're using bind variables, you must save the run control data before clicking the View Details link.

See Chapter 3, "Using PeopleSoft Data Archive Manager," Viewing Details, page 60.

Archive Process

Use this section to manage the processes that are associated with the selected archive template. As the archiving process runs, counters are inserted into work tables to indicate which records have been processed (for both set-based and row-based operations) and the number of rows processed (for row-based operations only).

For set-based processing, the database server commits only after each record is processed. If the process fails in the middle of processing a record (say, the database logs were full), it'll perform a rollback of everything that has not commited.

For row-based processing, if the process fails for any reason, the counters keep track of only those rows that have been committed to the database. When the Application Engine job is restarted, it skips all the rows that have been committed, and begins with the first uncommitted row.

Selection Select to copy data from transaction tables to history tables.

Use Query Specify the archive query defined within the archive template to use at runtime. If there are bind variables, you will be prompted to enter the bind variables when you click the Define Binds link.

Define Binds Click to access the Define Query Bind Variables page.

See Chapter 3, "Using PeopleSoft Data Archive Manager," Defining Archive Query Binds, page 61.

Delete Select to delete data from transactional tables. Data will only be deleted from the transactional tables if they have already been archived in the history tables.

Rollback Select to copy data from history tables back to transaction tables.

Remove from History Select to delete data from the history tables.

Batch Num For archiving processes that are based on data in the history tables (such as delete data from transactional tables, copy data from history tables to transactional tables, and delete data from history tables), you will be prompted to enter an Archive Batch Number.

Audit Row Count Select to audit the number of rows in the record that meet the criteria.

Commit Processing

By default, batch processing is performed by the Data Archive Manager using set-based processing. Unless specified using the check boxes below, a commit is issued to the database after each table is processed within the Archive Template

Do Archiving as Unit of Work

Data Archive Manager processes data using set-based processing, but doesn't issue any commits to the database server until the entire process has completed.

For example, if your Archive Template is defined with Pre- and Post-AE programs, the Data Archive Manager will first execute the Pre-AE program, then it will process all of the tables in the Archive Template, then it will execute the Post-AE program. Upon successful execution of all these steps, a commit will be issued to the database.

When you select this option, the set-based processing option is automatically selected as well.

Set-Based Processing

Data is processed by passing a single SQL statement per record to be archived to the database server. A commit is issued to the database server after successful completion of each SQL statement.

Row-Based Processing

Data Archive Manager processes data one row at a time using PeopleCode fetches. This method of archiving is more memory intensive and takes longer than set-based processing. However, for archiving processes that contain significant amounts of data, row-based processing could be used to reduce adverse affects on the database server

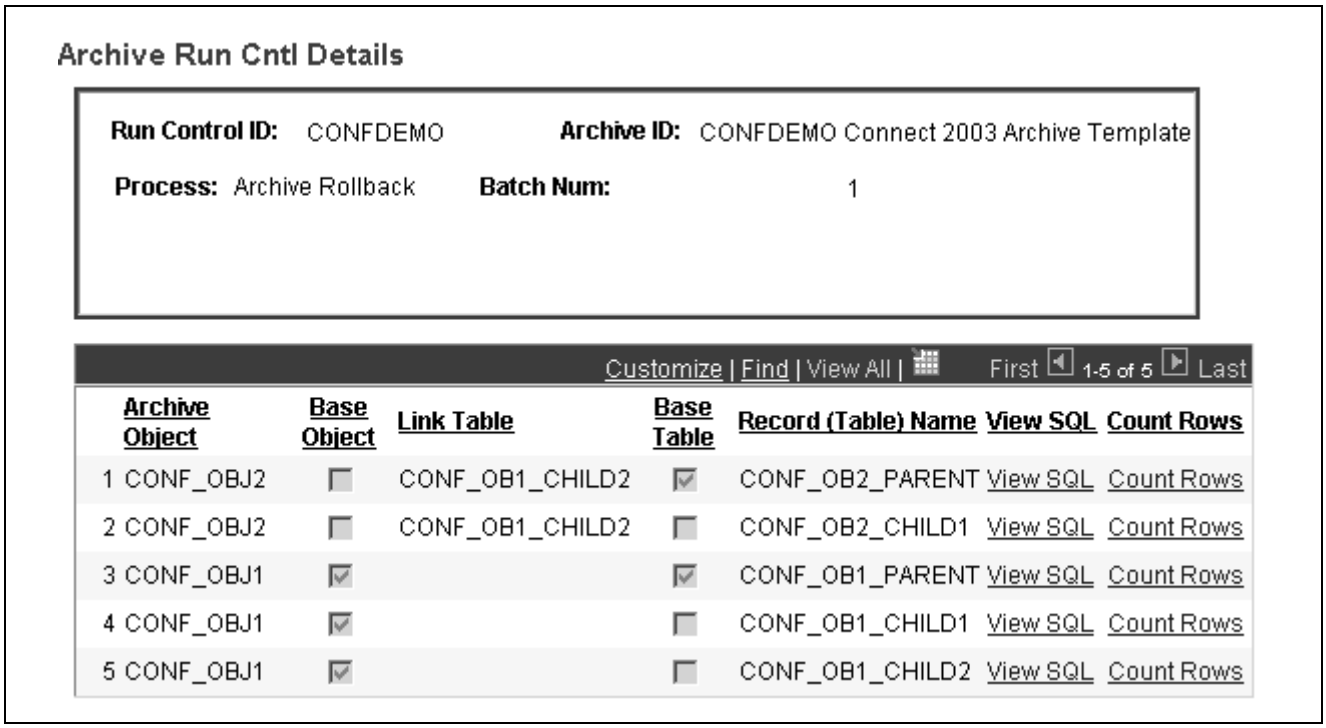
Row-based processing is appropriate when you're archiving large amounts of data from transactional tables and wish to issue commits more frequently. If you select this option, you must enter a commit frequency.

Commit Frequency

Specify the number of rows to process before issuing a commit to the database.

Viewing Details

Access the Archive Run Cntl Details page.



Archive Run Cntl Details page

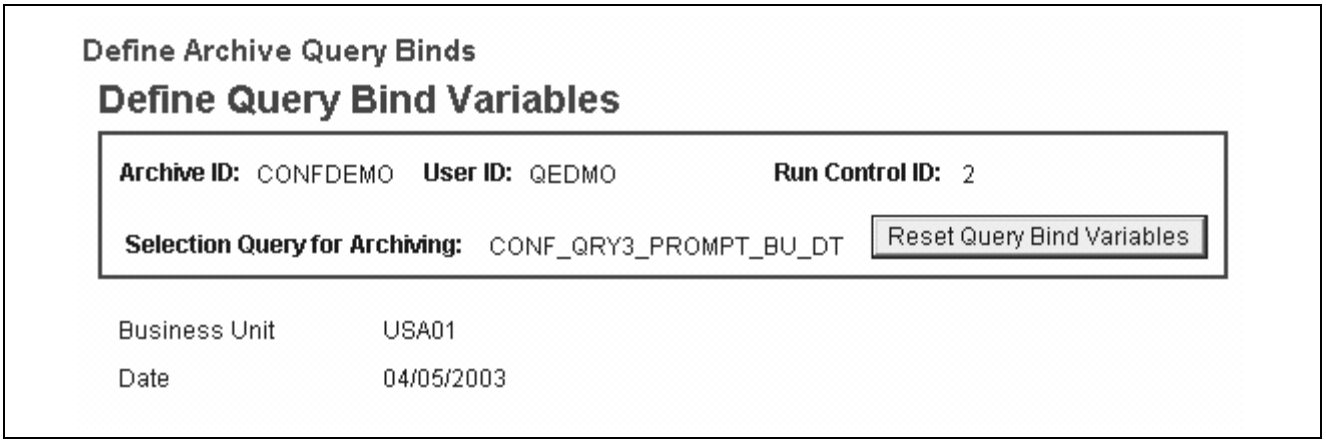
View SQL Select to view the archive selection SQL for the archive object. The View Details page appears, with a text box containing the SQL, for example:

```
%InsertSelect(CONF_OB2_PARENT, CONF_OB2_PAR_HS)⇒
 FROM PS_CONF_OB2_PAR_HS WHERE⇒
 PSARCH_ID = 'CONFDEMO' AND PSARCH_BATCHNUM = 1
```

Count Rows Select to view the number of rows of the archive object that the archiving process will affect in the related database. The View Details page appears, with a description of the number of rows that will be processed by Data Archive Manager.

Defining Archive Query Binds

Access the Define Query Bind Variables page.



Define Query Bind Variables page

Click the Reset Query Bind Variables button, and a prompt page appears where you can enter the new query bind values. The prompt page appears only if you have defined prompts for the selection query that you use for the job. When you enter the query bind values and click OK, they appear as read-only information on the Define Query Bind Variables page. Click OK to return to the Archive Data To History page.

Auditing Archive Processes

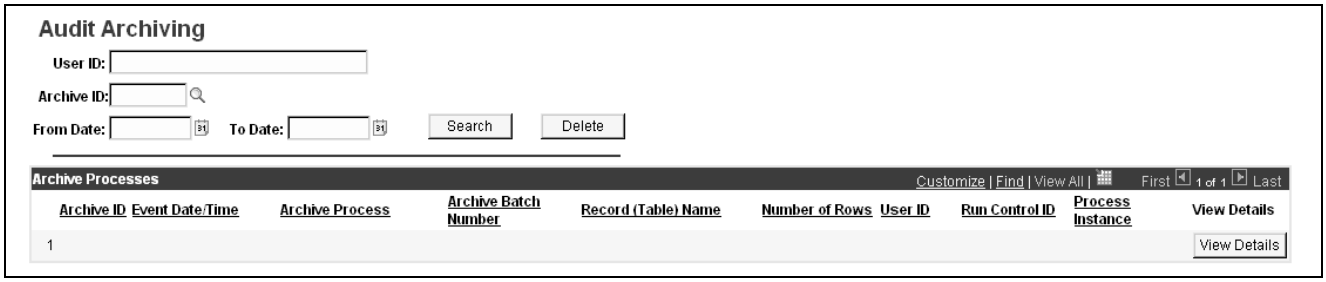
This section discusses how to audit the details of previous archiving processes:

Page Used to Audit Archive Processes

Page Name	Object Name	Navigation	Usage
Audit Archiving	PSARCHIVEAUDIT	PeopleTools, Data Archive Manager, Audit Archiving	Use this page to view details of previous archiving processes.

Audit Archiving

Access the Audit Archiving page.



Audit Archiving page

User ID Select which user to audit.

Archive ID Select an existing archive ID to audit.

From Date Select a start date for the audit.

To Date Select an ending date for the audit.

Search Click this button to have the system create the audit report and display the appropriate fields on the page.

Delete Click this button to purge audited rows based on the criteria specified.

Archive ID Select an existing archive ID.

Event Date/Time Displays the date and time that corresponds to the date when the data was archived for that particular archive number.

Archive Process Displays the archive process you want to run.

Archive Batch Number Displays the batch number of the archive process.

Record (Table) Name Displays the name of the table that you want to archive.

Number of Rows Displays the number of rows to be archived.

User ID Displays the user ID that you want to audit.

Run Control ID A unique ID to associate each user with his or her own run control table entries.

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.

View Details Click this button to view the SQL detail of previous archiving processes.

CHAPTER 4

Ensuring Data Integrity

This chapter provides an overview of data integrity tools and discusses how to:

- Run SQL Alter (Structured Query Language alter).
- Run DDDAudit.
- Run SYSAUDIT.

Understanding Data Integrity Tools

PeopleSoft provides several tools to ensure the integrity of the data that is stored in the PeopleSoft system, such as SQL Alter , SYSAUDIT, and DDDAUDIT. You may want to use these tools during upgrades and system customizations, to verify the PeopleSoft system and check how it compares to the actual SQL objects.

Running SQL Alter

The primary purpose of the PeopleSoft Application Designer SQL Alter function is to bring SQL tables into accordance with PeopleTools record definitions. You can run SQL Alter in an audit-only mode that alerts you to discrepancies between record definitions and SQL tables, but that doesn't actually perform an alter.

To audit tables or views:

1. In PeopleSoft Application Designer, choose the records that you want to audit.

   You have the option of auditing the active record definition, the selected records in the project workspace, or all the records that are in the current project.

2. Select the Build menu and select the appropriate option for the records that you want to audit.

   If you're auditing an open record definition, choose Build, Current Object. If you select one or more records in the project workspace, you can select Build, Selected Objects. If you want to audit all records in the current project, select Build, Project.

   The Build Scope shows a list of all the records that are affected, or audited in the case.

3. Select Alter tables as the Build Option and select Build script file as the Build Execute option.
4. Click Settings and choose the Alter tab in the Build Settings dialog.
5. In the Alter Any group box, select the situations for which you want an Alter performed.
6. Select the Scripts tab.

You use the Scripts tab to specify the output for the build scripts in one file, in two files, where the file is generated, and so on.

7. Select Write Alter comments to script.

   Performing alters with this option enabled adds comments to the SQL script about what fields are being manipulated.

8. Choose the other script file options.
9. Click OKto close the Build Settings dialog and return to the Build dialog.
10. Press Buildon the Build dialog.

Understanding Table and Column Audits

The SELECT statements that are produced by auditing with SQL Alter deal with inconsistencies between PeopleTools tables and SQL in the definition of tables or columns. A SQL table is equivalent to a record in PeopleSoft Application Designer, and a column is equivalent to a field.

To fix problems that are found in the system tables and columns, you need to know how PeopleSoft field types correspond to SQL data types:

Application Designer Field Type	SQL Data Type	SQL Description
Character	CHAR	Alphanumeric; fixed length.
Long character	LONGVAR	Alphanumeric; variable length.
Date	DATE	Dates; stored as fixed length; displayed in various formats.
Number or signed number	SMALLINT	Numeric; integers only (no decimals); 1 to 4 digits (and 5 digits if RawBinary).
Number or signed number	INTEGER	Numeric; integers only (no decimals); 5 to 9 digits (and 10 digits if RawBinary).
Number or signed number	DECIMAL	Numeric; either (1) 10 or more digits or (2) contains decimal positions.

Note. In PeopleSoft Application Designer, if a field is specified as required, or if a field is numeric and does not have a format of Phone, SSN (social security number), or SIN, you need to initialize the starting value of the column and specify the NOT NULL attribute in SQL.

Running DDDAUDIT

This section discusses DDDAUDIT queries.

The Database Audit Report (DDDAUDIT) finds inconsistencies between PeopleTools record and index definitions and the database objects. This audit consists of nine queries: four on tables, two on views, and three on indexes.

Note. This SQR refers to the Data Designer, the PeopleTool that allowed you to create record definitions in the PeopleTools releases prior to release 7. Now, all of the development tools are incorporated into one integrated development environment called the PeopleSoft Application Designer.

To run DDDAUDIT:

1. Using Windows Explorer, navigate to <PS_HOME>\sqr and locate DDDAUDIT.SQR.
2. Double-click it.
3. Type in the database name, username, and password.

   You probably need to use the database access ID and password to execute the DDDAUDIT properly.
4. Verify the Report arguments and click OK.

   The f argument indicates where the system writes the .LIS file.
5. At the Command Prompt, press ENTER.

   At the end of a successful run, you're prompted to press Enter again.

When you run DDDAUDIT.SQR, its results are written to a file called DDDAUDIT.LIS in the \TEMP folder. After running DDDAUDIT, view the .LIS file by using any text editor. Here's a sample excerpt of this file:

DDDAUDIT Queries

The following table lists the names of each query that DDDAUDIT performs on the PeopleSoft system, what it means if rows are returned, and how to resolve the inconsistency.

Note. The query names in this table are arranged alphabetically, and are not necessarily in the order in which they appear in DDDAUDIT.LIS:

Query	If Rows are Returned?	Resolution
INDEX-1	Indexes are defined in PeopleSoft Application Designer and not found in the database.	Use PeopleSoft Application Designer to create the index.
INDEX-2	Indexes are defined in the database and not found in PeopleSoft Application Designer.	If the index is valid, use PeopleSoft Application Designer to define the index. Otherwise, drop the index.
INDEX-3	Uniqueness or the number of keys in the Index Definition do not match between PeopleSoft Application Designer and the database.	See INDEX-1.

Query	If Rows are Returned?	Resolution
TABLE-1	SQL table names are defined in the Data Designer that are not blank and not the same as the record name.	Use Application Designer to enter the record name as the Non-Standard SQL Table Name.
TABLE-2	SQL tables are defined in the Data Designer and not found in the database.	If you want to delete the record definition, use PeopleSoft Application Designer (select File, Delete). Otherwise, to create the SQL table, use Application Designer. This command also creates the appropriate indexes for keys, duplicate order keys, alternate keys, and list items.
TABLE-3	SQL tables are defined in the database and not found in the Data Designer. SYSINDEXES and SYSTABLES can be ignored in these results. For Informix: PSALTERLONG can also be ignored.	If the table is not valid, drop it. Otherwise, define a new record in PeopleSoft Application Designer.
TABLE-4	Tablespace is not defined for the SQL table in PeopleSoft Application Designer.	If you're using or migrating to a relational database management system that uses table spaces, you should use PeopleSoft Application Designer to assign table spaces to these tables.
TABLE-5	Table contains more than 250 fields.	Use PeopleSoft Application Designer to adjust the number of fields on the table, as needed.
VIEWS-1	Views are defined in the Data Designer and not found in the database.	If you want to delete the view definition, use PeopleSoft Application Designer (select File, Delete). Otherwise, to create the SQL view, use Application Designer.
VIEWS-2	Views are defined in the database and not found in the Data Designer.	If the view is not valid, Drop it. Otherwise, define a new view in PeopleSoft Application Designer.
TRIGGER-1	Trigger defined in the PeopleSoft Application Designer and not found in the database.	Delete the definition if it is not needed. Otherwise, use PeopleSoft Application Designer to create the trigger in the database.

Running SYSAUDIT

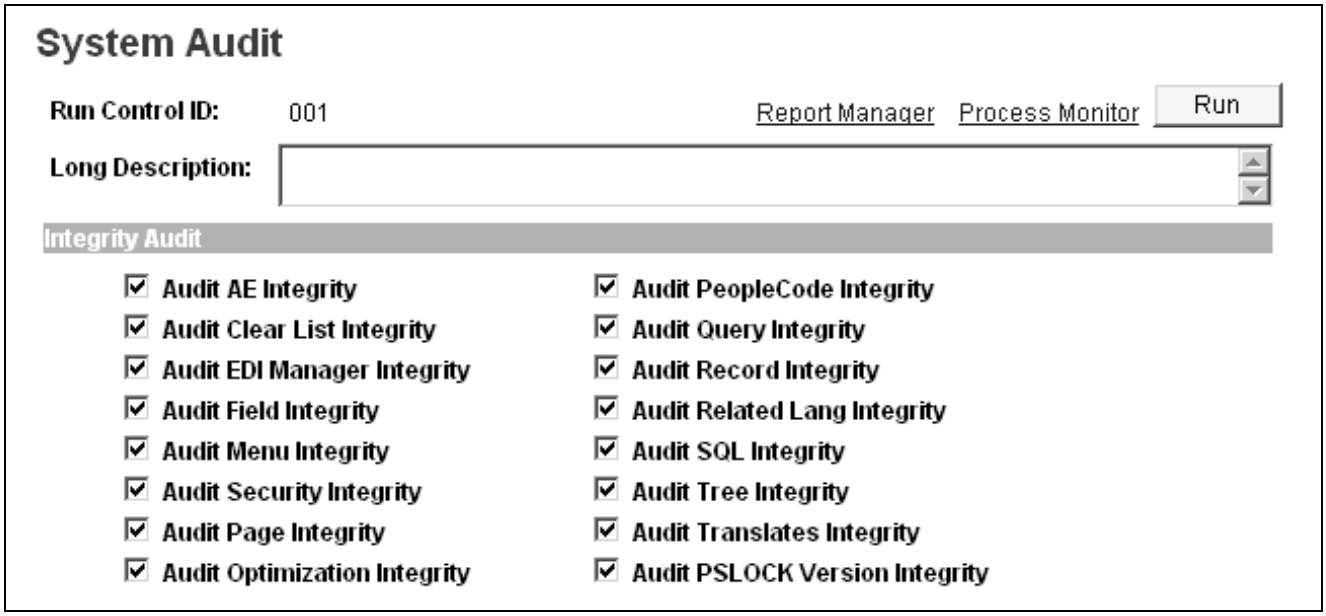
This section provides an overview of how to run SYSAUDIT and discusses audits for:

- Application Engine integrity.
- Clear list integrity.
- EDI Manager integrity.
- Field integrity.
- Menu integrity.
- Security integrity.
- Page integrity.
- Optimization integrity.
- PeopleCode integrity.
- Process Scheduler.
- Query integrity.
- Record integrity
- Related language integrity.
- SQL integrity.
- Tree integrity.
- Notes for TREE-09.
- Notes for TREE-22.
- Translate integrity.
- PSLOCK integrity.

Understanding How to Run SYSAUDIT

The System Audit (SYSAUDIT) identifies orphaned PeopleSoft objects and other inconsistencies within the system. An example of an orphaned object is a module of PeopleCode that exists, but which does not relate to any other objects in the system.

Select PeopleTools, Utilities, Audit, Perform System Audit.



System Audit page

Audit AE Integrity Audits PeopleSoft Application Engine program definitions and components.

Audit Clear List Integrity Audits the SYSCLRLIST* component.

Audit EDI Manager Integrity Audits the EC* component for EDI Manager.

Audit Field Integrity Audits the DBFLD* component for PeopleSoft Application Designer fields.

Audit Menu Integrity Audits the MENU* component for PeopleSoft Application Designer menus.

Audit Security Integrity Audits the AUTH*, OPRDF* components for PeopleTools Security.

Audit Page Integrity Audits the PNL* component for PeopleSoft Application Designer pages.

Audit Optimization Integrity Audits the definitions for Optimization Engine.

Audit PeopleCode Integrity Audits the PCM* and PRG* components for PeopleCode programs.

Audit Query Integrity Audits the QRY* component for PeopleSoft Query.

Audit Record Integrity Audits the REC* and VIEWT* components for PeopleSoft Application Designer records.

Audit Related Lang Integrity Audits Related Language Integrity. Query the *LANG component.

Audit SQL Integrity Audits the referential integrity of the tables supporting SQL objects in the db component.

Audit Tree Integrity Audits the TREE* component.

Audit Translates Integrity Audits the XLAT* component.

Audit PSLOCKS Version Integrity Audits the VERSN* component.

To run SYSAUDIT:

1. Select PeopleTools, Utilities, Audit, Perform System Audit.
2. When prompted, enter a new run control ID and click OK.
3. Select the desired Integrity Audit options.
4. Click Run.
5. Select the appropriate settings on the Process Scheduler Request page, and click .OK.

Understanding SYSAUDIT Output

When you run SYSAUDIT, the results are written to the Data Mover output file. For best viewing, PeopleSoft recommends opening the output file in a text editor.

The following table lists the names of each of the audit queries that SYSAUDIT performs on the PeopleSoft system, what it means if rows are returned, and how to resolve the discrepancies that the audit report uncovers.

Note. The query names in this table are arranged alphabetically, and are not necessarily in the order in which they appear in the output.

Application Engine Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
AE-01	AE programs without any sections.	If the affected program is delivered by PeopleSoft and is not modified, contact the GSC. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. Otherwise, use the Application Engine designer to either create valid sections for the program or remove the program. It is not possible to recover the missing sections.
AE-02	AE sections without AE programs.	If the affected program is delivered by PeopleSoft and is not modified, contact the GSC. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. If the affected program is a customization, it is not possible to recover the missing program. Restore it from a backup if needed. Use SysAECleanUp.dms (located in PS_HOME \scripts.) to remove any orphans remaining after you have followed the steps above.

Query	Description	Resolution
AE-03	AE state records without AE programs.	If the affected record is delivered by PeopleSoft, contact the GSC. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. Otherwise, ignore the warnings or restore the program from a backup. It is not possible to recover the missing program.
AE-04	AE state records without record definitions.	If the affected record is delivered by PeopleSoft, contact the GSC. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. Otherwise, using PeopleTools Application Designer, remove invalid records from the program definition or create record definitions.
AE-05	AE section details without base section definitions.	If the affected program is delivered by PeopleSoft and is not modified, contact the GSC. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. Otherwise, ignore the warnings or restore the program from a backup. It is not possible to recover the missing sections.
AE-06	AE steps without sections.	If the affected program is delivered by PeopleSoft and is not modified, contact the GSC. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. If the affected program is a customization, it is not possible to recover the missing program. Restore it from a backup if needed. Use SysAECleanUp.dms (located in PS_HOME \scripts.) to remove any orphans remaining after you follow the steps above.

Query	Description	Resolution
AE-07	AE Call Section actions referring to nonexistent sections	If the affected program is delivered by PeopleSoft and is not modified, contact the GSC. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. Otherwise, use the Application Engine either to open the program containing the Call Section and change it to call the correct section, or create the required section.
AE-08	AE Log Message actions without an AE step.	If the affected record is delivered by PeopleSoft, contact the GSC. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. If the affected program is a customization, it is not possible to recover the missing program; restore it from a backup if needed. Use SysAECleanUp.dms (located in PS_HOME \scripts.) to remove any orphans remaining after you follow the steps above.
AE-09	AE actions without an AE step.	If the affected record was delivered by PeopleSoft, contact the GSC. If the affected program was converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. If the affected program is a customization, it is not possible to recover the missing program; restore it from a backup if needed. Use SysAECleanUp.dms (located in PS_HOME \scripts.) to remove any orphans remaining after you follow the steps above.
AE-10	AE temp tables that are attached to invalid AE programs.	If the affected temp table was delivered by PeopleSoft, contact the GSC. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact the GSC. Otherwise, ignore the warnings or restore the program from a backup. It is not possible to recover the missing programs.

Query	Description	Resolution
AE-11	Orphaned AE PeopleCode.	Because of platform issues and Structured Query Report (SQR), this check may not be included in the audit report. But SysAECleanUp.dms cleans up these orphans.
AE-12	Orphaned AE SQL objects.	Because of platform issues and SQR, this check may not be included in the audit report. But SysAECleanUp.dms cleans up these orphans.
AE-13	Verify that enough rows are loaded into PS_AEONLINEINST.	Contact the GSC. This table is a critical component of the Application Engine Runtime.
AE-14	Verify that enough rows are loaded into PS_AEINSTANCENBR.	Contact the GSC. This table is a critical component of the Application Engine Runtime.
AE-15	Verify that a row is loaded into PS_AELOCKMGR.	Contact the GSC. This table is a critical component of the Application Engine Runtime.

Clear List Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
SYSCLRLIST-01	Entries in PSACTIVITYDEL and PSACTIVITYDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-02	Entries in PSAEAPPLDEL and PSAEAPPLDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-05	Entries in PSCOLORDEL and PSCOLORDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-06	Entries in PSFMTDEL and PSFMTDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-07	Entries in PSHOLIDAYDEL and PSHOLIDAYDEFN are not mutually exclusive.	Run the VERSION Application Engine program.

Query	Description	Resolution
SYSCLRLIST-09	Entries in PSIMPDEL and PSIMPDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-10	Entries in PSMENUDEL and PSMENUDEFN are not mutually exclusive	Run the VERSION Application Engine program.
SYSCLRLIST-11	Entries in PSPCMPROGDEL and PSPCMPROG are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-12	Entries in PSPNLDEL and PSPNLDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-13	Entries in PSPNLGRPDEL and PSPNLGRPDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-14	Entries in PSPRCSRUNCDEL and PSPRCSRUNCNTL are not mutually exclusive	Run the VERSION Application Engine program.
SYSCLRLIST-15	Entries in PSPROJECTDEL and PSPROJECTDEFN are not mutually exclusive	Run the VERSION Application Engine program.
SYSCLRLIST-16	Entries in PSQRYDEL and PSQRYDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-17	Entries in PSRECDEL and PSRECDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-18	Entries in PSRECURDEL and PS_PRCSRECUR are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-19	Entries in PSSTYLEDEL and PSSTYLEDEFN are not mutually exclusive.	Run the VERSION Application Engine program.

Query	Description	Resolution
SYSCLRLIST-20	Entries in PSTOOLBARDEL and PSTOOLBARDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-21	Entries in PSTREEBRADEL and PSTREEBRANCH are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-22	Entries in PSTREEDEL and PSTREEDEFN are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-23	Entries in PSTREESTRDEL and PSTREESTRCT are not mutually exclusive.	Run the VERSION Application Engine program.
SYSCLRLIST-24	Entries in XLATTABLEDEL and XLATTABLE are not mutually exclusive.	Run the VERSION Application Engine program.

EDI Manager Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
ECINMPFL-1	Inbound work records that are not found in the PSRECDEFN table.	Either modify the inbound map definition to not use the Inbound Row ID Work Record (ECINMAPFILE), or create the Work Record Definition.
ECINMPFL-2	Inbound work record EC Map ID is not found in the PS_ECMAPDEFN table.	Create an entry in the map definition table (ECMAPDEFN).
ECINMPFD-1	Inbound work record fields are not found with valid EC Map ID and EC File Row ID combination from the PS_ECINMAPFILE table	Either remove the invalid map ID from the Inbound Work Record (ECINMAPFLD), or create an Inbound Row ID Work Record entry.
ECINMPFD-2	Inbound work record fields from PS_ECINMAPFLD are not found in PSRECFIELD	Either remove the invalid entry in the inbound work record or create the record/field definition.

Query	Description	Resolution
ECINMPRC-1	Target inbound records are not found in the PSRECDEFN table	Either modify the inbound map definition to not use the Inbound Row ID Target Record (ECINMAPREC), or create the Work Record Definition.
ECINMPRC-2	Target inbound EC Map ID is not found in the PS_ECMAPDEFN table	Either remove the invalid map ID from the Inbound Row ID Target Record or create an entry in the Map Definition table.
ECINMPRF-1	EC Map ID/EC File Row ID combination is not found in PS_ECINMAPREC for the target inbound record field in PS_ECINMAPRECFLD	Remove the invalid map ID from the Inbound Target Record.
ECINMPRF-2	A Field for a Record in PS_ECINMAPRECFLD was not found in PSRECFIELD	Create the appropriate definitions in PSRECFIELD or remove the invalid map ID from the Inbound Target Record.
ECINMPRF-4	A related record in PS_ECINMAPRECFLD is not found in PSRECDEFN	Either create the record definition or remove the reference to the related record in the Inbound Target Record.
ECINMPRF-5	An EC Related Record in PS_ECINMAPRECFLD does not have a valid EC Related Row ID from PS_ECINMAPREC	Either remove the reference to the related record from the Inbound Target Record or create an appropriate entry in the Inbound Row ID Target Record.
ECINMPRF-6	A related field in PS_ECINMAPRECFLD is not found in PSRECFIELD	Either remove or correct the reference to the related field record from the Inbound Target Record or create the correct definition in PSRECFIELD.
ECOTMPRC-1	Target outbound records are not found in the PSRECDEFN table	Either modify the outbound map definition to not use the Outbound Target Record, or create the record definition.
ECOTMPRC-2	Outbound work record EC Map ID is not found in the PS_ECMAPDEFN table	Create an entry in the map definition table.
ECOTMPRC-3	Parent records from the outbound work record are not found in the PSRECDEFN table	Remove the reference to the parent record or create a record definition for the parent.

Query	Description	Resolution
ECOTMPRC-4	File records from the outbound work record are not found in the PSRECDEFN table	Create a record definition for the file record.
ECOTMPFD-1	Outbound work record fields are not found with a valid EC Map ID and EC File Row ID combination from the PS_ECOUTMAPREC table	Either remove the entry from the Outbound Work Record (ECOUTMAPFLD) or create an entry in the Outbound Target Record (ECOUTMAPREC).
ECOTMPFD-2	Outbound work record fields from PS_ECOUTMAPFLD are not found in PSRECFIELD	Create the appropriate definitions in PSRECFIELD or remove the invalid map ID from the Outbound Work Record.
SYSECMGR-1	Inbound work record field does not exist in the type definitions in PSDBFIELD	Call PeopleSoft GSC for resolution.

Field Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
FIELD-3	The following default fields are invalid.	Modify the default value in record field properties.
FIELD-4	Fields used in record definitions that do not exist in PSDBFIELD	Define the field in PeopleSoft Application Designer.
FIELD-5	Fields have multiple default field labels in PSDBFLDLABL.	Open the field, select default label, and resave.
FIELD-06	Deleted fields have orphaned field labels in PSDBFLDLABL.	`DELETE FROM PSDBFLDLABL WHERE FIELDNAME NOT IN (SELECT FIELDNAME FROM PSDBFIELD)`

Menu Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
MENU-01	A row in the MenuItem table has no corresponding row in the MenuDefinition table.	Issue the following SQL: `DELETE FROM PSMENUITEM WHERE MENUNAME = 'x';`
MENU-02	A component-type menu item specifies no component.	Use the Menu Designer to change each of these menu items to reference an existing component.
MENU-03	A menu item has a specified component, but that component has no corresponding row in the ComponentDefinition table.	Use the Menu Designer to change each of these menu items to reference an existing component.
MENU-04	A PeopleCode-type menu item has a specified enabling component, but that component is not specified for any component-type menu item within the same menu. (Such menu items never get enabled at runtime.)	Use the Menu Designer to change each of these menu items to reference a component that is associated with a component-type menu item within the same menu.
MENU-05	A menu has no rows in the MenuItem table.	Use the Menu Designer to add any appropriate menu items to each of these menus.

Security Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
SEC-1	Authorized Signon Operator does not exist in the Class Definition table. Incomplete permission list: Orphan signon times: (Verifies the existence of permission lists owning signon times.)	Delete the extra signon times. If this is a permission list that should exist, recreate it through PeopleTools Security. `DELETE FROM PSAUTHSIGNON WHERE CLASSID='x'`
SEC-2	Incomplete permission list: Orphan page permissions: (Verifies the existence of permission lists owning page permissions.)	Delete the extra page permissions. If this is a permission list that should exist, recreate it through PeopleTools Security. `DELETE FROM PSAUTHITEM WHERE CLASSID='x'`

Query	Description	Resolution

SEC-3	Incomplete permission list: Orphan process groups:
	
	(Verifies existence of permission lists owning process groups.)
		Delete the extra process group authorizations. If this is a permission list that should exist, recreate it through PeopleTools Security.
		
		`DELETE FROM PSAUTHPRCS`
		`WHERE CLASSID='x'`

SEC-4	Incomplete permission list: Orphan process profiles:
	
	(Verifies existence of permission lists owning process profiles.)
		Delete the extra process profiles. If this is a permission list that should exist, recreate it through PeopleTools Security.
		
		`DELETE FROM PSPRCSPRFL`
		`WHERE CLASSID='x'`

SEC-5	Permission list references a nonexistent process group:
	
	(Verifies the existence of process groups.)
		Delete the extraneous process groups. If this group should exist, recreate it.
		
		`DELETE FROM PSAUTHPRCS`
		`WHERE CLASSID='x' AND`
		`PRCSGRP = 'y'`

SEC-6	User profile references a role that does not exist:	Open the user profile in PeopleTools Security and remove the reference to the Role that does not exist.
SEC-7	Role references a permission list that does not exist:	Open the Role in PeopleTools Security and remove the reference to the Permission List that does not exist.
SEC-8	Role references a user that does not exist in the PSOPRDEFN table.	Remove the user from the PSROLEUSER table.
SEC-9	Permission list references a role that does not exist in the PSROLEDEFN table.	Remove the role from the PSROLECLASS table.

Query	Description	Resolution
SEC-28	Invalid entries in the PSAUTHITEM table. (Continues in next row)	Run the following SQL: DELETE FROM PSAUTHITEM WHERE (PSAUTHITEM.MENUNAME NOT LIKE 'WEBLIB_%'⇒ AND PSAUTHITEM.MENUNAME NOT IN⇒ ('CLIENTPROCESS', 'DATA_MOVER',⇒ 'IMPORT_MANAGER', 'OBJECT_SECURITY',⇒ 'QUERY', 'PERFMONPPMI')⇒ AND PSAUTHITEM.MENUNAME NOT LIKE⇒ ('APPLICATION_DESIGNER%')⇒ AND PSAUTHITEM.MENUNAME <>'REN'⇒ AND NOT EXISTS (SELECT 'X' FROM PSMENUITEM MI⇒ WHERE PSAUTHITEM.MENUNAME = MI.MENUNAME⇒ AND PSAUTHITEM.BARNAME = MI.BARNAME⇒ AND PSAUTHITEM.BARITEMNAME = MI.ITEMNAME⇒ AND (MI.ITEMTYPE IN (0, 1, 2, 3, 4, 6, 7, 8, 10, 11)⇒ OR (MI.ITEMTYPE = 5 AND EXISTS⇒ (SELECT 'X' FROM PSPNLGRPDEFN GD, PSPNLGROUP GI⇒ WHERE MI.PNLGRPNAME = GD.PNLGRPNAME⇒ AND MI.MARKET = GD.MARKET⇒ AND GD.PNLGRPNAME = GI.PNLGRPNAME⇒ AND GD.MARKET = GI.MARKET⇒ AND PSAUTHITEM.PNLITEMNAME = GI.ITEMNAME))⇒
SEC-28	(Continued)	OR (MI.ITEMTYPE = 9 AND EXISTS⇒ (SELECT 'X' FROM PSPCMNAME PCN, PSPCMPROG PCP⇒ WHERE PCN.OBJECTID1 = 3⇒ AND PCN.OBJECTVALUE1 = MI.MENUNAME⇒ AND PCN.OBJECTID2 = 4⇒ AND PCN.OBJECTVALUE2 = MI.BARNAME⇒ AND PCN.OBJECTID3 = 5⇒ AND PCN.OBJECTVALUE3 = MI.ITEMNAME⇒ AND PCN.OBJECTID4 = 12⇒ AND PCN.OBJECTVALUE4 = 'ItemSelected'⇒ AND PCN.OBJECTID1 = PCP.OBJECTID1⇒ AND PCN.OBJECTVALUE1 = PCP.OBJECTVALUE1⇒ AND PCN.OBJECTID2 = PCP.OBJECTID2⇒ AND PCN.OBJECTVALUE2 = PCP.OBJECTVALUE2⇒ AND PCN.OBJECTID3 = PCP.OBJECTID3⇒ AND PCN.OBJECTVALUE3 = PCP.OBJECTVALUE3⇒ AND PCN.OBJECTID4 = PCP.OBJECTID4⇒ AND PCN.OBJECTVALUE4 = PCP.OBJECTVALUE4))⇒ OR (MI.ITEMTYPE = 12⇒ AND EXISTS (SELECT 'X' FROM PSXFERITEM XI⇒ WHERE MI.MENUNAME = XI.MENUNAME⇒ AND MI.ITEMNAME = XI.ITEMNAME)))))⇒

Query	Description	Resolution
SEC-28	(Continued)	OR (PSAUTHITEM.MENUNAME LIKE 'WEBLIB_%'⇒ AND NOT EXISTS (SELECT 'X' FROM PSPCMPROG PCP⇒ WHERE PCP.OBJECTID1 = 1⇒ AND PCP.OBJECTVALUE1 = PSAUTHITEM.MENUNAME⇒ AND PCP.OBJECTID2 = 2⇒ AND PCP.OBJECTVALUE2 = PSAUTHITEM.BARNAME))⇒ OR (PSAUTHITEM.MENUNAME IN ('CLIENTPROCESS',⇒ 'DATA_MOVER', 'IMPORT_MANAGER',⇒ 'OBJECT_SECURITY', 'QUERY', 'PERFMONPPMI')⇒ AND (PSAUTHITEM.BARNAME <> ' '⇒ OR PSAUTHITEM.BARITEMNAME <> ' '⇒ OR PSAUTHITEM.PNLITEMNAME <> ' '))⇒ OR (PSAUTHITEM.MENUNAME LIKE⇒ ('APPLICATION_DESIGNER%')⇒ AND ((PSAUTHITEM.BARNAME <> ' '⇒ AND PSAUTHITEM.BARNAME NOT IN⇒ (SELECT OBJNAME FROM PS_APP_DES_OBJECTS⇒ WHERE PSAUTHITEM.BARNAME = OBJNAME))⇒ OR PSAUTHITEM.BARITEMNAME <> ' '⇒ OR PSAUTHITEM.PNLITEMNAME <> ' '))⇒ OR (PSAUTHITEM.MENUNAME = 'REN'⇒ AND ((PSAUTHITEM.BARNAME <> ' '⇒ AND PSAUTHITEM.BARNAME NOT IN⇒ (SELECT OBJNAME FROM PS_APP_DES_OBJECTS⇒ WHERE PSAUTHITEM.BARNAME = OBJNAME))⇒ OR PSAUTHITEM.BARITEMNAME <> ' '⇒ OR PSAUTHITEM.PNLITEMNAME <> ' '))
SEC-29	The displayed PSPRSMPERM rows contain invalid PORTAL_PERMTYPE values.	Run the following SQL: DELETE from PSPRSMPERM⇒ where PORTAL_PERMTYPE = ' '⇒ and exists (select 'x' from PSPRSMPERM PP2⇒ where PSPRSMPERM.PORTAL_NAME = PP2.PORTAL_NAME⇒ and PSPRSMPERM.PORTAL_REFTYPE⇒ = PP2.PORTAL_REFTYPE⇒ and PSPRSMPERM.PORTAL_OBJNAME⇒ = PP2.PORTAL_OBJNAME⇒ and PSPRSMPERM.PORTAL_PERMNAME⇒ = PP2.PORTAL_PERMNAME⇒ and PP2.PORTAL_PERMTYPE <> ' '); UPDATE PSPRSMPERM set PORTAL_PERMTYPE = 'P'⇒ where PORTAL_PERMTYPE = ' '⇒ and exists (select 'x' from PSCLASSDEFN⇒ where CLASSID = PSPRSMPERM.PORTAL_PERMNAME);

Page Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution

PAGE-01	Page definition's page field count is not equal to the count of its page fields in the PageField table, and there is at least one row in the PageField table for that page.	Enter the following SQL:
		`SELECT COUNT(*) FROM PSPNLFIELD WHERE PNLNAME = 'x';`
		`UPDATE PSPNLDEFN SET FIELDCOUNT = count WHERE PNLNAME = 'x';`

PAGE-02	Page definition's page field count is not equal to zero, but there are no rows in the PageField table for that page definition.	Enter the following SQL: `UPDATE PSPNLDEFN SET FIELDCOUNT = 0 WHERE PNLNAME = 'x';`
PAGE-03	A subpage contains itself as a page field.	Use the Page Designer to change each of these page fields to reference a different subpage.
PAGE-04	A row in the PageField has no corresponding row in the PageDefinition table.	Issue the following SQL: `DELETE FROM PSPNLFIELD WHERE PNLNAME = 'x';`
PAGE-05	A subpage-type page field has no corresponding row in the Page Definition table for its specified subpage.	Use the Page Designer to change each of these page fields to reference an existing subpage.
PAGE-06	A page field's specified record/field has no corresponding row in the RecordField table.	Use the Page Designer to change each of these page fields to reference an existing record/field.
PAGE-07	A row in the ComponentItem table has no corresponding row in the ComponentDefinition table.	Issue the following SQL: DELETE FROM PSPNLGROUP WHERE PNLGRPNAME = 'x';
PAGE-08	A component item's specified page has no corresponding row in the PageDefinition table.	Use the Component Designer to replace each of these component items with one that references an existing page.
PAGE-09	A component's specified access detail page has no corresponding row in the PageDefinition table.	Use the Component Designer to change each of these components to reference an access detail page that exists.

Query	Description	Resolution
PAGE -10	A component's specified search record has no corresponding row in the RecordDefinition table.	Use the Component Designer to change each of these components to reference a search record that exists.
PAGE-11	A component's specified add search record has no corresponding row in the RecordDefinition table.	Use the Component Designer to change each of these components to reference an add search record that exists.

Optimization Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution

OPTZN-01	Problem type records that do not have matching record definitions.	Execute the following SQL:
		`DELETE FROM PSOPTREC WHERE RECNAME = '<RECORD NAME>';`
		`DELETE FROM PSOPTFIELD WHERE RECNAME = '<RECORD NAME>';`

OPTZN-02	Optimization delete records that do not have matching definitions.	In PeopleSoft Application Designer, open the base record definition properties. Clear the optimization delete record name, and perform an alter.
OPTZN-03	Optimization base record has fields that delete record does not.	Using PeopleSoft Application Designer, delete the optimization delete record definition, drop the table, and recreate it by cloning the base record. Run Build. You may need to recreate triggers on the base record on some platforms where deferred processing is not done.
OPTZN-04	Optimization delete record has fields that base record does not.	Using PeopleSoft Application Designer, delete the optimization delete record definition, drop the table and recreate it by cloning the base record. Run Build. You may need to recreate triggers on the base record on some platforms where deferred processing is not done.
OPTZN-06	Optimization base record defn has the trigger flag set but has no delete record name, or vice versa.	Using PeopleSoft Application Designer, open the record definition properties, make sure that the optimization delete record name is set, and save. Build the record with the create triggers check box set to create optimization triggers.

Query	Description	Resolution
OPTZN-07	Optimization records that need to have trigger flag set and do not.	Using PeopleSoft Application Designer, open the record definition properties, make sure that the optimization delete record name is set, and save. Build the record with the create triggers check box set to create optimization triggers.
OPTZN-08	Optimization records that have trigger flag set but are not marked readable in any problem type.	Using PeopleSoft Application Designer, open the record definition properties, clear the optimization delete record name and alter the record to drop optimization triggers as they are no longer needed but affect performance.
OPTZN-09	Optimization Tools table PSOPTREC is empty for the listed problem types.	In PeopleSoft Application Designer, open the problem type definition and select the Records tab. Make sure that the problem type definition contains a list of record names. If there are no records listed, the problem type definition needs to be corrected in order to add the list of records for that problem type. Save the problem type definition.
OPTZN-10	Optimization Tools table PSOPTSYNC does not have an entry for the listed opt records, that are marked READABLE in PSOPTREC.	Open the problem type definition in Application Designer, make sure that the readable flags are set correctly for each readable record, and save the problem type definition.
OPTZN-11	Optimization Tools table PSOPTSYNC does not have an entry with PROBINST = ALL and is marked as NON SCENARIO_MANAGED and READABLE in PSOPTREC.	Using Problem Type Designer, make sure that the readable flags are set correctly for each readable record. Make sure that the scenario_managed flags are set correctly. Save the problem type definition.

OPTZN-12	Optimization Tools table PSOPTSYNC has extra entries for the listed record names that are not there in PSOPTREC.	Submit the following SQL to remove extra entries in PSOPTSYNC table.
		`Delete PSOPTSYNC`
		`Where RECNAME NOT IN`
		`(SELECT RECNAME`
		`FROM PSOPTREC)`

Query	Description	Resolution
OPTZN-13	The following record names in Optimization Tools table PSOPTREC do not have at least one field listed in the PSOPTFIELD table.	Open the problem type definition in Application Designer. Make sure that for every record in the problem type definition at least one field is selected to be loaded in the problem instance.
OPTZN-14	For the following transaction parameter of type RECARRAY, the default value contains an invalid record name.	Open the problem type definition in PeopleSoft Application Designer. Inspect the offending transaction parameter and make sure that the default value contains a valid record name.
OPTZN-15	PSOPTSOLVERCODE table is empty for the listed problem types.	You may ignore this if none of the problem types need a third-party solver. Otherwise, populate the PSOPTSOLVERCODE table with the third-party solver license key. Select PeopleTools, Utilities, Optimization, Solver Licenses.
OPTZN-16	PSOPTSOLVERCODE table has a null licence key for the listed problem types.	You may ignore this if the plugin does not need a third-party solver. Otherwise populate the PSOPTSOLVERCODE table with the third-party solver licence key. Select PeopleTools, Utilities, Optimization, Solver Licenses.

PeopleCode Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution

PEOPLECODE-1	The PeopleCode Name table contains a program name that does not exist in PcmProgram table	Run the following SQL:
		`DELETE FROM PSPCMNAME`
		`WHERE NOT EXISTS⇒`
		`(SELECT 'X'FROM PSPCMPROG B⇒`
		`WHERE B.OBJECTID1 = PSPCMNAME.OBJECTID1⇒`
		`AND B.OBJECTVALUE1 = PSPCMNAME.OBJECTVALUE1⇒`
		`AND B.OBJECTID2 = PSPCMNAME.OBJECTID2⇒`
		`AND B.OBJECTVALUE2 = PSPCMNAME.OBJECTVALUE2⇒`
		`AND B.OBJECTID3 = PSPCMNAME.OBJECTID3⇒`
		`AND B.OBJECTVALUE3 = PSPCMNAME.OBJECTVALUE3⇒`
		`AND B.OBJECTID4 = PSPCMNAME.OBJECTID4⇒`
		`AND B.OBJECTVALUE4 = PSPCMNAME.OBJECTVALUE4⇒`
		`AND B.OBJECTID5 = PSPCMNAME.OBJECTID5⇒`
		`AND B.OBJECTVALUE5 = PSPCMNAME.OBJECTVALUE5⇒`
		`AND B.OBJECTID6 = PSPCMNAME.OBJECTID6⇒`
		`AND B.OBJECTVALUE6 = PSPCMNAME.OBJECTVALUE6)`

Query	Description	Resolution

PEOPLECODE-2	The PeopleCode Program table contains a program name that does not exist in the PcmName table.	Run the following SQL:
		`DELETE FROM PSPCMPROG`
		`WHERE NAMECOUNT <> 0 AND NOT EXISTS⇒`
		`(SELECT 'X'FROM PSPCMNAME B⇒`
		`WHERE PSPCMPROG.OBJECTID1 = B.OBJECTID1⇒`
		`AND PSPCMPROG.OBJECTVALUE1 = B.OBJECTVALUE1⇒`
		`AND PSPCMPROG.OBJECTID2 = B.OBJECTID2⇒`
		`AND PSPCMPROG.OBJECTVALUE2 = B.OBJECTVALUE2⇒`
		`AND PSPCMPROG.OBJECTID3 = B.OBJECTID3⇒`
		`AND PSPCMPROG.OBJECTVALUE3 = B.OBJECTVALUE3⇒`
		`AND PSPCMPROG.OBJECTID4 = B.OBJECTID4⇒`
		`AND PSPCMPROG.OBJECTVALUE4 = B.OBJECTVALUE4⇒`
		`AND PSPCMPROG.OBJECTID5 = B.OBJECTID5⇒`
		`AND PSPCMPROG.OBJECTVALUE5 = B.OBJECTVALUE5⇒`
		`AND PSPCMPROG.OBJECTID6 = B.OBJECTID6⇒`
		`AND PSPCMPROG.OBJECTVALUE6 = B.OBJECTVALUE6)`

PEOPLECODE-3	The PeopleCode Program table name count does not match the record count in PcmName table.	Run the following SQL:
		`UPDATE PSPCMPROG⇒`
		`SET NAMECOUNT = (SELECT COUNT(*)FROM PSPCMNAME C⇒`
		`WHERE C.OBJECTID1 = PSPCMPROG.OBJECTID1⇒`
		`AND C.OBJECTVALUE1 = PSPCMPROG.OBJECTVALUE1⇒`
		`AND C.OBJECTID2 = PSPCMPROG.OBJECTID2⇒`
		`AND C.OBJECTVALUE2 = PSPCMPROG.OBJECTVALUE2⇒`
		`AND C.OBJECTID3 = PSPCMPROG.OBJECTID3⇒`
		`AND C.OBJECTVALUE3 = PSPCMPROG.OBJECTVALUE3⇒`
		`AND C.OBJECTID4 = PSPCMPROG.OBJECTID4⇒`
		`AND C.OBJECTVALUE4 = PSPCMPROG.OBJECTVALUE4⇒`
		`AND C.OBJECTID5 = PSPCMPROG.OBJECTID5⇒`
		`AND C.OBJECTVALUE5 = PSPCMPROG.OBJECTVALUE5⇒`
		`AND C.OBJECTID6 = PSPCMPROG.OBJECTID6⇒`
		`AND C.OBJECTVALUE6 = PSPCMPROG.OBJECTVALUE6)`

PEOPLECODE-4	PeopleCode contains invalid FILELAYOUT References.	Open the PeopleCode program in PeopleSoft Application Designer and correct the invalid reference.
PEOPLECODE-5	PeopleCode reference to an invalid record or field.	Open the PeopleCode program in PeopleSoft Application Designer and correct the invalid reference.
PEOPLECODE-6	PeopleCode reference to an invalid field.	Open the PeopleCode program in PeopleSoft Application Designer and correct the invalid field name.

Process Scheduler

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
PRCSSCHED-01	SQR-Related Process Definitions (PS_PRCSDEFN) that override the PARMLIST field from the Process Type Definition (PS_PRCSTYPEDEFN).	For the listed processes, select Process Scheduler, Processes, Override Options. Remove the value that is assigned to the Parameter List field. **Note.** This PRRSCHED-01 is intended to be a warning. If the override of the parameter list that is specified in the process type definition is intentional, then the above action can be bypassed.
PRCSSCHED-03	Process Definitions (PS_PRCSDEFN), where the OUTDESTTYPE should be set to NONE.	For the listed processes, select Process Scheduler, Processes Destination. In the Output Destination Options group, set the Type option to (None).
PRCSSCHED-04	Process Definitions, where the API AWARE should be set to true.	For the listed processes, select Process Scheduler, Processes, Process Definition. Select the check box that reads API Aware. If API Aware is not marked, this process gets an incorrect run status when it's viewed from Process Monitor. For additional information, please refer to the Process Scheduler PeopleBook.
PRCSSCHED-05	Process Definitions, where process type is not found in the Process Type Definition .	This occurs when a Process Definition is copied from another PeopleSoft database by using project upgrade. However, the Process Type definition that is associated with this Process Definition is not copied into the database. Review the project upgrade that is used to create the Process Definition. Create another project upgrade to copy Process Type definition from the database where the Process Definition originated.
PRCSSCHED-06	Process Job Item (PS_PRCSJOBITEM), where Process Type is listed as a job item, but is not found in the Process Definition (PS_PRCSDEFN).	This occurs when a PSJob is copied from another PeopleSoft database by using a project upgrade. However, the Process Definition for one or more job items in the PSJob is not copied from the database. Review the project upgrade that is used to create the PSJob. Create another project upgrade to copy the Process Definitions that are identified in this report from the database where the PSJob originated.
PRCSSCHED-07	Server Class List (PS_SERVERCLASS), where Process Type is not found in the Process Type Definition (PS_PRCSTYPEDEFN).	This occurs when a Server Definition is copied from another database by using a project upgrade. However, a process type in the Server Class list is not found in the Process Type Definition. Create another project upgrade to copy the Process Type definition from the database where the Server Definition is created.

Query	Description	Resolution
PRCSSCHED-08	Process Definitions, where the process category is invalid	For the listed processes, select Process Scheduler, Processes, Process Definition. Correct the Process Category.
PRCSSCHED-09	Job Definitions, where the process category is invalid.	For the listed jobs, select Process Scheduler, Jobs, Job Definition. Correct the Process Category.
PRCSSCHED-10	Process Definitions, where the process category is missing.	For the listed processes, select Process Scheduler, Processes, Process Definition. Specify a Process Category.
PRCSSCHED-11	Job Definitions, where the process category is missing.	For the listed jobs, select Process Scheduler, Jobs, Job Definition. Specify a Process Category.
PRCSSCHED-12	Server Categories, where a category defined for a server does not exist in process category definition.	For the listed servers, select Process Scheduler, Servers, Server Definition. Remove the invalid Process Category.
PRCSSCHED-13	Server Categories, where a server is missing a process category definition.	For the listed servers, select Process Scheduler, Servers, Server Definition. A warning message appears when you open the page, and the missing Process Category is added to the server when the page is saved.

PRCSSCHED-14	Process Scheduler Queue, where a queued/pending request specifies a category that does not exist in process category definition.	Run the following SQL:
		`DELETE FROM PSPRCSQUE S⇒`
		`WHERE S.RUNSTATUS IN('5', '16')⇒`
		`AND S.SERVERNAMERQST <> ''⇒`
		`AND S.PRCSCATEGORY NOT IN⇒`
		`(SELECT PRCSCATEGORY FROM PS_SERVERCATEGORY⇒`
		`WHERE SERVERNAME=S.SERVERNAMERQST⇒`
		`AND MAXCONCURRENT > 0)`

PRCSSCHED-15	Process Definitions, where a process specifies an invalid destination folder.	For the listed processes, select Process Scheduler, Processes, Destination. Correct the Destination Folder or blank it out.

Query	Description	Resolution
PRCSSCHED-16	Process Definitions, where a process definition specifies a recovery process that does not exist.	For the listed processes, select Process Scheduler, Processes, Process Definition Options. Correct the recovery process or blank it out.
PRCSSCHED-17	Job Definitions, where a job definition specifies a recovery process that does not exist.	For the listed jobs, select Process Scheduler, Jobs, Job Definition Options. Correct the recovery process or blank it out.

Query Integrity

The following table contains the audits and resolutions for this area.

Query	Description	Resolution

QUERY-01	Query Definition Select count does not match the record count that is in the Query Select table.	The query definition is corrupt. Run the following SQL to delete the entire query definition:
		`DELETE FROM PSQRYDEFN⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYSELECT⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYRECORD⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYFIELD⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYCRITERIA⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYEXPR⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYBIND⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`

QUERY-02	Query Definition Expression count does not match the record count in the Query Expression table.	Run the following SQL:
		`UPDATE PSQRYDEFN⇒`
		`SET EXPCOUNT = (SELECT COUNT(*) FROM PSQRYEXPR C⇒`
		`where oprid = 'x' AND qryname = 'y')⇒`
		`where oprid = 'x' AND qryname = 'y'`

QUERY-03	Query Definition Bind count does not match the record count in the Query Bind table	Run the following SQL:
		`UPDATE PSQRYDEFN⇒`
		`SET BNDCOUNT = (SELECT COUNT(*) FROM PSQRYBIND⇒`
		`where oprid = 'x' AND qryname = 'y')⇒`
		`where oprid = 'x' AND qryname = 'y'`

Query	Description	Resolution
QUERY-04	Query Definition Record name does not exist in the Record Definition table.	See resolution for QUERY-07.
QUERY-05	Query Definition Record JoinRecord name does not exist in the Query Record table	See resolution for QUERY-01.
QUERY-06	Query Definition Record JoinField name does not exist in the Query Field table.	See resolution for QUERY-01.
QUERY-07	Query Field Record Name does not exist in Record Definition Table	To salvage the query, you must use PeopleSoft Application Designer to re-create the record definition. Having re-created the record, run Query and open the offending query. Remove or repair the affected areas and save the query. Or, if the query is not important, you can delete the entire query definition by using the resolution for QRY-01.
QUERY-08	Query Definition Field name does not exist in the Field Definition table	If the record on which this field appears is deleted, you have seen errors for every referenced field that belongs to the deleted record. If this is the case, see the resolution for QUERY-1. If this is not the case, some fields that the query depends on are either deleted or renamed. Run Query and open the offending query. Query automatically repairs itself and updates the query definition in the database.
QUERY-09	Query Selection Record count does not match the record count in Query Record table.	See resolution for QUERY-01.
QUERY-10	Query Selection Field count does not match the record count in Query Field table.	See resolution for QUERY-01.
QUERY-11	Query Selection Criteria count does not match the record count in Query Criteria table.	See resolution for QUERY-01.

Query	Description	Resolution
QUERY-11A	Query Selection Criteria having count does not match the record count in Query Criteria table.	See resolution for QUERY-01.
QUERY-12	Query Selection Parent select number does not exist in Query Select table.	See resolution for QUERY-01.

QUERY-13	Query Criteria Selection-Left does not exist in the Query Selection table.	Run Query and delete the corrupted criteria entry. If you can't open the query, run the following SQL to delete the corrupted criteria entry:
		`DELETE FROM PSQRYCRITERIA⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'⇒`
		`AND CRTNUM = count`

QUERY-14	Query Criteria Selection-Right1 does not exist in the Query Selection table.	See resolution for QUERY-13.
QUERY-15	Query Criteria Selection-Right2 does not exist in the Query Selection table.	See resolution for QUERY-13.
QUERY-16	Query Criteria Field-Left does not exist in the Query Selection table.	See resolution for QUERY-13.
QUERY-17	Query Criteria Field-Right1 does not exist in the Query Selection table.	See resolution for QUERY-13.
QUERY-18	Query Criteria Field-Right2 does not exist in the Query Selection table.	See resolution for QUERY-13.
QUERY-19	Query Criteria Expression-Right1 does not exist in the Query Selection table.	See resolution for QUERY-13.
QUERY-20	Query Criteria Expression-Right2 does not exist in the Query Selection table.	See resolution for QUERY-13.

Query	Description	Resolution
QUERY-22	Following Queries Were Created Without PUBLIC Access.	This is normal; the audit insures that PeopleSoft does not deliver nonpublic queries as part of its standard delivered products.

QUERY-23	The listed queries reference non-existent database records.	This is an internal PeopleSoft audit. Call PeopleSoft GSC for resolution.
		The query definition is corrupt. Run the following SQL to delete the entire query definition:
		`DELETE FROM PSQRYDEFN⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYSELECT⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYRECORD⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYFIELD⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYCRITERIA⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYEXPR⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`
		`DELETE FROM PSQRYBIND⇒`
		`WHERE OPRID = 'X' AND QRYNAME = 'Y'`

QUERY-24	The listed queries were created with the name UNTITLED.	Queries should not be saved as UNTITLED. You must either rename or delete these queries.
QUERY-25	This audit identifies any blank query names.	You must either rename the query or delete the query.
QUERY-26	This audit identifies queries that contain unions but select an unequal number of fields.	Ensure that every select statement in the query has an equal number of fields selected. These fields must also match in display type and length.

Record Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution

RECORD-1	Record Definition Field count does not match the number of records in Record Field table.	Run the following SQL:
		`SELECT COUNT(*) FROM PSRECFIELD⇒`
		`WHERE RECNAME = 'X';`
		`UPDATE PSRECDEFN SET FIELDCOUNT = COUNT⇒`
		`WHERE RECNAME = 'X';`

Query	Description	Resolution

RECORD-2	Record Definition Fields do not exist in Record Field table.	Run the following SQL:
		`UPDATE PSRECDEFN SET FIELDCOUNT = 0⇒ WHERE RECNAME = 'X';`
		Or
		`DELETE FROM PSRECDEFN WHERE RECNAME = 'X';`

RECORD-3	Record Definition Parent Record does not exist in Record Definition table.	Use Application Designer to open the definition. Select File, Object Properties, Use and specify a valid parent record.
RECORD-4	Record Definition SubRecord does not exist in Record Definition table.	Use Application Designer to open the definition. Select File, Object Properties, Type and specify a valid subrecord.
RECORD-5	Record Definition Query Security Record does not exist in Record Definition table	Use Application Designer to open the definition. Select File, Object Properties, Use and specify a valid query security record.
RECORD-6	Record Field definitions contain record names that do not exist in the Record Definition table.	Run the following SQL: `DELETE FROM PSRECFIELD WHERE RECNAME = 'X'`
RECORD-7	DBField records do not exist for the following RecField table Fields.	Use PeopleSoft Application Designer to open the definition and fix the invalid fields.
RECORD-8	Record definitions do not exist for the following RecField table SubRecords.	Use PeopleSoft Application Designer to open the definition and fix the invalid fields.
RECORD-9	IDENTIFY INVALID RECORDS IN RECORD GROUP DEFINITIONS.	Run the following SQL: `DELETE FROM PS_REC_GROUP_REC WHERE RECNAME NOT IN (SELECT DISTINCT RECNAME FROM PSRECDEFN)`
RECORD-11	Records with more than two Longs defined.	This depends on whether the database platform supports it or not. If it does not, then those records must be modified.

Query	Description	Resolution

RECORD-12	Records with a blank or null RECNAME.	Run the following SQL:
		`DELETE FROM PSRECDEFN`
		`WHERE RECNAME = ''`

RECORD-13	Temp Records that specify a non-standard SQL table name.	Run the following SQL:
		`UPDATE PSRECDEFN SET SQLTABLENAME = ' '⇒`
		`WHERE RECTYPE = 7 AND SQLTABLENAME <> ' '`

Related Language Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution

SYSLANG-01	Base Language Records that are found in the PSRECDEFNLANG table.	Run the following SQL:
		`DELETE FROM PSRECDEFNLANG⇒`
		`WHERE LANGUAGE_CD =⇒`
		`(SELECT B.LANGUAGE_CD FROM PSOPTIONS B)`

SYSLANG-02	Base Language Fields that are found in the PSDBFIELDLANG table.	Check the value of LANGUAGE_CD on PSOPTIONS; this is the base language. Entries with this language code are found in PSDBFIELDLANG. Base language entries should only be in PSDBFIELD.
		After you establish that the base language entries in PSDBFIELD are correct, you delete them from PSDBFIELDLANG as follows:
		`DELETE FROM PSDBFIELDLANG⇒`
		`WHERE LANGUAGE_CD =⇒`
		`(SELECT LANGUAGE_CD FROM PSOPTIONS)`

SYSLANG-03	Foreign Language Records that are found in PSRECDEFNLANG table without related Base Records from PSRECDEFN.	Run the following SQL:
		`DELETE FROM PSRECDEFNLANG⇒`
		`WHERE NOT EXISTS⇒`
		`(SELECT 'X' FROM PSRECDEFN B⇒`
		`WHERE PSRECDEFNLANG.RECNAME = B.RECNAME)⇒`
		`AND PSRECDEFNLANG.LANGUAGE_CD <>⇒`
		`(SELECT C.LANGUAGE_CD FROM PSOPTIONS C)`

Query	Description	Resolution

SYSLANG-04	Foreign Language Fields that are found in the PSDBFIELDLANG table without related Base Fields from PSDBFIELD.	Run the following SQL:
		`DELETE FROM PSDBFIELDLANG⇒`
		`WHERE NOT EXISTS⇒`
		`(SELECT 'X' FROM PSDBFIELD B⇒`
		`WHERE PSDBFIELDLANG.FIELDNAME=B.FIELDNAME)⇒`
		`AND PSDBFIELDLANG.LANGUAGE_CD <>⇒`
		`(SELECT LANGUAGE_CD FROM PSOPTIONS)`

SYSLANG-05	Foreign Language Translate Fields that are found in the XLATTABLE table without related Base Language Translate Fields.	Either delete the offending entries via SQL, or use PeopleSoft Application Designer to add the equivalent entries in the base language of the database.
SYSLANG-07	Related Language Records Which Are Not Valid Records.	In PeopleSoft Application Designer, delete the specified Related Language Records.
SYSLANG-08	The displayed Related Language Records are effective-dated but do not have an EFFDT field defined.	In PeopleSoft Application Designer, add EFFDT to the specified related language table.
SYSLANG-09	The displayed Related Language Records point to another Related Language Record	In PeopleSoft Application Designer, delete the related language reference for each record that is listed.
SYSLANG-10	The displayed Related Language Records do not contain a LANGUAGE_CD as a key field.	In PeopleSoft Application Designer, make LANGUAGE_CD field a key on the specified Related Language Tables.
SYSLANG-11	The displayed Related Language Views Have The Wrong Structure Defined.	In PeopleSoft Application Designer, make the key structures that are on the specified Related Language view identical to the key structure that is on the Base Table/view.
SYSLANG-12	The displayed Related Language Records Have The Wrong Key Structure Defined.	In Application Designer, make the key structures on the specified Related Language Tables identical to the key structure on the Base Table.
SYSLANG-13	Identify related language records that have more than one base record defined.	In PeopleSoft Application Designer, remove the related language table link to one of the base records.

Query	Description	Resolution
SYSLANG-14	Identify related language RECORDS that have the wrong structure defined	In PeopleSoft Application Designer, make the key structures on the specified Related Language Tables identical to the key structure that is on the Base Table. Also remove any fields, except language_cd, on the related language record that do not exist on the base record.
SYSLANG-15	The displayed Related Language fields are orphaned. (Continues in next row)	For each row on the related language record there must be a single row on the base table with matching keys. An orphan row is a row of data on the related language record that does not have a corresponding parent row on the base table. Orphan rows must be deleted. Select against the related language table by using the key fields that are listed in the report to find discrepancies. To fix this problem, using the platform's query tools, select against the related language table, by using the fields that are listed in the report, where the values do not match a row that is on the base table. For every row on the report there is an orphan row on the table. Do a SELECT first to ensure you get the same row count as the report, then delete the selected rows. Following is example SQL for a Microsoft SQL Server database:

SYSLANG-15	(Continued)	`SELECT * FROM PSCONTDEFNLANG A⇒`
		`WHERE NOT EXISTS⇒`
		`(SELECT 'X' FROM PSCONTDEFN B⇒`
		`WHERE A.ALTCONTNUM = B.ALTCONTNUM⇒`
		`AND A.CONTNAME = B.CONTNAME⇒`
		`AND A.CONTTYPE = B.CONTTYPE)`
		`DELETE FROM PSCONTDEFNLANG WHERE NOT EXISTS⇒`
		`(SELECT 'X' FROM PSCONTDEFN B⇒`
		`WHERE PSCONTDEFNLANG.ALTCONTNUM = B.ALTCONTNUM⇒`
		`AND PSCONTDEFNLANG.CONTNAME = B.CONTNAME⇒`
		`AND PSCONTDEFNLANG.CONTTYPE = B.CONTTYPE)`
		For each field name listed:
		`SELECT * FROM` *`Related_Language_Record`* `A⇒`
		`WHERE NOT EXISTS⇒`
		`(SELECT 'X' FROM` *`BaseRecord`* `B⇒`
		`WHERE A.`*`FieldName`*` = B.`*`FieldName`*`)`

SQL Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
SQL-01	SQL text without a base definition.	Run the following SQL: `DELETE FROM PSSQLTEXTDEFN⇒ WHERE SQLID NOT IN⇒ (SELECT DISTINCT SQLID FROM PSSQLDEFN)`
SQL-02	SQL definitions without SQL text.	Run the following SQL: `DELETE FROM PSSQLDEFN WHERE SQLID NOT IN ( SELECT DISTINCT SQLID FROM PSSQLTEXTDEFN)`
SQL-03	SQL descriptions without a base definition.	Run the following SQL: `DELETE FROM PSSQLDESCR WHERE SQLID NOT IN ( SELECT DISTINCT SQLID FROM PSSQLDEFN)`
SQL-04	SQL descriptions without associated SQL text.	Run the following SQL: `DELETE FROM PSSQLDESCR WHERE SQLID NOT IN ( SELECT DISTINCT SQLID FROM PSSQLTEXTDEFN)`
SQL-05	AE SQL without SQL definitions.	This reveals Application Engine SQL Actions that do not contain any SQL code within them. Open the Application Engine program and complete the entry of the SQL before attempting to run the program. If the empty SQL actions are delivered by PeopleSoft, open an incident with the GSC to report the corrupt AE program.
SQL-06	AE SQL that’s not referenced.	This reveals an Application Engine SQL object that is not being referenced by an AE program. This indicates that the AE program is deleted but the associated SQL is not. The orphaned SQL does not cause issues other than consuming disk space. If the orphaned SQL is delivered by PeopleSoft, open an incident with GSC to make sure that it is not a symptom of a larger problem, such as a corrupted AE application.

Query	Description	Resolution
SQL-07	Record Views/Dynamic Views without SQL definitions.	Complete the entry of the record view or dynamic view before attempting to build or create the view. Each record should be opened, and the SQL should be entered as required.

SQL-08	View SQL that are not referenced by record or dynamic views.	Run the following SQL:
		`DELETE FROM PSSQLDEFN⇒`
		`WHERE SQLTYPE = 2 AND SQLID NOT IN⇒`
		`(SELECT DISTINCT RECNAME FROM PSRECDEFN⇒`
		`WHERE RECTYPE = 5 OR RECTYPE = 1)`
		`DELETE FROM PSSQLDESCR⇒`
		`WHERE SQLTYPE = 2 AND SQLID NOT IN⇒`
		`(SELECT DISTINCT RECNAME FROM PSRECDEFN⇒`
		`WHERE RECTYPE = 5 OR RECTYPE = 1)`
		`DELETE FROM PSSQLTEXTDEFN⇒`
		`WHERE SQLTYPE = 2 AND SQLID NOT IN⇒`
		`(SELECT DISTINCT RECNAME FROM PSRECDEFN⇒`
		`WHERE RECTYPE = 5 OR RECTYPE = 1)`

Tree Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
TREE-01	Tree Structure table contains Level Record name that does not exist in Record Definition table	Use Tree Manager to open the structure and fix the invalid fields.
TREE-02	Tree Structure table contains Level Page name that does not exist in Page Definition table.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-03	Tree Structure table contains Node Record name that does not exist in Record Definition table.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-04	Tree Structure table contains Node Field name that does not exist in RecordField table.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-05	Tree Structure table contains Node Page name that does not exist in Page Definition table.	Use Tree Manager to open the structure and fix the invalid fields.

Query	Description	Resolution
TREE-06	Tree Structure table contains Detail Record name that does not exist in Record Definition table.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-07	Tree Structure table contains Detail Record name that does not exist in Record Definition table.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-08	Tree Structure table contains Detail Page name that does not exist in Page Definition table.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-09	Tree Structure table contains Summary Tree that does not exist in Tree Level table.	See the following section on Notes for TREE-09.
TREE-10	Tree Structure table contains Level Menu-Menu Bar combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-11	Tree Structure table contains Node Menu-Menu Bar combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-12	Tree Structure table contains Detail Menu-Menu Bar combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-13	Tree Structure table contains Level Menu-Page combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-14	Tree Structure table contains Node Menu-Page combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-15	Tree Structure table contains Detail Menu-Page combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.

Query	Description	Resolution
TREE-16	Tree Definition Level count does not match the record count in Tree Level table.	Set the Count in the Tree Definition table to reflect the actual number of records that are in the PSTREELEVEL table for this tree branch. Note that a problem may occur if some levels are missing and there are still nodes referencing them. In this case, the nodes do not open the tree correctly. The third SELECT checks for the previous situation. If this is the problem, run PSTED, and define the missing levels, save the tree, and then close and reopen it. SELECT COUNT(*) FROM PSTREELEVEL WHERE TREE_NAME = 'tr ee_name' AND SETID = 'setid' AND EFFDT = 'effdt'; UPDATE PSTREEDEFN SET LEVEL_COUNT = $count WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt';
TREE-17	Tree Definition Node count does not match the record count in Tree Node table.	Set the count in the Tree Definition table to reflect the actual number of the records that are in the PSTREENODE table for this tree branch. SELECT COUNT(*) FROM PSTREENODE WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt' AND TREE_BRANCH = 'tree_branch_name'; UPDATE PSTREEDEFN SET NODE_COUNT = $count WHERE TREE_NAME = 'tr ee_name' AND SETID = 'setid' AND EFFDT = 'effdt' TREE_BRANCH = 'tree_b ranch_name';
TREE-18	Tree Definition Leaf count does not match the record count in Tree Leaf table.	Set the Count in the Tree Definition table to reflect the actual number of records that are in the PSTREELEAF table for this branch. SELECT COUNT (*) FROM PSTREELEAF WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt' TREE_BRANCH = 'tree_branch_name'; UPDATE PSTREEDEFN SET LEAF_COUNT = $count WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt' TREE_BRANCH = 'tree_branch_name';
TREE-19	Tree Definition contains Structure ID that does not exist in Tree Structure table	Use Tree Manager to create the structure that you desire by using the name that is reported in this audit.

Query	Description	Resolution
TREE-20	Tree Definition contains Query Access Group structure with undefined levels and leaves.	Query trees should have no leaves and no levels. This audit finds exceptions to that case in the definition counts. UPDATE pstreedefn SET level_count = 0, leaf_count = 0 WHERE tree_strct_id = 'ACCESS_GROUP' AND (level_count <> 0 OR leaf_count <> 0);
TREE-21	Tree Selector Control contains Tree name that is not defined in Tree Definition table.	This audit flags records in the PSTREESELCTL tables for records that don't have a corresponding record in the PSTREEDEFN table. DELETE FROM pstreeselctl A WHERE NOT EXISTS (SELECT 'x' FROM pstreedefn B WHERE B.setid = A.setid AND B.tree_name = A.tree_name AND B.effdt = A.effdt)
TREE-22	Tree Definition Level count does not match level use.	See the section titled "Notes for TREE-22" below.
TREE-23	Tree Level does not exist in Tree Definition table.	Tree Level records in the PSTREELEVEL table exist for trees that don't exist in the PSTREEDEFN table. DELETE FROM pstreelevel A WHERE NOT EXISTS (SELECT 'x' FROM pstreedefn B WHERE B.setid = A.setid AND B.tree_name = A.tree_ name AND B.effdt = A.effdt)
TREE-24	Tree Node does not exist in Tree Definition table.	Tree Node records in the PSTREENODE table exist for trees that don't exist in the PSTREEDEFN table. DELETE FROM pstreenode A WHERE NOT EXISTS (SELECT 'x' FROM pstreedefn B WHERE B.setid = A.setid AND B.tree_name = A.tree_name AND B.effdt = A.effdt)
TREE-25	Tree Leaf does not exist in Tree Definition table.	Tree Leaf records in the PSTREELEAF table exist for trees that don't exist in the PSTREEDEFN table. DELETE FROM pstreeleaf A WHERE NOT EXISTS (SELECT 'x' FROM pstreedefn B WHERE B.setid = A.setid AND B.tree_name = A.tree_name AND B.effdt = A.effdt)
TREE-26	Tree Leaf ranges are not valid in Tree Definition table.	Finds records in the PSTREELEAF table where RANGE_FROM is less than RANGE_TO. Use Tree Manager to open the tree and correct the invalid range values.

Query	Description	Resolution
TREE-27	Tree Leaf does not have parent Tree Node in Tree Definition table.	DELETE FROM pstreeleaf A WHERE NOT EXISTS (SELECT 'x' FROM pstreenode B WHERE B.setid = A.setid AND B.tree_name = A.tree_name AND B.effdt = A.effdt AND B.tree_node_num = A.tree_node_num)
TREE-28	Tree Branch does not exist in Tree Branch table.	Refer to the "Tree Audit and Repair Utilities" chapter in the Tree Manager PeopleBook and run the Unbranch Tree Repair Utility so that all branches are removed from the tree.
TREE-29	Tree Branch does not exist in Tree Branch table.	Refer to the "Tree Audit and Repair Utilities" chapter in the Tree Manager PeopleBook and run the Unbranch Tree Repair Utility so that all branches are removed from the tree.
TREE-30	Tree Branch Node count does not match the record count in Tree Node table.	See Resolution for Tree-29.
TREE-31	Tree Branch Leaf count does not match the record count in Tree Leaf table	See Resolution for Tree-29.
TREE-32	Tree Node Num, Node Num End, or Level Num is invalid in Tree Branch table.	See Resolution for Tree-29.
TREE-33	Identify all orphan access group definitions as well as invalid access group definitions in the access group security.	Open Query Access Group Tree in Query Access Group Manager and update the identified Access Group so that a record is created in the Access Group Table.
TREE-34	Tree Definition Node Count Does Not Equal 0 for a Branched Tree.	See Resolution for Tree-29.
TREE-35	Tree Definition Leaf Count Does Not Equal 0 for a Branched Tree.	See Resolution for Tree-29.

Notes for TREE-09

Lists any Summary Tree Structures that reference a level number that is on a Detail Tree that does not exist in the Tree Level table. Since a Summary Tree is a tree that is built off of the nodes from an existing Detail Tree at a given level, the level that is specified on the Summary Tree Structure must exist in the detail tree's PSTREELEVEL table. In this case, the Summary Tree is not usable from nVision or other reporting tools.

The situation could occur from several possible causes :

- Summary Tree is moved or imported into a new database but Detail Tree is not
- The levels on the Detail Tree are deleted after the Summary Tree structure is created.

To correct this :

1. First determine if Detail Tree exists and is in a valid state. This can be done by checking the name of the Detail Tree on the Summary Tree's Structure record; check the Summary Tree tab on the Tree Structure record for the Summary Tree. Note the tree name, setID, and level number.
2. If Detail Tree exists, check to see if the level number that is defined on the Summary Tree Structure (step 1) exists.
3. To correct the situation, either add missing level to detail tree or update Summary Tree Structure to refer to a valid detail tree and level number.

Notes for TREE-22

This audit flags the Level Use type with the Level Count for conflicts. When the Level Use is N, there should be no levels defined, and when it is not N, levels should be defined. A problem in this audit may also report problems in the TREE-16 audit.

When the Level Use is N and the Level Count is 0 and TREE-16 does not indicate an error on the same tree, do the following:

UPDATE PSTREEDEFN SET USE_LEVELS = 'S' WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt'

When the Level Use is S and the Level Count is 0 and TREE-16 does not indicate an error on the same tree, do the following (after checking the resolution on TREE-16 to clean up any level records):

UPDATE PSTREEDEFN SET LEVEL_COUNT = 0 WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt'

When the Level Use is not N and the Level Count is 0 and TREE-16 does not indicate an error on the same tree, do the following (after checking the resolution on TREE-16 to clean up any level records):

UPDATE PSTREEDEFN SET USE_LEVELS = 'N' WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt' and

UPDATE PSTREENODE SET TREE_LEVEL_NUM = 0 WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt'

When TREE-23 indicates an error on the same Tree with the Level Count on the PSTREEDEFN = number of PSTREELEVEL records (when the PSTREELEVEL has no levels for this tree, count is 0 for the following):

SELECT COUNT(*) FROM PSTREELEVEL WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt'

UPDATE PSTREEDEFN SET LEVEL_COUNT = 0 WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt'

UPDATE PSTREEDEFN SET USE_LEVELS = 'N' WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt'

UPDATE PSTREENODE SET TREE_LEVEL_NUM = 0 WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt'

Translate Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
XLATT-1	Translate table Field does not exist in database Field.	Create the field by using PeopleSoft Application Designer.
XLATT-3	Translate fields do not have associated translate values defined.	Edit translate field and enter translate value.

PSLOCK Integrity

The following table contains the audits and resolutions for this area:

Query	Description	Resolution
Manager-< XXX > Where XXX is the associated three-letter code of the object type.	Version Check of listed table against PSVERSION.	Run the Version Application Engine program.

CHAPTER 5

Employing Database Level Auditing

This chapter provides an overview of database level auditing and discusses how to:

- Create audit record definitions.
- Work with auditing triggers.
- View audit information.
- Create queries to view audit records details.
- Use Microsoft SQL Server trigger information.
- Use DB2 UDB for OS/390 and z/OS trigger information.
- Use Oracle trigger information.
- Use Sybase trigger information.

Understanding Database Level Auditing

PeopleSoft provides trigger-based auditing functionality as an alternative to the record-based auditing that PeopleSoft Application Designer provides. Some countries require that you audit changes to certain data, while some companies audit who is making changes to sensitive data. This level of auditing is not only for maintaining the integrity of the data, but it is also a heightened security measure. PeopleSoft takes advantage of database triggers (offered by most database vendors), and when a user makes a change to a specified field that you are monitoring, the changed data triggers the audit.

The information that a trigger records could include the user that made a change, the type of change that is made, when the change is made, and so on. Because the trigger records the user ID of the user who is modifying the base table, it is essential that you have the Enable DB Monitoring parameter set in PSADMIN. (This feature is not supported for Informix or DB2 LUW.)

Note. If you implement trigger-based auditing, be aware that there is an unavoidable amount of additional overhead that is associated with auditing that can effect the system's overall performance.

The components that are involved with database level auditing are:

Base Records — The base record is the record that you want to monitor, or audit, as in PS_ABSENCE_HIST. Presumably, the base record contains fields that you want to monitor. Limit the auditing of tables to the application tables and avoid auditing PeopleTools tables.

Audit Record — The audit record is a custom record that you create with PeopleSoft Application Designer. It stores the audit information for the fields on the base record that the trigger collects. Audit records begin with an AUDIT_ prefix.

Trigger The trigger is the mechanism that a user invokes upon making a change to a specified field. The trigger stores the audit information in the audit table. PeopleSoft enables you to create triggers. A sample name for a trigger might be PS_ABSENCE_HIST_TR.

Note. If you modify the record definition of the base record, then you must modify to the audit record and re-create the associated trigger.

Creating Audit Record Definitions

To audit a record using triggers, you must create a record definition and SQL table in which you store audit information. When creating the audit record, remove any attributes such as PARENT records, Query Security Records, and PeopleCode.

The easiest way to create an audit table is to open the record definition of the base record that you want to audit. Save it as a new record, prefaced with AUDIT_.

Note. When you create a new audit record definition, be sure to name it with an AUDIT_ prefix. Some processes, such as the Employee ID Change and Employee ID Delete in PeopleSoft HRMS product line, make changes to certain fields, such as EMPLID. These processes do not affect any record definitions that begin with the AUDIT_ prefix, leaving the audit data secure.

Remove the all edit and key attributes from the newly saved record. Add to the top of the audit record the following three special audit-specific fields:

- AUDIT_OPRID
- AUDIT_STAMP
- AUDIT_ACTN

Make these fields required and keys. The following table explains the purpose of each audit-specific field.

Note. When you add these fields to the audit record, add them in the same order that they appear in the following table.

Audit Field Name	Purpose
AUDIT_OPRID	Identifies the user who causes the system to trigger the audits, either by performing an add, change, or delete to an audited field.
AUDIT_STAMP	Identifies the date and time that the audit is triggered.
AUDIT_ACTN	Indicates the type of action the system audited. Possible action values include: • *A* – Row inserted. • *D* – Row deleted. • *K* – Row updated, snapshot before update. • *N* – Row updated, snapshot after update.

The audit table does not have to include all the columns of the base table. In fact, for performance reasons, it's best to only include those fields in the audit record that are deemed sensitive or significant. When adding fields to the audit record, PeopleSoft recommends that you conform to the order that they appear in the base record.

Note. This functionality allows SQL Server requirement of not including ntext, text columns in the trigger syntax. Oracle also has a requirement to exclude longs from audit records.

The following example compares the base table to the audit table, showing the audit-specific fields and the fields that are to be audited in the audit table.

Base Table PS_ABSENCE_HIST	Audit Table PS_AUDIT_ABSENCE
	AUDIT_OPRID
	AUDIT_STAMP
	AUDIT_ACTN
EMPLID	EMPLID
ABSENCE_TYPE	ABSENCE_TYPE
BEGIN_DT	
RETURN_DT	
DURATION_DAYS	
DURATION_HOURS	
REASON	REASON
PAID_UNPAID	
EMPLOYER_APPROVED	
COMMENTS	

Once you save the record definition, you need to run the SQL Build procedure to build the SQL table in the relational database management system (RDBMS).

The following is an example of a SQL script for an audit record that audits the PS_ABSENCE_HIST table:

```
-- WARNING:
--
```

```
-- This script should not be run in Data Mover.  It may contain platform
-- specific syntax that Data Mover is unable to comprehend.  Please use the
-- SQL query tool included with your database engine to process this script.
--
USE PT8A
go
SET IMPLICIT_TRANSACTIONS ON
go
IF EXISTS (SELECT 'X' FROM SYSOBJECTS WHERE TYPE = 'U' AND NAME =
 'PS_AUDIT_ABSENCE') DROP TABLE PS_AUDIT_ABSENCE
go
CREATE TABLE PS_AUDIT_ABSENCE (AUDIT_OPRID CHAR(8) NULL,
   AUDIT_STAMP PSDATETIME NOT NULL,
   AUDIT_ACTN CHAR(1) NOT NULL,
   AUDIT_RECNAME CHAR(15) NOT NULL,
   EMPLID CHAR(11) NOT NULL,
   ABSENCE_TYPE CHAR(3) NOT NULL,
   BEGIN_DT PSDATE NULL,
   RETURN_DT PSDATE NULL,
   DURATION_DAYS SMALLINT NOT NULL,
   DURATION_HOURS DECIMAL(2,1) NOT NULL,
   REASON CHAR(30) NOT NULL,
   PAID_UNPAID CHAR(1) NOT NULL,
   EMPLOYER_APPROVED CHAR(1) NOT NULL)
--   COMMENTS TEXT NULL) Text and Image Fields are not allowed
go
COMMIT
Go
```

If COMMENTS is not allowed during the actual creation of the audit table, drop the column or do not choose the column when you create the audit table definition.

Delete the generated index script; this eliminates the possibility of the duplicates causing trigger to fail.

Working With Auditing Triggers

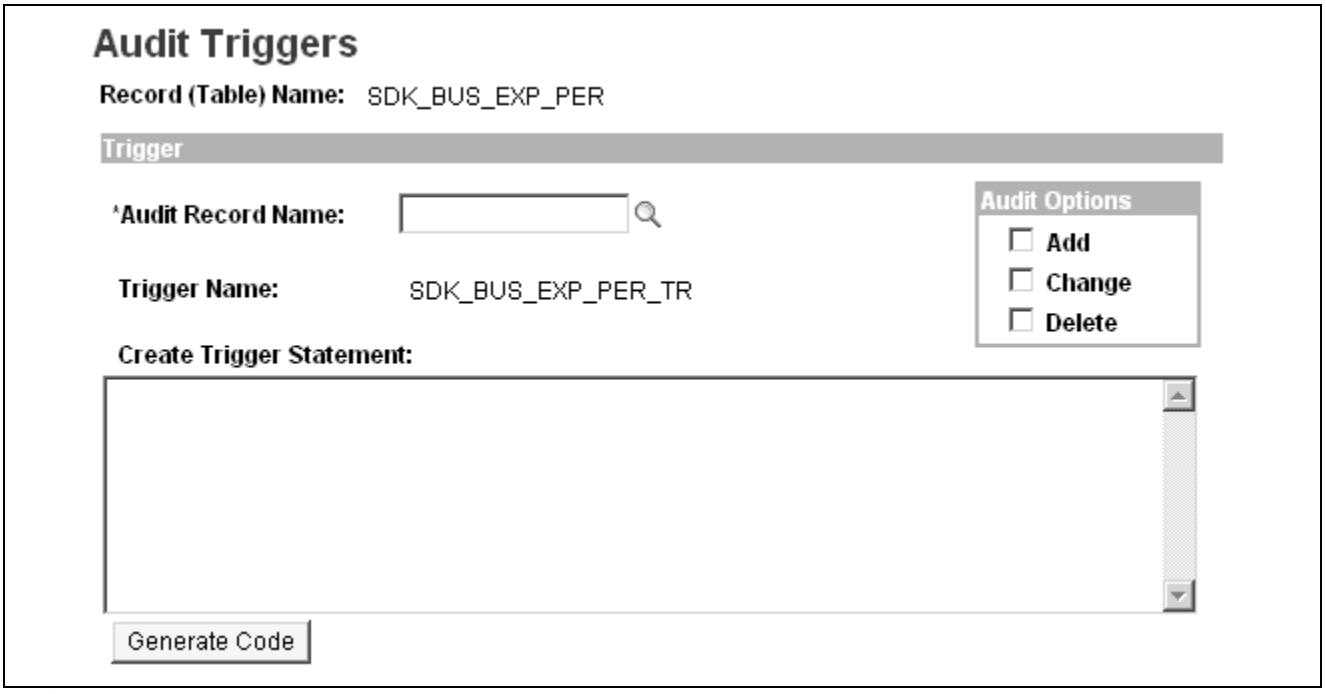
This section discusses how to:

- Define auditing triggers.
- Create and run the auditing triggers script.
- Delete auditing triggers.

A trigger is a database level object that the system initiates based on a specified event occurring on a table. Most RDBMS platforms support a form of database triggers.

Defining Auditing Triggers

Select PeopleTools, Utilities, Audit, Update Database Level Auditing.



Audut Triggers page

The Audit Triggers page contains the following options.

Audit Record Name	Use the Browse button to search the PSRECDEFN table. The Audit name must exist before a trigger can be created.
Trigger name	By default, the system names audit triggers by using the following naming convention <base record>_TR For example, ABSENCE_HIST_TR
Audit Options	Select from the options Add, Change or Delete.
Create Trigger Statement	The statement is populated when the Generate Code button is clicked. You can customize the script if you need to. It depends on your preference. One of the following sections contains RDBMS information to help you view the contents of the script.
Generate Code	Click this button when you complete the previous fields to generate the Trigger Statement.

To define an audit trigger

1. Select PeopleTools, Utilities, Audit, Update Database Level Auditing.
2. Click Add a New Value.
3. Enter an existing base record.
4. On the Audit Trigger page, you need to choose the record to hold the auditing data, the audit record.
5. Select the events to audit, as in when data is added, changed, or deleted. You can select all of the options.
6. Click Generate Code.

   This generates the SQL that ultimately creates the trigger.

7. Click Save.

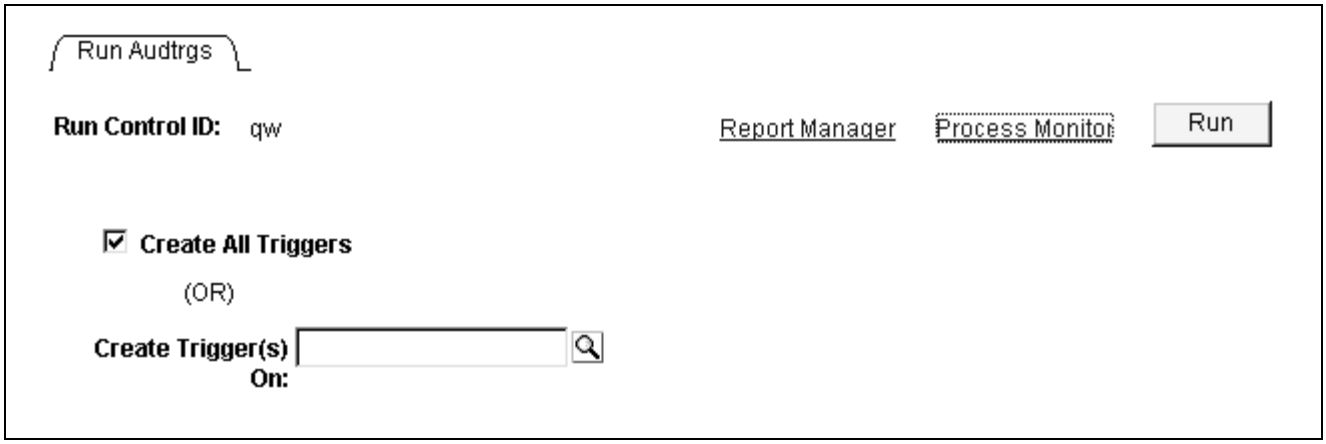
   All of this information, Record Name, Audit Record Name, Trigger Name, and Create Trigger Statement, gets saved to the PeopleSoft table, PSTRIGGERDEFN.

   Perform these steps for each trigger that you want to create. After you create all the trigger statements, then you create and run the trigger script, which is described in the following section.

Creating and Running the Auditing Triggers Script

After you create and modify all of the trigger statements, you need to create and run the trigger script against the database to physically create the triggers.

Select PeopleTools, Utilities, Audit, Perform Database Level Audit.



Run Audtrgs (Run Audit Triggers) page

Create All Triggers If you select this check box, the Application Engine writes the Create Trigger statement to a file for every row in PSTRIGGERDEFN.

Create Triggers on Specify the particular table that the Trigger statement should be created for.

To create and run a trigger script:

1. Select PeopleTools, Utilities, Auditing, Perform Database Level Auditing.
2. Indicate the triggers that you want to be included in the script, all in PSTRIGGERDEFN or just those that are related to a specific table.
3. Click Run.

   This process invokes an Application Engine program that writes the Create Trigger statement to a file for every row in PSTRIGGERDEFN that you select (all or for a specific table).

   The system writes the file to the location that is determined by the run location of the process. If it's run on the server, the file is created in the PS_SRVRDIR directory. If it's run on a Windows workstation, the file is created in the directory that the %TEMP% environment variable specifies.

   The file name is TRGCODEX.SQL, where X represents a digit that is determined by the number of files by the same name that already exist in the output directory.
4. After you create the SQL script, use the native SQL utility to run the script against the database.

Deleting Auditing Triggers

To delete a trigger:

1. Select PeopleTools, Utilities, Audit, Update Database Level Auditing.
2. Open the trigger that you want to delete.
3. Clear all the Audit options Add, Change, and Delete.
4. Click Generate Code.
5. Click Save.
6. Drop the trigger name from the database.

Viewing Audit Information

Viewing the data that is in the audit record is important. That's why you're storing the information. Because the information resides in a table within the RDBMS, you can extract it and manipulate it to suit your reporting needs. This section provides samples of how the information appears in an audit record and some sample queries that you can construct with PeopleSoft Query.

The following example presents the contents of PS_AUDIT_ABSENCE after a trigger test:

```
AUDIT_OPRID AUDIT_STAMP                AUDIT_ACTN EMPLID     ⇒
 ABSENCE_TYPE BEGIN_DT
----------- -------------------------- ---------- -----------⇒
 ------------ -------------------------
BARNEY07    2000-01-11 16:25:13.380     A          GORD       ⇒
 CNF          1981-09-12 00:00:00.000
BARNEY07    2000-01-11 16:25:36.123     K          8001       ⇒
 CNF          1981-09-12 00:00:00.000
BARNEY07    2000-01-11 16:25:36.123     K          8001       ⇒
 CNF          1983-03-02 00:00:00.000
BARNEY07    2000-01-11 16:25:36.123     K          8001       ⇒
 CNF          1983-08-26 00:00:00.000
BARNEY07    2000-01-11 16:25:36.133     N          8001       ⇒
 VAC          1981-09-12 00:00:00.000
BARNEY07    2000-01-11 16:25:36.133     N          8001       ⇒
 VAC          1983-03-02 00:00:00.000
BARNEY07    2000-01-11 16:25:36.133     N          8001       ⇒
 VAC          1983-08-26 00:00:00.000
BARNEY07    2000-01-11 16:25:40.790     D          GORD       ⇒
 CNF          1981-09-12 00:00:00.000
RETURN_DT                  DURATION_DAYS DURATION_HOURS⇒
 REASON                        PAID_UNPAID
--------------------------- ------------- --------------⇒
 ------------------------------ -----------
1981-09-26 00:00:00.000     14            .0            ⇒
 None                          P
1981-09-26 00:00:00.000     14            .0            ⇒
```

```
                                P
1983-03-07 00:00:00.000     6              .0            ⇒
                                P
1983-09-10 00:00:00.000     13             2.0           ⇒
                                P
1981-09-26 00:00:00.000     14             .0            ⇒
                                P
1983-03-07 00:00:00.000     6              .0            ⇒
                                P
1983-09-10 00:00:00.000     13             2.0           ⇒
                                P
1981-09-26 00:00:00.000     14             .0            ⇒
 None                           P
EMPLOYER_APPROVED COMMENTS
----------------- ------------------------------
Y                 This is the comments field
Y
Y
Y
Y                 This is an update
Y                 This is an update
Y                 This is an update
Y
```

Note. For Microsoft SQL Server 7 the AUDIT_OPRID field value will be NULL.

Creating Queries to View Audit Records Details

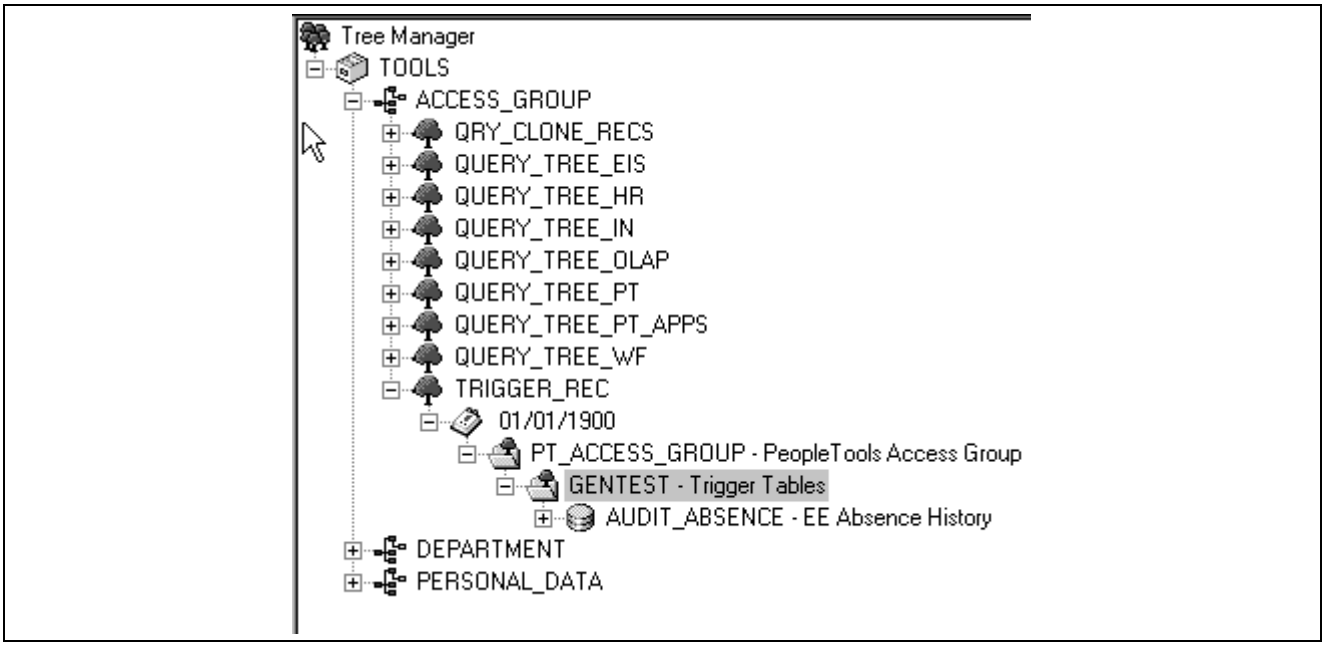
One way to view the information is to use PeopleSoft Query. This section assumes a working knowledge of PeopleSoft Query, and provides some sample queries that show the type of information that you can expect to view.

This section discusses how to:

- Create an access group.
- List all audit records in PS_AUDIT_ABSENCE.
- List all audit records for a specified user ID.
- List all audit records that contain an invalid OPRID.
- List all audit records for a specified time period.

Creating an Access Group

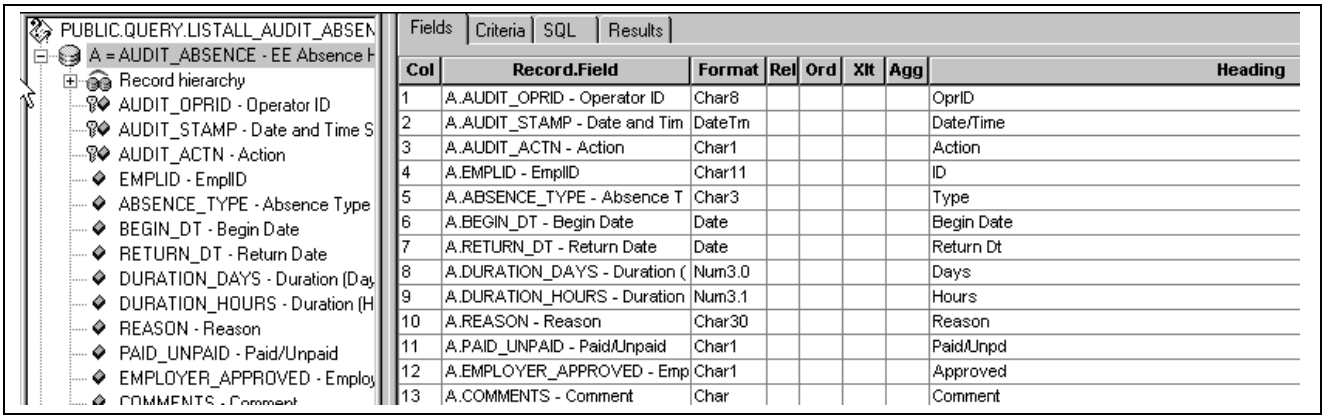
To track audit records, it's useful to create an Access Group in PeopleSoft Tree Manager that contains all audit records. This makes it easier to access the audit records under PeopleSoft Query:



Creating an Access Group

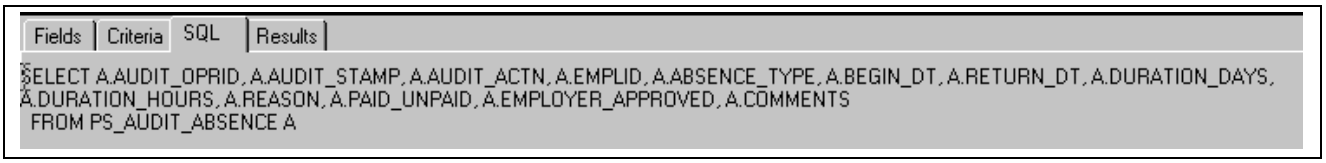
Listing All Audit Records in PS_AUDIT_ABSENCE

Select all the fields from AUDIT_ABSENCE. There are no extra criteria to add:



Query Criteria

The SQL is:



Query SQL

This example illustrates the results:

Fields | Criteria | SQL | Results

OprID	Date/Time	Action	ID	Typ	Begin Date	Return Dt	Day	Hour	Reason
BARNEY	2000-01-19-11.41.16.043000	I	G205	VAC	01/19/2000	09/09/2000	234	0.0	for fun
PS	2000-01-19-10.27.52.190000	I	GH07	CNF	01/19/2000	08/08/2000	202	0.0	
PTDMO	2000-01-19-10.27.02.760000	I	8665	DSC	01/19/2000	09/09/2000	234	0.0	
PTDMO	2000-01-19-10.27.19.713000	D	8665	DSC	01/19/2000	09/09/2000	234	0.0	
VP1	2000-01-19-10.20.32.630000	B	8001	DSC	09/12/1981	09/26/1981	14	0.0	
VP1	2000-01-19-10.20.32.640000	A	8001	VAC	09/12/1981	09/26/1981	14	0.0	
VP1	2000-01-19-10.20.39.660000	B	8001	PER	03/02/1983	03/07/1983	5	0.0	
VP1	2000-01-19-10.20.39.670000	A	8001	VAC	03/02/1983	03/07/1983	5	0.0	
VP1	2000-01-19-10.20.53.840000	D	8001	VAC	01/19/2000		45	0.0	For Fun in the Sun
VP1	2000-01-19-10.22.50.027000	I	8001	VAC	01/19/2000		34	0.0	For Fun
VP1	2000-01-19-10.24.30.423000	D	8001	VAC	01/19/2000		34	0.0	For Fun

Query Results

Listing All Audit Records for a Specified User ID

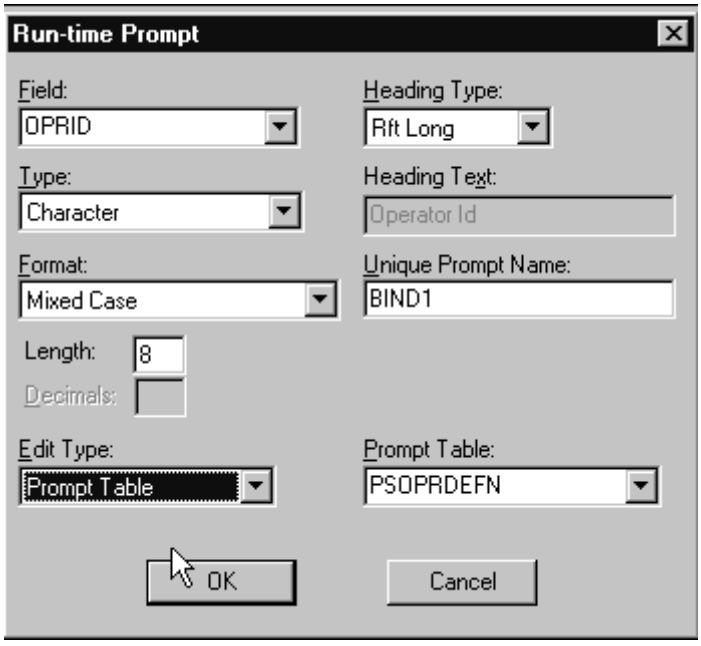
This query is similar to the previous one but with the following criteria added:

Fields | Criteria | SQL | Results

Logical	Expression 1	Operator	Expression 2
	A.AUDIT_OPRID - Operator ID	in list	(:1)

Query Criteria

The example shows the definition of the prompt for AUDIT_OPRID:

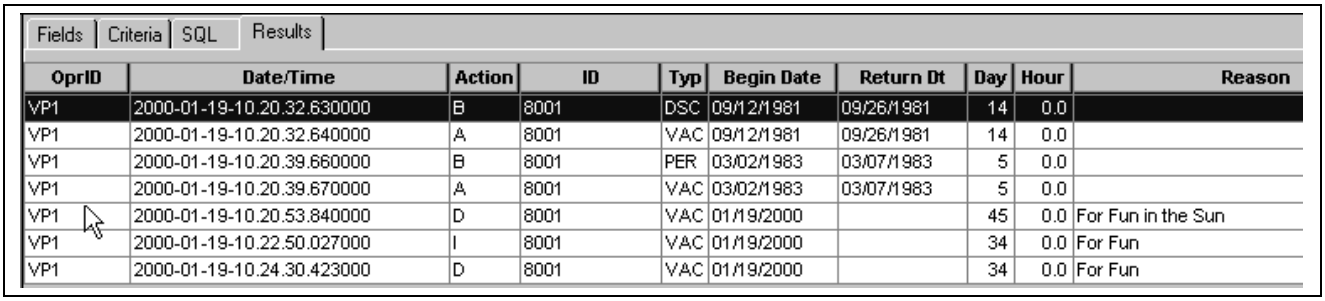


Query Run-time Prompt Dialog

Set up a prompt for User ID against the PSOPRDEFN table. That way, when you run the query, you can specify a particular User ID. In this case, the query focuses on User ID *VP1:*

Specifying a User ID

This example shows the results:

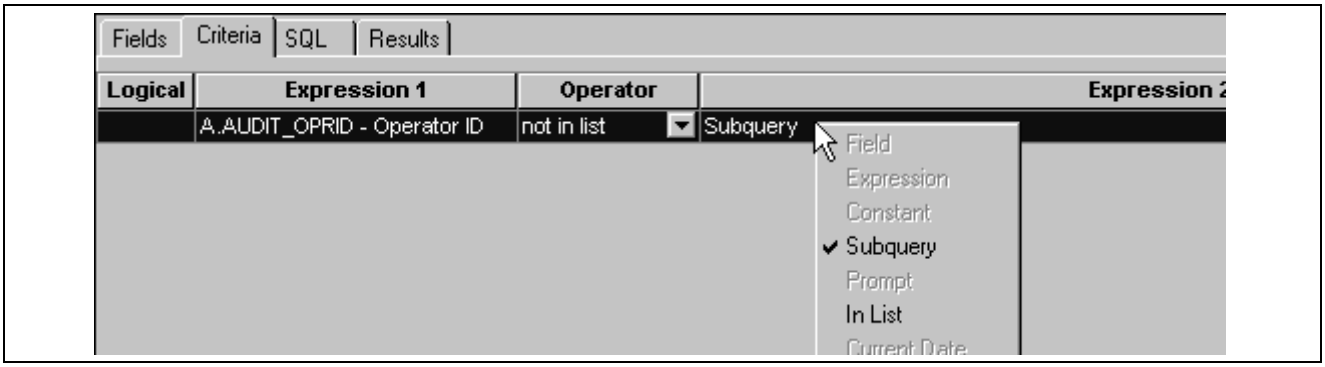


OprID	Date/Time	Action	ID	Typ	Begin Date	Return Dt	Day	Hour	Reason
VP1	2000-01-19-10.20.32.630000	B	8001	DSC	09/12/1981	09/26/1981	14	0.0	
VP1	2000-01-19-10.20.32.640000	A	8001	VAC	09/12/1981	09/26/1981	14	0.0	
VP1	2000-01-19-10.20.39.660000	B	8001	PER	03/02/1983	03/07/1983	5	0.0	
VP1	2000-01-19-10.20.39.670000	A	8001	VAC	03/02/1983	03/07/1983	5	0.0	
VP1	2000-01-19-10.20.53.840000	D	8001	VAC	01/19/2000		45	0.0	For Fun in the Sun
VP1	2000-01-19-10.22.50.027000	I	8001	VAC	01/19/2000		34	0.0	For Fun
VP1	2000-01-19-10.24.30.423000	D	8001	VAC	01/19/2000		34	0.0	For Fun

Viewing Query Results

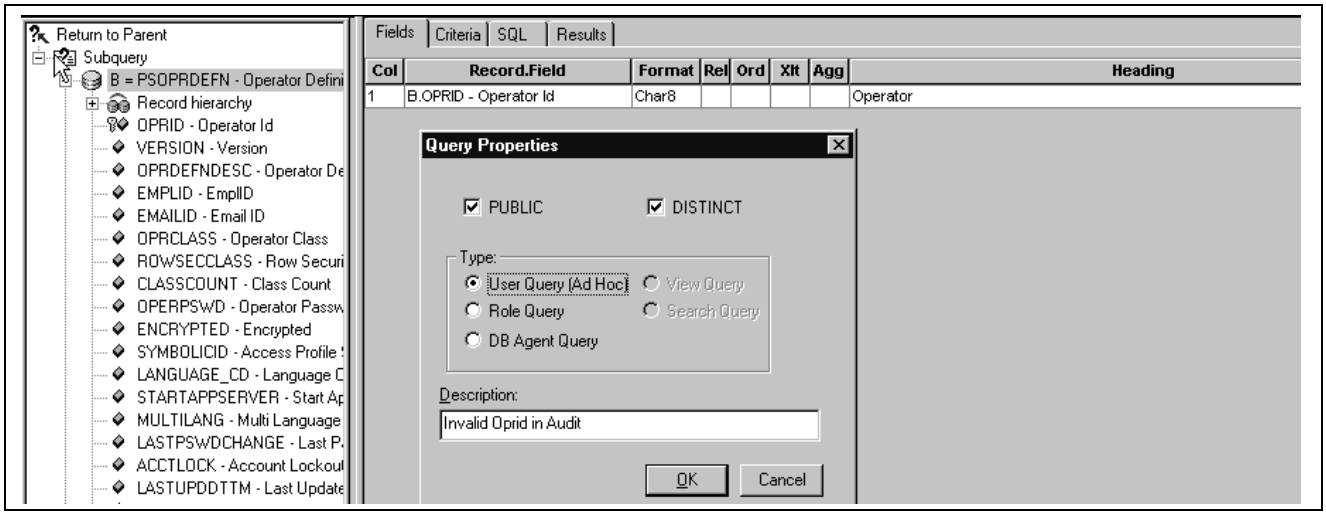
Listing All Audit Records Containing an Invalid OPRID

This query is similar to the previous one, but you specify different criteria:



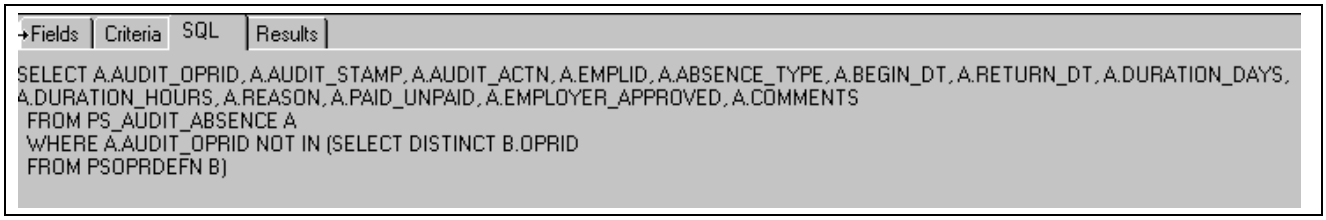
Query Criteria

Right click on Expression 2:



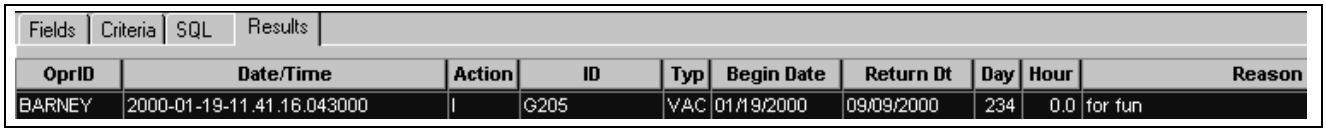
Viewing Query Properties

The subquery selects distinct User ID from PSOPRDEFN. This example shows the SQL for the query:



Query SQL

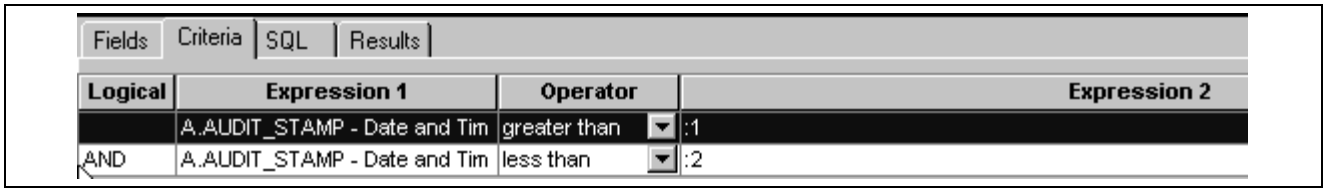
This example shows the results of the query:



Query Results

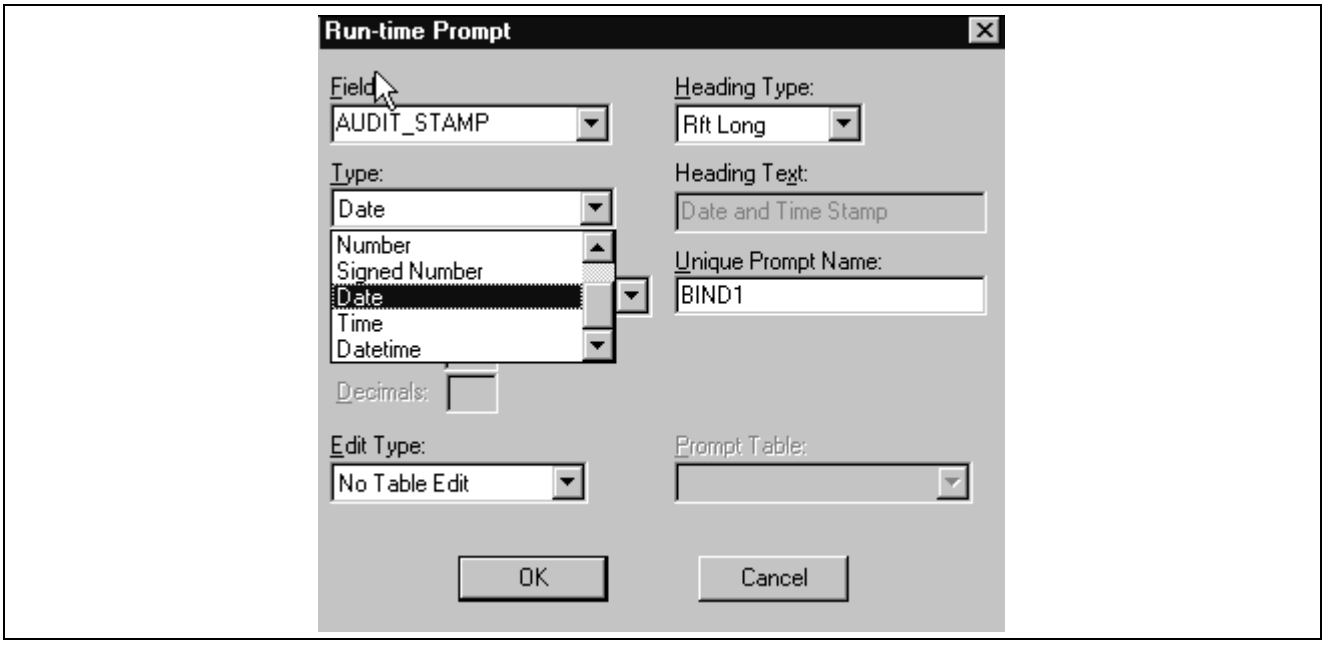
Listing All Audit Records for a Specified Time Period

This example shows a query containing the same fields as in the previous queries above, with different criteria:



Query Criteria

Set the run-time prompts to follow this example.



Run-time Prompt dialog box

The prompts for the AUDIT_STAMP need to have the type changed from date and time to date. This enables the query to take advantage of the calendar control as a prompt mechanism.

This example shows the results of the query:

Fields | Criteria | SQL | Results

OprID	Date/Time	Action	ID	Typ	Begin Date	Return Dt	Day	Hour	Reason
BARNEY	2000-01-19-11.41.16.043000	I	G205	VAC	01/19/2000	09/09/2000	234	0.0	for fun
PS	2000-01-19-10.27.52.190000	I	GH07	CNF	01/19/2000	08/08/2000	202	0.0	
PTDMO	2000-01-19-10.27.02.760000	I	8665	DSC	01/19/2000	09/09/2000	234	0.0	
PTDMO	2000-01-19-10.27.19.713000	D	8665	DSC	01/19/2000	09/09/2000	234	0.0	
VP1	2000-01-19-10.20.32.630000	B	8001	DSC	09/12/1981	09/26/1981	14	0.0	
VP1	2000-01-19-10.20.32.640000	A	8001	VAC	09/12/1981	09/26/1981	14	0.0	
VP1	2000-01-19-10.20.39.660000	B	8001	PER	03/02/1983	03/07/1983	5	0.0	
VP1	2000-01-19-10.20.39.670000	A	8001	VAC	03/02/1983	03/07/1983	5	0.0	
VP1	2000-01-19-10.20.53.840000	D	8001	VAC	01/19/2000		45	0.0	For Fun in the Sun
VP1	2000-01-19-10.22.50.027000	I	8001	VAC	01/19/2000		34	0.0	For Fun
VP1	2000-01-19-10.24.30.423000	D	8001	VAC	01/19/2000		34	0.0	For Fun

Query Results

Using Microsoft SQL Server Trigger Information

This section discusses how to:

- Use Microsoft SQL Server trigger syntax.
- Use Microsoft SQL Server to capture text/image columns.
- Administer Microsoft SQL Server trigger maintenance.

Note. For Microsoft SQL Server, Image and Text Columns in tables can't be selected from the trigger tables INSERTED and DELETED.

Using Microsoft SQL Server Trigger Syntax

To audit INSERTS, UPDATES, and DELETES of the records, use trigger with the following format:

Replace the names in emphasized text with the appropriate names for the trigger that you are constructing.

```
CREATE TRIGGER PS_ABSENCE_HIST_TR ON PS_ABSENCE_HIST
FOR DELETE , INSERT , UPDATE
AS
SET NOCOUNT ON
DECLARE @XTYPE CHAR(1), @OPRID CHAR(8)
SET @OPRID = NULL
The code in brackets below will only be generate on SQL Server
2000 platforms.
Context_info is new functionality in SQL Server 2000.
[SELECT @OPRID = substring(cast(context_info as char(128)),
1,(charindex(',',cast(context_info as char(128)))-1))
          FROM master..sysprocesses
          WHERE spid = @@spid]
-- Determine Transaction Type
IF EXISTS (SELECT * FROM DELETED)
BEGIN
SET @XTYPE = 'D'
END
IF EXISTS (SELECT * FROM INSERTED)
BEGIN
IF (@XTYPE = 'D')
 BEGIN
  SET @XTYPE = 'U'
 END
ELSE
 BEGIN
  SET @XTYPE = 'I'
 END
END
-- Transaction is a Delete
IF (@XTYPE = 'D')
BEGIN
INSERT INTO PS_AUDIT_ABSENCE
(AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED)
SELECT @OPRID,getdate(),'D',
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED FROM deleted
END
-- Transaction is a Insert
IF (@XTYPE = 'I')
BEGIN
INSERT INTO PS_AUDIT_ABSENCE
(AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
```

```
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED)
SELECT @OPRID,getdate(),'A',
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED FROM inserted
END
-- Transaction is a Update
IF (@XTYPE = 'U')
BEGIN
-- Before Update
INSERT INTO PS_AUDIT_ABSENCE
(AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED)
SELECT @OPRID,getdate(),'K',
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED FROM deleted
-- After Update
INSERT INTO PS_AUDIT_ABSENCE
(AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED)
SELECT @OPRID,getdate(),'N',
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED FROM inserted
END
```

Using Microsoft SQL Server to Capture Text/Image Columns

If you want to audit text or image columns with SQL Server you will have to manually alter the trigger scripts that are generated. The trigger scripts that generated through the online pages do not support text or image columns. Below is an example of how a join against the base table can capture the value of the COMMENTS field after an insert, or update is performed.

```
CREATE TRIGGER PS_ABSENCE_HIST_TR ON PS_ABSENCE_HIST
FOR DELETE , INSERT , UPDATE
AS
SET NOCOUNT ON
DECLARE @XTYPE CHAR(1), @OPRID CHAR(8)
SET @OPRID = NULL
The code in brackets below will only be generate on SQL
Server 2000 platforms.
Context_info is new functionality in SQL Server 2000.
[SELECT @OPRID = substring(cast(context_info as char(128)),1,
(charindex(',',cast(context_info as char(128)))-1))
          FROM master..sysprocesses
          WHERE spid = @@spid]
IF EXISTS (SELECT * FROM DELETED)
BEGIN
SET @XTYPE = 'D'
```

```
END
IF EXISTS (SELECT * FROM INSERTED)
BEGIN
IF (@XTYPE = 'D')
 BEGIN
  SET @XTYPE = 'U'
 END
ELSE
 BEGIN
  SET @XTYPE = 'I'
 END
END
-- Transaction is a Delete
IF (@XTYPE = 'D')
BEGIN
INSERT INTO PS_AUDIT_ABSENCE
(AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED,COMMENTS)
SELECT @OPRID,getdate(),'D',
A.EMPLID,A.ABSENCE_TYPE,A.BEGIN_DT,A.RETURN_DT,A.DURATION_DAYS,
A.DURATION_HOURS,A.REASON,A.PAID_UNPAID,A.EMPLOYER_APPROVED,''
FROM deleted A
END
-- Transaction is a Insert
IF (@XTYPE = 'I')
BEGIN
INSERT INTO PS_AUDIT_ABSENCE
(AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED,COMMENTS)
SELECT @OPRID,getdate(),'A',
A.EMPLID,A.ABSENCE_TYPE,A.BEGIN_DT,A.RETURN_DT,A.DURATION_DAYS,
A.DURATION_HOURS,A.REASON,A.PAID_UNPAID,A.EMPLOYER_APPROVED,B.COMMENTS
FROM inserted A, PS_ABSENCE_HIST B
WHERE A.EMPLID = B.EMPLID
  AND A.ABSENCE_TYPE = B.ABSENCE_TYPE
  AND A.BEGIN_DT = B.BEGIN_DT
END
-- Transaction is a Update
IF (@XTYPE = 'U')
BEGIN
-- Before Update
INSERT INTO PS_AUDIT_ABSENCE
(AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED,COMMENTS)
SELECT @OPRID,getdate(),'K',
A.EMPLID,A.ABSENCE_TYPE,A.BEGIN_DT,A.RETURN_DT,A.DURATION_DAYS,
A.DURATION_HOURS,A.REASON,A.PAID_UNPAID,A.EMPLOYER_APPROVED,''
```

```
FROM deleted A

-- After Update
INSERT INTO PS_AUDIT_ABSENCE
(AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED,COMMENTS)
SELECT @OPRID,getdate(),'N',
A.EMPLID,A.ABSENCE_TYPE,A.BEGIN_DT,A.RETURN_DT,A.DURATION_DAYS,
A.DURATION_HOURS,A.REASON,A.PAID_UNPAID,A.EMPLOYER_APPROVED,B.COMMENTS
FROM inserted A, PS_ABSENCE_HIST B
WHERE A.EMPLID = B.EMPLID
  AND A.ABSENCE_TYPE = B.ABSENCE_TYPE
  AND A.BEGIN_DT = B.BEGIN_DT
END
```

Administering Microsoft SQL Server Trigger Maintenance

The following commands may be helpful when administering triggers.

List All Triggers in a Database

This command lists all triggers in a database:

```
SELECT name  FROM sysobjects WHERE type = 'TR'
```

List the Trigger Definition

This command lists the trigger definition:

```
sp_helptext TRIGGERNAME
```

List Trigger Information

This command lists trigger information:

```
sp_helptrigger BASE TABLE NAME
```

Returns the type or types of triggers that are defined on the specified table for the current database.

```
sp_help TRIGGERNAME
```

Reports information about a database object (any object listed in the sysobjects table), a user-defined data type, or a data type that Microsoft SQL Server supplies.

To Remove a Trigger

To remove a trigger:

```
drop trigger TRIGGERNAME
```

To Modify an Existing Trigger

To modify a trigger:

```
ALTER trigger ...
```

See the full definition in SQL Server Books Online.

This alters the definition of a trigger that is created previously by the CREATE TRIGGER statement.

To Disable a Trigger

To disable a trigger:

```
ALTER TABLE table     | (ENABLE | DISABLE) TRIGGER  (ALL | trigger_name[,...n])
```

(ENABLE | DISABLE) TRIGGER - Specifies that trigger_name is enabled or disabled. When a trigger is disabled, it is still defined for the table; however, when INSERT, UPDATE or DELETE statements are executed against the table, the actions in the trigger are not performed until the trigger is reenabled.

- ALL: Specifies that all triggers in the table are enabled or disabled.
- trigger_nam: Specifies the name of the trigger to disable or enable.

Using DB2 UDB for OS/390 and z/OS Trigger Information

This section provides an overview of DB2 z/OS trigger information and discusses:

- Assembler program AUDIT01 for DB2 z/OS.
- User-defined function (external scalar) requirements.
- Sample DB2 UDB syntax to create UDF function AUDIT01.
- Verifying status of UDF function.
- Verifying monitor trace setting.
- DB2 z/OS trigger syntax.
- DB2 z/OS trigger maintenance

Understanding DB2 z/OS Trigger Information

The following topics describe the syntax, commands, and additional tasks, such as the assembler program, involved with DB2 z/OS triggers.

Before the Trigger Audit can be implemented on DB2 z/OS, several components must be implemented in the appropriate sequence:

- Work Load Manager (WLM) must be enabled.
- Assembler module AUDIT01 must be assembled.
- User Defined Function (UDF) AUDIT01 must be defined in DB2 UDB.
- Monitor Trace Class 1 must be enabled in DB2 UDB.
- Trigger statement must be defined.

Assembler Program AUDIT01 for DB2 z/OS

To implement triggers for DB2 z/OS, you need to create an assembler program. Here's a sample of it if it's needed.

```
         MACRO                                                  00010000
```

```
&LABEL   CLEAR &DATAREA,               DATA AREA TO BE CLEARED        *00020000
               &LEN=,                  LENGTH OF DATA AREA            *00040000
               &FILL=                  FILL CHARACTER                  00030000
&LABEL   LA    14,&DATAREA             ADDR OF FIELD TO BE CLEARED     00160000
         L     15,=A(&LEN)             REQUESTED LENGTH OF CLEAR       00340000
         LA    1,&FILL                 CLEAR VALUE                     00450000
         SLL   1,24                    SHIFT CLEAR VALUE TO FILL       00500000
         MVCL  14,0                    CLEAR FIELD                     00510000
         MEND
*
         MACRO
         C2X   &IN,&IN_L,                      INPUT & INPUT LENGTH    *
               &OUT,&OUT_L,                    OUTPUT & OUTPUT LENGTH  *
               &WORK=WORK,&TRTABLE=TRTABLE     WORK AREA AND TRAN TBL
         MVO   &WORK.(&IN_L.+1),&IN.(&IN_L)    MOVE WITH OFFSET
         UNPK  &OUT.(&OUT_L.),&WORK.(&IN_L.+1) UNPACK TO CREATE BASE
         NC    &OUT.(&OUT_L.),=8X'0F'          LEAVE LOW NIBBLE
         TR    &OUT.(&OUT_L.),&TRTABLE         TRANSLATE CHARS
         MEND
         EJECT
*
*
*Program audit01
*
*
AUDIT01  CEEENTRY AUTO=WORKSIZE,MAIN=YES,PLIST=OS
         USING WORKAREA,R13
*
         LR    R6,R1              SAVE THE PARMLIST ADDRESS
*
         LA    R10,XIFCA          ADDRESS FOR IFI COMM AREA
         LA    R9,XWQAL           ADDRESS FOR IFI QUALIFICATION AREA
         USING IFCA,R10
         USING WQAL,R9
*
         CLEAR XIFCA,LEN=L'XIFCA,FILL=X'00'
         CLEAR XWQAL,LEN=L'XWQAL,FILL=X'00'
*
         MVC   IFCAID,=CL4'IFCA'   EYE CATCHER
         MVC   IFCALEN,=AL2(L'XIFCA) LENGTH OF IFCA
         MVC   IFCAOWNR,=CL4'PSFT'
*
         MVC   WQALEYE,=CL4'WQAL' EYE CATCHER
         MVC   WQALLEN,=AL2(WQALLN5)  LENGTH
         MVC   WQALACE,=XL4'5C'   ACE TOKEN = '*', GET YOURS ONLY
*                                 ACE TOKEN = 0, GET ALL
*                                 ACE TOKEN = TOKEN, GET SPECIFIC
*
         MVC   RETALEN,=F'32004'  LENGTH FOR RETURN AREA
*
```

```
         MVC   AIFICMD,=A(READSCMD) BUILD
         LA    R3,XIFCA               THE
         ST    R3,AIFCA                 PARMLIST
         LA    R4,RETAREA                 FOR
         ST    R4,ARETAREA                  THE
         MVC   AIFCID,=A(XIFCID)              IFI
         LA    R5,XWQAL                         READS
         ST    R5,AWQAL                           FOR
         OI    AWQAL,X'80'                          IFCID=148
*
         LA    R1,IFIPARMS        PARMLIST FOR READS CALL
         L     R15,=V(DSNWLI)
         BALR  R14,R15
*
*
         EJECT
*
*
         CLEAR OPRIDVAL,LEN=L'OPRIDVAL,FILL=X'40'
         MVC   OPRIDLEN,=YL2(L'OPRIDVAL)   RETURN LENGTH OF FIELD
         MVC   OPRIDIND,=H'0'     INDICATOR FOR RESULT ALWAYS RETURNED
*
         CLC   IFCARC1,=F'0'      RETURN CODE = ZERO
         BNE   AUDIT30            NO, RETURN WITH ERROR INDCATOR
*
         CLC   IFCABM,=F'0'       ANY DATA RETURNED
         BE    AUDIT35            NO, RETURN WITH ERROR INDCATOR
*
         LA    R4,4(,R4)          ADDRESS THE RETURNED DATA
         USING QWIW,R4
         CLC   QWIWLEN,=H'0'      DATA INDICATED BY IFI WRITER HDR
         BE    AUDIT40            NO, RETURN WITH ERROR INDICATOR
         DROP  R4
*
         LA    R5,4(0,R4)         ADDRESS SELF DEFINING SECTION
         USING QWT0,R5
*
         CLC   QWT02R5O,=F'0'     PRODUCT SECTION OFFSET
         BE    AUDIT50            NO, RETURN WITH ERROR INDICATOR
         CLC   QWT02RAO,=F'0'     ACCOUNTING FACILITY DATA
         BE    AUDIT55            NO, RETURN WITH ERROR INDICATOR
*
         LR    R7,R4              BASE FOR ALL OFFSETS
         A     R7,QWT02R5O        ADDRESS OF CORRELATION INFO
         USING QWHC,R7            ADDRESSABILITY
*
         LR    R8,R4              BASE FOR ALL OFFSETS
         A     R8,QWT02RAO        ADDRESS OF ACCOUNTING FACILITY DATA
         USING QMDA,R8            ADDRESSABILITY
*
```

```
         CLC   QWHCLEN,=H'0'      STANDARD HEADER HAVE DATA
         BE    AUDIT60            NO
*
         MVC   HDRTYPE,QWHCTYP    SAVE CORR HEADER TYPE
         CLI   QWHCTYP,X'02'      CORRELATION HEADER TYPE = 2
         BNE   AUDIT63            NO
*
*
         EJECT
*
*
         MVC   OPRIDVAL(8),QWHCAID     DEFAULT TO THE LOCAL ACCESS ID
         CLC   QWHCATYP,=AL4(QWHCRUW)  IF DRDA PROTOCOL
         BE    PROCESS_DRDA               GO THERE
         CLC   QWHCATYP,=AL4(QWHCTSO)  IF TSO ACCESS
         BE    PROCESS_TSO_CAF            GO THERE
         CLC   QWHCATYP,=AL4(QWHCDB2C) IF OTHER THAN DRDA,TSO,CAF
         BNE   AUDIT66                    EXIT WITH ERROR
*
*
PROCESS_TSO_CAF EQU *
         CLC   QMDAPTYP,=C'DSN'        IF LOCAL ACCESS TYPE IS NOT DSN
         BNE   AUDIT70                    ERROR
         LA    R4,QMDAASTR             ADDR OF THE ACCOUNT STRING
         USING QMDAINFO,R4             ADDRESSABILITY ACCT STRING
         LA    R4,QMDAACCT             ADDR OF THE MVS ACCOUNT STRING
         DROP  R4
         SR    R5,R5                   CLEAR FOR LENGTH INSERT
         IC    R5,QMDAASLN             LENGTH OF QMDAAINF
         AR    R5,R8                   ESTABLISH END OF THE QMDA AREA
         SH    R5,=YL2(L'PS_USERID-1)  STOP SEARCH WHEN CAN'T BE FOUND
*
SEARCH_PS_USERID EQU *
         CLC    0(L'PS_USERID,R4),PS_USERID   LOOK FOR PS_USERID
         BE     FOUND_PS_USERID               LEAVE LOOP IF FOUND
         LA     R4,1(0,R4)           NEXT CHAR IN MVS ACCT STRING
         CR     R4,R5                IF MORE OF THE STRING TO SEARCH
         BNH    SEARCH_PS_USERID        CONTINUE
         B      AUDIT20              ELSE PS_USERID NOT FOUND - LEAVE
FOUND_PS_USERID EQU *
         LA     R4,L'PS_USERID(0,R4)       BUMP PAST PS_USERID TO DATA
         AH     R5,=YL2(L'PS_USERID-1)     ADJ TO ACTUAL END ACCT STR
         CR     R4,R5                      IF A VALUE AFTER PS_USERID
         BNH    SETUP_SEARCH_END_PS_USERID    CONTINUE
         LR     R4,R5                      ELSE -- SET VALUE=END
SETUP_SEARCH_END_PS_USERID EQU *
         LR     R2,R4                      SAVE BEGIN OF ACTUAL USERID
*
SEARCH_END_PS_USERID EQU *
         CLI    0(R4),X'FF'           IF END OF MVS ACCOUNT SUB-FIELD
```

```
         BE     FOUND_END_PS_USERID      LEAVE SEARCH - GOT THE END
         LA     R4,1(0,R4)             NEXT CHAR IN MVS ACCT STRING
         CR     R4,R5                  IF MORE OF THE STRING TO SEARCH
         BNH    SEARCH_END_PS_USERID     CONTINUE
FOUND_END_PS_USERID EQU *
         LR     R5,R4                  END OF ACTUAL USERID
         SR     R5,R2                  LENGTH OF USERID = END - BEGIN
         LR     R4,R2                  END OF THE USERID STRING
         ICM    R5,B'1000',=X'40'      SPACE AS PAD CHAR IN MVCL
         LA     R14,OPRIDVAL           UDF OUTPUT AREA
         L      R15,=A(L'OPRIDVAL)     LENGTH OF UDF OUTPUT AREA
         MVCL   R14,R4                 MOVE THE PROPER LENGTH
         B      AUDIT20                USE THIS RETURN VALUE
*
*
PROCESS_DRDA EQU *
         MVC   OPRIDVAL(16),QWHCEUID USE THE OPERID FROM CORR HDR
*                                    THIS IS THE OLD TOOLS API
         CLC   QMDAPTYP,=C'SQL'      CHECK FOR DB2 CLIENT/SERVER
         BNE   AUDIT75               ERROR - S/B SQL
         LA    R8,QMDAASTR           ACCESS THE ACCOUNTING STRING
         DROP  R8
         USING QMDASQLI,R8           CLIENT PLATFORM ACCTING STRING
         CLI   QMDASFLN,X'00'        IF DDCS ACCT SUF LEN=0
         BE    AUDIT20                  USE 16 BYTE VALUE FROM CORR
*
         LA    0,QMDASUFX            DDCS ACCT SUFFIX STRING BEGIN
         SR    R1,R1                 INIT LENGTH FIELD
         IC    R1,QMDASFLN           LENGTH OF DDCS ACCT STRING
         ICM   R1,B'1000',=X'40'     PAD WITH BLANKS
         LA    R14,OPRIDVAL          TARGET OF THE MOVE - OPRIDVAL
         L     R15,=A(L'OPRIDVAL)    LENGTH OF OPRIDVAL
         MVCL  R14,R0                MOVE IT BASED ON DDCS STR LEN
         DROP  R8
*
*
AUDIT20  EQU   *
         L     R5,4(,R6)          ADDRESS OF THE INDICATOR FOR RETURN
         L     R6,0(,R6)          ADDRESS OF THE RESULT FOR RETURN
         MVC   0(L'OPRIDVAL+2,R6),OPRID     RESULT IS VARCHAR(30)
         MVC   0(2,R5),OPRIDIND   INDICATE A RESULT IS RETURNED
*
         CEETERM  RC=0
*
*
         EJECT
*
*
AUDIT30  EQU   *
         MVC   OPRIDVAL(4),=CL4'RC1='
```

```
         MVC   OPRIDVAL+4(4),IFCARC1
         MVC   OPRIDVAL+8(4),=CL4'RC2='
         MVC   OPRIDVAL+12(4),IFCARC2
         B     AUDIT30_35_EXIT
AUDIT35  EQU   *
         MVC   OPRIDVAL(16),=CL16'IFCABM=0'
AUDIT30_35_EXIT EQU *
 wto 'xifca'
 lr r2,r3
 la r3,12
 bal r14,wto
         B     AUDIT20
*
AUDIT40  EQU   *
         MVC   OPRIDVAL(16),=CL16'QWIWLEN=0'
 wto 'qwiw'
 lr r2,r4
 la r3,1
 bal r14,wto
         B     AUDIT20
*
AUDIT50  EQU   *
         MVC   OPRIDVAL(16),=CL16'QWT02R50=0'
         B     AUDIT50_55_EXIT
AUDIT55  EQU   *
         MVC   OPRIDVAL(16),=CL16'QWT02RA0=0'
AUDIT50_55_EXIT EQU *
 wto 'qwt0'
 lr r2,r5
 la r3,10
 bal r14,wto
         B     AUDIT20
*
AUDIT60  EQU   *
         MVC   OPRIDVAL(16),=CL16'NO STD HDR LEN'
         B     AUDIT60_63_66_EXIT
AUDIT63  EQU   *
         MVC   OPRIDVAL(16),=CL16'QWHC BAD TYPE=X'
         MVC   OPRIDVAL+14(1),HDRTYPE
         B     AUDIT60_63_66_EXIT
AUDIT66  EQU   *
         MVC   OPRIDVAL(16),=CL16'S/B TSO,CAF,DRDA'
AUDIT60_63_66_EXIT EQU *
 wto 'qwhc'
 lr r2,r7
 la r3,12
 bal r14,wto
         B     AUDIT20
*
AUDIT70  EQU   *
```

```
         MVC   OPRIDVAL(16),=CL16'NO DSN-LOCAL'
         B     AUDIT70_75_EXIT
AUDIT75  EQU   *
         MVC   OPRIDVAL(16),=CL16'NO SQL-DRDA'
AUDIT70_75_EXIT EQU *
 wto 'qmda'
 lr r2,r8
 la r3,12
 bal r14,wto
         B     AUDIT20
*
*
 eject
*
*
WTO equ *
 st r14,WTO_return
*
 clear output,len=l'output,fill=x'40'
WTO_loop equ *
 st r2,temp
 c2x temp+00,4,output_addr,8
 mvc temp,0(r2)
 c2x temp+00,4,output_hex1,8
 c2x temp+04,4,output_hex1+08,8
 c2x temp+08,4,output_hex2,8
 c2x temp+12,4,output_hex2+08,8
 mvc output_disp(l'temp),temp
 mvc output_ll,=yl2(l'output)
 wto text=output_ll
 la r2,l'temp(0,r2)
 bct r3,WTO_loop
*
 l r14,WTO_return
 br r14
*
*
         EJECT
*
*
*****************************************************************
*  VARIABLE DECLARATIONS AND EQUATES                            *
*****************************************************************
         YREGS
PPA      CEEPPA  ,                 CONSTANTS DESCRIBING THE CODE BLOCK
READSCMD DC    CL8'READS'
XIFCID   DC    AL2(XIFCIDL)      SET LENGTH OF BLOCK
         DC    H'0'               RESERVED
         DC    H'148'             READS FOR IFCID=148
XIFCIDL  EQU   *-XIFCID
```

```
*
PS_USERID DC C'PS_USERID='        *** USE FOR ACCOUNT IDENTIFIER ***
*
trtable        dc  c'0123456789ABCDEF'
*
         LTORG ,                  PLACE LITERAL POOL HERE
*
WORKAREA DSECT
         ORG   *+CEEDSASZ         LEAVE SPACE FOR DSA FIXED PART
*
OPRID    DS    0F                 RESULT AREA FOR OPRID
OPRIDLEN DS    H                  LENGTH
OPRIDVAL DS    CL30               OPRID FROM COORELATION HEADER
OPRIDIND DS    H                  OPRID INDICATOR FOR UDF INTERFACE
*
HDRTYPE  DS    X
*
IFIPARMS DS    0F                 PARMS FOR IFI READS CALL
AIFICMD  DS    A(READSCMD)        READS COMMAND
AIFCA    DS    A(XIFCA)           IFCA COMMUNICATION AREA
ARETAREA DS    A(RETAREA)         RETURN AREA FOR OUR 147 IFCID
AIFCID   DS    A(XIFCID)          IFI AREA TO SELECT 147 ONLY
AWQAL    DS    A(XWQAL)           IFI QUAL AREA TO SET OUR ACEADDR
*
XIFCA    DS    XL(IFCEND-IFCA)    STORAGE FOR IFCA
XWQAL    DS    XL(WQALEND-WQAL)   STORAGE FOR WQAL
*
work           ds  d
WTO_return     ds  f
temp           ds  cl16
output_ll      ds  yl2
output         ds  0cl(8+1+8+8+1+8+8+1+16)
output_addr    ds  cl8
               dc  cl1' '
output_hex1    ds  2cl8
               dc  cl1' '
output_hex2    ds  2cl8
               dc  cl1' '
output_disp    ds  cl16
*
*
RETAREA  DS    0F
RETALEN  DS    F
RETRECS  DS    XL32000
WORKSIZE EQU   *-WORKAREA
*
         DSNDIFCA ,               MAPPING OF IFI-COMM-AREA
         DSNDWQAL ,               MAPPING OF IFCID-QUAL-AREA
         DSNDQWIW ,               MAPPING OF IFI-WRITER-HEADER
         DSNDQWT0 ,               MAPPING OF TRACE SELF DEFINING SECT
```

```
DSNDQW02 ,              INCLUDES IFC 148 MAPPING
DSNDQWHC ,              MAPPING FOR CORRELATION INFORMATION
DSNDQMDA ,              MAPPING FOR ACCOUNTING FACILITY DATA
CEEDSA ,                MAPPING OF THE DYNAMIC SAVE AREA
CEECAA ,                MAPPING OF THE COMMON ANCHOR AREA
END   AUDIT01
```

Sample JCL to Compile AUDIT01

This is a sample JCL compile:

```
//PROC  JCLLIB ORDER=(PSHLQ.PPVVV.PROCLIB)
//AUDIT01     EXEC PSASM,PSLIST='*',MEM=AUDIT01,
//            PSCOPY='PSHLQ.PPVVV.COPYLIB',
//            PSSRCE='PSHLQ.PPVVV.SOURCE',⇒
   --> Where you have the AUDIT program Source Code
//            PSLOAD='SYS2.WLMDSND.LOAD',⇒
   --> Load Library defined in WLM for the subsystem
//            MACLIB2='CEE.SCEEMAC',
//            MACLIB3='DSN610.SDSNMACS',
//            LKEDLIB='CEE.SCEELKED',
//            LKEDLIB2='DSN610.SDSNLOAD'
//*
//ASM.SYSLIB  DD  DISP=SHR,DSN=&MACLIB
//            DD  DISP=SHR,DSN=&MACLIB2
//            DD  DISP=SHR,DSN=&MACLIB3
//            DD  DISP=SHR,DSN=&PSCOPY
//LKED.SYSLIB DD DISP=SHR,DSN=&LKEDLIB
//            DD DISP=SHR,DSN=&LKEDLIB2
//LKED.SYSLIN DD
//            DD  DDNAME=SYSIN
//LKED.SYSIN  DD *
  INCLUDE SYSLIB(DSNRLI)
  ENTRY AUDIT01
  NAME AUDIT01(R)
/*
```

Note. The DB2 UDB –Display Thread command displays the first 16 bytes of the PeopleSoft User ID in the IFC field. However, the AUDIT01 UDF returns the full 30–byte PeopleSoft User ID if PeopleTools utilizes the proper API to record all 30 bytes.

User-Defined Function (External Scalar) Requirements

This sample shows the user-defined function requirements:

```
Module Name:   AUDIT01
Description:   Upon being invoked from the trigger, it makes IFI READS call⇒
 to get an operator ID and pass the information back to the caller.
```

```
Note:   This program need to run under WLM managed Address Spaces.
Module Type:   IFI READS call program.
Language Type: Assembler
Entry Point:   AUDIT01
Input:      None
Output:      Operator ID
Ext Serv:   WLMAPPLENV - WLM Application Environment
```

Sample DB2 UDB Syntax to Create UDF Function AUDIT01

This example shows the syntax that is used to create the UDF function:

```
CREATE FUNCTION AUDIT01()
       RETURNS VARCHAR(30)
       EXTERNAL NAME 'AUDIT01'
       NO EXTERNAL ACTION
       WLM ENVIRONMENT WLMDSNZ
       PARAMETER STYLE DB2SQL
       LANGUAGE ASSEMBLE         ;
```

Verifying Status of UDF Function

After the User Defined Function is created, verify that the status is STARTED by using this command:

```
-DIS FUNCTION SPECIFIC(PSOFT.*)
DSNX975I < DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS -
------- SCHEMA=PSOFT
FUNCTION           STATUS ACTIVE QUEUED MAXQUE TIMEOUT WLM_ENV
AUDIT01            STARTED     0      0      1       0 WLMDSND
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I < DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

If the function is not in Started status, you can start it by using this command:

```
-START FUNCTION SPECIFIC(PSOFT.AUDIT01)
```

Verifying Monitor Trace Setting

The monitor class 1 must be enabled for the UDF to issue an instrumentation facility interface (IFI) READS request to the IFI facility.

The following is the command:

```
-DIS TRACE(MONITOR)
DSNW127I  " CURRENT TRACE ACTIVITY IS -
TNO TYPE   CLASS        DEST QUAL
01  MON    01           OP1  NO
*********END OF DISPLAY TRACE SUMMARY DATA*********
DSN9022I  " DSNWVCM1 '-DIS TRACE' NORMAL COMPLETION

If the monitor class 1 is not enabled, you can start it using the Command:
 -START TRACE(MONITOR) CLASS(1)
```

DB2 z/OS Trigger Syntax

A trigger for each SQL operation type, as in INSERT, UPDATE and DELETE, needs to be defined separately with a different trigger name for a given triggering table. The allowable trigger name length is eight characters long. S user-defined scalar function returns a single value of User ID by making an IFI READS call each time that it is invoked. The following is a sample of the trigger syntax.

```
CREATE TRIGGER _AUD_TRI
 AFTER  INSERT ON PS_ABSENCE_HIST
 REFERENCING NEW AS C_ROW
 FOR EACH ROW MODE DB2SQL
 INSERT INTO PS_AUDIT_ABSENCE
 VALUES (PSOFT.AUDIT01(),CURRENT TIMESTAMP,'A',
C_ROW.EMPLID,
C_ROW.ABSENCE_TYPE,
C_ROW.BEGIN_DT,
C_ROW.RETURN_DT,
C_ROW.DURATION_DAYS,
C_ROW.DURATION_HOURS,
C_ROW.REASON,
C_ROW.PAID_UNPAID,
C_ROW.EMPLOYER_APPROVED);
CREATE TRIGGER _AUD_TRD
 AFTER DELETE ON PS_ABSENCE_HIST
 REFERENCING OLD AS C_ROW
 FOR EACH ROW MODE DB2SQL
 INSERT INTO PS_AUDIT_ABSENCE
 VALUES (PSOFT.AUDIT01(),CURRENT TIMESTAMP,'D',
C_ROW.EMPLID,
C_ROW.ABSENCE_TYPE,
C_ROW.BEGIN_DT,
C_ROW.RETURN_DT,
C_ROW.DURATION_DAYS,
C_ROW.DURATION_HOURS,
C_ROW.REASON,
C_ROW.PAID_UNPAID,
C_ROW.EMPLOYER_APPROVED);
CREATE TRIGGER AUD_TRUB
 AFTER UPDATE ON PS_ABSENCE_HIST
 REFERENCING OLD AS C_ROW
 FOR EACH ROW MODE DB2SQL
 INSERT INTO PS_AUDIT_ABSENCE
 VALUES (PSOFT.AUDIT01(),CURRENT TIMESTAMP,'K',
C_ROW.EMPLID,
C_ROW.ABSENCE_TYPE,
C_ROW.BEGIN_DT,
C_ROW.RETURN_DT,
C_ROW.DURATION_DAYS,
C_ROW.DURATION_HOURS,
C_ROW.REASON,
C_ROW.PAID_UNPAID,
```

```
C_ROW.EMPLOYER_APPROVED);
```

DB2 z/OS Trigger Maintenance

These commands may be useful for administering triggers.

List All Triggers in a Database

To list all triggers:

```
SELECT name  FROM SYSIBM.SYSTRIGGERS
```

List the Trigger Definition

To list the trigger definition:

```
SELECT text FROM SYSIBM.SYSTRIGGERS WHERE NAME = <trigger name>
```

List Trigger Information

To list the trigger information:

```
SELECT text FROM SYSIBM.SYSTRIGGERS WHERE NAME = <trigger name>
```

To Remove a Trigger

To remove a trigger:

```
drop trigger TRIGGERNAME restrict
```

To Modify an Existing Trigger

This command alters the definition of a trigger that was created previously by the CREATE TRIGGER statement.

```
ALTER trigger ...
```

See *DB2 Universal Database for OS/390 and z/OS SQL Reference.*

See *DB2 Universal Database for OS/390 and z/OS Application Programming and SQL Guide.*

Using Oracle Trigger Information

This section discusses how to:

- Use Oracle trigger syntax.
- Maintaining Oracle triggers.

The triggers that are generated on the Oracle platform reference a function that PeopleSoft delivers to obtain the PS_OPRID. This function must be installed into the Oracle database schema for the PeopleSoft database prior to creating the trigger. This function can be installed by executing the following SQL as the PeopleSoft database owner ID:

```
$PS_HOME\scripts\getpsoprid.sql
```

Using Oracle Trigger Syntax

This example shows the Oracle trigger syntax.

```
drop function GET_PS_OPRID
/
create function GET_PS_OPRID (v_client_info VARCHAR2 )
        return VARCHAR2 is
        i integer;
/* Title: GET_PS_OPRID                                    */
/* Purpose: Retrieves the operator id (OPRID)             */
/*          from a VARCHAR2 comma separated field         */
/*          of the format 'OPRID,OS_USER,MACHINE'         */
/*          If no OPRID is found, it returns '!NoOPRID'   */
/* Limitations: (any grants, privileges, etc)             */
/* Who: PeopleSoft Inc.                                   */
/* Date: 2000-04-07                                       */
begin
        if ( length(v_client_info) IS NULL ) then
                return('!NoOPRID');
        end if;
        if ( substr(v_client_info,1,1) = ',' ) then
                return('!NoOPRID');
        end if;
        i := 1;
        while ( (substr(v_client_info,i,1)) <> ',' and i < 10) loop
                i := i + 1;
        end loop;
        if ( i > 9 ) then
                return('!NoOPRID');
        else
                i := i - 1;
                return (substr (v_client_info, 1, i));
        end if;
end GET_PS_OPRID;
/
grant execute on GET_PS_OPRID to public
/
/* If Transaction is an Insert Or Update              */
/*    Capture After Values                            */
/* If Transaction is a Delete or Update               */
/*    Capture Before Values                           */
CREATE OR REPLACE TRIGGER PS_ABSENCE_HIST_TR
AFTER INSERT OR UPDATE OR DELETE ON PS_ABSENCE_HIST
FOR EACH ROW
DECLARE
        V_AUDIT_OPRID VARCHAR2(64);
BEGIN
    DBMS_APPLICATION_INFO.READ_CLIENT_INFO(V_AUDIT_OPRID);
  IF :OLD.EMPLID IS NULL
```

```
THEN
      INSERT INTO PS_AUDIT_ABSENCE
      VALUES (
              GET_PS_OPRID(V_AUDIT_OPRID) ,
              SYSDATE                                    ,
              'A'                                        ,
              :NEW.EMPLID                                ,
              :NEW.ABSENCE_TYPE                          ,
              :NEW.BEGIN_DT                              ,
              :NEW.RETURN_DT                             ,
              :NEW.DURATION_DAYS                         ,
              :NEW.DURATION_HOURS                        ,
              :NEW.REASON                                ,
              :NEW.PAID_UNPAID                           ,
              :NEW.EMPLOYER_APPROVED
              );
ELSE
    IF :NEW.EMPLID IS NULL
    THEN
      INSERT INTO PS_AUDIT_ABSENCE
      VALUES (
              GET_PS_OPRID(V_AUDIT_OPRID) ,
              SYSDATE                                 ,
              'D'                                     ,
              :OLD.EMPLID                             ,
              :OLD.ABSENCE_TYPE                       ,
              :OLD.BEGIN_DT                           ,
              :OLD.RETURN_DT                          ,
              :OLD.DURATION_DAYS                      ,
              :OLD.DURATION_HOURS                     ,
              :OLD.REASON                             ,
              :OLD.PAID_UNPAID                        ,
              :OLD.EMPLOYER_APPROVED
              );
    ELSE
      INSERT INTO PS_AUDIT_ABSENCE
      VALUES (
              GET_PS_OPRID(V_AUDIT_OPRID) ,
              SYSDATE                                 ,
              'K'                                     ,
              :OLD.EMPLID                             ,
              :OLD.ABSENCE_TYPE                       ,
              :OLD.BEGIN_DT                           ,
              :OLD.RETURN_DT                          ,
              :OLD.DURATION_DAYS                      ,
              :OLD.DURATION_HOURS                     ,
              :OLD.REASON                             ,
              :OLD.PAID_UNPAID                        ,
              :OLD.EMPLOYER_APPROVED
              );
```

```
        INSERT INTO PS_AUDIT_ABSENCE
        VALUES (
                GET_PS_OPRID(V_AUDIT_OPRID) ,
                SYSDATE                                  ,
                'N'                                      ,
                :NEW.EMPLID                              ,
                :NEW.ABSENCE_TYPE                        ,
                :NEW.BEGIN_DT                            ,
                :NEW.RETURN_DT                           ,
                :NEW.DURATION_DAYS                       ,
                :NEW.DURATION_HOURS                      ,
                :NEW.REASON                              ,
                :NEW.PAID_UNPAID                         ,
                :NEW.EMPLOYER_APPROVED
                );
      END IF;
  END IF;
END PS_ABSENCE_HIST_TR;
/
```

Maintaining Oracle Triggers

The following command may be helpful with triggers.

List All Triggers in a Database

To list triggers:

```
SELECT TRIGGERNAME FROM USER_TRIGGERS;
```

Executed from Schema_owner_id

```
SELECT TRIGGERNAME FROM ALL_TRIGGERS;
```

Executed from SYSTEM

The following data dictionary views reveal information about triggers:

- USER_TRIGGERS
-

```
   SQL> descr user_triggers;
 Name                            Null?          Type
 ------------------------------  --------       ----
 TRIGGER_NAME                    NOT NULL       VARCHAR2(30)
 TRIGGER_TYPE                                   VARCHAR2(16)
 TRIGGERING_EVENT                               VARCHAR2(26)
 TABLE_OWNER                     NOT NULL       VARCHAR2(30)
 TABLE_NAME                      NOT NULL       VARCHAR2(30)
 REFERENCING_NAMES                              VARCHAR2(87)
 WHEN_CLAUSE                                    VARCHAR2(4000)
 STATUS                                         VARCHAR2(8)
 DESCRIPTION                                    VARCHAR2(4000)
 TRIGGER_BODY                                   LONG
```

ALL_TRIGGERS

```
SQL> desc all_triggers;
Name                                Null?          Type
----------------------------------  --------       ----
OWNER                               NOT NULL       VARCHAR2(30)
TRIGGER_NAME                        NOT NULL       VARCHAR2(30)
TRIGGER_TYPE                                       VARCHAR2(16)
TRIGGERING_EVENT                                   VARCHAR2(26)
TABLE_OWNER                         NOT NULL       VARCHAR2(30)
TABLE_NAME                          NOT NULL       VARCHAR2(30)
REFERENCING_NAMES                                  VARCHAR2(87)
WHEN_CLAUSE                                        VARCHAR2(4000)
STATUS                                             VARCHAR2(8)
DESCRIPTION                                        VARCHAR2(4000)
TRIGGER_BODY                                       LONG
```

DBA_TRIGGERS

```
SQL> descr dba_triggers;
Name                                Null?          Type
----------------------------------  --------       ----
OWNER                               NOT NULL       VARCHAR2(30)
TRIGGER_NAME                        NOT NULL       VARCHAR2(30)
TRIGGER_TYPE                                       VARCHAR2(16)
TRIGGERING_EVENT                                   VARCHAR2(26)
TABLE_OWNER                         NOT NULL       VARCHAR2(30)
TABLE_NAME                          NOT NULL       VARCHAR2(30)
REFERENCING_NAMES                                  VARCHAR2(87)
WHEN_CLAUSE                                        VARCHAR2(4000)
STATUS                                             VARCHAR2(8)
DESCRIPTION                                        VARCHAR2(4000)
TRIGGER_BODY                                       LONG
```

The new column, BASE_OBJECT_TYPE, specifies whether the trigger is based on DATABASE, SCHEMA, table, or view. The old column, TABLE_NAME, is null if the base object is not table or view.

The column ACTION_TYPE specifies whether the trigger is a call type trigger or a PL/SQL trigger.

The column TRIGGER_TYPE includes two additional values: BEFORE EVENT and AFTER EVENT, which are applicable only to system events.

The column TRIGGERING_EVENT includes all system and DML events.

List the Trigger Definition

To list the trigger definition:

```
Select Trigger_Name,  Trigger_Body  from USER_TRIGGERS⇒
 where Trigger_name=< TRIGGERNAME>;
```

List Trigger Information

To list trigger information:

```
Select Trigger_Name, Trigger_Type, Triggering_Event, Table_Owner,
```

```
Table_Name, Referencing_Names,When_Clause, Status, Description,
Trigger_Body from USER_TRIGGERS where Trigger_name=< TRIGGERNAME>;
```

To Remove a Trigger

To remove a trigger:

```
drop trigger TRIGGERNAME
```

To Modify an Existing Trigger

On Oracle, to explicitly alter a trigger definition, use the CREATE OR REPLACE option. See a full explanation in the Oracle SQL Reference (CREATE TRIGGER).

To Disable a Trigger

By default, triggers are enabled when they're first created. Disable a trigger by using the ALTER TRIGGER statement with the DISABLE option.

For example, to disable the trigger named REORDER of the INVENTORY table, enter the following statement:

```
ALTER TRIGGER Reorder DISABLE;
```

All triggers that are associated with a table can be disabled with one statement by using the ALTER TABLE statement with the DISABLE clause and the ALL TRIGGERS option. For example, to disable all triggers that are defined for the INVENTORY table, enter the following statement:

```
ALTER TABLE Inventory
   DISABLE ALL TRIGGERS;
```

Using Sybase Trigger Information

This section discusses how to:

- Use Sybase trigger syntax.
- Use Sybase trigger maintenance.

Using Sybase Trigger Syntax

This example shows the syntax for creating triggers on Sybase:

```
CREATE TRIGGER PS_ABSENCE_HIST_TR
ON PS_ABSENCE_HIST
FOR INSERT, UPDATE, DELETE AS
BEGIN
  DECLARE @XTYPE CHAR(1), @OPRID CHAR(8)
  IF EXISTS (SELECT 'X' FROM deleted)
    SELECT @XTYPE = 'D'
  IF EXISTS (SELECT 'X' FROM inserted)
  BEGIN
    IF (@XTYPE = 'D')
      SELECT @XTYPE = 'U'
  ELSE
```

```
      SELECT @XTYPE = 'I'
  END
  SELECT @OPRID = substring(clientname, 1, charindex(',', clientname) - 1)
    FROM master..sysprocesses
    WHERE spid = @@spid
  -- Transaction is a Delete and the Delete Part of an Update
  IF (@XTYPE = 'D') OR (@XTYPE = 'U')
  BEGIN
    IF (@XTYPE = 'U')
      SELECT @XTYPE = 'B'
    INSERT INTO PS_AUDIT_ABSENCE
    (AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
    EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED)
    SELECT @OPRID, getdate(), @XTYPE,
    EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED
    FROM deleted
  END
  -- Transaction is a Insert and the Insert Part of an Update
  IF (@XTYPE = 'I') OR (@XTYPE = 'B')
  BEGIN
    IF (@XTYPE = 'B')
      SELECT @XTYPE = 'A'
    INSERT INTO PS_AUDIT_ABSENCE
    (AUDIT_OPRID,AUDIT_STAMP,AUDIT_ACTN,
    EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED)
    SELECT @OPRID,getdate(), @XTYPE,
    EMPLID,ABSENCE_TYPE,BEGIN_DT,RETURN_DT,DURATION_DAYS,
DURATION_HOURS,REASON,PAID_UNPAID,EMPLOYER_APPROVED
    FROM inserted
  END
END
```

Using Sybase Trigger Maintenance

Commands that are useful with the trigger feature include:

List All Triggers in a Database

To list all triggers:

```
SELECT name  FROM sysobjects WHERE type = 'TR'
```

List the Trigger Definition

To list trigger definition

```
sp_helptext TRIGGERNAME
```

List Trigger Information

To list trigger information:

```
sp_help TRIGGERNAME
```

This command reports information about a database object (any object that is listed in the sysobjects table), a user-defined data type, or a data type that Microsoft SQL Server supplies.

To Remove a Trigger

To remove a trigger:

```
drop trigger TRIGGERNAME
```

To Disable a Trigger

To disable a trigger:

```
ALTER TABLE table     | (ENABLE | DISABLE) TRIGGER ( trigger_name)
```

(ENABLE | DISABLE) TRIGGER specifies that trigger_name is enabled or disabled. When a trigger is disabled, it is still defined for the table; however, when INSERT, UPDATE or DELETE statements are executed against the table, the actions that are in the trigger are not performed until the trigger is enabled.

- ALL. Specifies that all triggers in the table are enabled or disabled.
- trigger_name. Specifies the name of the trigger to disable or enable.

CHAPTER 6

Running Diagnostics with Diagnostic Framework

This chapter provides an overview of Diagnostic Framework and discusses how to:

- Set up security for Diagnostic Framework.
- Run diagnostics.
- Import post-release plug-ins.

Understanding Diagnostic Framework

This section discusses:

- The purpose and benefits of Diagnostic Framework.
- The architecture of Diagnostic Framework.

The Purpose and Benefits of Diagnostic Framework

PeopleSoft provides a framework for defining and retrieving application data diagnostics within the PeopleSoft Internet Architecture (PIA) environment. Diagnostic Framework retrieves diagnostic information from a PeopleSoft database. With this diagnostic information, you can:

- Discover problematic application-related data.
- Explore setup details.
- Present information to PeopleSoft support in a common format.

Using Diagnostic Framework, you can perform diagnostic tests on your system with minimal instructions from the PeopleSoft Global Support Center. These tests answer application-specific questions to help development and user support teams diagnose and troubleshoot any problems that you may be experiencing.

The tests can request additional parameters to tailor the diagnostics to your situation. They output HTML pages that you can open using any PeopleSoft-supported browser, and XML documents containing the same information in a form suitable for programmatic processing. You can email the HTML or XML documents to an application expert.

Diagnostic Framework is not designed to be a reporting tool, such as Query or Crystal Reports. Diagnostic Framework should not be used to return large amounts of data. Use it only to get small sets of diagnostic data, for example 100 rows of data or fewer.

The Architecture of Diagnostic Framework

Diagnostic Framework includes:

- Delivered base classes in application packages.
- Delivered application diagnostic plug-ins developed from the base classes and application packages.
- The capability to extend delivered base classes to develop additional diagnostic plug-ins and to register the new plug-ins.
- A common user interface for all diagnostic plug-ins.

Diagnostic Framework is installed automatically when you install PeopleTools. Use standard PeopleSoft security administration to grant access to the user interface.

See Chapter 6, "Running Diagnostics with Diagnostic Framework," Setting Up Security for Diagnostic Framework, page 143.

Application Classes and Packages

By definition, each application class is responsible for asking one diagnostic question. Each class has a method that is called by Diagnostic Framework. This method, in turn, gathers the information and calls methods in extended base classes that return the information to Diagnostic Framework. An application package is a container for application classes or other application packages.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference*, "Application Classes".

Diagnostic Plug-ins

Application packages that are used in Diagnostic Framework are referred to as diagnostic application packages. They are a collection of application classes and methods encapsulated within an application package. The metadata that defines a diagnostic application package is referred to as a diagnostic plug-in. In this documentation, we refer to diagnostic application packages as *diagnostic plug-ins* or simply as *plug-ins*.

Diagnostic plug-ins probe the application for diagnostic information. When you perform diagnostic tests on your PeopleSoft system, Diagnostic Framework executes programs within these plug-ins and then returns the information from those programs in an HTML or XML document. Because of Diagnostic Framework, these plug-ins supply a consistent method of gathering relevant diagnostic information from your system.

There are three categories of diagnostic plug-ins:

- Delivered diagnostic plug-ins that are automatically installed when you install PeopleTools and PeopleSoft applications.

  The available diagnostic plug-ins depend on which applications you have installed. Appropriate plug-ins are automatically available after your application installation is complete.
- Post-release diagnostic plug-ins that PeopleSoft Global Support Center might send to you for specific diagnostic purposes.

  You import these plug-ins to Diagnostic Framework using PeopleSoft Application Designer.
- Custom diagnostic plug-ins that you develop.

  You must register custom plug-ins before you can use them.

All registered plug-ins appear in Diagnostic Framework user interface. Delivered plug-ins are grouped according to installed PeopleSoft applications and functional areas within the applications. From here, you can select which plug-ins to run. A user support person may ask you to run a particular plug-in, depending on the problem that you are reporting.

See Also

Chapter 7, "Diagnostic PeopleCode," page 151

Setting Up Security for Diagnostic Framework

This section provides an overview of security for Diagnostic Framework and discusses how to:

- Set up security access to pages.
- Set up security access to web libraries.

Understanding Security for Diagnostic Framework

Before you can gather information using Diagnostic Framework, you must have access to the WEBLIB_PTDIAG web library and to Diagnostic Framework components.

Once you have been granted access to Diagnostic Framework, you can access the pages through the portal.

Security for Diagnostic Framework is handled through regular PeopleTools security (roles and permission lists).

Pages Used to Set Up Security for Diagnostic Framework

Page Name	Object Name	Navigation	Usage
Pages	ACL_GENERAL	PeopleTools, Security, Permission & Roles, Permission Lists, Pages	Set up security access to pages.
Component Permissions	ACL_COMPONENT2	Click Edit Components on the Pages page.	Set up security access to pages.
Page Permissions	ACL_PAGES2	Click Edit Pages on the Component Permissions page.	Set up security access to pages.
Web Libraries	ACL_WEBLIBS	PeopleTools, Security, Permission & Roles, Permission Lists, Web Libraries	Set up security access to web libraries.
Web Lib Permissions	ACL_WEBLIB_ACCESS	Click Edit on the Web Libraries page.	Set up security access to web libraries.

See Also

Chapter 4, “Ensuring Data Integrity,” page 63

Setting Up Security Access to Pages

To set up security access for Diagnostic Framework pages:

1. Access the Pages page for the relevant permission list.
2. Click Edit Components for the menu PT_DIAGNOSTICS.
3. For each component displayed on the Component Permissions page, click Edit Pages.
4. For each page, select the desired security permission settings on the Page Permissions page.
5. Click OK until you return to the Pages page and click Save.

Setting Up Security Access to Web Libraries

To set up security access for Diagnostic Framework web library:

1. Access the Web Libraries page for the relevant permission list.
2. Click Edit for the web library named WEBLIB_PTDIAG.
3. Select the desired access permission settings on the Web Lib Permissions page.
4. Click OK until you return to the Web Libraries page and click Save.

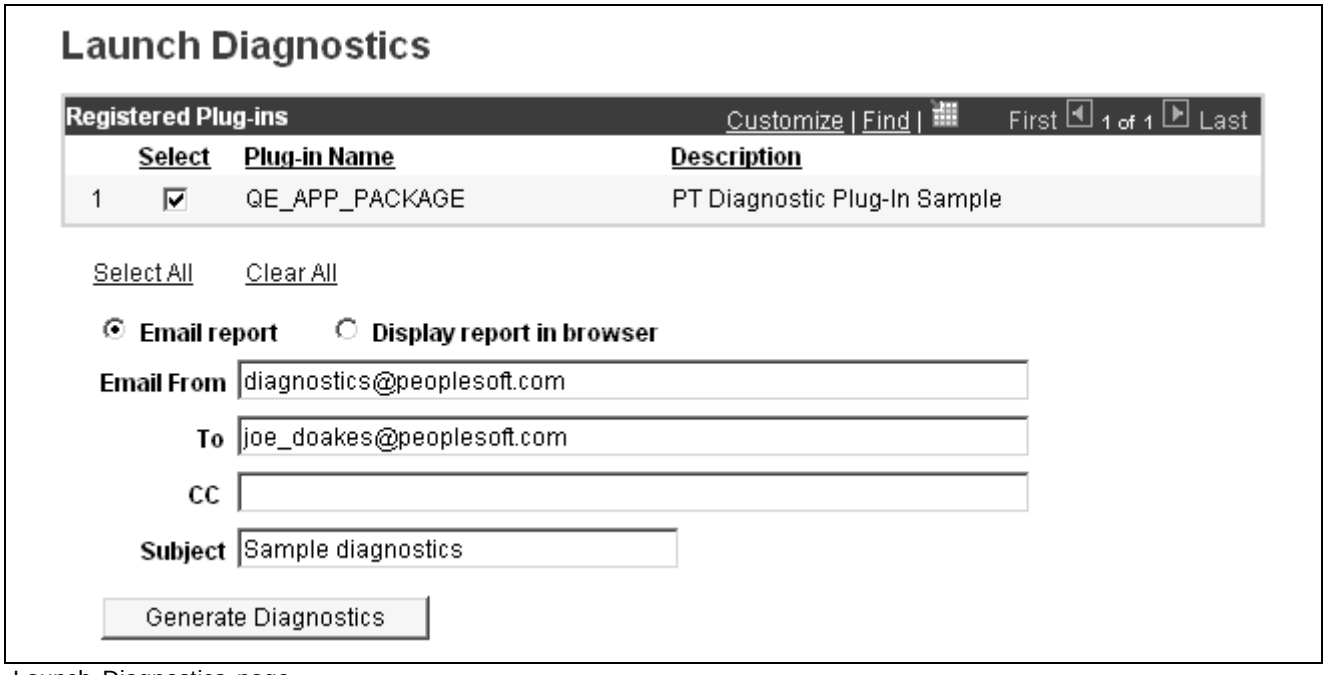
Running Diagnostics

This section discusses how to:

- Launch diagnostic plug-ins.
- Provide additional information.
- Obtain diagnostic results.

Launching Diagnostic Plug-Ins

Select Application Diagnostics, Launch Diagnostics to access the Launch Diagnostics page.



Launch Diagnostics page

This page displays a list of available diagnostic plug-ins. Only registered plug-ins appear.

Plug-In Name Displays the name of the application package that defines each diagnostic plug-in.

Select Select this check box for each diagnostic plug-in package that you want to run.

Click the Select All link to include all of the listed plug-ins, or the Clear All link to exclude all of the listed plug-ins from the diagnostics

Note. You must select at least one diagnostic plug-in.

Display report in browser Select to display the generated HTML diagnostic report in a new browser window.

Email report Select to generate an email containing HTML and XML copies of the generated diagnostic report. The following standard email fields appear:

- Email From
- To
- CC (optional)
- Subject (optional)

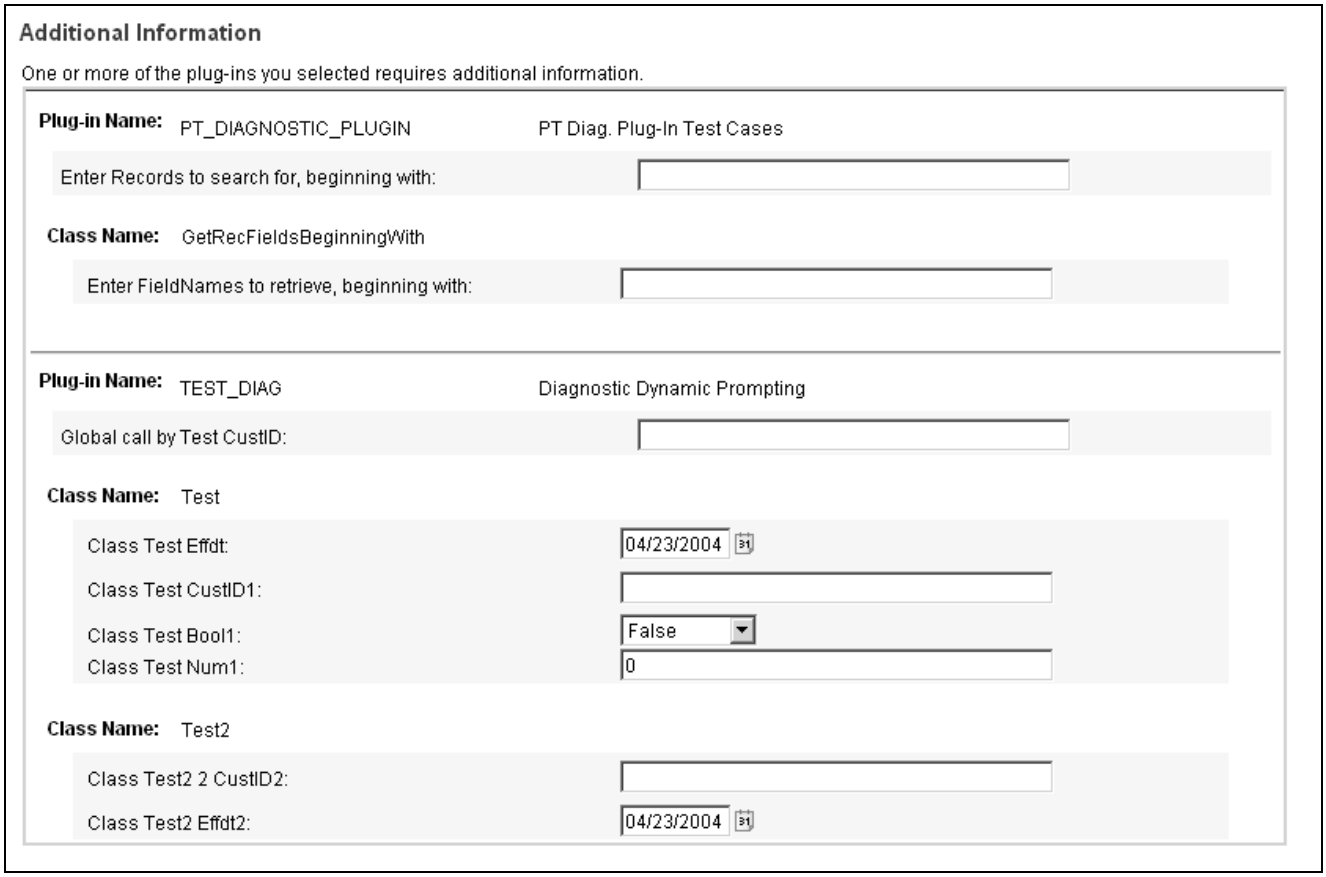
Note. Before you can use this option, you must configure the application server domain to handle SMTP email.

See *Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration*, "Setting Application Server Domain Parameters," SMTP Settings.

Generate Diagnostics Click to launch the selected diagnostics, and either display or email the resulting report.

Providing Additional Information

When you click the Generate Diagnostics button on the Launch Diagnostics page, one or more of the diagnostic plug-ins you selected might have been designed to dynamically prompt you for relevant parameters. The Additional Information page appears to enable you to enter the required parameters.



Additional Information page

The fields that appear on this page depend on the diagnostic plug-ins that you specified on the Launch Diagnostics page. The Additional Information page includes a section for each plug-in that requires information. Each section can contain fields that are specific to individual classes, or fields that apply globally for the plug-in. For the diagnostic plug-ins delivered with your PeopleSoft application, your application documentation explains what values are required for each field.

Obtaining Diagnostic Results

When all of the diagnostic results have been gathered, they're disseminated based on the option you selected on the Launch Diagnostics page.

If you selected Display report in browser, the resulting PeopleSoft Diagnostics page appears in HTML format in a new browser window. Following is an example of the PeopleSoft Diagnostics page in HTML format.

PeopleSoft Diagnostics

Database Name: **QE845DVL**
User ID: **QEDMO**
Date Created: **2004-01-30-16.04.52.000000**
Database Type: **MICROSFT**

Plug-in Name: PT_DIAGNOSTIC_PLUGIN
Description: PT Diag. Plug-In Test Cases

Purpose: This is a diagnostic to determine all of the languages installed in your PeopleSoft Database. This diagnostic tests AddRowset functionality.

The following rows were retrieved:

	*LANGUAGE_CD	CHARSET	INSTALLED	VERITY_LOCALE	SCLANG	WINDOWS_CHARSET	VERITY_CHARSET	ISO_LOCALE
1	CFR	ISO_8859-1	0	frenchx	SC16	CP1252	CP1252	fr-ca
2	DAN	ISO_8859-1	0	danishx	SC09	CP1252	CP1252	da
3	DUT	ISO_8859-1	1	dutchx	SC11	CP1252	CP1252	nl
4	ENG	ISO_8859-1	1	englishx	SC00	CP1252	CP1252	en
5	ESP	ISO_8859-1	0	spanishx	SC34	CP1252	CP1252	es
6	FRA	ISO_8859-1	0	frenchx	SC16	CP1252	CP1252	fr
7	GER	ISO_8859-1	0	germanx	SC18	CP1252	CP1252	de
8	GRK	ISO_8859-7	0	englishx	SC21	CP1253	CP1253	el
9	ITA	ISO_8859-1	0	italianx	SC25	CP1252	CP1252	it
10	JPN	Shift_JIS	0	japanb	SC00	CP932	Shift_JIS	ja
11	KOR	CP949	0	koreab	SC00	CP949	CP949	ko
12	MAY	ISO_8859-1	0	englishx	SC00	CP1252	CP1252	ms
13	POL	ISO_8859-2	0	polish	SC28	CP1250	CP1250	pl
14	POR	ISO_8859-1	0	portugx	SC31	CP1252	CP1252	pt
15	SVE	ISO_8859-1	0	swedishx	SC35	CP1252	CP1252	sv
16	THA	ISO_8859-11	0	uni	SC00	CP874	UTF8	th
17	ZHS	GB2312	0	simpcb	SC00	CP936	GB2312	zh-cn
18	ZHT	Big5	0	tradcb	SC00	CP950	Big5	zh-tw

Plug-in Name: PT_DIAGNOSTIC_PLUGIN
Description: PT Diag. Plug-In Test Cases

Purpose: This is a diagnostic to print out a listing of fields from records in your PeopleSoft database that matches search criteria. This diagnostic tests globalType and classType prompting. The global prompt is retrieved from inputs defined by a different class in this plug-in.

Additional Information
Enter Records to search for, beginning with: MAINT
Enter FieldNames to retrieve, beginning with: REL

The following values were retrieved:
Record: MAINTENANCE_LOG has the following field that matches your criteria: RELEASEDTTM
Record: MAINTENANCE_LOG has the following field that matches your criteria: RELEASELABEL
Record: MAINTLOGREL_VW has the following field that matches your criteria: RELEASEDTTM
Record: MAINTLOGREL_VW has the following field that matches your criteria: RELEASELABEL
Record: MAINTLOGSRCH_VW has the following field that matches your criteria: RELEASEDTTM
Record: MAINTLOGSRCH_VW has the following field that matches your criteria: RELEASELABEL

PeopleSoft Diagnostics page in HTML format

Rowset information is presented on the page in tabular form, and non-rowset information is presented in list form. You can use your browser's Save As functionality to save the page to your local machine.

If you selected Email report, the resulting PeopleSoft Diagnostics page is emailed as HTML and XML attachments to the address you specified. Following is an example of the XML that comprises a PeopleSoft Diagnostics page.

```
<?xml version="1.0"?>
<PeopleSoftDiagnostics>
  <UserInformation>
 <Database_Name>QE845DVL</Database_Name>
 <User_ID>QEDMO</User_ID>
 <Date_Created>2004-01-30-16.04.54.000000</Date_Created>
 <Database_Type>MICROSFT</Database_Type>
  </UserInformation>
  <ApplicationDiagnostics>
    <PT_DIAGNOSTIC_PLUGIN>
      <GetLanguages>
        <Purpose>This is a diagnostic to determine all⇒
 of the languages installed in your PeopleSoft Database.⇒
 This diagnostic tests AddRowset functionality.</Purpose>
        <Result>
          <LANGUAGE_CD>CFR</LANGUAGE_CD>
          <CHARSET>ISO_8859-1</CHARSET>
          <INSTALLED>0</INSTALLED>
          <VERITY_LOCALE>frenchx</VERITY_LOCALE>
          <SCLANG>SC16</SCLANG>
          <WINDOWS_CHARSET>CP1252</WINDOWS_CHARSET>
          <VERITY_CHARSET>CP1252</VERITY_CHARSET>
          <ISO_LOCALE>fr-ca</ISO_LOCALE>
        </Result>
        <Result>
          <LANGUAGE_CD>DAN</LANGUAGE_CD>
          <CHARSET>ISO_8859-1</CHARSET>
          <INSTALLED>0</INSTALLED>
          <VERITY_LOCALE>danishx</VERITY_LOCALE>
          <SCLANG>SC09</SCLANG>
          <WINDOWS_CHARSET>CP1252</WINDOWS_CHARSET>
          <VERITY_CHARSET>CP1252</VERITY_CHARSET>
          <ISO_LOCALE>da</ISO_LOCALE>
        </Result>
        <Result>
          <LANGUAGE_CD>ENG</LANGUAGE_CD>
          <CHARSET>ISO_8859-1</CHARSET>
          <INSTALLED>1</INSTALLED>
          <VERITY_LOCALE>englishx</VERITY_LOCALE>
          <SCLANG>SC00</SCLANG>
          <WINDOWS_CHARSET>CP1252</WINDOWS_CHARSET>
          <VERITY_CHARSET>CP1252</VERITY_CHARSET>
          <ISO_LOCALE>en</ISO_LOCALE>
        </Result>
      </GetLanguages>
      <GetRecFieldsBeginningWith>
```

```
       <Purpose>This is a diagnostic to print out a listing⇒
of fields from records in your PeopleSoft database that⇒
matches search criteria.  This diagnostic tests globalType⇒
and classType prompting.  The global prompt is retrieved⇒
from inputs defined by a different class in this plug-in.</Purpose>
       <AdditionalInformation>
         <Question>Enter Records to search for, beginning with:</Question>
         <Answer>MAINT</Answer>
       </AdditionalInformation>
       <AdditionalInformation>
         <Question>Enter FieldNames to retrieve, beginning with:</Question>
         <Answer>REL</Answer>
       </AdditionalInformation>
       <Result>
         <Descr>Record: MAINTENANCE_LOG has the following⇒
field that matches your criteria: </Descr>
         <Type>String</Type>
         <Answer>RELEASEDTTM</Answer>
       </Result>
       <Result>
         <Descr>Record: MAINTENANCE_LOG has the following⇒
field that matches your criteria: </Descr>
         <Type>String</Type>
         <Answer>RELEASELABEL</Answer>
       </Result>
       <Result>
         <Descr>Record: MAINTLOGREL_VW has the following⇒
field that matches your criteria: </Descr>
         <Type>String</Type>
         <Answer>RELEASEDTTM</Answer>
       </Result>
       <Result>
         <Descr>Record: MAINTLOGREL_VW has the following⇒
field that matches your criteria: </Descr>
         <Type>String</Type>
         <Answer>RELEASELABEL</Answer>
       </Result>
     </GetRecFieldsBeginningWith>
   </PT_DIAGNOSTIC_PLUGIN>
 </ApplicationDiagnostics>
</PeopleSoftDiagnostics>
```

Importing Post-Release Plug-Ins

If information that you generate from the delivered plug-ins is not sufficient to diagnose your problem, the PeopleSoft Global Support Center (GSC) might give you additional plug-ins. You import these plug-ins into your database using the Copy Project from File command in PeopleSoft Application Designer.

A plug-in project is an upgrade project, and it must contain the following definitions:

- Application Packages.
- Diagnostic Plug-Ins.
- Application Package PeopleCode.

If you need to send a file to GSC or move a file between databases, use the Copy Project to File command in PeopleSoft Application Designer.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Application Designer*, "Upgrading with PeopleSoft Application Designer".

Plug-ins that you import are registered automatically; they become immediately available on the Launch Diagnostics page.

CHAPTER 7

Diagnostic PeopleCode

This chapter provides an overview of diagnostic PeopleCode and discusses how to:

- Develop diagnostic plug-ins.
- Make diagnostic plug-ins available.
- Use the PTDiagnostics application class.
- PTDiagnostics Class Reference.

Understanding Diagnostic PeopleCode

PeopleSoft delivers Diagnostic Framework base classes in an application package that can be extended to create your own diagnostic plug-in. Methods in these base classes can be called by application classes in the extended application packages, to return diagnostic information back into Diagnostic Framework.

A diagnostic plug-in can be comprised of a single application class or multiple application classes within the extended application package. Each of the application classes within the plug-in must be extended from the PTDiagnostics base application class.

How Diagnostic Plug-Ins Are Used

You define diagnostic plug-ins using application classes, but you don't use them in the same way that other PeopleCode application classes are used:

- Diagnostic plug-in classes are instantiated only by Diagnostic Framework, and can't be called from any other location, including PeopleCode programs.
- Diagnostic plug-in classes must contain certain methods that are recognized and used by Diagnostic Framework.

Developing Diagnostic Plug-Ins

This section provides an overview of plug-in structure and discusses how to:

- Import the PTDiagnostics class.
- Define the IsPlugIn method.
- Define the GetDiagnosticInfo method.
- Define the GetDynamicPrompt method.

Understanding Plug-In Structure

You use the Application Packages editor in PeopleSoft Application Designer to create application packages that are treated as diagnostic plug-ins by the framework.

Each of the application classes within the diagnostic plug-in asks one diagnostic question, and can return multiple answers to that question.

Each application class you define within a diagnostic plug-in must:

- Inherit from the delivered PT_DIAGNOSTICS application package by using the following in the import section:

```
import PT_DIAGNOSTICS:*;
```

- Contain a constructor method that includes the following within the body of the constructor:

```
%Super = create PTDiagnostics();
```

- Contain a public method called IsPlugIn to identify the class as part of a diagnostic plug-in.
- Contain a public method called GetDiagnosticInfo to run the diagnostic.

The application class can also contain an optional public method called GetDynamicPrompt to prompt users for additional information.

Note. You can also define your own private methods within the application class, which you can call only within the class.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, "Application Classes"

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode Developer's Guide, "Creating Application Packages and Classes"

Importing the PTDiagnostics Class

The PTDiagnostics Class is not a built-in class, like Rowset, Field, Record, and so on. It's an application class. Before you can use this class in your PeopleCode program, you must import it to your program.

An import statement names the application class. The application package PT_DIAGNOSTICS contains the PTDiagnostics class.

The import statement you should use is as follows:

```
import PT_DIAGNOSTICS:*;
```

Using the asterisk after the package name makes all the application classes directly contained in the named package available. Application classes contained in subpackages of the named package are not made available.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, "Application Classes"

Defining the IsPlugIn Method

This required public method is invoked by Diagnostic Framework when you register the plug-in, to verify that the class is part of a diagnostic plug-in. You don't need to define the method beyond the following:

```
method IsPlugIn
end-method
```

Defining the GetDiagnosticInfo Method

Use this required public method to define the code that retrieves diagnostic information and returns it to Diagnostic Framework for presentation to the user. This method is invoked by Diagnostic Framework to initiate information collection, then output the results.

The GetDiagnosticInfo method uses the base class InsertData method to pass the results of the diagnostic to Diagnostic Framework for presentation to the users, for example:

```
&status = %Super.InsertData("Number", "Number of Records: ", &rs1.RowCount);
```

InsertData can pass output data using the following data types:

- String
- Number
- Date
- Boolean
- Rowset

Before you can pass rowset data as output, you must first use the base class SetProperty method to set the base class hasRowset property to True. Following is an example of GetDiagnosticInfo that passes rowset data as ouput:

```
method GetDiagnosticInfo
   Local boolean &status;
   Local number &rc1;
   Local Rowset &rs1;
   Local string &sError;

   &rs1 = CreateRowset(Record.PSLANGUAGES);
   &rc1 = &rs1.Fill();
   &status = %Super.SetProperty(%This, "hasRowset", "Boolean", True);
   &status = %Super.InsertData("Rowset", "LANGUAGES description,⇒
 not used in output", &rs1);

end-method;
```

Note. You can pass only one data type in each diagnostic plug-in application class. To return multiple data types, define multiple application classes. Results that are passed to the framework are retained in memory.

Other Considerations for GetDiagnosticInfo

If you're also defining the GetDynamicPrompt method to prompt users for additional information, use the base class GetUserInputByKey method to retrieve the user responses, for example:

```
&status = %Super.GetUserInputByKey("Recs", &sVal);
```

For more readable output, use the base class SetProperty method to insert a description into the base class Purpose property, for example:

```
&status = %Super.SetProperty(%This, "Purpose", "String",⇒
 "This is a diagnostic to determine your license code.");
```

Note. You can also set the Purpose property in the constructor or in the GetDynamicPrompt method.

See Also

Chapter 7, “Diagnostic PeopleCode,” GetUserInputByKey, page 156

Chapter 7, “Diagnostic PeopleCode,” InsertData, page 156

Chapter 7, “Diagnostic PeopleCode,” SetProperty, page 158

Defining the GetDynamicPrompt Method

If you want a diagnostic application class to prompt users for additional information that you can use as dynamic criteria for the diagnostic, you must define a public method called GetDynamicPrompt within the class.

Before you can use the GetDynamicPrompt method, you must first use the base class SetProperty method within the constructor to set the base class Where property to True , for example:

```
&status = %Super.SetProperty(%This, "Where", "Boolean", True);
```

Note. If the Where property is False, Diagnostic Framework ignores the GetDynamicPrompt method.

Within the GetDynamicPrompt method, use the base class InsertQuestion method to define the questions used to prompt the users. Following is an example of GetDynamicPrompt that prompts users for record names:

```
method GetDynamicPrompt
   Local boolean &status;

   &status = %Super.InsertQuestion("Recs",⇒
 "Enter Records to search for, beginning with: ", "String", True);
end-method;
```

See Also

Chapter 7, “Diagnostic PeopleCode,” InsertQuestion, page 157

Chapter 7, “Diagnostic PeopleCode,” SetProperty, page 158

Making Diagnostic Plug-Ins Available

This section discusses how to:

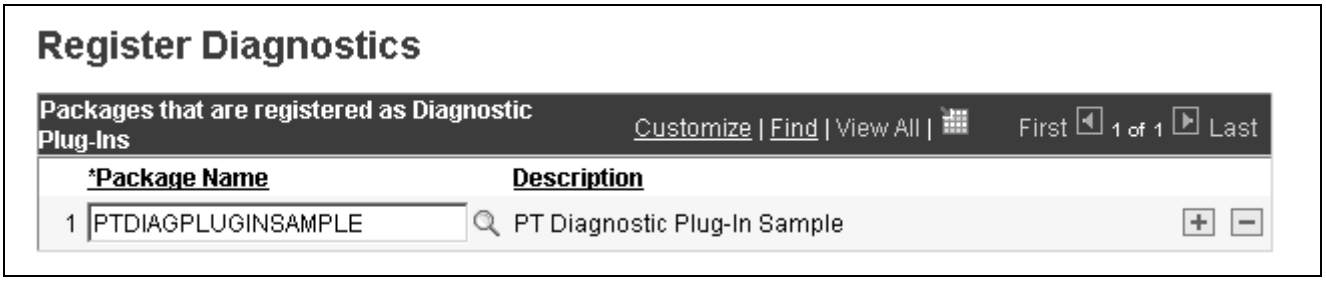
- Register diagnostic plug-ins.
- Insert plug-ins into projects.

Page Used to Register Diagnostic Plug-ins

Page Name	Object Name	Navigation	Usage
Register Diagnostics	PT_DIAG_FRAME_REG	Application Diagnostics, Register Diagnostics	Register new diagnostic plug-ins.

Registering Diagnostic Plug-Ins

Access the Register Diagnostics page.



Register Diagnostics page

After you create a new diagnostic package in PeopleSoft Application Designer, define the plug-in by registering the application package on the Register Diagnostics page.

Package Name Add a row and select the new plug-in from the drop down list.

Note. Available values include application packages that are not for diagnostic purposes. If you select an application package that is not specifically written as a diagnostic plug-in, the system returns an error message.

Inserting Plug-Ins Into Projects

Once the diagnostic plug-ins have been inserted into a project, they can be copied or compared between databases and can be exported to and imported from files, using PeopleSoft Application Designer. This step would be necessary for plug-ins that you might want to send to the PeopleSoft Global Support Center for their review.

To insert a plug-in into a project:

1. In Application Designer, open the Insert into Project dialog box.
2. Select *Diagnostic Plug-Ins* as the definition type.
3. Select the plug-in and click Insert.
4. After inserting the plug-in, make sure to include the underlying application packages and application package PeopleCode in the project as well.

See Also

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Application Designer, "Working With Projects"

PTDiagnostics Application Class

This PeopleCode application class is part of the PT_DIAGNOSTICS application package. It establishes the basic framework for developing your diagnostic plug-ins. To define your plug-in, you develop a new application package containing one or more application classes that extend the PTDiagnostics application class.

The PTDiagnostics application class contains methods and properties that you can extend to develop your diagnostic plug-ins.

PTDiagnostics Class Methods

This section discusses the diagnostic methods for the PTDiagnostics PeopleCode class. The methods are listed in alphabetical order.

GetUserInputByKey

Syntax

GetUserInputByKey(*sKeyID*, *&data*)

Description

The GetUserInputByKey method retrieves the user response to a question, which can then be used as an input parameter in the diagnostic. You invoke this method within the GetDiagnosticInfo method.

Parameters

Parameter	Description
sKeyID	Specify as a string the key that identifies the question for which you're retrieving the user response.
&data	Provide a variable to contain the retrieved user response.

Returns

A Boolean value: True if the user response was retrieved successfully, False otherwise.

See Also

Chapter 7, "Diagnostic PeopleCode," Defining the GetDiagnosticInfo Method, page 153

InsertData

Syntax

InsertData(*propFormat*, *propDescr*, *&data*)

Description

The InsertData method passes data to Diagnostic Framework to be presented as the output of the diagnostic. This enables you to pass any information you want without having to hardcode base class methods in the plug-in. You invoke this method within the GetDiagnosticInfo method.

Parameters

Parameter	Description
propFormat	Specify as a string the data type of the data to be presented. Select from the following: • String • Number • Date • Boolean • Rowset
propDescr	Specify a string of text to describe or introduce the output data.
&data	Provide the output data value, in a variable of the data type specified by the *propFormat* parameter.

Returns

A Boolean value: True if the data has been inserted into Diagnostic Framework, False if the data can't be inserted into the framework, or if the data type specified by *propFormat* doesn't exist in the current framework.

See Also

Chapter 7, "Diagnostic PeopleCode," Defining the GetDiagnosticInfo Method, page 153

InsertQuestion

Syntax

```
InsertQuestion(sKeyID, sQuestion, sType, GblBool)
```

Description

The InsertQuestion method passes a question to Diagnostic Framework, which then presents it to the user to obtain an input parameter. You invoke this method within the GetDynamicPrompt method.

Parameters

Parameter	Description
sKeyID	Specify as a string a key to identify the question. This value must be unique across all plug-ins that are made available to a user.
sQuestion	Specify as a string the question you want the user to answer.
sType	Specify as a string the data type of the response required from the user. Select from the following: • String • Number • Date • Boolean
GblBool	Specify a Boolean value indicating the scope of the question: • True: The question applies globally to the plug-in. • False: The question applies only to the current class. Global questions are asked once per plug-in on the Additional Information prompt page. For example, a plug-in could be defined to gather employee information. The plug-in might contain many application classes that gather specific information (for example, one application class for getting employee paycheck information, and one application class for getting employee addresses). Class level questions are asked only for the current application class. For example, for the paycheck information, you might want to prompt for specific pay periods and for the address information, you might want to prompt for an effective date.

Returns

A Boolean value: True if the method is successful, False otherwise.

See Also

Chapter 7, "Diagnostic PeopleCode," Defining the GetDynamicPrompt Method, page 154

SetProperty

Syntax

```
SetProperty(&obj, propName, propFormat, &propValue)
```

Description

The SetProperty method sets a property of an instantiated PTDiagnostics object to the value that you specify.

Parameters

Parameter	Description
&obj	Specify the PTDiagnostics object for which you want to set a property. Typically, you'll specify %This.
propName	Specify as a string the name of the property that you want to set. The values are: • hasRowset • Purpose • Where
propFormat	Specify as a string the data type of the property that you want to set. For hasRowset and Where, specify Boolean. For Purpose, specify string.
&propValue	Provide the property value, in a variable that has the data type specified by the *propFormat* parameter.

Returns

A Boolean value: True if the property specified by *propName* exists and can be set in the base class, False if the property can't be set (for example, if the current plug-in is used in a previous release of Diagnostic Framework where that property isn't defined).

PTDiagnostics Class Properties

This section lists the properties for the PTDiagnostics PeopleCode class. The properties are listed in alphabetical order.

hasRowset

Description

Use the hasRowset property to indicate whether the InsertData method passes output data to Diagnostic Framework as a rowset. This property only needs to be defined for classes that use rowsets. This property takes a Boolean value:

- True: InsertData will pass data in rowset format.
- False: InsertData will pass data in string, number, date, or Boolean format. This is the default value.

Note. You must use the SetProperty method to set the value of this property.

See Also

Chapter 7, "Diagnostic PeopleCode," InsertData, page 156
Chapter 7, "Diagnostic PeopleCode," SetProperty, page 158

Purpose

Description

Use the Purpose property to specify as a string the text that introduces and describes the purpose of the diagnostic that this application class performs. This text will be displayed as part of the diagnostic output.

The default value of this property is **Unknown**.

Note. You must use the SetProperty method to set the value of this property.

See Also

Chapter 7, "Diagnostic PeopleCode," SetProperty, page 158

Where

Description

Use the Where property to indicate whether this application class should dynamically prompt the user for relevant parameters. This property only needs to be defined for classes that prompt the user.

This property takes a Boolean value:

- True: The application class should dynamically prompt the user.
- False: The application class should not dynamically prompt the user. This is the default value.

Note. You must use the SetProperty method to set the value of this property, and you must set it from within the constructor method. If you set this property to True, you must define the GetDynamicPrompt method in your application class to prompt the user.

See Also

Chapter 7, "Diagnostic PeopleCode," SetProperty, page 158
Chapter 7, "Diagnostic PeopleCode," Defining the GetDynamicPrompt Method, page 154

PTDiagnostics Class Examples

The following are examples of typical actions you might perform using the notification classes.

Determining Installed Languages

The following example demonstrates how to return the list of languages in the database.

```
import PT_DIAGNOSTICS:*;

class GetLanguages extends PTDiagnostics
   /* Constructor */

   method GetLanguages();

   /* Public Method */
```

```
   method GetDiagnosticInfo();
   method IsPlugIn();

private

end-class;

method GetLanguages;
   Local boolean &status;
   %Super = create PTDiagnostics();
   &status = %Super.SetProperty(%This, "hasRowset", "Boolean", True);
   &status = %Super.SetProperty(%This, "Purpose", "String",⇒
 "This is a diagnostic to determine all of the languages⇒
 installed in your PeopleSoft Database.");
end-method;

method GetDiagnosticInfo
   Local boolean &stat;
   Local number &rc1;
   Local Rowset &rs1;
   Local string &sError;

   &rs1 = CreateRowset(Record.PSLANGUAGES);
   &rc1 = &rs1.Fill();
   &stat = %Super.InsertData("Rowset", "LANGUAGES description,⇒
 not used in output", &rs1);

end-method;

method IsPlugIn
end-method;
```

Determining the User's License Code

The following example demonstrates how to return a user's license code:

```
import PT_DIAGNOSTICS:*;

class GetLicenseCode extends PTDiagnostics
   /* Constructor */

   method GetLicenseCode();

   /* Public Method */
   method GetDiagnosticInfo();
   method IsPlugIn();

private

end-class;
```

```
method GetLicenseCode;
   Local boolean &status;
   Local string &sError;
   %Super = create PTDiagnostics();
   &status = %Super.SetProperty(%This, "Purpose", "String",⇒
 "This is a diagnostic to determine your license code");
end-method;

method GetDiagnosticInfo
   Local string &sLicenseCode, &sLicenseGroup;
   Local boolean &status;
   Local string &sError;

   SQLExec("SELECT LICENSE_CODE, LICENSE_GROUP FROM PSOPTIONS",⇒
 &sLicenseCode, &sLicenseGroup);
   &status = %Super.InsertData("String",⇒
 "Your License Code is: ", Upper(&sLicenseCode));

end-method;

method IsPlugIn
end-method;
```

Determining Record Counts

The following example demonstrates how to return row counts:

```
import PT_DIAGNOSTICS:*;

class GetPSRECDEFNCount extends PTDiagnostics
   /* Constructor */

   method GetPSRECDEFNCount();

   /* Public Method */
   method GetDiagnosticInfo();
   method IsPlugIn();

private

end-class;

method GetPSRECDEFNCount;
   Local boolean &status;
   Local string &sError;
   %Super = create PTDiagnostics();
   &status = %Super.SetProperty(%This, "Purpose", "String",⇒
 "This is a diagnostic to count the number of records, views,⇒
 derived work records, and sub-records in your PeopleSoft Database.");
```

```
end-method;

method GetDiagnosticInfo
   Local boolean &status;
   Local number &rc1;
   Local Rowset &rs1;
   Local string &sError;

   &rs1 = CreateRowset(Record.PSRECDEFN);
   &rc1 = &rs1.Fill("where RECTYPE = 0");
   &status = %Super.InsertData("Number",⇒
 "Number of Records: ", &rs1.RowCount);

   &rs1 = CreateRowset(Record.PSRECDEFN);
   &rc1 = &rs1.Fill("where RECTYPE = 1");
   &status = %Super.InsertData("Number",⇒
 "Number of Views: ", &rs1.RowCount);

   &rs1 = CreateRowset(Record.PSRECDEFN);
   &rc1 = &rs1.Fill("where RECTYPE = 2");
   &status = %Super.InsertData("Number",⇒
 "Number of Derived/Work Records: ", &rs1.RowCount);

   &rs1 = CreateRowset(Record.PSRECDEFN);
   &rc1 = &rs1.Fill("where RECTYPE = 3");
   &status = %Super.InsertData("Number",⇒
 "Number of sub-records: ", &rs1.RowCount);

end-method;

method IsPlugIn
end-method;
```

Prompting for Global Information

The following example demonstrates the use of global prompting.

```
import PT_DIAGNOSTICS:*;

class GetRecordsBeginningWith extends PTDiagnostics
   /* Constructor */
   method GetRecordsBeginningWith();

   /* Public Method */
   method GetDiagnosticInfo();
   method GetDynamicPrompt();
   method IsPlugIn();

private
```

```
end-class;

method GetRecordsBeginningWith;
   Local boolean &status = False;
   %Super = create PTDiagnostics();
   &status = %Super.SetProperty(%This, "Where", "Boolean", True);
   &status = %Super.SetProperty(%This, "Purpose", "String",⇒
 "This is a diagnostic to print out a listing of records in your⇒
 PeopleSoft database that matches the following search criteria.⇒
 This diagnostic tests globalType prompting.");
end-method;

method GetDynamicPrompt
   Local boolean &status;

   &status = %Super.InsertQuestion("Recs",⇒
 "Enter Records to search for, beginning with: ", "String", True);
end-method;

method GetDiagnosticInfo
   Local boolean &status = False;
   Local string &sVal, &sError;
   Local number &iCount = 0;
   Local Record &REC;
   Local SQL &SQL1;

   &REC = CreateRecord(Record.PSRECDEFN);
   &SQL1 = CreateSQL("%SelectAll(:1) where RECNAME LIKE :2");

   &status = %Super.GetUserInputByKey("Recs", &sVal);
   &SQL1.Execute(&REC, Upper(&sVal | "%"));
   While &SQL1.Fetch(&REC)
      &iCount = &iCount + 1;
      &status = %Super.InsertData("String", "Record #" | &iCount,⇒
 &REC.RECNAME.Value | " (" | &REC.RECDESCR.Value | ")");
   End-While;

end-method;

method IsPlugIn
end-method;
```

Prompting for Global and Class-Level Information

The following example demonstrates the use of global and class-level prompts.

```
import PT_DIAGNOSTICS:*;
```

```
class GetRecFieldsBeginningWith extends PTDiagnostics
   /* Constructor */
   method GetRecFieldsBeginningWith();

   /* Public Method */
   method GetDiagnosticInfo();
   method GetDynamicPrompt();
   method IsPlugIn();
private

end-class;

method GetRecFieldsBeginningWith;
   Local boolean &status;
   %Super = create PTDiagnostics();
   &status = %Super.SetProperty(%This, "Where", "Boolean", True);
   &status = %Super.SetProperty(%This, "Purpose", "String",⇒
 "This is a diagnostic to print out a listing of fields from records⇒
 in your PeopleSoft database that matches search criteria.⇒
 This diagnostic tests globalType and classType prompting.⇒
 The global prompt is retrieved from inputs defined by⇒
 a different class in this plug-in.");
end-method;

method GetDynamicPrompt
   Local boolean &status;
   Local string &sError;

   /* define prompt for this class */
   &status = %Super.InsertQuestion("Flds", "Enter FieldNames to retrieve,⇒
 beginning with:", "String", False);
end-method;

method GetDiagnosticInfo
   Local boolean &status;
   Local string &sValRecs, &sValFlds, &sError;
   Local number &iCount = 0;
   Local Record &REC;
   Local boolean &bReturn;
   Local SQL &SQL1;

   &REC = CreateRecord(Record.PSRECFIELD);
   &SQL1 = CreateSQL("%SelectAll(:1)⇒
 where RECNAME LIKE :2 and FIELDNAME LIKE :3");

 /* get global prompt */
   &status = %Super.GetUserInputByKey("Recs", &sValRecs);
 /* get class prompt */
   &status = %Super.GetUserInputByKey("Flds", &sValFlds);
   &SQL1.Execute(&REC, Upper(&sValRecs | "%"), Upper(&sValFlds | "%"));
```

```
   While &SQL1.Fetch(&REC)
      &iCount = &iCount + 1;
      &status = %Super.InsertData("String",⇒
 "Record: " | &REC.RECNAME.Value | " has the following field⇒
 that matches your criteria: ", &REC.FIELDNAME.Value);
   End-While;

end-method;

method IsPlugIn
end-method;
```

Handling Constructor Failure

The following example demonstrates error handling when the constructor fails.

```
import PT_DIAGNOSTICS:*;

class TestFailedConstructor extends PTDiagnostics
   /* Constructor */

   method TestFailedConstructor();

   /* Public Method */
   method GetDiagnosticInfo();
   method IsPlugIn();

private
   instance boolean &constructorFailed;
end-class;

method TestFailedConstructor
   Local boolean &status;
   Local string &sError;
   %Super = create PTDiagnostics();

   &status = %Super.SetProperty(%This, "Purpose", "String",⇒
 "This is a diagnostic to show how developers can trap a failure⇒
 in the constructor and print out results to the web page.");

   /* introduce unknown propName of Where1 rather than Where */
   &status = %Super.SetProperty(%This, "Where1", "Boolean", True);

   If Not &status Then
      %This.constructorFailed = True;
   Else
      %This.constructorFailed = False;
   End-If;
```

```
end-method;

method GetDiagnosticInfo
   Local string &sError, &sLicenseCode, &sLicenseGroup;
   Local boolean &status;

   If %This.constructorFailed Then
      &status = %Super.InsertData("String", "Status Failed!",⇒
 "This message will be printed out in the HTML page if something⇒
 fails in the constructor.  This is expected behaviour.");
   Else
      SQLExec("SELECT LICENSE_CODE, LICENSE_GROUP FROM PSOPTIONS",⇒
 &sLicenseCode, &sLicenseGroup);

      &status = %Super.InsertData("String",⇒
 "Your License Code is: ", Upper(&sLicenseCode));

   End-If;
end-method;

method IsPlugIn
end-method;
```

Handling InsertData Method Failure

The following example demonstrates error handling when InsertData fails.

```
import PT_DIAGNOSTICS:*;

class TestFailedGetDiagnosticInfo extends PTDiagnostics
   /* Constructor */

   method TestFailedGetDiagnosticInfo();

   /* Public Method */
   method GetDiagnosticInfo();
   method IsPlugIn();

private

end-class;

method TestFailedGetDiagnosticInfo;
   Local boolean &status;
   %Super = create PTDiagnostics();
   &status = %Super.SetProperty(%This, "Purpose", "String",⇒
 "This is a diagnostic to show how developers can trap a failure in⇒
 GetDiagnosticInfo method and print out results to the web page.");
end-method;
```

```
method GetDiagnosticInfo
   Local string &sError, &sLicenseCode, &sLicenseGroup;
   Local boolean &status;

   SQLExec("SELECT LICENSE_CODE, LICENSE_GROUP FROM PSOPTIONS",⇒
 &sLicenseCode, &sLicenseGroup);
   /* introduce unknown propFormat of String1 */
   &status = %Super.InsertData("String1",⇒
 "Your License Code is: ", Upper(&sLicenseCode));

   If Not &status Then
      &status = %Super.InsertData("String", "Status Failed!",⇒
 "This message will be printed out in the HTML page if⇒
 something fails here.  This is expected behaviour.");
   End-If;
end-method;

method IsPlugIn
end-method;
```

Handling Dynamic Prompting Failure

The following example demonstrates error handling when retrieving user prompts.

```
import PT_DIAGNOSTICS:*;

class TestFailedGetUserInputByKey extends PTDiagnostics
   /* Constructor */
   method TestFailedGetUserInputByKey();

   /* Public Method */
   method GetDiagnosticInfo();
   method IsPlugIn();

private

end-class;

method TestFailedGetUserInputByKey;
   Local boolean &status = False;
   %Super = create PTDiagnostics();
   &status = %Super.SetProperty(%This, "Purpose", "String",⇒
 "This is a diagnostic to test getting a 'False' from GetUserInputByKey.");
end-method;

method GetDiagnosticInfo
   Local boolean &status = False;
   Local string &sVal, &sError;
```

```
   Local number &iCount = 0;
   Local Record &REC;
   Local SQL &SQL1;

   &REC = CreateRecord(Record.PSRECDEFN);
   &SQL1 = CreateSQL("%SelectAll(:1) where RECNAME LIKE :2");

   /*
 sKeyID "Recs" has already be defined elsewhere in the package
 see if we can get RecJY.  should get a False
*/
   &status = %Super.GetUserInputByKey("RecsJY", &sVal);
   If &status Then
      &SQL1.Execute(&REC, Upper(&sVal | "%"));
      While &SQL1.Fetch(&REC)
         &iCount = &iCount + 1;
         &status = %Super.InsertData("String", "Record #" | &iCount,⇒
 &REC.RECNAME.Value | " (" | &REC.RECDESCR.Value | ")");
      End-While;
   Else
      &status = %Super.InsertData("String", "Status Failed!",⇒
 "Failed status encountered in retrieving RecsJY key.⇒
 This is expected behaviour.");
   End-If
end-method;

method IsPlugIn
end-method;
```

APPENDIX A

Administering PeopleSoft Databases on Microsoft SQL Server

This appendix discusses:

- Server options.
- Required database configuration.
- Other considerations.

Server Options

This section discusses:

- Delivered configuration.
- Access ID.
- Service Packs and QFE.

Delivered Configuration

PeopleSoft deliver a minimal configuration for its applications through the file %PS_HOME%\scripts \spconfig.sql. The configuration values within this file are intended for installation only. It is a good practice to review and modify the parameters if necessary and adjust them to your site requirements.

The configuration delivered with this file may change from site to site and from application to application. The parameters modified with the file are:

- lightweight pooling
- network packet size
- priority boost

Don't use "priority boost" when running additional applications like PeopleSoft Process Scheduler on your database server box.

The delivered configuration remarks the importance of the parameters listed at the configuration file. A DBA should adjust them to the appropriate values if required. Advanced knowledge of the server behavior is necessary for their modification.

See Also

Microsoft SQL Server documentation

Access ID

PeopleSoft does not recommend the use of the SQL Server system administrator login *sa* as an access ID.

Service Packs and QFE

PeopleSoft always runs certifications on the latest service packs as they become available. PeopleSoft does not run certification tests for any particular SQL Server QFE, but considers them supported when recommended by Microsoft to solve specific problems. PeopleSoft does not distribute SQL Server QFE software; please contact Microsoft to make sure a QFE is required and to download the software.

Required Database Configuration

PeopleSoft applications require a standard database configuration that's not optional and should not be changed. This section discusses the options that you must enable:

- ANSI nullability.
- Quoted Identifier, Arithabort, and functional index.
- Database collation settings.

ANSI Nullability

Make sure your database uses ansi nulls by default. This is a database option that will be set up at installation time. The configuration occurs automatically when using the database configuration wizard and is enabled by the SQL script addobj.sql when installed manually.

The following line shows how to enable this parameter at SQL Server 2000 using query analyzer:

```
EXEC sp_dboption <DBNAME>, 'ansi null default', true
```

Quoted Identifier, Arithabort, and Functional Index

PeopleSoft uses computed columns that allow the creation of functional indexes. A functional index is an index created to keep uniqueness in a table when the number of keys exceeds the SQL Server limits of 16 key columns for an index.

PeopleSoft implements the functional index creating an index over a computed column. The computed column MSSCONCATCOL is the result of adding up all the key columns required to keep uniqueness. What makes a functional index special is that it's required only when the number of key columns exceeds the SQL Server 2000 limits, which is 16 key columns maximum for an index.

In order to create indexes on computed columns, SQL Server 2000 requires the *Quoted Identifier* option to be enabled in the database. This is the default configuration, but this option could be overridden as a connection option from any client. If you are using Query Analyzer to run SQL scripts, look at Tools, Options, Connection Properties on your Query Analyzer menu and make sure the *Quoted Identifier* option is selected, which will activate it for that particular connection.

Another important option that needs to be enabled to operate computed columns is the database property *Arithabort.* Make sure this option is enabled for your PeopleSoft database. Both options are explicitly set during installation automatically by the database configuration wizard or when running the script *createdb.sql* at the database installation.

See Also

Microsoft SQL Server documentation

Database Collation Settings

The use of the right collation is very important for PeopleSoft applications. PeopleSoft delivers its applications with a standard collation of *Latin1_General_Bin* on SQL Server 2000. This collation was selected for being compatible with the binary sort order used on SQL Server 7.

However, PeopleSoft supports other sort orders with some applications. The application installation manual will point out whether this is permitted for a particular application. The sort order supported must be Kana sensitive, case sensitive and accent sensitive. Therefore a collation such as *Latin1_CS_AS_KS* is supported. Note that the *Latin1_General_Bin* collation also satisfies this requirement.

Please consult your application installation manual for further details since it is not recommendable to change the delivered collation for all applications.

We recommend the use of the collation delivered as default on PeopleSoft installations. This collation option will be set when running the *creatdb.sql* script at installation. The script runs automatically when using the database configuration wizard. It is a requirement to run the script when installing the database manually.

Other Considerations

This section discusses:

- Recovery model.
- Auto create and auto update statistics.
- Automatic file growth.
- Trace flags.
- Database monitoring.
- File management.
- Tuning Tempdb.
- Moving Tempdb.

Recovery Model

PeopleSoft recommends using the Full recovery model on SQL Server databases. All production databases should use this model for better reliability. The PeopleSoft applications do not require any particular recovery model but using the Full recovery model is considered the best practice.

See Also

Microsoft SQL Server documentation

Auto Create and Auto Update Statistics

SQL Server 2000 allows you to create statistics and update them automatically. We recommend that you leave this feature enabled for PeopleSoft applications.

See Microsoft SQL Server documentation.

Automatic File Growth

SQL Server 2000 enables you to automatically grow a database device when it's full. PeopleSoft recommends that you leave this feature enabled for all the devices used by PeopleSoft applications, however this feature should be in use only to avoid errors during transactions in the database due the lack of space on the database devices.

The automatic file growth doesn't prevent the DBA from determining the appropriate device size required for each database hosting a PeopleSoft application. Consult your application installation manual for further information on database sizing. Don't use the automatic file growth on a database device with no appropriate sizing. Every time the device grows SQL Server is interrupted, which causes severe performance problems. Use this only as a way to prevent interruption of the database activity.

The usage of the automatic file growth feature is enforced at database creation through the createdb.sql script. This option is optional when installing the database through the database configuration wizard. When installing the database manually it is necessary to uncomment the following lines at the script and complete the script with appropriate device and database names:

```
-- ALTER DATABASE <DBNAME> MODIFY FILE (NAME = <DATANAME>, MAXSIZE = UNLIMITED)
-- go
-- ALTER DATABASE <DBNAME> MODIFY FILE  (NAME = <LOGDATANAME>, MAXSIZE = UNLIMITED)
-- go
```

Trace Flags

When reporting problems to customer support, it is advisable to generate files with traces of the problem that you want to report. Use the trace flags incorporated in PeopleSoft tools to generate these files. The trace flags are accessible through the configuration files for the Process Scheduler and the Application Server and through the selection of several flags when using the PeopleSoft Configuration Manager on your developer workstation.

Use "TRACESQL=63" to display the SQL statements executed when using PeopleSoft applications. This trace flag is very useful to identify problems in the SQL being executed against a database that hosts a PeopleSoft application.

The trace flag will show the details about the execution of a sql statement such as: if the statement was recompiled or if it's using an old query plan, the time it took to execute, the time between executions, if the SQL was parametrized, among other things.

Once you find the SQL with problems, you can use the SQL Server profiler to reproduce this outside of your PeopleSoft application.

See Also

Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration, "Setting Application Server Domain Parameters"

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler, "Using the PSADMIN Utility"

Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration, "Using PeopleSoft Configuration Manager"

Database Monitoring

Available through the configuration files for the Process Scheduler and the Application Server, the activation of the “EnableDBMonitoring” flag allows you to populate context information of the query executed against the database. This is particularly useful to gather information about the PeopleSoft user running a particular SQL statement.

Examples of SQL Statements

The following are examples of SQL statements that will display the context information of a user once “EnableDBMonitoring” is enabled. Modify the scripts according to your necessities.

```
--SQL to get OPRID only
select (substring(cast(context_info as varchar(128)),0,PATINDEX('%,%',cast(context_⇒
info as varchar(128)))))
from master..sysprocesses where spid=<spid>


--SQL to select the network id if it is there
select substring(cast(context_info as varchar(128)),
 len(substring(cast(context_info as varchar(128)),0,PATINDEX('%,%',cast(context_⇒
info as varchar(128)))))+2,
 PATINDEX('%,%',substring(cast(context_info as varchar(128)),len(substring(cast⇒
(context_info as varchar(128)),0,PATINDEX('%,%',cast(context_info as varchar⇒
(128)))))+2,128))-1)
from master..sysprocesses where spid=<spid>

--SQL to select network host
select
substring(substring(cast(context_info as varchar(128)),
len(substring(cast(context_info as varchar(128)),0,PATINDEX('%,%',cast(context_⇒
info as varchar(128)))))+2
+PATINDEX('%,%',substring(cast(context_info as varchar(128)),len(substring(cast⇒
(context_info as varchar(128)),0,PATINDEX('%,%',cast(context_info as varchar⇒
(128)))))+2,128))
,128),0,PATINDEX('%,%',substring(cast(context_info as varchar(128)),
len(substring(cast(context_info as varchar(128)),0,PATINDEX('%,%',cast(context_⇒
info as varchar(128)))))+2
+PATINDEX('%,%',substring(cast(context_info as varchar(128)),len(substring(cast⇒
(context_info as varchar(128)),0,PATINDEX('%,%',cast(context_info as varchar⇒
(128)))))+2,128))
,128))) from master..sysprocesses where spid=<spid>

--SQL to select App server domain
select reverse(substring(reverse(cast(context_info as varchar(128))),0,PATINDEX⇒
('%,%',reverse(cast(context_info as varchar(128))))))
from master..sysprocesses where spid=<spid>

--SQL to select all the information trimming blanks
select
substring(cast(context_info as varchar(128)),0,128-PATINDEX('%,%',reverse(cast⇒
```

```
(context_info as varchar(128)))+10)
from master..sysprocesses where spid=<spid>
```

File Management

PeopleSoft recommends using separate physical disks for the MS SQL Server devices. Ideally, you should use separate physical disks or disk arrays for the Tempdb database, master database, as well as for the operating system and the paging file (in case you run some additional application other than the database software on the server). This will improve performance and offer protection in case of a device failure. The more spindles the better, so always choose more disks over larger sizes. If you do not have separate physical disks for each of the items above, you should at least place your tempdb, data, and log files on separate physical devices for recovery and performance purposes.

Make sure your log device is using its own disk controller and is not accessed by anything else.

Note. Always consider disk fault tolerance when deciding on your server configuration.

Using Filegroups

Microsoft SQL Server 2000 maps each database using a set of operating system files. All database objects and data are stored within these files. A database can have one or more data files (*.mdf* and *.ndf* extensions) and transaction log files (*.ldf* extension).

Filegroups are logical containers that enable the database files (.mdf, .ndf, and .ldf) to be grouped together for administrative and data placement purposes. While a filegroup can contain more than one database file, each database file can be a member of only one filegroup.

Note. While the number and placement of data files may have an impact on system performance, the number and organization of filegroups has no direct correlation to performance.

Because of the large number of tables and the complex IO patterns of a PeopleSoft database, you must consider the placement of the data files carefully to maximize performance. The best approach is to use a RAID-10 disk configuration and spread the data over as many disks as possible. Use a large number of smaller sized disks, rather than a small number of larger disks.

In addition to the main database, give careful consideration to the configuration and placement of the SQL Server Tempdb database, because PeopleSoft applications use it heavily. Given the unusual input/output characteristics of this database (on average, 50% read, 50% write), you should create your Tempdb database on a separate RAID-10 disk with multiple database files. Generally, it's appropriate to make the number of data files equal to the number of processors used.

See Also

Microsoft SQL Server documentation

Microsoft Windows documentation

Tuning Tempdb

The tempdb database is used for sorting result sets, either because of an 'order by' clause in a query or to organize pre-result sets needed to execute a given query plan. If tempdb is being used extensively (evidenced by many work tables being created per second, or else heavy I/O to tempdb files), performance may be improved by tuning it.

First, consider moving tempdb to its own set of disks. Do this with ‘alter database’ using the ‘modify file’ option to specify a new location for tempdb’s data file and log file. You may also want to increase the SIZE option to a larger value, such as 500MB and increase the FILEGROWTH option to, say, 50MB. This will require you to shut down and restart SQL Server.

Note. The estimated size of a tempdb database for PeopleSoft applications is about 15% of the total size of your application database.

Moving Tempdb

During installation of Microsoft SQL Server, tempdb is put in the default data directory. If you wish to move it to a separate disk and resize it, the following scripts are an example of how this can be accomplished.

```
-- To find out where tempdb resides:
-- The following stored procedure will show on which drive tempdb
-- data and log files reside.

sp_helpdb tempdb

-- This example script moves tempdb to drive f:

alter database tempdb
modify file ( name = 'tempdev' , filename = 'f:\data\tempdb.mdf'  )
go
alter database tempdb
modify file ( name = 'templog' , filename = 'f:\log\tempdblog.ldf' )
go

-- This example script resizes the tempdb data file to 500MB
-- and the tempdb log file to 500MB

alter database tempdb
modify file ( name = 'tempdev' , size = 500MB )
go
alter database tempdb
modify file ( name = 'templog' , size = 500MB  )
go
```

Also, consider adding several data files to tempdb rather than just one. This strategy can reduce contention on tempdb. Do this by using alter database using the add file option. This approach will require you to shut down and restart SQL Server.

APPENDIX B

Administering PeopleSoft Databases on DB2 UDB for OS/390 and z/OS

This appendix provides an overview of administration on DB2 UDB for OS/390 and z/OS, and discusses how to:

- Monitor batch programs.
- Associate PeopleSoft operators with DB2 UDB threads.
- Run COBOL.
- Administer SQR for OS/390 and z/OS.
- Update statistics.
- Set the number of temporary tables.

Note. *DB2 UDB for OS/390 and z/OS* is the official IBM name for the DBMS. In the current PeopleTools release, PeopleSoft no longer supports the OS/390 operating system, only z/OS, its replacement.

For the sake of brevity, this appendix sometimes refers to DB2 UDB for OS/390 and z/OS as *DB2 z/OS.*

Understanding DB2 UDB for OS/390 and z/OS Administration

This section discusses:

- Database considerations.
- Concurrency.

Database Considerations

The section discusses:

- Tablespace strategy
- Locksize tablespace

Tablespace Strategy

Tablespaces named xxLARGE—where xx is a product identifier, such as HR—contain tables that grow substantially and/or experience high update activity. You should track the growth and extents for tablespaces and indexes, as well as monitor for page splits in the indexes.

Each of the tables in xxLARGE is a candidate for partitioning or for a separate tablespace. Tables defined in tablespaces other than xxLARGE are relatively stable and can be defined in shared tablespaces with little, if any, freespace.

As a general rule of thumb, the xxLARGE tablespaces grow substantially large with application data and contain the largest tables in your database. From a PeopleTools perspective, there are several delivered tablespaces that may grow in size. For example, tablespace PTTLRG contains PeopleTools tables (XLATTABLE, PSPCMNAME, and others) that my grow large in size. The “Tree” tables are delivered in tablespace PTTREE—tables prefixed with PSTREE%. These tables may grow substantially early on as you add branches and nodes in the Tree Manager, then plateau once the tree structure is fully defined.

Customers with large amounts of data may require that the larger tables be partitioned, and as a result must be moved to their own tablespace. This improves concurrency and also allow DB2 UDB utilities such as backup, reorg, and Runstats to be run in parallel.

With PeopleSoft 8, new tablespaces were introduced for tables requiring row level locking to avoid deadlock and timeout errors. Those tablespaces are: PTLOCK, PTAMSG, PTPRC, PTRPTS, PTAUDIT, , PTPRJWK PTCMSTAR and PSIMGR. Note that PSIMGR and PSIMAGE both require a 32K page size. If redistributing any of the tables delivered in these tablespaces, it is critical for performance to carry over the row level locking attribute and buffer pool assignment for the new tablespace.

Locksize Tablespace

You can avoid reaching lock escalation thresholds by ALTERing tablespaces from LOCKSIZE ANY (or PAGE) to LOCKSIZE TABLESPACE for the duration of batch jobs. This technique also improves batch program performance.

The ALTERed LOCKSIZE specification is effective immediately. Plan rebinds are not needed since PeopleSoft uses dynamic SQL. The simplest way to implement this technique is to ALTER all of the application tablespaces. If that is not desirable, determine the tables accessed in a particular job by examining SQL statements in PS_SQLSTMT_TBL and finding their corresponding tablespaces.

Note. Tablespaces should be ALTERed back to the original value after job completion. Tablespace locks will lock out online users until LOCKSIZE is reset to PAGE or ANY. If online users are active during the time you are running batch jobs, you may not want to ALTER LOCKSIZE to TABLESPACE.

Concurrency

This section discusses:

- CursorHold
- Isolation levels and CURRENTDATA
- RELCURHL

CursorHold

Enabling CURSORHOLD was required in pre 8.0x releases of PeopleTools; however, this is now the default for DB2 Connect (CURSORHOLD=1). For PeopleTools 8 and beyond, the use of Cursor With Hold, or "persistent cursors", with PeopleSoft applications is controlled entirely by PeopleTools. Consequently, there is no reason to use anything other than the IBM default for CURSORHOLD.

Isolation Levels and CURRENTDATA

PeopleSoft batch processes interface to DB2 UDB either through PTPSQLRT (for Cobol and AE), or through SQRPLAN for SQRs. Both of these are bound with the defaults—that is, CS (cursor stability) and CURRENTDATA NO. Using CURRENTDATA NO results in less lock contention in DB2 UDB and potentially reduce deadlock situations. It also provides two extra benefits:

- Block fetch is enabled for ambiguous cursors.

• DB2 UDB considers parallelism for ambiguous cursors.

RELCURHL

A DB2 z/OS subsystem parameter RELCURHL lets you indicate that you want DB2 UDB to release a data page or row lock after a COMMIT is issued for cursors defined WITH HOLD. This lock is not necessary for maintaining cursor position.

The default for DB2 z/OS is YES. In prior releases, the value was NO, which causes DB2 UDB to hold a data page or row lock for the row on which the cursor is positioned. This lock is not necessary for maintaining cursor position and could cause deadlocks. The PeopleSoft recommendation is Yes to improve concurrency.

Monitoring Batch Programs

This section provides an overview of batch program monitoring tools and discusses how to:

• Enable DB2 CLI/ODBC trace.

• Enable the PTPSQLRT Mainframe Statistics report.

• Enable dynamic explains.

• Enable parallelism.

• Enable PeopleSoft SQL trace.

• Enable SQR monitoring.

Understanding Batch Program Monitoring Tools

This section discusses the utilities provided by PeopleSoft and IBM to monitor and help you tune the performance of PeopleSoft batch programs.

Utility	Description
SQL Trace - Client	This PeopleSoft client utility records the actual SQL statements that PeopleTools and batch COBOL send to the database, along with their relative processing times. This trace can increase response time significantly.
DB2 CLI/ODBC Trace	The DB2 CLI/ODBC trace is a trace provided by IBM that can be very helpful in debugging DB2 Connect problems. For PeopleSoft, this trace can be very useful for tracking down problems when running client COBOL or AE. It can also be used for debugging the PeopleSoft on-line as well. Like many of these other traces, enabling this trace slows processing down and results in large output files on the client.

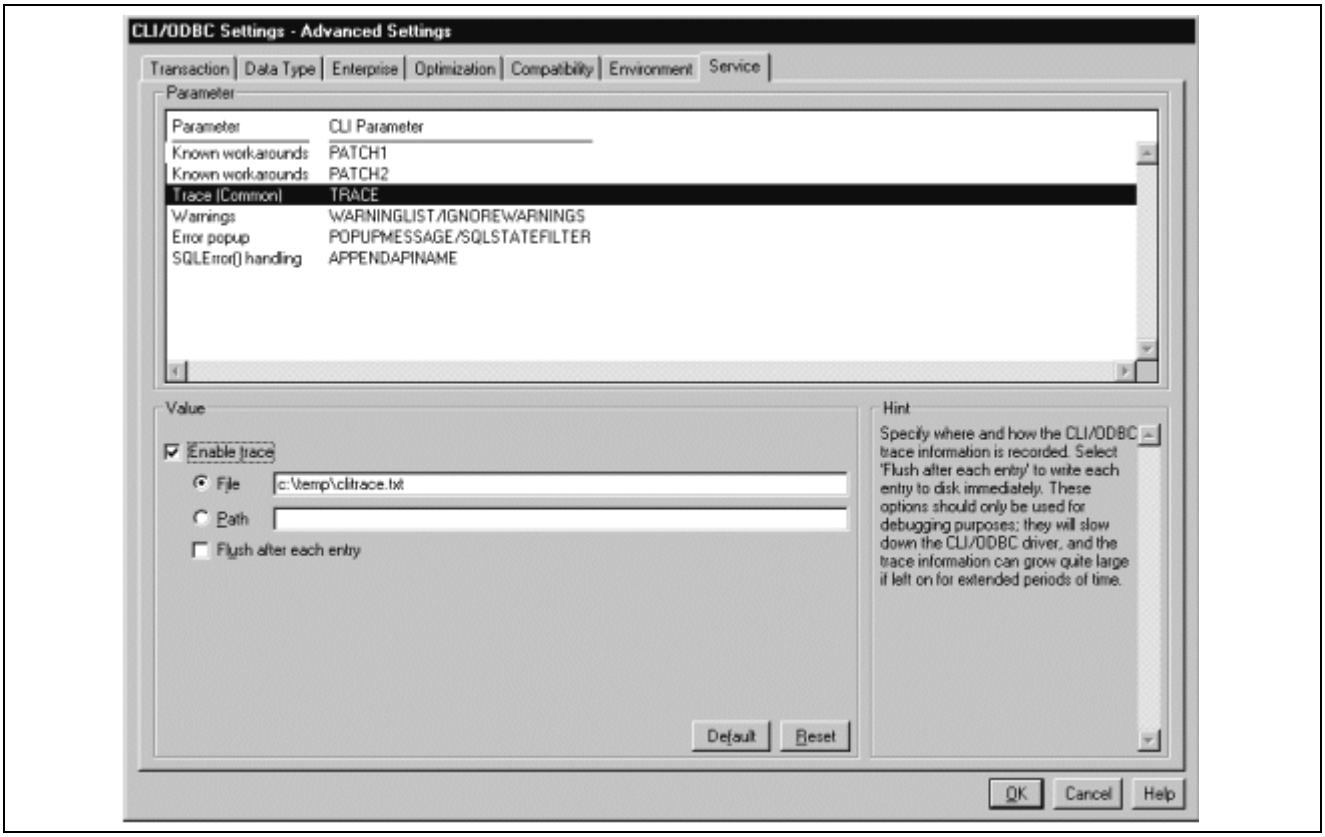
Utility	Description
PTPSQLRT Statistics Report	The PeopleSoft COBOL API, PTPSQLRT, provides a report called “PTPSQLRT Statistics” that shows frequency and elapsed times for SQL statements executed in batch processing. This report provides you with an effective and easy-to-use tool to monitor and evaluate SQL performance. No DBA involvement is required, and impact on batch performance is negligible. This report can track both client or mainframe COBOL. For client COBOL, you enable this report by selecting a SQL Trace option. On the mainframe, you modify the JCL to set the trace option to *Y*.
Dynamic Explains	The PTPSQLRT API program allows you to capture access path information of the SQL statements executed during batch processing. This information can help you tune any problem queries identified by the Timings Statistics Report.
Parallelism	The PTPSQLRT API program allows you to capture access path information of the SQL statements executed during batch processing. This information can help you tune any problem queries identified by the Timings Statistics Report.
Parallelism	You can specify the degree of parallelism for the execution of queries that are dynamically prepared by the application process.
SQL Trace - Server	You can run a PeopleSoft SQL trace on the z/OS server that displays similar results as the SQL Trace on the client. You can see SQL times, return codes and SQL that is executed.
SQR flag	You can use the `-S` flag to generate a SQL script consisting of fully resolved DB2 z/OS statements. This trace doesn’t provide timings for the SQL statements.

Enabling DB2 CLI/ODBC Trace

The DB2 CLI/ODBC trace can be enabled in two places. The first method is to update the DB2CLI.INI directly. These are the recommended settings for turning Trace on:

```
[COMMON]
TRACEFLUSH=1
TRACEPATHNAME=C:\TEMP\DB2TRACE\ (or use TRACEFILENAME=C:\TEMP\<file> to direct to⇒
 file)
TRACECOMM=1
TRACE=1   (trace=0 turns the trace OFF)
TRACEREFRESHINTERVAL=60
```

The second method is to enable the trace using the Client Configuration Assistant.



DB2 UDB Client Configuration Assistant

Enabling the PTPSQLRT Mainframe Statistics Report

Control over enabling and disabling Statistics Reports on the z/OS batch server is now done primarily through the PSOS390 Process Scheduler configuration. The JCL shell SHELCBL.JCT file located in the /u/datax/psvvv/appserv/prcs/<process scheduler name>shelljcl directory on USS contains symbolic variables for each parameter that is resolved by Process Scheduler when a COBOL job is submitted. As an option, you can replace the symbolic with the "hard coded" value of *Y* to enable or *N* to disable the particular parameter. The section in the JCL shell appears as follows:

```
//* PARMFILE - PARM 1 IS OPRID - LEAVE AS SYMBOLIC
//*            PARM 2 IS RUN CONTROL NAME
//*            PARM 3 IS A YES/NO SWITCH FOR PERFORMANCE STATISTIC
//*           PARM 4 IS PROCESS INSTANCE; 0 TRIGGERS PROC INST LOGIC
//*            BLANK IF NON-PROCESS SCHEDULER JOB
//*            PARM 5 IS A YES/NO SWITCH FOR DYNAMIC EXPLAINS
//*            PARM 5 REQUIRES THAT PARM 3 IS SET TO YES
//*          PARM 6 IS A YES/NO SWITCH TO ENABLE PARALLEL PROCESSING
//*            PARM 7 IS A YES/NO SWITCH TO ENABLE SQL TRACE
//*            PARM 8 IS A YES/NO SWITCH TO ENABLE RUN STATISTICS
//*         PARM 9 IS A REMOTE-CAL INDICATOR - ALWAYS "BATCH" IN JCL
//*            PARM 10 IS THE FULL DIRECTORY PATH OF PS_HOME
//*            PARM 11 IS THE FULL DIRECTORY PATH OF PS_SERVDIR
//*            PARM 12 IS THE FULL PATH OF THE PROCESS SCHEDULER
//*              CONFIGURATION FILE
//*            PARM 13 IS THE USERID WITH FULL AUTHORIZATION IN USS
```

```
//*             OF ALL HFS DIRECTORY WHERE AE WILL WRITE THE LOGS TO
//*            PARM 14 IS THE JCL JOB NAME
//*            PARM 15 IS THE REGION SIZE SETTING (DEFAULT IS -1)
//*          PARM 16 IS THE MAX CPU TIME ALOTTED FOR AN AE SUBROUTINE
//*
//*
//*****************************************************************
//*  NOTE  ON ENABLE RUN STATISTICS:  IF YOU HAVE CHANGED THE
//*         SETTINGS TO RUN STATISTICS ON TABLESPACES, YOU MUST BIND
//*         THE PLAN FOR PTPSQLRT USING BINDAREP AND BINDEREP IN THE
//*        JCLLIB LIBRARY WITH THE BIND OPTION:  PKLIST (DSNUTILS.*)
//*         INCLUDED IN THE OPTIONS LIST
//*
//*         PARAMETERS 10-16 ARE REQUIRED WHEN RUNNING A COBOL
//*         PROGRAM THAT TRIGGERS AN AE SUBROUTINE
//*********************************************************************
//*
//PARMFILE DD *
%OPRID%
%RUNID%
%PERFSTAT%
%INSTANCE%
%DYNEXPLN%
%PARALLEL%
%SQLTRACE%
%RUNSTAT%
BATCH
%PS_HOME%
%PS_SERVDIR%
%PS_SERVERCFG%
%PS_CONFIG%
%HFS_USERID%
%JOBNAME%
%REGION_SIZE%
%CPU_TIME%
/*
```

Viewing the PTPSQLRT Report

Timings represent elapsed (wall clock) times expressed in seconds. For example, the value 2.383 means 2 seconds, 383 milliseconds. Where a value for COUNT exists, and TIME = .000, indicates an elapsed time of less than 1 millisecond. Because the results of this report are in elapsed time, care should be taken in interpreting the results. If you are using the report to identify poorly performing SQL, then multiple executions should be analyzed and the results compared before drawing any conclusions. Further analysis requires Explains and CPU timings of possible problem statements.

- STATEMENT NAME

  The name associated with a single SQL statement to be executed dynamically by program PTPSQLRT. It can be either a Select, Update, Insert or Delete statement as indicated by the “S,” “U,” “I,” or “D” designation in the statement name. Each statement may be executed once, or many times during a PeopleSoft batch program.

A statement name is made up of the program name and the type of SQL DML. For example, “PSPAGERT_S_AGERT” is in program PSPAGERT and is a SELECT; AGERT is a unique name within the application.

To examine a statement’s contents, you may either:

- Select * From <your_id>.PS_SQLSTMT_TBL Where PGM_NAME=’PSPAGERT’, STMT_TYPE=’S’ and STMT_NAME=’AGERT’
- On a LAN, locate the \PS\SRC\CBL\BASE subdirectory; statements are contained in files having the “DMS” extension and a filename of program name.

- RETRIEVE

  Reports the number of times and total time it takes for PTPSQLRT to select SQL statement text from PS_SQLSTMT_TBL. Retrieve count of zero indicate a Dynamic Statement that is not stored in PS_SQLSTMT_TBL.

- PREPARE

  Reports the number of times and total time it takes PTPSQLRT to do “DECLARE CURSOR” and prepare the SQL statement.

- CLOSE

  Reports the number of times and total time to CLOSE an open cursor. Applies to Cursor SELECT statements only

- FETCH

  Reports the number of FETCHes as well as the total time spent

  For example, if Count=2 and Time=.040, it means that the program issued 2 fetch statements totaling .040 seconds (40/1000), or an average of .020 per fetch. If EXECUTION Count = 1, it means 2 FETCHes were done in a single OPEN Cursor. (This is true for columns 2 through 6.)

- STMT TOTALS

  Shows sum of all timings for each statement, calculated by adding all TIME accumulations horizontally. Also shown under “% SQL” is the percentage of the total processing time that a particular SQL statement represents. For example, a “% SQL” time of 1.03 indicates a statement used just over 1% of all SQL processing time.

- TOTAL IN SQL CALLS

  Total time spent by PTPSQLRT making SQL calls (sum of all SQL activity).

- TOTAL IN SQLRT STATS

  Total time spent by PTPSQLRT producing statistics (Assembler calls, COBOL processing).

- TOTAL IN SQLRT OTHER

  Total time spent by application programs (as in PSPTCALC.CBL) processing COBOL statements.

- TOTAL IN SQLRT

  Total in SQL Calls + Total in SQLRT stats + Total in SQLRT other. Total time spent by PTPSQLRT (SQL, STATS, COBOL).

- TOTAL IN APPL COBOL

  Total time processing COBOL programs.

- TOTAL IN APPL

  Total in SQLRT + Total in APPL COBOL. Grand total processing of all COBOL, PTPSQLRT, and SQL processing.

- TOTAL SQLRT CALLS

  Total number of calls made to PTPSQLRT called by COBOL programs.

- TOTAL SQLRT STATEMENTS

  Total number of distinct SQL statements executed.

- MAXIMUM CURSORS CONNECTED

  Largest number of active concurrent connection paths to DB2 UDB during program execution; "cursor" refers not only to open cursors, but deletes, inserts and updates as well.

There are three ways to enable the statistics report generation. The following steps take you through the three possibilities.

Enabling the Statistics Report on the DB2 UDB for OS/390 and z/OS Server

To enable the statistics report on the DB2 UDB for OS/390 and z/OS server:

1. Initialize the PSADMIN program on Unix System Services to administer the Process Scheduler PSOS390.
2. You may either select Option 3 - Configure a Process Scheduler Server, or Option 6 - Edit a Process Scheduler Configuration File.
3. If you select Option 3, you would set the value for TraceSQL to 128.
4. If you select Option 6, you need to locate the related section in the file and change the TraceSQL flag to 128.
5. As a third option, you could chose Option 9 - Edit a Shell JCL from the PSADMIN menu and select JCL Shell shelcbl.jct (selection 1)

   This brings the file up in the VI editor.
6. Locate the PARMFILE section in the JCL Shell, and replace the symbolic %PERFSTAT% (Performance Statistic-PARM 3) parameter with the value Y for Yes.
7. Save the file and stop and restart the Process Scheduler.

   The Statistics Report is written to a Sequential Dataset HLQ.ppvvv.<program name>.

Enabling Dynamic Explains

The PTPSQLRT API program enables you to capture access path information of the SQL statements executed during batch processing. This information can help you tune any problem queries identified by the Timings Statistics Report.

Note. The Dynamic Explains feature is a performance tool for DBAs and other PeopleSoft product support personnel to be used for performance tuning. We recommend that this feature be disabled when in production mode.

There are three ways to enable generating a dynamic explain. The following steps take you through the three possibilities.

To enable Dynamic Explains in the JCL:

1. Initialize the PSADMIN program on Unix System Services to administer the Process Scheduler PSOS390.
2. You may either select Option 3 - Configure a Process Scheduler Server, or Option 6 - Edit a Process Scheduler Configuration File.
3. If you select Option 3, you need to set the value for TraceSQL to 256.

4. If you select Option 6, you need to locate the related section in the file and change the TraceSQL flag to 256.
5. As a third option, you could chose Option 9 - Edit a Shell JCL from the PSADMIN menu and select JCL Shell shelcbl.jct (selection 1)

   This brings the file up in the VI editor.
6. Locate the related section in the JCL Shell, and replace both the symbolic %PERFSTAT% (Performance Statistic-PARM 3) and %DYNEXPLN% (Dynamic Explains-PARM 5) parameters with the value Y.
7. Save the file. It is not necessary to stop and re-start the Process Scheduler for the change in the JCL shell to take effect.

Enabling Parallelism

The PTPSQLRT API program provides a feature that allows you to enable DB2 UDB parallelism. If enabled, PTPSQLRT issues the following command to DB2 UDB:

```
SET CURRENT DEGREE = 'ANY'
```

The CURRENT DEGREE parameter specifies the degree of parallelism for the execution of queries that are dynamically prepared by the application process. For PeopleSoft, this applies to all queries run in batch. While setting CURRENT DEGREE =ANY enables DB2 UDB parallelism, this does not necessarily mean that the statements in the application programs uses parallelism. What it means is that DB2 UDB's optimizer considers parallelism as a possible option.

There are also three ways to enable parallel processing. The following steps takes you through the three possibilities.

To enable parallelism in the JCL

1. Initialize the PSADMIN program on Unix System Services to administer the Process Scheduler PSOS390.
2. You may either select Option 3 - Configure a Process Scheduler Server, or Option 6 - Edit a Process Scheduler Configuration File.
3. If you select Option 3, you need to set the value for Enable Parallel Processing to *1*.
4. If you select Option 6, you need to locate the related section in the file and change the Enable Parallel Processing flag to 1.
5. As a third option, you could chose Option 9 - Edit a Shell JCL from the PSADMIN menu and select JCL Shell shelcbl.jct (selection 1).

   This brings the file up in the VI editor.
6. Locate the section in the JCL Shell, and replace the symbolic %PARALLEL% (Parallel Processing-PARM 6) parameter with the value *Y*.
7. Save the file. It is not necessary to stop and re-start the Process Scheduler for this change to be recognized.

Enabling PeopleSoft SQL Trace

The PTPSQLRT API program enables you to capture a PeopleSoft SQL trace. This trace has been improved in PeopleSoft 8 to mirror the familiar SQL trace used on the client. The functionality of this trace has been improved to include all dynamic SQL statements that have been captured in the past by the DYSQLOG trace.

Note. The SQL Trace feature is a tool for DBAs and other PeopleSoft product support personnel to use for performance tuning. We recommend that this feature be disabled in production mode.

There are three ways to enable the SQL Trace. The following steps take you through the three possibilities.

To enable PeopleSoft SQL trace in the JC:

1. Initialize the PSADMIN program on Unix System Services to administer the Process Scheduler PSOS390.
2. You may either select Option 3 - Configure a Process Scheduler Server, or Option 6 - Edit a Process Scheduler Configuration File.
3. If you select Option 3, you need to set the value for TraceSQL to 1.
4. If you select Option 6, you need to locate the related section in the file and change the TraceSQL flag to *1*.
5. As a third option, you could chose Option 9 - Edit a Shell JCL from the PSADMIN menu and select JCL Shell shelcbl.jct (selection 1).

   This brings the file up in the VI editor.
6. Locate the section in the JCL Shell, and replace the symbolic %SQLTRACE% parameter with the value *Y*.
7. Save the file. If you originally configure the Process Scheduler with “Allow Dynamic Changes = Y” it isn’t necessary to stop and re-start the Process Scheduler for this change to take effect.

The following is Sample PeopleSoft SQL trace from z/OS output queue:

```
Elapsed     SQL
    Time       Time  Crsr Return  DB API Statement
    ------    ------ ---- ------  ----------------
     0.044     0.044      RC=   0 CEX Stmt=SELECT OWNERID FROM PSSTATUS
     0.000     0.000      RC=   0 CEX Stmt=SET CURRENT SQLID = 'PT800RB'
     0.000     0.000      RC=   0 CEX Stmt=SET CURRENT DEGREE= '1'
     0.039     0.029      RC=   0 GETSTMT Stmt=PTPRUNID_U_UPDID
     0.003     0.002 #001 RC=   0 Prepare=UPDATE PS_PRCSSYSTEM SET⇒
 LASTPRCSINSTANCE = LASTPRCSINSTANCE + 1
     0.013     0.000 #001 RC=   0 Execute
     0.013     0.000 #001 RC=   0 Row Count=000000001
     0.000     0.000      RC=   0 GETSTMT Stmt=PTPRUNID_S_GETID
     0.002     0.002 #001 RC=   0 COM=SELECT LASTPRCSINSTANCE FROM PS_PRCSSYSTEM
     0.006     0.000 #001 RC=   0 SSB=0001 TYPE=SQLPSLO LEN=0004
     0.000     0.000 #001 RC=   0 Execute
     0.000     0.000 #001 RC=   0 Fetch
     0.002     0.000 #001 RC=   0 Commit
     0.011     0.000      RC=   0 GETSTMT Stmt=PTPLOGMS_S_OPRDEFN
     0.002     0.002 #002 RC=   0 COM=SELECT LANGUAGE_CD FROM PSOPRDEFN WHERE⇒
 OPRID = :1
     0.007     0.000 #002 RC=   0 SSB=0001 TYPE=SQLPBUF LEN=0003
     0.000     0.000 #002 RC=   0 Bind=0001 Type=SQLPBUF Len=0002 Data=PS
     0.000     0.000 #002 RC=   0 Execute
     0.000     0.000 #002 RC=   0 Fetch
     0.000     0.000 #001 RC=   0 Close Cursor for PTPRUNID_S_GETID
     0.000     0.000      RC=   0 GETSTMT Stmt=PTPLOGMS_I_LOGMSG
     0.002     0.002 #001 RC=   0 Prepare=INSERT INTO PS_MESSAGE_LOG ( PROCESS_⇒
INSTANCE, MESSAGE_SEQ, JOBID,
 PROGRAM_NAME, MESSAGE_SET_NBR, MESSAGE_NBR, MESSAGE_SEVERITY, DTTM_STAMP_SEC )⇒
 VALUES (:1,:2,:3,:4,:5,:6,:7,:8)
     0.006     0.000 #001 RC=   0 Bind=0001 Type=SQLPSLO Len=0004 Data=000000044
```

```
     0.000     0.000 #001 RC=   0 Bind=0002 Type=SQLPSLO Len=0004 Data=000000001
     0.000     0.000 #001 RC=   0 Bind=0003 Type=SQLPBUF Len=0008 Data=PTPTEDIT
     0.000     0.000 #001 RC=   0 Bind=0004 Type=SQLPBUF Len=0008 Data=PTPTEDIT
     0.000     0.000 #001 RC=   0 Bind=0005 Type=SQLPSLO Len=0004 Data=000000104
     0.000     0.000 #001 RC=   0 Bind=0006 Type=SQLPSLO Len=0004 Data=000000101
     0.000     0.000 #001 RC=   0 Bind=0007 Type=SQLPSLO Len=0004 Data=000000010
     0.000     0.000 #001 RC=   0 Bind=0008 Type=0392 Len=0026 Data=1999-10-12-⇒
17.40.35.580000
     0.000     0.000 #001 RC=   0 Execute
     0.000     0.000 #001 RC=   0 Row Count=000000001
     0.000     0.000      RC=   0 GETSTMT Stmt=PTPLOGMS_S_GETMSG
     0.002     0.002 #003 RC=   0 COM=SELECT MESSAGE_TEXT FROM PS_MESSAGE_CATALOG⇒
 WHERE LANGUAGE_CD = :1 AND
 MESSAGE_SET_NBR = :2 AND MESSAGE_NBR = :3
     0.006     0.000 #003 RC=   0 SSB=0001 TYPE=SQLPBUF LEN=0100
     0.000     0.000 #003 RC=   0 Bind=0001 Type=SQLPBUF Len=0003 Data=ENG
     0.000     0.000 #003 RC=   0 Bind=0002 Type=SQLPSLO Len=0004 Data=000000104
     0.000     0.000 #003 RC=   0 Bind=0003 Type=SQLPSLO Len=0004 Data=000000101
     0.000     0.000 #003 RC=   0 Execute
     0.000     0.000 #003 RC=   0 Fetch
 > 1999-10-12-17.40.35.580000 INFO(104,101) PI(44) Program(PTPTEDIT)
 TSE Application Edits:  Begin Job.
     0.006     0.000 #003 RC=   0 Commit
     0.013     0.000 #003 RC=   0 Commit
     0.001     0.001      RC=   0 GETSTMT Stmt=PTPUSTAT_U_PRCRQSB
     0.004     0.003 #001 RC=   0 Prepare=UPDATE PSPRCSRQST SET RUNSTATUS = :1⇒
 ,MSGNUM = :2 ,MSGSET = :3 ,PRCSRTNCD = :4
 ,BEGINDTTM = CURRENT TIMESTAMP ,LASTUPDDTTM = CURRENT TIMESTAMP ,MSGPARM1 = :5⇒
 ,MSGPARM2 = :6 ,MSGPARM3 = :7 ,MSGPARM4
 = :8 ,MSGPARM5 = :9 ,CONTINUEJOB = :10 WHERE PRCSINSTANCE = :11
     0.008     0.000 #001 RC=   0 Bind=0001 Type=SQLPBUF Len=0001 Data=7
     0.000     0.000 #001 RC=   0 Bind=0002 Type=SQLPSLO Len=0004 Data=000000000
     0.000     0.000 #001 RC=   0 Bind=0003 Type=SQLPSLO Len=0004 Data=000000104
     0.000     0.000 #001 RC=   0 Bind=0004 Type=SQLPSSH Len=0002 Data=0000
     0.000     0.000 #001 RC=   0 Bind=0005 Type=SQLPBUF Len=0001 Data=
     0.000     0.000 #001 RC=   0 Bind=0006 Type=SQLPBUF Len=0001 Data=
     0.000     0.000 #001 RC=   0 Bind=0007 Type=SQLPBUF Len=0001 Data=
     0.000     0.000 #001 RC=   0 Bind=0008 Type=SQLPBUF Len=0001 Data=
     0.000     0.000 #001 RC=   0 Bind=0009 Type=SQLPBUF Len=0001 Data=
     0.000     0.000 #001 RC=   0 Bind=0010 Type=SQLPSSH Len=0002 Data=0000
     0.000     0.000 #001 RC=   0 Bind=0011 Type=SQLPSLO Len=0004 Data=000000044
     0.000     0.000 #001 RC= 100 Execute
     0.000     0.000 #001 RC=   0 Row Count=000000000
```

Some PeopleSoft COBOL programs utilize the stored statement technique of fetching and executing dynamic SQL. Programs fetch SQL statements—commonly known as stored SQL statements—from PS_SQLSTMT_TBL, then processes them using dynamic SQL (Prepares and Executes). Other COBOL programs are designed to generate their own SQL text inside the program, rather than fetching the SQL text from a table. This technique is sometimes referred to as "dynamic-dynamic," and is more commonly known as "dynamic statement" owing to its ability to generate dynamic SQL text and then to execute a dynamic SQL statement.

In the past, the timings of these dynamically generated statements have been recorded to the DYSQLLOG. In PeopleSoft 8 we have included the information on 'dynamic-dynamic' SQL statements into the PeopleSoft trace.

For example:

```
DYSQLLOG from previous versions of PeopleTools:

*********************************************************************
DYNAMIC SQL-STATEMENT (Len=  90)
UPDATE PS_TSE_EDITTBL SET TSE_EDIT_ERROR = ' ' WHERE (SETID = 'USA' AND
COMPANY = 'CCB')
Begin Time      End Time           Stmt Run Time           Total Run Time
08:59:42        08:59:42           0.00.00.000000          0.00.00.000000
*********************************************************************
```

Same dynamic SQL represented in the new PeopleTools trace for PeopleSoft appears as follows:

```
0.010     0.000 #001 RC=   0 DYNAMICSTMT Stmt=PTPEDIT_U_HE000
0.003     0.003 #001 RC=   0 Prepare=UPDATE PS_TSE_EDITTBL SET TSE_EDIT_ERROR = '⇒
 '
WHERE (SETID = 'USA' AND  COMPANY = 'CCB')
0.007     0.000 #001 RC=   0 Execute
0.007     0.000 #001 RC=   0 Row Count=000000008
```

Note. The PeopleSoft SQL trace can grow very large, so do your initial testing on smaller processes—for example, a small number of journals to EDIT or POST.

Enabling SQR Monitoring

For SQR programs, there is no report available that shows statement timings like the one provided by PTPSQLRT. However, you can generate a SQL script consisting of fully resolved DB2 UDB for OS/390 and z/OS statements by running the SQR with the -S option.

There are three areas that SQR monitoring can be introduced and four ways that it can be enabled.

Adding SQR Flag to SQRSAMP

The first area is in the JCL member SQRSAMP. The SQR flag can be added to the PARMLIB(SQRPARMS) directly if you plan to use sample JCL member JCLLIB(SQRSAMP).

```
DSN SYSTEM(DSND)
RUN PROG(SQR)  -
    PLAN(SQR84) -
    LIB('<PSHLQ>.SQR.UNICODE.LOAD') -
    PARMS('SP DSN/PT84 -FSQROUT –S -GPRINT=NO -ISI( -TBZ –
```

```
              -PRINTER:LP ')
END
```

Configuring Process Scheduler

The second area is within the Process Scheduler configuration, as follows:

To include the –S flag when configuring the PSOS390 Process Scheduler:

1. Initialize the PSADMIN program on Unix System Services to administer the Process Scheduler PSOS390.
2. You may either select Option 3 - Configure a Process Scheduler Server, or Option 6 - Edit a Process Scheduler Configuration File.
3. If you select Option 3, you would set the value for PSSQRFLAGS to -GPRINT=NO -TBZ -S.
4. If you select Option 6, you need to locate the related section in the file and change the PSSQRFLAGS value to -GRPINT=NO -TBZ -S.

Amending the Process Definition

The third area is from within the Process Definition. If you are running the SQR via the PSOS390 server, you have to append the –S flag to the process definition.

This can be accomplished by selecting PeopleTools, Process Scheduler Manager, Use, Process Definitions. Enter ‘SQR Report’ or ‘SQR Process’ for the process type and enter the SQR name (for example, XRFWIN in the screen below). Select the Override Options tab and then select Append from the drop down Parameter list and enter the –S. This appends the –S flag to the SQR list when you run the SQR. Output from the –S flag is directed to SYSOUT in the SHELSQR JCL, which is the output queue by default.

Sample output from SQR with the –S flag enabled:

```
Cursor Status:

Cursor #1:
   SQL = SET CURRENT PRECISION = 'DEC31'
Compiles = 1
Executes = 1
Rows     = 0

Cursor #2:
   SQL = select A.RECNAME, A.SQLTABLENAME  FROM PSRECDEFN A WHERE
         A.SQLTABLENAME <> ' ' AND A.SQLTABLENAME <> A.RECNAME ORDER BY
         RECNAME
Compiles = 1
Executes = 1
Rows     = 0

Cursor #3:
  SQL = select A.RECNAME, A.SQLTABLENAME  FROM  PSRECDEFN A WHERE A.RECTYPE
        AND A.RECNAME <> 'PSDUMMY' ORDER BY A.RECNAME
Compiles = 2
Executes = 1
Rows     = 1284
```

```
Cursor #4:
  SQL = select 'X' FROM SYSIBM.SYSTABLES B WHERE B.CREATOR = CURRENT SQLID
        AND B.NAME = ? AND B.TYPE = 'T'
Compiles = 2
Executes = 1284
Rows     = 1280
```

Note. The -S option produces output that shows the frequency in which all SQL statements are compiled and executed.

Associating PeopleSoft Operators with DB2 UDB Threads

DBAs who have worked with PeopleSoft in the past know that it is a daunting task to associate PeopleSoft operators to DB2 UDB distributed threads. This is because all threads are displayed (via DISPLAY THREAD) with the access ID as the ID. If my access ID is PSOFT then there is the potential to see hundreds or even thousands of DB2 UDB distributed threads all signed on with PSOFT. One workaround is to give each PeopleSoft operator a unique access ID. This method works fine for PeopleSoft two-tier users. But it involves extra overhead to create and maintain these new access IDs, and it doesn't work for PIA connections, because all PIA connections are displayed with the access ID that was used to boot the application server regardless of whether each operator ID has a unique access ID.

With enhancements made to DB2 Connect's sqleseti API Function, you can now associate the following items with each database connection or transaction:

- User ID.

  A 16-character field reserved for the client's ID. This ID entered in this field is not used for connecting in DB2 UDB; it's only intended for identification purposes.

- Workstation name.

  An 18-character field reserved for providing the workstation name for the client.

- Application name.

  A 32-character field reserved for the application name. Set to blanks for two tier connections. Set to the Domain Name for three tier connections. May appear to be database name but is really the Domain Name. Most customers set Domain Name equal to database name.

In PeopleSoft 8.4 these fields are populated with PeopleSoft-specific information.

Example #1: The following example shows a connection to a Peoplesoft/DB2 UDB database via two tier. I have signed onto Data Mover with Oprid = PTDMO. Notice that the Workstation Name of the client and PeopleSoft operator are displayed when using the DISPLAY THREAD command.

```
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
SERVER   RA *  3251 PSDMT        PSOFT    DISTSERV 00D0  3009
 V437-WORKSTATION=EPRESZ050499, USERID=PTDMO,
      APPLICATION NAME=*
```

Example #2: The following example shows a three-tier connection with DB2 UDB Client Monitoring enabled for the application server domain PT800T5. I am signing onto a Peoplesoft/DB2 UDB database with Oprid = VP1. The DISPLAY THREAD command shows the Workstation Name of the client, the PeopleSoft Operator as the user ID, and the application name is set to the Domain Name from the application server.

```
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
SERVER   RA * 13237 PSQCKSRV.exe PSOFT    DISTSERV 00D0  3097
 V437-WORKSTATION=EPRESZ050499, USERID=VP1,
      APPLICATION NAME= PT812
```

Note. DB2 UDB Client Monitoring is enabled by default for PeopleTools 8.44 and above. For previous releases, DB2 UDB Client Monitoring was automatically enabled for two-tier PeopleSoft connections. For three-tier connections, customers must enable the DB2 UDB Client Monitoring feature by overriding the EnableDBMonitoring flag in the App Server configuration (psappsrv.cfg) file. To confirm that DB2 UDB Client Monitoring is enabled, check for the following setting in psappsrv.cfg: [Database Options] EnableDBMonitoring=1

Running COBOL

This section provides an overview of COBOL API and Meta SQL and discusses how to:

- Run COBOL outside of Process Scheduler.
- Disable persistent cursors.

Understanding COBOL API and Meta SQL

PTPSQLRT is the COBOL API program called by application COBOL programs to prepare and execute dynamic SQL statements. The program fetches SQL statements—known as stored SQL statements—from PS_SQLSTMT_TBL, then processes them using dynamic SQL. (Prepares and Executes.) Except for PTPSQLRT, PeopleSoft application COBOL programs do not contain a direct DB2 UDB interface and therefore need only be compiled and link-edited.

Stored SQL statements contain META SQL statements mainly to support date and time functions, for example:

```
Select %currentdatetimeout from PSLOCK ;
```

PTPSQLRT resolves META SQL statements by calling PTPSQLGS which translates the META SQL function into DB2 UDB syntax. Stored statements are delivered in directory \SRC\CBL\BASE of the installation file server.

Running COBOL Outside of Process Scheduler

COBOL jobs may run outside process scheduler by specifying a 0 (a numeric zero) for the process instance, as shown in the fourth parameter below:

```
//PARMFILE DD  *
%OPRID%
%RUNID%
N
0
N
N
Y
```

Sample COBOL JCL is provided in HLQ.PPVVV.JCLLIB(CBLSAMP). Be aware that some application processes are designed to run only through Process Scheduler.

Disabling Persistent Cursors

The z/OS version of the COBOL API program called PTPSQLRT uses Cursor With Hold by default.

In PeopleSoft terminology, the field CURSOR_SW in PTPSQLRT is used to define Persistent Cursors (CURSOR-PERSISTENT) and Normal Cursors (CURSOR-NORMAL). CURSOR-PERSISTENT adds the WITH HOLD keyword to SQL selects in DB2 UDB. This maintains cursor position after a commit, so that repositioning (i.e. reopening and re-fetching) does not need to occur.

The DB2 UDB version of PTPSQLRT is shipped with Persistent Cursors enabled. If you don't want to use this feature, you can disable it by editing PTPSQLRT as follows:

To disable Cursor with Hold (i.e. Persistent Cursors):

1. Edit PTPSQLRT and do a "find" on 'CURSOR WITH HOLD' => f 'CURSOR WITH HOLD'.
2. For each of the 254 pairings of Cursor statements, remove the asterisk (*) from line that creates the cursor without the WITH HOLD option (column 7) and place it in column 7 of the line that creates the cursor with the WITH HOLD option above it. For example:

   Before:

```
EXEC SQL
           DECLARE CURSOR_01
           CURSOR WITH HOLD FOR SQLSTMT_01
  *       CURSOR FOR SQLSTMT_01
       END-EXEC
```

   After:

```
EXEC SQL
           DECLARE CURSOR_01
  *       CURSOR WITH HOLD FOR SQLSTMT_01
           CURSOR FOR SQLSTMT_01
       END-EXEC
```

Administering SQR for OS/390 and z/OS

This section provides an overview of SQR on z/OS and discusses how to:

- Run SQRs outside of Process Scheduler.
- Specify input and output files.
- Print SQRs.

Understanding SQR on z/OS

The SQR product is available for z/OS server platforms. The z/OS version of SQR is compatible with your existing SQR reports that you currently run from your client machines. The ability to run SQRs on the mainframe means a significant performance enhancement for SQR execution. All SQL is dynamic, therefore no precompiling is necessary for running SQRs.

You can execute SQRs on z/OS by either by using PeopleSoft Process Scheduler or submitting them as traditional z/OS jobs. Process Scheduler dynamically generates SQR JCL.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler*.

Be aware that some application processes are designed to run only through Process Scheduler.

SQR for z/OS is now delivered on the PeopleTools Installation CD. The DB2 UDB for OS/390 and z/OS Installation guide includes instructions for performing the installation of SQR on the z/OS server. SQR must be installed prior to running PeopleSoft SQRs on the z/OS server. Please allow at least 12 cylinders of 3390 DASD or equivalent disk space to complete the installation.

Running SQRs Outside of Process Scheduler

Three run time parameters are supplied to SQRs initiated by the Process Scheduler: PROCESS_INSTANCE, OPRID and RUN_CNTL_ID. (RUN_CNTL_ID is needed by FS SQRs only.) When submitted by Process Scheduler, run time parameters are obtained from a Process Scheduler table and dynamically inserted into the generated JCL.

For an SQR submitted as a traditional z/OS batch job, two run time parameters are required, but it is not necessary to pass any specific values. It is only necessary to include a blank line for each parameter in the JCL specified in the SYSIN DD. The appropriate segment of the JCL is shown below:

```
//SQRNAME  EXEC  SQRPROC,SQRID=SQRNAME
//SQR.SYSIN  DD  *




/*
```

Sample SQR JCL is provided in HLQ.PPVVV.JCLLIB(SQRSAMP).

Specifying Input and Output Files

Input and output files are required by SQR when using commands such as OPEN and NEW-REPORT. Remember to consult the appropriate Application User Guide for important application specific information concerning SQR for z/OS. For instance, Accounts Payable naming conventions used to build DD names are discussed in the PeopleSoft Payables PeopleBook.

There are two ways to specify input and output files in SQR for z/OS:

- Add DD statements to the JCL.

  This method allows you to maintain a common set of SQR that can execute on any operating system by requiring the modification of Process Scheduler Shell JCL and/or z/OS Batch JCL.

- Add a DSN style filename to the SQR.

  This method alleviates the need to modify JCL but creates z/OS operating system specific SQR.

Use the first method if your SQR should execute on any operating system. Use the second method if your SQR executes only on the z/OS operating system.

Adding DD Statements to the JCL

You may specify a file name for commands such as OPEN and NEW-REPORT using DOS or UNIX file naming conventions. The SQR for z/OS documentation states that SQR will use up to 8 alpha-numeric characters preceding the file extension as a DD name in JCL. This has been found to be untrue.

SQR for z/OS does not find 8 alpha-numeric characters as documented. To get around this problem, the FILEPREFIX and FILESUFFIX environment variables in $PSHLQ$.SQRINC(SETENV) are used when coding filenames in SQR. The example below shows that SETENV does not contain values for FILEPREFIX and FILESUFFIX when running on the z/OS operating system:

```
! File prefixes and suffix
!
#ifdef NT
#define FILEPREFIX C:\TEMP\
#define FILESUFFIX
#endif
!
#ifdef MVS
#define FILEPREFIX
#define FILESUFFIX
#endif
!
#ifdef UNIX
#define FILEPREFIX /usr/tmp/
#define FILESUFFIX
#endif
!
```

This coding standard enables DOS or UNIX file naming conventions to be used with the OPEN or NEW-REPORT SQR commands. The root portion of the file name is used as a JCL DD name. The Process Scheduler Shell JCL or z/OS Batch JCL must contain this DD name. Each file used for input or output requires a separate DD statement in the JCL.

In the following example, SQR for z/OS uses VIEWTBL as the DD name in JCL:

```
let $outputfile = '{FILEPREFIX}VIEWTBL{FILESUFFIX}'
open $outputfile as 1 for-writing record=132
```

The DD statement in the execution JCL requires the same DD name:

```
//VIEWTBL DD  DSN=&PSHLQ..OUTFILES(VIEWTBL),DISP=SHR
```

The data set name specified in the JCL may be either sequential or partitioned dataset. If the SQR requires multiple input or output files, you must add a separate DD statement to the JCL for each file.

Note. DD names must reference either sequential datasets or separate partitioned datasets due to the z/OS restriction on writing to more than one PDS member simultaneously.

While modifications to Process Scheduler Shell JCL or z/OS Batch JCL are required using this method, the resulting SQR is not operating system specific.

Adding a DSN Style Filename to the SQR

Filenames are preceded by DSN: (dataset name). For example:

```
OPEN 'DSN: $PSHLQ$.SQR.DAT(VIEWTBL)' FOR-READING RECORD=133
NEW-REPORT 'DSN: $PSHLQ$.SQR.DAT(VIEWTBL)'
```

Modifications to Process Scheduler Shell JCL or z/OS Batch JCL are not required when using the above option. However, this option results in operating system specific SQR that executes only on the z/OS operating system.

Printing SQRs

The SQR language supports a DECLARE PRINTER command so that reports can be directed to HPLASERJET and POSTSCRIPT type printers. However, if the printer is not mainframe-connected, the TYPE=LINEPRINTER option is required.

The TYPE=LINEPRINTER specification produces a basic text type report which can be redirected to a system printer. Normally, print output lines don't exceed 124 print positions.

SQRs containing the SETUP02 statement (refers to a member in the SQC library) allow print lines up to 177 positions. If SQROUT DD prints to SYSOUT, then supply a DCB override for SYSOUT and set the LRECL to 178, this is given in SQRSAMP JCL under your JCLLIB PDS. It would be a good practice to have DCB override with LRECL=178 for SQROUT DD, as this fits both landscape and portrait mode.

Formatted reports cannot be downloaded, then printed from a workstation connected printer. Only selected SQRs utilize the DECLARE PRINTER feature.

Note. A "formatted" report is one produced with the TYPE=HPLASERJET /POSTSCRIPT specification.

Many customers customize SQRs to tailor information to unique business requirements.

Another reason to customize SQRs is due to the way SQR formats reports when the TYPE=LINEPRINTER option is used. Column header and data field alignment may not be optimal, so modifications may be required.

Updating Statistics

This section provides an overview of %UpdateStats and discusses how to:

- Set up the IBM stored procedure DSNUTILS.
- Install the database following the enhanced installation path.
- Update system tables with database and tablespace information.
- Activate %UpdateStats.

Understanding %UpdateStats

The meta-SQL %UpdateStats is introduced in PeopleSoft 8 to allow an Application Engine and COBOL programs to update statistics in the DB2 UDB catalog table. Updating the statistics in the DB2 UDB catalog table is particularly helpful for the overall performance of Application Engine programs that heavily use PeopleSoft temporary tables.

Often, these tables are delivered empty or the Application Engine program contains code to purge the content of these tables prior to termination. So even if you periodically perform a REORG or RUNSTATS against all the tablespaces in a PeopleSoft database, the DB2 UDB catalog does not reflect accurate statistical information on these temporary tables. The %UpdateStats meta-SQL was specifically written to get around this issue.

To fully implement the %UpdateStats functionality in DB2 UDB in your environment, you need to be aware of the following key items:

- Setting up the DB2 z/OS stored procedure: DSNUTILS.
- Installing the database following the Enhanced Installation Path.
- Updating the PeopleSoft System Tables with Database and Tablespace Information
- Activating the %UpdateStats in Application Engine.

Setting Up the IBM System Stored Procedure: DSNUTILS

DSNUTILS is required to enable the %UpdateStats meta-SQL function on DB2 z/OS.

The DSNUTILS procedure has been integrated with Application Engine specifically to invoke the Runstats utility; hence DSNUTILS is required if you intend to use the %UpdateStats meta-SQL function.

Please refer to your IBM manuals for more information on enabling the DSNUTILS stored procedure for DB2 z/OS.

The %UpdateStats meta-SQL function can be enabled and disabled through the Process Scheduler configuration. If the command is disabled, the Application Engine and COBOL programs ignore any %UpdateStats coded within the program, and the Runstats utility will not execute.

Note. In PeopleTools 8.44 and beyond, the %UpdateStats meta-SQL is enabled by default for both z/OS and Windows NT/2000 Process Schedulers.

See *PeopleTools 8.45 Installation for DB2 UDB for OS/390 and z/OS.*

Note. A document from IBM entitled “Enabling the DSNUTILS Stored Procedure for DB2 for OS/390” is available on the PeopleSoft Customer Connection website. This document outlines the minimum requirements to run Workload Manager in goal mode which is required by the DSNUTILS stored procedure.

See Also

IBM DB2 for OS/390 and z/OS documentation.

Installing the Database Following the Enhanced Installation Path

The lowest level of granularity for running RUNSTATS on DB2 z/OS is at the tablespace level. The %UpdateStats function processes at the table level. For this reason, it is critical to the success of implementing the %UpdateStats functionality that those temporary tables which are the object of the %UpdateStats meta-SQL are placed in their own, unique tablespaces, rather than a shared tablespace. The performance of the Runstats utility itself can be negatively affected if these tables are not segregated. There is also risk of invalidating the catalog statistics of other objects that reside in the shared tablespace.

To assist with this process, we deliver two Installation paths for our System and Demo databases. The “Traditional” installation path follows our pre-PeopleSoft 8 strategy of combining multiple tables into a single tablespace. The “Enhanced” installation path has segregated the PeopleSoft temporary tables into separate tablespaces.

See *PeopleTools 8.45 Installation for DB2 UDB for OS/390 and z/OS, "Creating a Database."*

If you plan to use the %UpdateStats functionality, it is critical that you use the Enhanced Installation path for optimal performance of the %UpdateStats function and the Runstats utility.

Note. The %UpdateStats meta-SQL function is only intended to be invoked with PeopleSoft temporary tables. It is not intended to be used to update catalog statistics for permanent Application or PeopleTools tables.

Updating System Tables with Database and Tablespace Information

When issuing %UpdateStats meta-SQL in your program, you specify the temporary table on which you intend to have the statistics updated. For PeopleTools releases prior to 8.44, code within PeopleTools determines the tablespace and database to which the table belongs and passes these values to the DSNUTILS stored procedure. Database and tablespace name is retrieved from the PeopleTools meta data. Therefore, it is imperative that these tables reflect accurate information as contained in the DB2 UDB catalog.

Running the following SQRs ensures that the PeopleTools tables are in sync with the DB2 UDB system catalog.

SQR Program	Purpose
SETSPACE.SQR	Extracts the database/tablespace values from the SYSIBM.SYSTABLES and updates the PSRECTBLSPC table with this information. The SQR also inserts valid database/tablespace combinations into PSTBLSPCCAT
SETTMPIN.SQR	Inserts Temporary Table instance information into PSRECTBLSPC to provide values necessary for processing Runstats on the instance.

Note. DB2 RUNSTATS is run at the tablespace level. From a performance perspective, it is recommended that you move tables that are the object of the %UpdateStats to a separate tablespace.

For PeopleTools 8.44 and later, it is not mandatory to run SETSPACE or SETTMPIN to use the %UpdateStats meta-SQL function because %UpdateStats retrieves the correct database and tablespace name directly from the DB2 UDB catalog. You should, however, still run SETSPACE and SETTMPIN to keep the PeopleTools metadata synchronized with the DB2 UDB Catalog.

See *IBM's Installation Guide for DB2 UDB for OS390 and z/OS.*

Activating %UpdateStats

%UpdateStats can be disabled by setting the DBFLAGS switch.

This switch has two valid values:

- 0 – enable the %UpdateStats
- 1 – disable use of the %UpdateStats

For PeopleTools 8.44 and later, the DBFLAGS default for NT/2000, OS/390, and z/OS Process Schedulers is 0 (%UpdateStats enabled).

Enabling/Disabling %UpdateStats for Batch Processing

To enable/disable %UpdateStats for batch processing:

1. Initialize the PSADMIN program on Unix System Services to administer the Process Scheduler PSOS390.
2. Select either Option 3, Configure a Process Scheduler Server, or Option 6, Edit a Process Scheduler Configuration File.
3. If you select Option 3, set the value for DbFlags to 0 to enable and 1 to disable %UpdateStats.
4. If you select Option 6, locate the related section in the file and change DbFlags to 0 to enable and 1 to disable %UpdateStats.
5. To fully enable %UpdateStats for COBOL to run on the mainframe (Server PSOS390), you need to modify the program PSPTSQLRT. Note that several lines are delivered commented out in the program,

```
--
      *    ELSE
      *        PERFORM VX000-EXECUTE-RUNSTATS
                     END-IF

--.
--.
      *    EXEC SQL
      *      CALL DSNUTILS(:DSNUTIL-WK.UID, :DSNUTIL-WK.RESTART,
      *                    :DSNUTIL-WK.UTSTMT,
      *                    :RETCODE, :DSNUTIL-WK.UTILITY,
      *                    :DSNUTIL-WK.RECDSN, :DSNUTIL-WK.RECDEVT,
      *                    :DSNUTIL-WK.RECSPACE,
      *                    :DSNUTIL-WK.DISCDSN, :DSNUTIL-WK.DISCDEVT,
      *                    :DSNUTIL-WK.DISCSPACE,
      *                    :DSNUTIL-WK.PNCHDSN, :DSNUTIL-WK.PNCHDEVT,
      *                    :DSNUTIL-WK.PNCHSPACE,
      *                    :DSNUTIL-WK.COPYDSN1, :DSNUTIL-WK.COPYDEVT1,
      *                    :DSNUTIL-WK.COPYSPACE1,
      *                    :DSNUTIL-WK.COPYDSN2, :DSNUTIL-WK.COPYDEVT2,
      *                    :DSNUTIL-WK.COPYSPACE2,
      *                    :DSNUTIL-WK.RCPYDSN1, :DSNUTIL-WK.RCPYDEVT1,
      *                    :DSNUTIL-WK.RCPYSPACE1,
      *                    :DSNUTIL-WK.RCPYDSN2, :DSNUTIL-WK.RCPYDEVT2,
      *                    :DSNUTIL-WK.RCPYSPACE2,
      *                    :DSNUTIL-WK.WORKDSN1, :DSNUTIL-WK.WORKDEVT1,
      *                    :DSNUTIL-WK.WORKSPACE1,
      *                    :DSNUTIL-WK.WORKDSN2, :DSNUTIL-WK.WORKDEVT2,
      *                    :DSNUTIL-WK.WORKSPACE2,
      *                    :DSNUTIL-WK.MAPDSN, :DSNUTIL-WK.MAPDEVT,
      *                    :DSNUTIL-WK.MAPSPACE,
      *                    :DSNUTIL-WK.ERRDSN, :DSNUTIL-WK.ERRDEVT,
      *                    :DSNUTIL-WK.ERRSPACE,
      *                    :DSNUTIL-WK.FILTERDSN, :DSNUTIL-WK.FILTERDEVT,
      *                    :DSNUTIL-WK.FILTERSPACE )
      *    END-EXEC.
--
```

6. Uncomment the lines noted above so they are activated in the program code.
7. Compile the program PTPSQLRT by submitting the two JCL members found in $PSHLQ.JCLLIB:
   - PSCOBDA
   - PSCOBDE
8. Determine whether you want to bind or rebind two DB2 UDB plans for PeopleSoft. Add the following line to the Bind Parameter list in the appropriate JCL members noted below, before submitting them:

   PKLIST (DSNUTILS.*)
9. For first time Binding modify the following two members in $PSHLQ.JCLLIB:
   - BINDAADD
   - BINDEADD
10. For rebinding an existing plan, modify the following two members in $PSHLQ.JCLLIB
    - BINDAREP
    - BINDEREP

Setting the Number of Temporary Tables

Normally you may leave the number of temporary tables set to the default established at installation. You may need to change this setting for optimal performance, depending on various aspects of your implementation, including account transaction volumes, benchmark numbers for the current hardware and database platform, as well as your service-level requirements. Use the following procedure if you need to adjust the number of temporary tables to improve performance in your implementation.

To set the number of temporary tables:

1. Select PeopleTools, Utilities, Administration, PeopleTools Options.
2. Set the Temp Table Instances (Total) and Temp Table Instances (Online) fields to the desired settings.

   Note. Temp Table Instances (Total) should always be set to the same values as Temp Table Instances (Online), unless you have been instructed otherwise in the application documentation.
3. Scroll to the bottom of the page and select the Save icon to save the newly edited PeopleTools options.

   Note. The total number of instance generated consists of the allocations specified on the PeopleTools Options panel plus the allocations specified on each individual Application Engine program. (To modify these allocations, open an Application Engine program in Application Designer, open the Properties dialog box for the object, and click the Temporary Tables tab.)
4. Recreate all temp tables in your database.

   See *PeopleTools 8.45 Installation for DB2 UDB for OS/390 and z/OS, "Creating a Database."*

Warning! If you change the number of online temporary table instances as described above, it is critical that you recreate all temporary tables in your database, particularly if you are increasing the number of instances. The parameter above is global to all temporary tables and is used by all on-line processes to determine the number of temporary table instances that should be available to a given process. If you don't recreate all temporary tables, a process may try to use an instance that has not been created on the database, and will subsequently fail.

APPENDIX C

Administering PeopleSoft Databases on DB2 UDB for Linux, Unix, and Windows

This appendix provides an overview of administration on DB2 UDB for Linux, Unix, and Windows, and discusses:

- Instances.
- Configuration parameters.
- Tablespaces.
- Client database catalog.
- Meta-SQL %TruncateTable().
- DB2 UDB for Linux, Unix, and Windows administration.
- Checklists and troubleshooting.

Note. PeopleSoft supports a number of versions of UNIX and Linux. Throughout this appendix, the word UNIX refers to all UNIX-like operating systems, including Linux.

For the sake of brevity, this appendix sometimes refers to DB2 UDB for Linux, Unix, and Windows as *DB2 LUW.*

Understanding Administration on DB2 UDB for Linux, Unix, and Windows

A PeopleSoft DB2 for UNIX database must be configured, monitored, and tuned to achieve optimum performance. In this section, we offer concepts, procedures, and tips to help you plan and implement the installation of PeopleSoft system and demonstration databases.

Recognizing that many DB2 LUW database administrators have DB2 z/OS backgrounds, this documentation includes references and comparisons to DB2 z/OS to help bridge understanding of concepts and procedures.

For UNIX systems, DB2 UDB publications can be accessed online from AIX using the system command *db2help;* this will bring up the browser and display the DB2 UDB information in HTML format.

Note. You need to install the supported browser and the DB2 UDB HTML information for the chosen language (locale) before running the *db2help* command. By default, the HTML files are copied from the CD-ROM to the hard disk in compressed form; you will have to decompress it using the *db2insthtml* command under the DB2 UDB-installed directory. In AIX, it is similar to /usr/lpp/db2_06_01/doc/db2insthtml. In Windows NT, the above steps are not needed because all the HTML files are uncompressed during normal DB2 UDB installation.

See Also

DB2 Administration Guide

DB2 System Monitor Guide and Reference

Instances

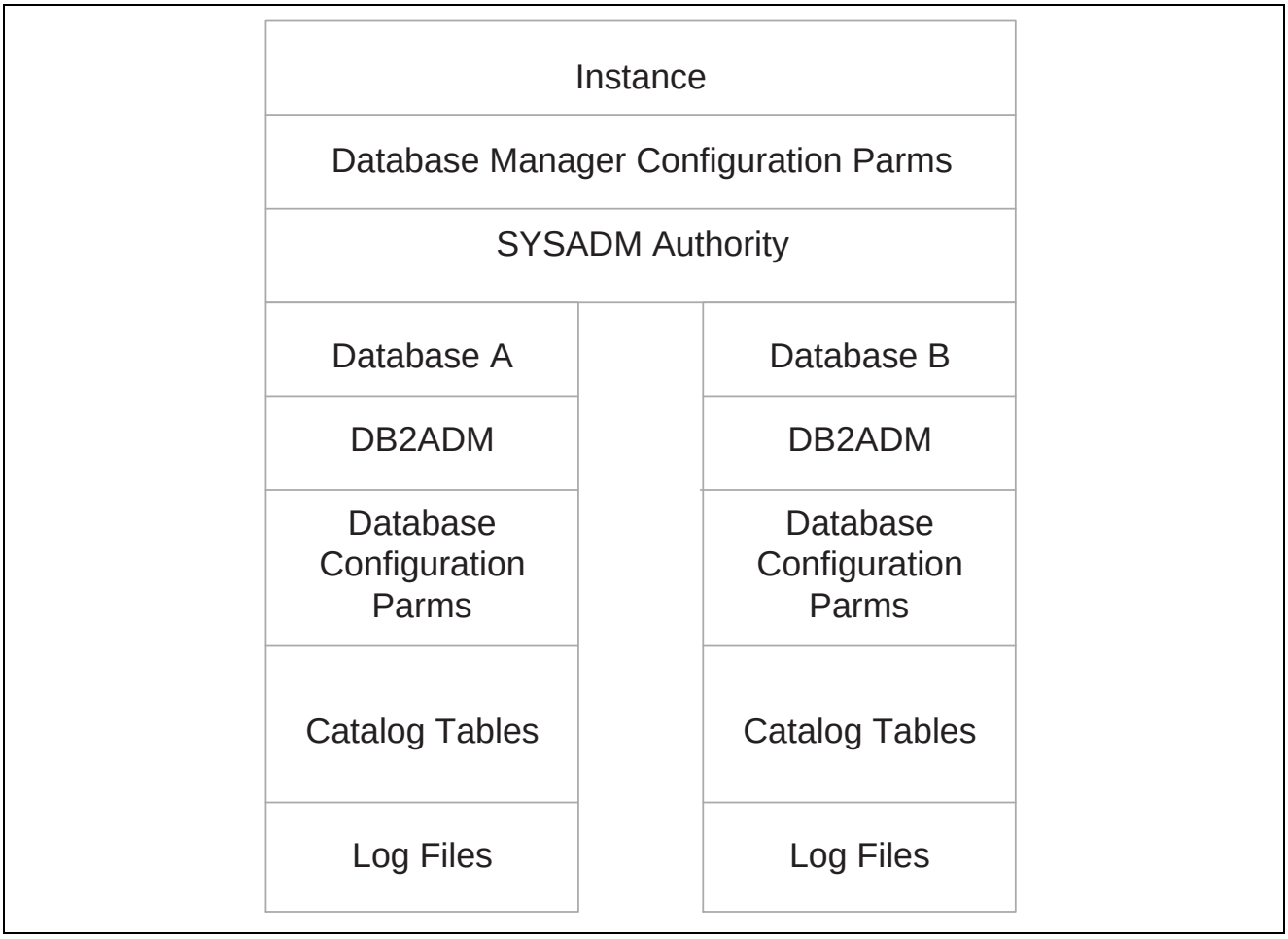
This section discusses:

- Definition of instance.
- SYSADM authority and security.
- Instances and Connectivity.
- Other considerations.

Definition of Instance

Operating System administrators (for UNIX logged in as root, for NT logged on as Administrator) may create one or more DB2 LUW instances to support their PeopleSoft environment. If you only have one computer to house both production and development database, PeopleSoft recommends that you create at least two DB2 UDB instances, one for your development database(s) and one for production. If you have more than one computer for the PeopleSoft environment use, PeopleSoft further recommends that the production instance be created on a separate machine for performance and security reasons.

A conceptual view of a DB2 LUW instance is shown in the graphic below. Each instance is a collection of databases sharing the same DB2 UDB engine and set of configuration tuning parameters called “database manager parameters.” These parameters control a variety of system resources such as communication buffer sizes, TCP/IP service name, and memory allocations. SYSADM authority controls all databases in an instance. DB2ADM controls the resources within a particular database.



One instance housing two databases

Each database within an instance is, to a large extent, self-contained, having its own set of system catalog tables, configuration tuning parameters, tablespaces, and log files.

Each PeopleSoft application is installed entirely into a single DB2 LUW database. To simplify administration, it is recommended that you created all the PeopleSoft nonproduction databases (such as Upgrade, Demo and Development) within one DB2 LUW instance. Setting up the production database in a separate instance by itself will provide you with greater flexibility in administration.

SYSADM Authority and Security

In DB2 LUW, SYSADM owns all databases in an instance. For this reason, to secure access to the production database, consider maintaining separate production and development instances. In this way, you can restrict SYSADM in production and be less restrictive in development.

If your site uses a single instance standard, you must restrict SYSADM authority—keeping in mind the additional burden this places on your DBA to support development and production environments.

Note. Administrators with DB2 UDB for OS/390 and z/OS experience should note the difference between DB2 z/OS and DB2 LUW in the way SYSADMs are created. In DB2 z/OS, an "Install SYSADM" is specified during DB2 UDB installation and other SYSADMs are granted using an SQL Grant SYSADM statement.

In DB2 LUW, the instance owner is the de facto "Install SYSADM" and other SYSADMs are created by assigning its group ID to the same primary group as the instance owner.

Instances and Connectivity

To create an instance, you can use the command db2icrt. Once the instance is created, you need to assign different TCP/IP port numbers for the respective DB2 UDB instance. For DB2 LUW Version 7, each instance should have its unique pair of TCP/IP ports, one for regular communication and one for interrupt use.

To register the pair of TCP/IP ports, you have to edit the services file. Both the UNIX Server (/etc/services) and Windows (\windows\system32\drivers\etc\services) must specify the following:

```
db2dudb 50000/tcp#DB2 Client Application Enabler-Dev
db2dudbi 50001/tcp#DB2 Client Application Enabler Interrupt
db2pudb 50010/tcp#DB2 Client Application Enabler-Prod
db2pudbi 50011/tcp#DB2 Client Application Enabler-Interrupt
```

The names db2dudb and db2pudb are user-defined. The interrupt ports are obselete in DB2 LUW Version 8.

Each instance requires a SERVICENAME, which points to a unique entry in the service file.

On the DB2 LUW Server, update the Database Manager SVCENAME Configuration Parameter:

```
db2 update dbm cfg using SVCENAME db2dudb
```

If you created a second instance, you'd have to update the Database Manager SVCENAME Configuration Parameter on that instance using another service name:

```
db2 update dbm cfg using SVCENAME db2pudb
```

Other Considerations

The following are other considerations related to instances:

- For AIX, DB2 LUW Version 7 products are installed under the directory /usr/lpp/db2_*version_release*. DB2 LUW Version 8 products are installed under the directory /usr/opt/db2_*version_release*. For Solaris DB2 LUW products are installed in the /opt/IBM/db2/V*v.r* where *v* is the version, and *r* is the release.
- sqllib is the root of the DB2 UDB directories created for each DB2 UDB instance. Enter `echo $DB2INSTANCE` to find the currently attached DB2 UDB instance.
- Use db2ilist to list all the instances configured for this machine.
- DB2 UDB system commands begin with the prefix db2 (for example, db2start and db2stop for starting and stopping DB2 UDB) and are entered at the command window.
- DB2 Command Line Processor (CLP) is a DB2 UDB interface for executing utilities, updating the system configuration, executing SQL, and for getting online help. CLP is the functional equivalent of SPUFI in the DB2 z/OS environment.
  - Type "db2 ?" at the command prompt for general CLP syntax.
  - Use db2 "? <command>" to get help on a particular CLP command.
  - If you need to execute several lengthy SQLs, you can create a text file which contains all the SQLs and invoke the db2 CLP with the –f switch, such as *db2 –tvf job.txt*.

- As an alternative to the Command Line Processor to issue SQL commands, IBM provides the Control Center, a graphical interface for database administrative tasks. For DB2 LUW v6.1 and later versions, the Control Center is a Java application or can be run as a Java applet within a browser. To start the Control Center, type the command *db2cc*.

Configuration Parameters

This section discusses:

- Definition of configuration parameters.
- Useful configuration commands.
- Parameters overview.

Definition of Configuration Parameters

Database manager configuration parameters are those which apply for all databases managed by the current instance. You can update database manager configuration parameters using DB2 CLP or Control Center. New database manager configuration parameters take effect after DB2 UDB is stopped and restarted using DB2 UDB commands, *db2stop* and *db2start*, successively.

Most database configuration parameter changes take effect immediately. Some take effect only after all current users disconnect from a database, or after you forcefully disconnect them with the db2 force application all command, then execute the db2 terminate command to flush the database's directory cache and remove the db2bp (backend process).

Useful Configuration Commands

Useful DB2 CLP commands for Database Manager configuration:

- get dbm cfg.

  Lists the current setting of database manager config parms.

- update dbm cfg using <parm-name> <new-value>.

  Updates the configuration parameter <parm-name>. For example:

```
update dbm cfg using numdb 4
```

Useful DB2 CLP commands for Database configuration:

- get db cfg for <db-name>.

  Lists the database config parms for <db-name> database.

- update db cfg for <db-name> using <parm-name> <new-value>.

  Updates the parm <parm-name> for database <db-name>. For example:

```
db2 update db cfg for hr800dmo using  locktimeout  60
```

Parameters Overview

Following is an overview of the more important configuration parameters with tips for tuning them. Fine tuning these parameters for optimal performance gain on your system requires careful benchmarking techniques.

- BUFFPAGE

  This database-level parameter controls the database buffer pool (cache) size. It is allocated for the respective database and shared among all the connected client applications. The rule for setting this parameter is: the more the better. Too high a value can cause the system to page (check with vmstat). Database system monitor can also give you hints about the "hit-ratio" and I/O elapsed times, that can help you set a proper value. This is a database level parm. The maximum BUFFPAGE for each database without using ESTORE is about 1.5 gigabytes. However, the total BUFFPAGEs allocated for all the databases running on the same computer should not exceed 75% of the amount of real memory available for the computer.

Note. The BUFFPAGE is related to the SYSCAT.BUFFERPOOLS table entry. DB2 LUW has the capability to attach individual buffer pool for each tablespaces. For simplicity, the user can also attach a generic buffer pool named IBMDEFAULTBP for all the tablespaces use. To set up the size of the IBMDEFAULTBP to use the DBM configuration parameter, BUFFPAGE, the user can run the given script ALTRDB.SQL or the following command:

```
db2 alter bufferpool   ibmdefaultbp size -1
```

- SORTHEAP

  This database-level parameter is the capacity of in-memory sort. If a sort size exceeds this, the sort has to be written to a temp tables and merged. Make sure this and the SHEAPTHRES parameter (see next item) are large before any large index-creation, since it presorts the data. This parameter is also important for batch jobs, such as payroll, that do large sorts.

- SHEAPTHRES

  This is a database manager level parm. If all the sort-heaps in the system at any given time exceed this value, the available memory to any further sorts is reduced. Make sure it’s greater than SORTHEAP*N, where N is the number of sorts you expect to occur at any given time.

- LOCKLIST

  This database-level parameter, along with MAXLOCKS, determines the maximum memory used for database locks. When this runs out (due to a large number of row locks acquired by transactions), lock escalation will occur to minimize the lock usage. Many Row locks will be escalated to one Table lock. The drawback of Table level lock is the reduction of concurrency among multiple applications, which need access to the same tables. Check the database event monitor for lock escalations.

- NUM_IOSERVERS

  This database-level parameter stores the Number of Processes/threads used for prefetch/parallel I/O, as well as for backup/restore. A good value is the number of physical disk drives uses to house the database plus 2.

- NUM_IOCLEANERS

  This database-level parameter stores the number of page-cleaners that do "write-behind" of dirty pages. The recommended value for this parameters is to match the number of CPUs inside the computer.

- RQRIOBLK

  This database-level parameter stores the size of the cache used for row blocking for remote, cursor-based applications. Rows are placed in this cache in anticipation of their use and retrieved from here for the next FETCH request.

- ASLHEAP

Size of the cache used for row blocking for local applications.

- LOGPRIMARY

  This database-level parameter controls the number of DB2 UDB Log files that will be pre-allocated for regular database transaction logging use. Setting this number to a high value will minimize the need to allocate log files on demand, which will improve runtime performance.

Note. You should separate log files on separate disks from the actual database data. You can use the database configuration parameter NEWLOGPATH to do this.

- DFT_QUERYOPT

  This database-level parameter determines the query optimization class used for the SQL compilation. The higher the level represents a more detailed study of the potential access path. Since PeopleSoft applications comprises dynamic SQLs only, setting this parameter to a high value will affect the compilation time for all SQLs. The recommended value for this parameter is the default value 5.

Note. For SQL that is complex and can benefit from the use of higher optimization class, users can alter the optimization class for that SQL alone with the following SQL construct:

```
SET CURRENT QUERY OPTIMIZATION = 7
[ complex SQL ]
SET CURRENT QUERY OPTIMIZATION = 5
```

Most of these memory-related parameters are allocated out of one of the many heaps. You may need to adjust the heap parameters accordingly.

As a point of reference, PeopleSoft used the following database parameter values when running HRMS/Payroll benchmark tests processing a 100,000+ employee payroll: BUFFPAGE (125000); SORTHEAP (9000).

Tablespaces

This section discusses:

- DDL scripts.
- Using the PeopleSoft DMS tablespace DDL.
- DMS tablespaces: Cooked or raw.
- System catalog tablespace and other initial tablespaces.
- Capacity planning.

DDL Scripts

PeopleSoft provides DDL scripts to create a database, and set database manager and database tuning parameters. These scripts are on the PeopleSoft installation file server in the \scripts directory. Run the following scripts:

```
\scripts\createdb.sql -- creates DB2 LUW database.
.
\scripts\xxddldms.sql -- creates DMS (Data Managed Storage) tablespaces
```

Where xx is the product identifier, such as HR for PeopleSoft HRMS or FS for PeopleSoft Financials and Supply Chain Management.

Using the PeopleSoft DMS Tablespace DDL

Create all tables and indexes in Data Managed Storage (DMS) tablespaces using PeopleSoft standard tablespace names as described in the installation guide. This storage option, as oppose to System Managed Storage (SMS), is appropriate for a database that you plan to change and grow. DMS is appropriate for a system test or production database.

Note. DROPPED TABLE RECOVERY feature is turned off in the xxddldms.sql script to avoid performance issue when dropping large number of tables. This feature can be turn on again with ALTER TABLESPACE command.

Here are some installation guidelines for manually creating your PeopleSoft database and tablespaces:

- On the database server, edit and run CREATEDB.SQL to create a database and default tablespace USERSPACE1. Note that this script assumes you will use Circular Logging; if archival logging is desired, you must make the necessary changes.
- On the database server, edit the DMS script /sql/hrddldms.sql. Instructions for editing this file are contained inside the file. This script creates all the PeopleSoft standard tablespaces.
- In Windows, use Data Mover to create and populate tables and indexes. In Data Mover, the command line below—if it exists in the Data Mover script—should either be removed or commented out (disabled) in the Data Mover script (the ‘;’ in position 1 disables the command):

```
; set space * as USERSPACE1 ;
```

Note. Disabling the above command causes Data Mover to use PeopleSoft’s standard tablespace grouping strategy.

See Also

PeopleTools 8.45 Installation for DB2 UDB for UNIX/NT, “Preparing for Installation”

DMS Tablespaces: Cooked or Raw

DMS tablespaces may be created as either COOKED Files System or RAW Storage Devices. PeopleSoft provides DDL script /sql/hrddldms.sql to support DMS COOKED Files System.

PeopleSoft does not provide a tablespace script to support the Raw device, but you can create the RAW device with the proper Operating System command and the following DB2 UDB command:

```
CREATE TABLESPACE PSAPP MANAGED BY DATABASE USING
        (device ’/dev/data1_lv’ 20000)
```

In AIX, the COOKED File System refers to the Journal File System (JFS). In NT, the COOKED Files System refers to NTFS.

Note. There is a roughly 5-10% performance gain on RAW device over COOKED file system on tablespaces which are frequently being updated. However, it is generally much easier to administer a COOKED file system than a RAW device.

System Catalog Tablespace and Other Initial Tablespaces

For system test and production databases, PeopleSoft recommends that you consider tailoring the Create Database statement to override the DB2 LUW default tablespace definitions for SYSCATSPACE and TEMPSPACE1. An example of this is provided below, where CATALOG Tablespace defines the SYSCATSPACE and TEMPORARY Tablespace defines TEMPSPACE1:

```
CREATE DATABASE <db2-database-name>  ON  <dir-name|drive> COLLATE USING IDENTITY  \
  CATALOG TABLESPACE MANAGED BY SYSTEM USING
   (‘/<cat-dir-name>‘ )
   EXTENTSIZE 16  PREFETCHSIZE 32
  TEMPORARY TABLESPACE MANAGED BY SYSTEM USING
   (‘/temp-dir-name’)
           EXTENTSIZE 8
```

Note. The above tablespaces may be defined as DMS tablespaces. If you omit these tablespace definitions, DB2 LUW will create these tablespaces in the file system directory denoted by *dir-name*.

Capacity Planning

If significant growth is expected as you conduct functional, system, and volume testing, you must plan to accommodate such growth when using DMS tablespaces. High growth tables will be created in tablespace xxLARGE, where xx represents an application product code, such as TLLARGE for Time and Labor, HRLARGE for HRMS, and so on.

Client Database Catalog

When cataloging databases on a client machine, always use AUTHENTICATION clause and match the authentication algorithm with the one specified on the server by the database manager parameter (AUTHENTICATION). For example:
`db2 catalog database <database-name> at node <node-name> AUTHENTICATION SERVER` This will avoid additional network traffic between client and server generated to resolve the authentication algorithm discrepancy.

Meta-SQL %TruncateTable()

In DB2 LUW, there’s no SQL implementation of a Truncate Table command as the one found in Oracle. PeopleSoft has implemented a DB2 UDB utility to achieve the same effect as the Truncate Table command. This utility is available through the PeopleCode function %TruncateTable().

You might wish to disable this meta-SQL function because of the performance overhead incurred by bufferpool flushing. The effect of bufferpool flushing is that when you truncate large tables using the DB2 LUW API, the process can run much longer than a SQL “Delete From” clause. If you’re experiencing this problem, a workaround is now available to convert %TruncateTable into a SQL “Delete From” clause.

To enable this workaround, there is a new flag setting in psprcs.cfg and psappsrv.cfg. If DbFlags is a bitmap value and if it contains the value of *2*, then SQL is used rather than the DB2 UDB API to set the table to zero rows. The default value for DbFlags is *zero*.

The following is an example:

DbFlags=3 will enable the workaround, since 3=2+1.

DbFlags=1 doesn’t enable the workaround and the Truncate is done similar to the Oracle’s Truncate command.

Handling Errors

During the execution of the %TruncateTable() meta-SQL, error information is written to a disk file. The location of this disk file varies depending on which platform type is used.

Windows 2000

If you are running %TruncateTable on Windows 2000, then the directory name format is "%TEMP%\PS \DB2Truncate\PS_TruncateLogFile_<pid id>.txt” , where <pid id> is a variable depending on the process ID.

UNIX

If you are running %TruncateTable on UNIX, then the directory name format is “$PS_HOME/log /DB2Truncate/PS_TruncateLogFile_<pid id>.txt”, where <pid id> is a variable depending on the process ID.

In most cases, error files might be created under the following circumstances:

- %TruncateTable(Table_name), where Table_name doesn’t exist.
- Internal errors in DB2 UDB.

DB2 UDB for Linux, Unix, and Windows Administration

This section discusses:

- Updating statistics.
- Performing queries on a Windows client.
- Object restrictions.
- Administrative tools.
- Connectivity using ODBC/CLI.

Updating Statistics

We recommend that you update the database statistics on a periodic basis, typically weekly, to account for ongoing data changes. You do this by running runstats for tables and indexes in the database. This allows DB2 UDB's cost based optimizer to generate efficient access plans for your stored and dynamic SQL statements. Using the SHRLEVEL CHANGE keywords together with the runstats command will enable the application to access the table while the statistics are being computed. An example of the command is shown below:

```
db2 runstats on table sysibm.systables with distribution and indexes all SHRLEVEL⇒
 CHANGE
```

Runstats can be executed from the Database Control Center or DB2 CLP. Type "db2? runstats" for more information.

PeopleSoft provides an SQR program, RUNSTATS.SQR, to execute runstats on all your System and PeopleSoft tables. This script is located in the database server's /SQR directory, and can be executed using the instructions found in installation guide. If desired, for efficiency's sake, you can modify this script to limit running the runstats command against only those tables that experience high growth or high update. To identify such tables, modify RUNSTATS.SQR to join tables to SYSCAT.SYSTABLES and only select those tables belonging to tablespace xxLARGE.

Use *explain* to determine the access path chosen by the DB2 UDB optimizer. You can either use the Visual Explain utility or the db2expln tool to get access path information.

See Also

PeopleTools 8.45 Installation for DB2 UDB for UNIX/NT, "Creating a Database"

Performing Queries on a Windows Client

Query capability on Windows clients can be accomplished via multiple products:

- IBM's DB2 Connect provides connectivity to a DB2 LUW database server (and other DB2 UDB Family servers) as well as SQL support.
- DB2 UDB "Command Window" or the graphical "Command Center". SQL issued from the Command Center can be stored as Scripts and then be retrieved for later use via the Script Center.
- Third-party vendor tools such as Business Object (Forest and Trees), Information Advantage, and so forth.
- Use Lotus Approach 97 with DB2 LUW. Full GUI interface or most ODBC Query products, such as MS Query.

Object Restrictions

PeopleSoft applications contain many table and index objects. The number of objects in a DB2 LUW database does not pose a problem as it would in DB2 z/OS.

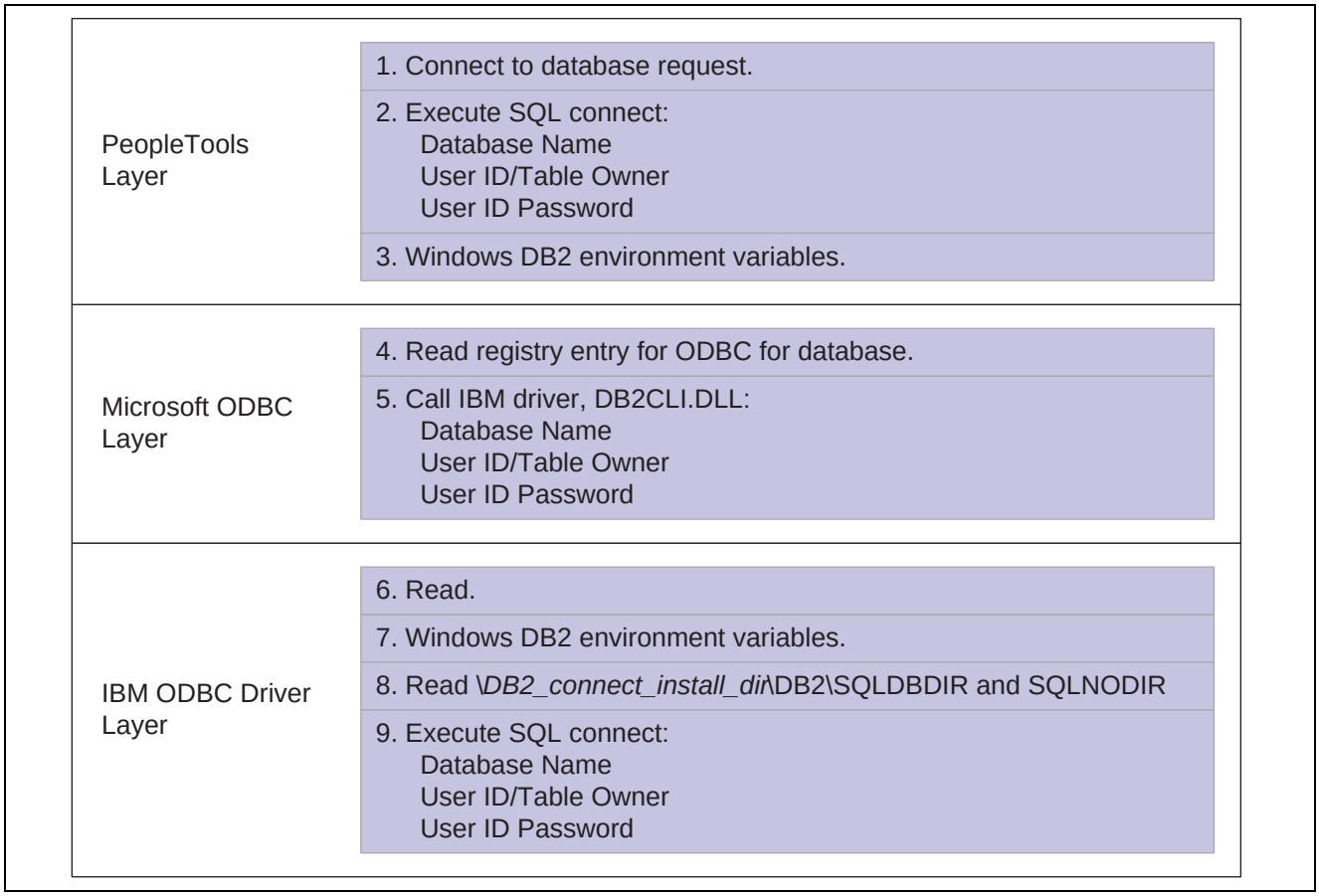
Unlike DB2 z/OS, which places a restriction on the number of database objects in a single DB2 z/OS database (not a subsystem), the number of objects in a DB2 LUW database is not of concern. The DB2 z/OS DBD (DataBase Descriptor), which limits the number of objects in a single database to 25% of the DBD memory allocation, has no exact counterpart in DB2 LUW.

Administrative Tools

Database Control Center is an easy to use, graphical interface that the DBA can use to configure database manager instances, databases, backup/recovery and media management. The Control Center is fully Java enabled and can be executed as a Java application or a Java applet via a standard browser.

Connectivity Using ODBC/CLI

The following is a conceptual view illustrating how a PeopleSoft client establishes connectivity with the DB2 LUW database server using ODBC and the Call Level Interface (CLI):



Connectivity using ODBC/CLI

In the PeopleTools Layer, a PeopleTools program (1) issues an SQL Connect request. PeopleTools program PSODBC.DLL (2) processes the request and formats the SQL request in an ODBC-compliant format and invokes the ODBC SQLConnect function (3).

In the Microsoft ODBC Layer, ODBC.DLL reads the registry entry for the ODBC (4) checking for the data source name (in our case, the database name). It finds this entry and loads the associated vendor driver (5), \WINDOWS\SYSTEM32\DB2CLI.DLL.

Note. ODBC.DLL determines by reading ODBC.INI which of several possible vendor ODBC-compliant drivers to load (e.g. Microsoft's SQL Server (sqlsrv32.dll), IBM DB2 LUW, IBM DB2 z/OS, and so forth). In this case, it loads the IBM driver, DB2CLI.DLL.

In the IBM ODBC Driver Layer, (6) DB2CLI.DLL reads the WINDOWS Environment Variable (DB2PATH) to obtain the path for db2cae executables and (7) reads *db2 connect install dir*\db2\sqldbdir\sqldbdir and *db2 connect install dir*\db2\sqlnodir\sqlnodir to obtain database directory and node directory information, respectively, as it formats and submits its connect request to the database server (8).

IBM CLI (Call Level Interface) on the Client

IBM's Call Level Interface (CLI) programs, unlike embedded SQL programs, are not precompiled and bound to a database and, therefore, do not produce PLANs or Packages.

PeopleSoft uses the Call Level Interface for online client connectivity (as well as database server batch processing). Both CLI interfaces operate in a similar manner, executing SQL statements one at a time, at runtime, caching prepared statements in a package cache buffer controlled by DB2 LUW. Again, no PLAN or Package is produced, as happens in a DB2 z/OS environment using embedded SQL.

Mapping Client and Server IP Addresses

In a two-tier architecture, processes on the database server displayed using the DB2 UDB list application command can be mapped back to particular clients using the Application ID field. The ability to map a server process to a client is important since all PeopleSoft client tasks are connected using the identical table owner ID.

To map a server process back to a client, issue a DB2 UDB list application on the database server, then convert the value in the Application ID to a client's IP address. The Application ID is displayed in hexadecimal representation with each two characters representing a node in IP's dotted notation format. In the example below, Auth ID PTDVL is connected from the client at x'C65D379E', or IP Address 198.93.55.158.

Auth ID	Application Name	Application ID	DB Name
PTDVL	PSIDE.EXE	*TCPIP.C65D379E.960305015712	HR800DMO

Note. In the preceding example, Auth ID shows PTDVL in uppercase, even though the table owner is defined in the respective operating system as a login ID in lowercase.

Checklists and Troubleshooting

This section discusses:

- Connectivity checklist.
- Diagnosing transaction hangs.
- DB2DIAG.LOG.
- ODBC Trace.
- db2trc.
- DB2 UDB Help facility.

Connectivity Checklist

This checklist is provided to help diagnose online connectivity problems.

- On PeopleSoft signon screen, the database name must be specified in upper case.
- On PeopleSoft signon screen, the user/password is case-sensitive (examine table PSOPRDEFN).
- Is the ConnectID and ConnectPSWD specified properly in the PeopleTools Configuration Manager?
- Is the DB2 LUW database on server running? To check it:

```
Database Server, type "db2 connect to <database-name>"
```

- Does table PS.PSDBOWNER contain the database name (in uppercase) and ownerid (in lowercase)? It should contain 1 row only. If it contains more than 1 row, drop it and recreate it using /sql/dbowner.sql.
- Can you ping the server? (This will test to see if the network is operating successfully.)
- Can you connect to the database using the client Command Line Processor?
- Did you specify an ODBC data source for your database? You can do this with CAE's CLI-ODBC Administrator.
- DB2 Connect on Windows NT requires that the user ID which is cataloging databases and nodes be an NT Administrator (not just a user with administrative authorization). The Administrator's user ID must not exceed 8 characters. Log in as the Windows NT administrator, go to Administrative Tools, User Manager, New User. Complete the New User Information, Click the Groups pushbutton, Add Administrators, OK.

Diagnosing Transaction Hangs

One way to check to see whether the SQL is hung or if it is still executing due to a long unit of work or bad access path is to use DB2 LUW's Snapshot Monitor. Other diagnostic tools include vmstat and iostat to determine server CPU and I/O activity.

The "Snapshot Monitor" requires that database monitor switches be turned on. Unfortunately, these switches must be turned on before a process is started.

To use Snapshot Monitor:

1. Logon to the Command Line Processor on the server.
2. Issue the following statements:
   - db2 update monitor switches using bufferpool on
   - db2 update monitor switches using table on
   - db2 update monitor switches using uow on
3. Wait a mninute to allow statistics to compile.
4. Issue the following statement(s):

```
db2 "get  snapshot for database on hr800dmo" > snapsht1.dbx
```

5. Wait a minute to allow additional statistics to compile.
6. Issue the following statement:

```
db2 "get  snapshot  for  database on hr800dmo" > snapsht2.dbx
```

7. Compare the two files and identify any changes, such as:
   - Bufferpool logical reads
   - Bufferpool physical reads
   - Commit Statistics
   - Dynamic SQL Statements Attempted
   - Rows Selected

If parm values are the same for both snapshots, then the transaction may be hung. If the most logical explanation is that the transaction is hung, perform this step to retry the transaction:

```
db2 force application (<agent-id>)
```

For example:

```
db2 force application (3265)   (parenthesis required)
```

Note. If an application is terminated using the above “force” command, the user will have to reconnect to the database server.

DB2DIAG.LOG

The <instance-owner-home-dir>/sqllib/db2dump/db2diag.log file contains diagnosis information related to instance, utility, and connectivity problems. The full name of the directory may also be obtained by issuing a get dbg cfg command in the command line processor on the database server, then checking the DIAGPATH configuration setting. This file contains diagnosis information related to instance, utility, and connectivity problems.

ODBC Trace

Go to Control Panel. Open ODBC Administrator, select the appropriate Data Source, and then press Options. Select the ODBC Trace option.

db2trc

If the problem is repeatable, you can use db2trc to trace the database internal logic. Although this trace is mostly used by DB2 UDB service personnel, it may give you some clues. To obtain the help information of db2trc, type the following command:

```
db2trc -h
```

The following is a simple example of how to use db2trc to obtain DB2 LUW internal tracing information:

```
db2trc -l 1000000 on
[repeat the failing process]
db2trc flw > trc.flw
db2trc fmt > trc.fmt
db2trc off
```

DB2 UDB Help Facility

DB2 UDB Message Reference can give you detailed information about your SQL error-code. A quick way to get similar information online is to do db2 "? <sqlcode>". For example:

```
db2 "?  Sql1042"
```

Note. DB2 UDB requires a 4-digit error code suffix.

APPENDIX D

Administering PeopleSoft Databases on Informix

This appendix discusses:

- Database terminology.
- DBspace strategy.
- Database server directory structure.
- Troubleshooting model.

Database Terminology

PeopleSoft uses the following technical naming conventions for databases:

Database	A set of data tables accessed and managed as a group. Informix manages the database at the system level.
Informix Database Server	A cooperating set of host processes and shared memory capable of managing one or more databases—a running online engine. Corresponds to a specific INFORMIXSERVER value and a matching entry in the sqlhosts file. May contain many databases each with its own catalog.
Object Set	A collection of database objects. PeopleSoft uses tables, indexes. An object set corresponds to the PeopleSoft owner ID.

DBspace Strategy

The following strategies can be used for DBspace:

- Separate the root dbspace, physical logs, logical logs, and the temporary dbspace from one another and from the application dbspaces.

  Place the root dbspace, logical log, and physical log in separate dbspaces on separate disks.

- Separate certain high volume application tables to optimize performance.

  A minimum configuration for production systems is four physical drives (at least one each for database files, physical logs, logical logs, and temporary area).

- Separate the dbspace for data on one drive and dbspace for indexes on a separate drive.

Database Server Directory Structure

The environment variable $INFORMIXDIR points to the directory where Informix is installed on your machine; normally this is set to /usr/INFORMIX. The standard Informix directory structure is built under INFORMIX directory by the Informix install process.

The standard Informix architecture uses the "Two-Task Model." In this architecture, when a user connects to the database server a network thread is created to handle the network processing for that user.

Each Informix server instance consists of the following pieces:

- At least eight database processes that operate the database.
- At least 3 shared memory segments, through which the database processes communicate.
- The $INFORMIXDIR/etc/sqlhosts file holds networking parameters for each accessible server instance.

  Entries include server name, network protocol, host name and tcp-ip service.
- The $INFORMIXDIR/etc/$ONCONFIG file.

  This file primarily holds shared memory configuration parameters for the local server instance. These include the number of buffers, number of locks, size of the initial shared memory segment, and so on The "onconfig" file also includes pointers to the root dbspace, the temporary dbspace, and the physical log dbspace, as well as the names of the backup tape devices. By convention, these files are often given a name such as onconfig.inf11, where inf11 is the name of the server. This is helpful when managing multiple server instances on one host.
- Database "chunks."

  Hold the data stored in the database. Under UNIX, these may be either "raw" or "cooked" files. In either case they should be stored in a common directory, with links pointing to their physical locations, if necessary.

See Also

Administrator's Guide for Informix Dynamic Server

Troubleshooting Model

This section discusses some steps you can take to diagnose system signon problems. Understanding basic operations and process flow is essential when you are troubleshooting connectivity errors. Use the following model as a reference for this section.

1. Test terminal connection.

   Try using TELNET, or a similar network utility, to get a terminal connection to your database server. If this succeeds, you probably have a problem with the way Informix-Connect or Informix is set up. Check to see if the Informix database server is active. If Step 2 fails, then the problem is within the networking layer.

2. Consult your networking experts.

   The problem has been isolated to something within the network layer. Try to isolate the network problem. Can you log on to other servers? Are other terminals still able to connect? Try lowering level network diagnostics, if they exist

APPENDIX E

Administering PeopleSoft Databases on Oracle

This appendix provides an overview of Oracle administration and discusses how to:

- Monitor PeopleSoft client database connections.
- Set the number of temporary tables.

Understanding Oracle Administration

This section discusses:

- NET8i/9i.
- PeopleSoft servers and the Oracle connection string.

NET8i/9i

NET8i/9i offers peer to peer connectivity and a multi-protocol interchange (MPIC). The product is installed as two or more components; the Transparent Network Substrate (TNS), the Oracle Protocol Adapter and the multi-protocol interchange.

NET8i/9i uses a few configuration files (SQLNET.ORA and TNSNAMES.ORA). Configuration files can be created using a system editor, or with the Net8i/9i Assistant

PeopleSoft Servers and the Oracle Connection String

Beginning with PeopleTools 7, PeopleSoft changed the Oracle connect string used to connect to the database. Prior to this, the format of the connect string was userid/password for the batch processes and userid /password@service_name for the online processes. With PeopleTools 7, the connect string was consistently changed to be userid/password@service_name, for all PeopleSoft processes. This includes online, batch, and application server processes.

This modification made setup and configuration easier for those platform configurations that could support PeopleSoft batch server processes and/or application server processes. However, performance for the batch processes and application server processes on a server that also functions as the database server was slightly degraded. This was due to the overhead involved in routing through SQL*NET.

Starting with PeopleTools 7.5 and continuing through PeopleTools 8.4x, we have addressed the aforementioned issue by providing a mechanism for the PeopleSoft Administrator to indicate which connect string to use. This mechanism is a flag that is set while configuring the application server or the Process Scheduler. The flag is set in the respective configuration files for application server or the Process Scheduler processes.

Database Options

When configuring an application server or the Process Scheduler, you can modify the parameters in the Database Options section if desired.

```
Values for config section - Database Options
UseLocalOracleDB=0
;ORACLE_SID=
EnableDBMonitoring=0

Do you want to change any values (y/n)? [n]:
```

UseLocalOracleDB Indicates if the PS database that you are connecting to is in a Local Oracle SID. The default is *0*, meaning that the DB you are connecting to is remote. The resultant connect string is in the following format: userid/password@service_name. If you set this to *1*, then you are indicating to our processes to use the following connect string when attempting to connect to the target database: userid/password. This implies a local connection.

If you decide to use UseLocalOracleDB, then you must add the BEQUEAH_DETACH=YES parameter to the SQLNET.ORA file of the machine where you are setting up the App or Batch servers. This enables Oracle to cleanup any zombie DB processes spawned on behalf of PeopleSoft transactions left over from aborted transactions

Oracle SID Indicates for a Local Oracle connection only, the name of the Local ORACLE_SID you wish the PeopleSoft processes to connect to. Many sites set up more than one ORACLE_SID on their servers. This parameter gives you the ability to choose which ORACLE_SID you wish to connect to when connecting in Local mode.

EnableDBMonitoring This parameter enables or disables DB monitoring of 3-Tier connections. This feature is covered later on in this chapter. See Monitoring PeopleSoft Client Database Connections.

The following tables describe the relationship between the UseLocalOracleDB parameter and the ORACLE_SID environment variable.

UseLocalOracleDB Flag	The target database is local	The target database is remote
0 is the default setting Internally Psoft will generate the following connect string when attaching to the target database: UID/PW@TNS_ALIAS	Access will be made via TNSNAMES	Access will be made via TNSNAMES
1 is the setting you use if you intend to use a Local Oracle DB. Internally Psoft will generate the following connect string when attaching to the target database: UID/PW (Note the omission of the TNS_ALIAS.)	Access will default to the Local DB as designated by the ORACLE_SID environment variable If the ORACLE environment variable TWO_TASK is set to a valid TNS_ALIAS, then this would also work. The existence of the TWO_TASK environment variable is in effect overriding the generated connect string.	To choose this option does not make sense if it is your intention to use a Local Oracle DB. This combination will work if the ORACLE environment variable TWO_TASK is set to a valid TNS_ALIAS. You are in effect overriding the generated connect string.

ORACLE_SID Parameter	UseLocalOracleDB Flag	UseLocalOracleDB Flag
This parameter is delivered in the application server and Process Scheduler configuration file commented out. This indicates that the default setting is however the current ORACLE_SID environment variable is set.	0 The target database is remote	1 The target database is local The ORACLE_SID parameter is not enabled (commented out) therefore ORACLE_SID for this process will default to the current ORACLE_SID environment variable.
ORACLE_SID=xxxxxxx where xxxxxxxx equals a valid ORACLE_SID for the server that this process is running on.	If UseLocalOracleDB Flag is set to zero, then enabling ORACLE_SID is invalid since you are indicating a remote connection, the value associated with the ORACLE_SID parameter will be ignored.	If UseLocalOracleDB Flag is set to one, and ORACLE_SID is enabled, the value associated with the ORACLE_SID parameter will be exported as an OS environment variable thus overriding the current ORACLE_SID environment variable.

Monitoring PeopleSoft Client Database Connections

This section provides an overview of PeopleSoft client database connections and discusses how to:

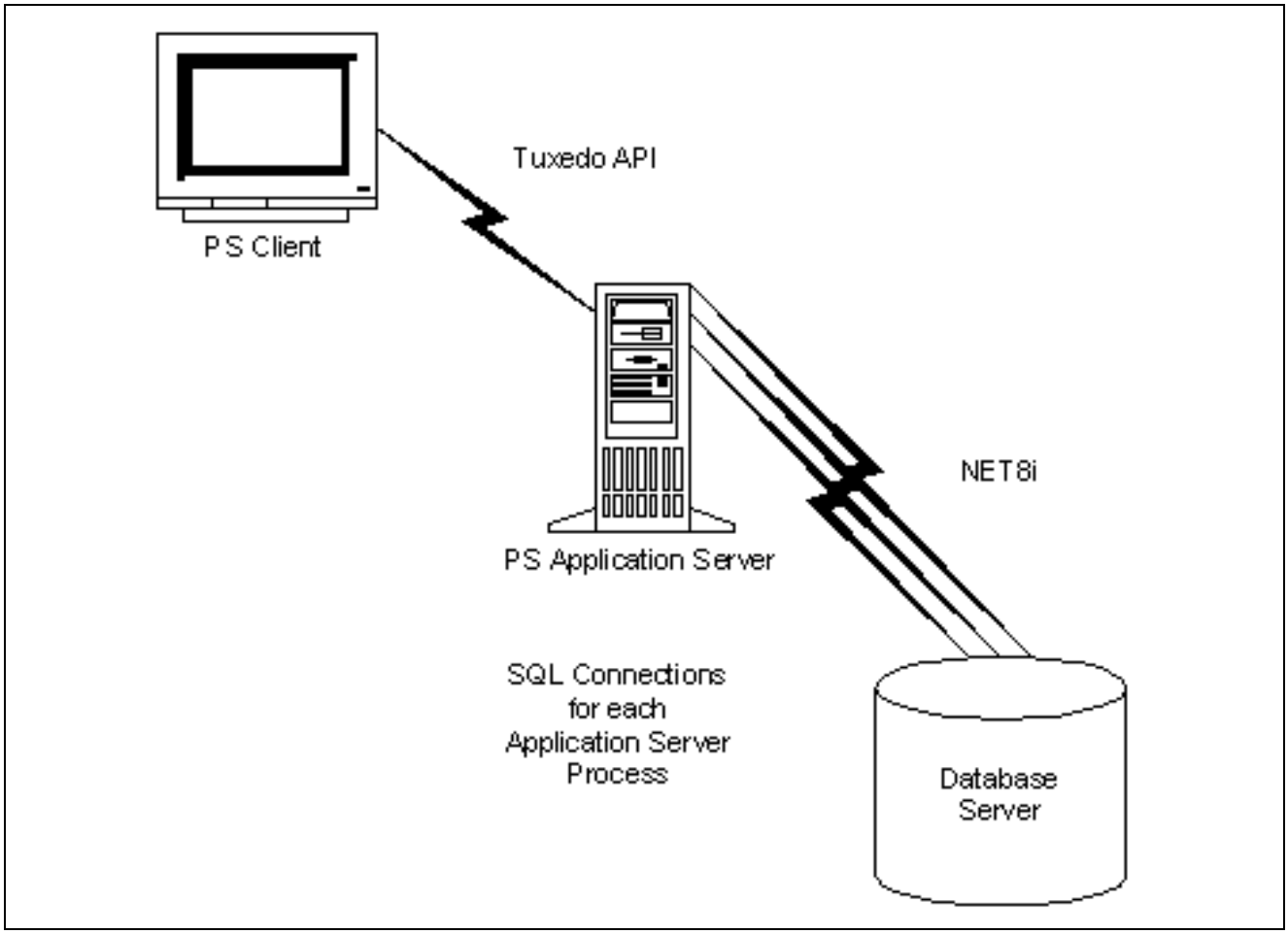
- Use client monitoring.
- Track PeopleSoft database connections by PeopleSoft User ID.

Understanding PeopleSoft Client Database Connections

PeopleSoft provides the ability to monitor PeopleSoft Window clients connected to the database server from both two-tier and three-tier workstations with much improved informational content.

Without this feature, an administrator could not obtain any specific information regarding the user and the associated transaction when a user accessed the database. This was mainly due to the fundamentals of our security implementation, and the inherent nature of three-tier architecture wherein many clients connect to one application server, but only the application server maintains connections to the database server. Consequently, when querying the database, an administrator would see application server processes and two-tier clients connected to the database as the schema owner, not by specific workstations or PeopleSoft operators ID's.

To illustrate this concept, examine the following graphic. Notice the individual application server processes maintain the connection to the database server, not the individual client workstations.



PeopleSoft client database connections

Note. In this example, assume that the number of workstations is n, where n may equal anywhere from 10 to 1000 clients.

To illustrate this concept further, suppose one of your users has executed an extremely inefficient query that severely impacts the rest of the database users. A Database Administrator would want to identify that user and take appropriate action. Without an ability to monitor users you would probably have to terminate the physical connection, which in a three-tier scenario would mean dropping the connection between the application server and the database server. This could potentially affect hundreds of users.

We now associate the PeopleSoft User ID (and some other information, such as OSUserName, MachineName, AppServerDomainName, ProgramEexecutable) with each connection or transaction. This allows us to associate activity on the database server with a particular workstation and operator. This information is stored in the CLIENT_INFO column of the V$SESSION dynamic view. Monitoring enabled is the new default behavior for two-tier connections.

For three-tier connections, the PeopleSoft systems administrator has the option to enable this feature by setting the EnableDBMonitoring flag to ‘1’ in the Application Server configuration file. (PSAPPSRV.CFG).

Some possible uses of this feature include system-wide troubleshooting, performance monitoring, "chargeback" accounting, and security audits for your system.

Using Client Monitoring

You can use client monitoring for the following:

- COBOL Programs are always monitored.
- SQR programs are always monitored.
- Process Scheduler is always monitored.
- The two-tier client is always monitored.
- For the three-tier client (whether it be a Window-based workstation or Browser) the ability to monitor is configurable through PSADMIN in the Database Options configuration section.

Tracking PeopleSoft Database Connections by PeopleSoft User ID

This section provides an overview of tracking database connections by user ID, a legend for interpreting illustrations, and discusses the following:

- Oracle process connections.
- Two-tier client connections.
- Application server process connections.
- Three-tier client connections on Windows.
- Three-tier client connections on PIA clients.
- Process Scheduler connections.
- SQR connections.
- COBOL connections
- Window and browser connections multithreaded through the application server.

Understanding Tracking PeopleSoft Database Connections by PeopleSoft User ID

To view the information associated with client connections, sign on to SQLPlus for the appropriate SID and execute the following SQL Query:

Note. This is a sample query that ties the OS PID and PeopleSoft CLIENT_INFO to the process connected to the Oracle database.

```
set linesize 200
```

```
select p.spid,
   substr(s.osuser,1,10) osuser,
   substr(s.username,1,8) username,
   substr(s.program,1,24) program,
   substr(s.client_info,1,60) ClientInfo
from v$session s, v$process p
where s.paddr=p.addr
and s.osuser is not null
order by s.osuser
/
```

The result of this query will differ somewhat for two-tier and three-tier connections. The following sections describe the information returned for various scenarios.

Legend

JZARATE (uppercase) is a NETWORK login ID for the window workstations and JZARATE123199 is the window client MACHINENAME.

TMJONES (uppercase) is a NETWORK login ID for the window workstations and TMJONES110299 is the window client MACHINENAME.

JRSMITH (uppercase) is a NETWORK login ID for the window workstations and JRSMITH031198 is the window client MACHINENAME.

PREILLY (uppercase) is a NETWORK login ID for the window workstations and PREILLY060499 is the window client MACHINENAME.

PT844P01 is the PS schema (PS SYSADM ID or Access ID).

PT81 is an example of a Tuxedo domain name.

PTDMO, VP1, and PS are PeopleSoft User IDs used to signon to the database from the various clients

oracle (lowercase) is the owner id of all of the Oracle processes

certora (lowercase) is the Unix login id of the PS administrator that started the Application Server and Process Scheduler.

Oracle Process Connections

Execution of the sample query noted above shows the Oracle Processes for the SID in which PeopleSoft database PT844P01 resides and this SQL*Plus session used to monitor the client info. There is no client info because no PeopleSoft client connections currently exist.

```
Oracle Processes and this SQLPLUS session used to monitor the client info from⇒
 network user
JZARATE

SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO               ⇒

-------- ------- -------- ---------------------------- ---------------------------⇒
----------
15276    JZARATE PT844P01 SQLPLUSW.EXE                                             ⇒
```

```
8364     oracle           oracle@st-sun01 (PMON)                                ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                ⇒

8372     oracle           oracle@st-sun01 (SMON)                                ⇒

8374     oracle           oracle@st-sun01 (RECO)                                ⇒


7 rows selected.
```

Two-Tier Client Connections

For the two-tier connection, you can expect to monitor the following client information: "%oprid%,%osusername%,%machinename%,,%executable%,"

Adding to what was previously displayed, this a two-tier client connection from workstation TMJONES110299, Peoplesoft OPRID PS, executing PSIDE.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO                ⇒

-------- ------- -------- ------------------------------ --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                      PS,TMJONES,TMJONES110299,,p⇒
side.exe,
15276    JZARATE PT844P01 SQLPLUSW.EXE                                             ⇒

8364     oracle           oracle@st-sun01 (PMON)                                ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                ⇒

8372     oracle           oracle@st-sun01 (SMON)                                ⇒

8374     oracle           oracle@st-sun01 (RECO)                                ⇒


8 rows selected.
```

Application Server Process Connections

For the application server connection, you can retrieve the following information from the database: "%oprid%,%osusername%,%machinename%,%tuxedo_domain%,%executable%,"

Adding to what was previously displayed, this shows the Application Server process connections for Domain PT81, from server st-sun01, using UNIX login ID certora, with the appserver processes connecting to the database as OPRID PTDMO.

Keep in mind that each application server process maintains an individual connection to the database. If your application server is up and running, you should see the following information after executing the session query:

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO                ⇒

-------- ------- -------- ---------------------------- ---------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                    PS,TMJONES,TMJONES110299,,p⇒
side.exe,
15276    JZARATE PT844P01 SQLPLUSW.EXE                                           ⇒

15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSAPPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSSAMSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSAPPSRV,
8364     oracle           oracle@st-sun01 (PMON)                                 ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                 ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                 ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                 ⇒

8372     oracle           oracle@st-sun01 (SMON)                                 ⇒

8374     oracle           oracle@st-sun01 (RECO)                                 ⇒


11 rows selected.
```

Three-Tier Client Connections – Window Workstations

For the three-tier connections, you can retrieve the following client information: "%oprid%,%osusername%,%machinename%,%tuxedo_domain%,%executable%,""

When the three-tier client is connected, then you should see the application server process that is executing the transaction for the client. For example, the PSAPPSRV handles the large queries executed by three-tier clients. Let’s assume for this example that the PSAPPSRV is processing the current client request.

Adding to what was previously displayed, this a three-tier client workstation JRSMITH031198, signing on as PSOFT oprid VP1, to Domain PT81 and utilizing two application server processes PSAPPSRV.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                          CLIENTINFO                ⇒

-------- ------- -------- ------------------------------ --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                      PS,TMJONES,TMJONES110299,,p⇒
side.exe,
15276    JZARATE PT844P01 SQLPLUSW.EXE                                             ⇒

15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) VP1,,JRSMITH031198,PT81,PSA⇒
PPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSSAMSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) VP1,,JRSMITH031198,PT81,PSA⇒
PPSRV,
8364     oracle           oracle@st-sun01 (PMON)                                   ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                   ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                   ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                   ⇒

8372     oracle           oracle@st-sun01 (SMON)                                   ⇒

8374     oracle           oracle@st-sun01 (RECO)                                   ⇒


11 rows selected.
```

Three-Tier Client Connections – PIA Clients

Similarly, for the three-tier PIA client connections, you can retrieve the following client information: "%oprid%,%osusername%,%machinename%,%tuxedo_domain%,%executable%,""

When the three-tier client is connected, then you should see the application server process that is executing the transaction for the client. For example, the PSAPPSRV handles the large queries executed by three-tier clients. Let's assume for this example that the PSAPPSRV is processing the current client request.

Adding to what was previously displayed, this is a three-tier (PIA) client PREILLY060499 (connecting through a web browser), signing on as PSOFT/PTDMO, to Domain PT81 and utilizing two application server processes PSAPPSRV. From a monitoring perspective, there is no difference between a three-tier window client and a PIA client.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                          CLIENTINFO                ⇒

-------- ------- -------- ------------------------------ --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                      PS,TMJONES,TMJONES110299,,p⇒
side.exe,
```

```
15276    JZARATE PT844P01 SQLPLUSW.EXE                                          ⇒

15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSSAMSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
8364     oracle           oracle@st-sun01 (PMON)                                ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                ⇒

8372     oracle           oracle@st-sun01 (SMON)                                ⇒

8374     oracle           oracle@st-sun01 (RECO)                                ⇒


11 rows selected.
```

Process Scheduler Connections

For the Process Scheduler connection, you can expect to see the following information: "%oprid%,%osusername%,%machinename%,,%executable%,"

Adding to what was previously displayed, this is the Process Scheduler running, started by OSUSER certora, from server st-sun01, logged in as PSOFT oprid PTDMO.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO              ⇒

-------- ------- -------- ---------------------------- --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                    PS,TMJONES,TMJONES110299,,p⇒
side.exe,
15276    JZARATE PT844P01 SQLPLUSW.EXE                                          ⇒

15435    certora PT844P01 psprcsrv@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,,psprcsrv,
15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSSAMSRV,
8364     oracle           oracle@st-sun01 (PMON)                                ⇒
```

```
8366     oracle           oracle@st-sun01 (DBW0)                                ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                ⇒

8372     oracle           oracle@st-sun01 (SMON)                                ⇒

8374     oracle           oracle@st-sun01 (RECO)                                ⇒


12 rows selected.
```

SQR Connections

For the SQR Program connections, you can expect to see the following information: "%oprid%,%spid%"

Adding to what was previously displayed, this is a SQR report run from client workstation JZARATE123199, submitted from PSOPRID PS and having a PID of 15449.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO                ⇒

-------- ------- -------- ------------------------------ --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                      PS,TMJONES,TMJONES110299,,p⇒
side.exe,
15276    JZARATE PT844P01 SQLPLUSW.EXE                                             ⇒

15449    JZARATE PT844P01 sqrw.exe                       PS,15449                  ⇒

15435    certora PT844P01 psprcsrv@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,,psprcsrv,
15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSSAMSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
8364     oracle           oracle@st-sun01 (PMON)                                ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                ⇒

8372     oracle           oracle@st-sun01 (SMON)                                ⇒

8374     oracle           oracle@st-sun01 (RECO)                                ⇒
```

```
13 rows selected.
```

COBOL Connections

For the Cobol Program connections, you can expect to see the following information: "%oprid%,%osusername%,%machinename%,,%executable%,"

Adding to what was previously displayed, this a COBOL process PTPTEDIT run from client workstation JZARATE123199, submitted from PSOPRID PS.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO                ⇒

-------- ------- -------- ---------------------------- --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                    PS,TMJONES,TMJONES110299,,p⇒
side.exe,
15276    JZARATE PT844P01 SQLPLUSW.EXE                                           ⇒

15449    JZARATE PT844P01 sqrw.exe                     PS,329                    ⇒

15451    JZARATE PT844P01 PTPTEDIT.exe                 PS,JZARATE,JZARATE123199,,P⇒
TPTEDIT,
15435    certora PT844P01 psprcsrv@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,,psprcsrv,
15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSSAMSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
8364     oracle           oracle@st-sun01 (PMON)                                 ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                 ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                 ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                 ⇒

8372     oracle           oracle@st-sun01 (SMON)                                 ⇒

8374     oracle           oracle@st-sun01 (RECO)                                 ⇒


14 rows selected.
```

Window and Browser Connections Multithreading Through the Application Server

For the three-tier connections, you can retrieve the following client information: "%oprid%,%osusername%,%machinename%,%tuxedo_domain%,%executable%,""

The application server multithreads the incoming three-tier client through the application server processes already connected to the database. The next several displays illustrate a continual changing of the monitoring information displayed through the application server 'thread' based on incoming three-tier client activity.

Adding to what was previously displayed, accessing the database again from the three-tier window client JRSMITH031198 reflects a change in the User ID (VP1) and client machinename for the both application server processes.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO                ⇒

-------- ------- -------- ------------------------------ --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                      PS,TMJONES,TMJONES110299,,p⇒
side.exe,
15276    JZARATE PT844P01 SQLPLUSW.EXE                                             ⇒

15449    JZARATE PT844P01 sqrw.exe                       PS,329                    ⇒

15451    JZARATE PT844P01 PTPTEDIT.exe                   PS,JZARATE,JZARATE123199,,P⇒
TPTEDIT,
15435    certora PT844P01 psprcsrv@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,,psprcsrv,
15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) VP1,,JRSMITH031198,PT81,PSA⇒
PPSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) VP1,,JRSMITH031198,PT81,PSA⇒
PPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSSAMSRV,
8364     oracle           oracle@st-sun01 (PMON)                                   ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                   ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                   ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                   ⇒

8372     oracle           oracle@st-sun01 (SMON)                                   ⇒

8374     oracle           oracle@st-sun01 (RECO)                                   ⇒


14 rows selected.
```

Adding to what was previously displayed, accessing the database from the three-tier (PIA) client PREILLY060499 illustrates a change in the User ID (PTDMO) and client machinename for one of the application server processes.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO                ⇒

-------- ------- -------- ---------------------------- --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                    PS,TMJONES,TMJONES110299,,p⇒
side.exe,
15276    JZARATE PT844P01 SQLPLUSW.EXE                                           ⇒

15449    JZARATE PT844P01 sqrw.exe                     PS,329                    ⇒

15451    JZARATE PT844P01 PTPTEDIT.exe                 PS,JZARATE,JZARATE123199,,P⇒
TPTEDIT,
15435    certora PT844P01 psprcsrv@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,,psprcsrv,
15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSSAMSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) VP1,,JRSMITH031198,PT81,PSA⇒
PPSRV,
8364     oracle           oracle@st-sun01 (PMON)                                 ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                 ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                 ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                 ⇒

8372     oracle           oracle@st-sun01 (SMON)                                 ⇒

8374     oracle           oracle@st-sun01 (RECO)                                 ⇒


14 rows selected.
```

Adding to what was previously displayed, accessing the database from the three-tier window client JRSMITH031198 reflects a change in the User ID (VP1) and client machinename for one of the application server processes.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO                ⇒

-------- ------- -------- ---------------------------- --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                    PS,TMJONES,TMJONES110299,,p⇒
side.exe,
```

```
15276    JZARATE PT844P01 SQLPLUSW.EXE                                    ⇒

15449    JZARATE PT844P01 sqrw.exe                      PS,329            ⇒

15451    JZARATE PT844P01 PTPTEDIT.exe                  PS,JZARATE,JZARATE123199,,P⇒
TPTEDIT,
15435    certora PT844P01 psprcsrv@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,,psprcsrv,
15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) VP1,,JRSMITH031198,PT81,PSA⇒
PPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,PT81,PSSAMSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPPSRV,
8364     oracle           oracle@st-sun01 (PMON)                          ⇒

8366     oracle           oracle@st-sun01 (DBW0)                          ⇒

8368     oracle           oracle@st-sun01 (LGWR)                          ⇒

8370     oracle           oracle@st-sun01 (CKPT)                          ⇒

8372     oracle           oracle@st-sun01 (SMON)                          ⇒

8374     oracle           oracle@st-sun01 (RECO)                          ⇒


14 rows selected.
```

Adding to what was previously displayed, accessing the database from the three-tier window client JRSMITH031198 executing some functional process that requires use of all of the application server processes. This is reflected in the change in the User (VP1) and client machinename for all of the application server processes. Note the absence of the SQR information. The SQR has completed so no information is available.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO              ⇒

-------- ------- -------- ---------------------------- --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                     PS,TMJONES,TMJONES110299,,p⇒
side.exe,
15276    JZARATE PT844P01 SQLPLUSW.EXE                                    ⇒

15451    JZARATE PT844P01 PTPTEDIT.exe                  PS,JZARATE,JZARATE123199,,P⇒
TPTEDIT,
15435    certora PT844P01 psprcsrv@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,,psprcsrv,
15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) VP1,,JRSMITH031198,PT81,PSA⇒
PPSRV,
15409    certora PT844P01 PSSAMSRV@st-sun01 (TNS V1-V3) VP1,,JRSMITH031198,PT81,PSS⇒
```

```
AMSRV,
15402    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) VP1,,JRSMITH031198,PT81,PSA⇒
PPSRV,
8364     oracle           oracle@st-sun01 (PMON)                                  ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                  ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                  ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                  ⇒

8372     oracle           oracle@st-sun01 (SMON)                                  ⇒

8374     oracle           oracle@st-sun01 (RECO)                                  ⇒


13 rows selected.
```

Adding to what was previously displayed, the application server has been shut down and the Cobol process PTPTEDIT has completed. All clients have logged off. The Process Scheduler is still active.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO              ⇒

-------- ------- -------- ---------------------------- ----------------------------⇒
----------
15276    JZARATE PT844P01 SQLPLUSW.EXE                                            ⇒

15435    certora PT844P01 psprcsrv@st-sun01 (TNS V1-V3) PTDMO,certora,st-⇒
sun01,,psprcsrv,
8364     oracle           oracle@st-sun01 (PMON)                                  ⇒

8366     oracle           oracle@st-sun01 (DBW0)                                  ⇒

8368     oracle           oracle@st-sun01 (LGWR)                                  ⇒

8370     oracle           oracle@st-sun01 (CKPT)                                  ⇒

8372     oracle           oracle@st-sun01 (SMON)                                  ⇒

8374     oracle           oracle@st-sun01 (RECO)                                  ⇒


8 rows selected.
```

Setting the Number of Temporary Tables

Normally you will leave the number of temporary tables set to the default of three. You may need to change this setting for optimal performance, depending on various aspects of your implementation, including account transaction volumes, benchmark numbers for the current hardware and database platform, as well as your service-level requirements. Use the following procedure if you need to adjust the number of temporary tables to improve performance in your implementation.

To set the number of temporary tables:

1. Select PeopleTools, Utilities, Administration, PeopleTools Options.
2. Set the Temp Table Instances (Total) and Temp Table Instances (Online) fields to the desired settings.

Note. Temp Table Instances (Total) should always be set to the same values as Temp Table Instances (Online), unless you have been instructed otherwise in the application documentation.

3. Save your changes.

Note. The total number of instances generated consists of the allocations specified on the PeopleTools Options panel plus the allocations specified on each individual Application Engine program.

This chapter assumes that you have Oracle DBA experience, have attended Oracle's Introductory DBA classes, or have read the Oracle's Server Concepts and DBA Manuals. You should also review the Oracle Flexible Architecture (OFA) standard, and adopt as much of this as possible during your initial Oracle install.

See Also

PeopleSoft Application Engine and PeopleSoft Red Paper "PeopleSoft Batch Performance Tuning on Oracle"

APPENDIX F

Administering PeopleSoft Databases on Sybase

This appendix discusses:

- Required configuration.
- Trace options.
- Other considerations.

Required Configuration

PeopleSoft applications require certain standard configuration at the server and the database that are not optional and cannot be changed. This section discusses the following options that you must have enabled:

- Server Options.
- Database Options.

Server Options

This section discusses the following options:

- Lock scheme to datarows
- Lock promotion
- Language options
- Page size
- EBFs

Lock Scheme to Datarows

Row level locking is preferred and required with PeopleSoft applications. The following command enables the necessary configuration for your server:

```
sp_configure 'lock scheme', 0, 'datarows'
go
```

Lock Promotion

The following is the configuration delivered for the installation:

```
sp_setrowlockpromote server, NULL, 2147483647, 2147483647, 100
go
```

The purpose of setting this parameter to this value is to avoid the promotion of row level locking, which could potentially increase considerably the amount of locks required on certain activities such as loading a database through datamover. You can modify the configuration in order to allow the promotion of locks earlier when loading databases, but remember to change the configuration back after the process ends.

Language Options

When installing your server use iso_1 character set as the default for your server.

If you want to install your database using unicode (must use tools 8.44 and above) use UTF8 as your default character set.

Page Size

PeopelSoft stopped supporting the 2k page size for tools 8.44 and above. Select a 4k or 8k page size during server installation. The 16k page size is not supported.

EBFs

Starting with PeopleTools 8.44, PeopleSoft certifies Sybase ASE by its interim release. All of the EBF's above the certified interim release are certified until the next interim release is reached.

Database options

Make sure your database uses ansi nulls by default. This is a database option that is set up at installation. The configuration occurs automatically when using the database configuration wizard and is enabled by the SQL script *createdb.sql* when installed manually.

The following line shows how to enable this:

```
sp_dboption <DBNAME>, 'allow nulls',true
go
```

Another option that needs to be enabled is the following:

```
sp_dboption <DBNAME>, 'ddl in tran',true
go
```

During the database load it is recommended to truncate the transaction log. The following command is executed at installation:

```
sp_dboption <DBNAME>, 'trunc',true
go
```

Remember to disable the truncation of the transaction log if desired. This option should not be enabled for production databases.

Trace Options

This section discusses:

- Trace flags in PeopleSoft tools.
- Sybase API-specific tracing.

Trace Flags in PeopleSoft Tools

When reporting problems to customer support, it is advisable to generate files with traces of the problem that you want to report. Use the trace flags incorporated in PeopleSoft tools to generate these files. The trace flags are accessible through the configuration files for the Process Scheduler and the Application Server and through the selection of several flags when using the PeopleSoft Configuration Manager on your developer workstation.

Use "TRACESQL=63" to display the SQL statements executed when using PeopleSoft applications. This trace flag is very useful to identify problems in the SQL being executed against a database that hosts a PeopleSoft application.

The trace flag will show the details about the execution of a sql statement such as: if the statement was recompiled or if it's using an old query plan, the time it took to execute, the time between executions, if the SQL was parametrized, among other things.

See Also

Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration, "Setting Application Server Domain Parameters"

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler, "Using the PSADMIN Utility"

Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration, "Using PeopleSoft Configuration Manager"

Sybase API-Specific Tracing

When using Sybase, you can select from three SQL Trace options that you can use to enable very detailed Sybase tracing:

- Database API-specific Calls
- Sybase Bind Information
- Sybase Fetch Information

To set these options, use the Trace tab in the PeopleSoft Configuration Manager. Keep in mind that online performance will be affected.

The output of the Sybase tracing can be found in %TEMP%/SYBxxx.TMP, where xxx is a random integer that is different for each file based on each connection.

If you select any options other than Database API, Sybase Bind Information, and Sybase Fetch Information, the tracing output will be found in %TEMP%/DBG1.TMP.

Note. Only use tracing for debugging purposes, since performance will be affected. Depending on what level of tracing you select, a very large file can be created. To turn tracing off for Windows, clear the boxes in Configuration Manager.

See Also

Enterprise PeopleTools 8.45 PeopleBook: System and Server Administration, "Using PeopleSoft Configuration Manager"

Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler

Other Considerations

This section discusses:

- Database monitoring.
- Device management.
- Caches.
- Segments.
- Tempdb.
- Network packet size.
- Updating statistics.

Database Monitoring

Activation of the "EnableDBMonitoring" flag, available through the configuration files for the Process Scheduler and the Application Server, allows you to populate context information of the query executed against the database. This is particularly useful when looking for information about the PeopleSoft user running a particular SQL statement.

Example of SQL Statement

The following is an example of SQL statements that will display the context information of a user once "EnableDBMonitoring" is enabled. Modify the scripts according to your needs.

```
select clientname, clienthostname, clientapplname from master..sysprocesses where⇒
 spid=<spid>
```

If you don't know the SPID of the user you are trying to monitor, start with the DBID.

Device Management

Try to use separate physical devices for the various servers in your system. Ideally, you should have one device for database data, one device for tempdb, one device for master, one device for syslogs, and one device for the operating system.

Note. Tempdb and transaction logs (syslogs) are used very heavily. PeopleSoft highly recommends using a separate device and allocating adequate space. Mirror the syslogs for recovery.

Caches

Consider using named caches on tempdb and syslogs. Also, consider experimenting with different private log cache sizes as this can improve performance. You can definitely reduce the contention for the last page of syslogs by increasing the size of the Private Log Cache for the users so that they will write to the syslogs table less frequently. Your Database Administrator should determine required memory to support your number of users.

Segments

Consider using segments to separate data, non-clustered indexes, and heavily used tables onto separate devices.

Tempdb

Tempdb is heavily used for sorting (order by statements) and to create worktables for "OR" and "GROUP BY" statements. It is rebuilt every time the dataserver is booted; no permanent data is stored in it. Because of this, the normal considerations for recoverability do not apply to tempdb.

Note. You should consider binding tempdb to its own named cache.

Sizing

Tempdb should be sized according the number of concurrent users, the size of the sorts or group by statements, and the largest possible sort that might be done in tempdb. You will need to consider all databases running on your dataserver because they all share tempdb.

Placement

Whether to place tempdb on a journaled file system, a logical volume, a raw device or a solid-state device is platform-dependent. Following are some considerations for each:

- Journaled File System

  Placing tempdb on a journaled file system is faster on many platforms because the OS buffers the writes so that Sybase Adaptive Server doesn't have to wait for physical disk writes. If you have enough OS memory to buffer all of the writes, then tempdb can essentially stay in memory and never have to write to disk.

  When you initialize a file system device(file), Sybase Adaptive Server does not know that there is really enough space in the file system for the size you specify. At run time, the server may well run out of space if the file system was—or became—too full for the tempdb file.

  A precaution would be to configure a separate file system solely for the tempdb file, and set permissions so that only the server can write to this device.

  Note. Do not place user databases on a journaled file system as Adaptive Server cannot guarantee recoverability if the system shuts down abnormally.

- Raw Device

  There are no special considerations for placing tempdb on a raw device.

- Solid-State Device

  This is a special device that essentially runs in memory. If tempdb is an I/O bottleneck, then placing tempdb on a solid-state device can improve performance

Another consideration is placing the tempdb syslogs on a separate device.

Network Packet Size

You may be able to improve performance for large result sets by matching the Sybase data packet size to your network packet size and reclaiming unused network bandwidth. Larger packets will also improve network performance by reducing the number of packets sent between the client and server.

From within Configuration Manager, on the Common tab of the Edit Profile dialog box, you can increase the TCP Packet Size for Sybase. Sybase uses a default of 512 bytes and it accepts packet sizes in increments of 512.

The Sybase server must also be configured to accept the larger packet size. To increase the packet size at the server level, issue the following command using Sybase ISQL or a similar SQL utility:

```
1> sp_configure 'max network packet', 1024
2> go
```

Note. Increase the network memory allocated per connection using the sp_configuration additional network memory command when increasing the max network packet size.

These server commands will require the Sybase dataserver to be rebooted before the configurations will take effect

---

Note. PeopleSoft does not recommend increasing the dataserver default network packet size from the default value of 512. This will ensure that all PeopleSoft clients are able to connect. If the TCP Packet Size is increased on the client with Configuration Manager and the max network packet size is not increased on the server, Signon failure will occur.

See Also

Sybase Adaptive Server Enterprise Reference Manual

Sybase Adaptive Server Enterprise Performance and Tuning Guide

Updating Statistics

When an index gets created, the system gathers statistics about the table. These statistics help to determine the best search method for accessing a table. Each time an index gets created, the statistics are updated for that table. When an index is dropped, the statistics are not removed. In this case you will want to delete the statistics for the dropped index. Use the Sybase DELETE STATISTICS command follow by an UPDATE STATISTICS to rebuild your existing index and column statistics.

You should also update database statistics if there have been significant changes to the index—such as adding or deleting a large number of rows in a table. To do this, use the Sybase UPDATE STATISTICS command. You can run this command against tables and indexes in a database.

Note. There is no command to delete and update statistics for an entire database.

APPENDIX G

Configuring Remote Data Access

This appendix provides an overview of remote data access and discusses how to:

- Configure Informix application or batch servers for remote data access.
- Configure Oracle application or batch servers for remote data access.
- Configure DB2 UDB application or batch servers for remote data access.
- Configure Sybase application or batch servers for remote data access.
- Install and configure the Microsoft SQL Server JDBC driver.

Understanding Remote Data Access

JDBC drivers are either supplied with the database product that you already have, or can be obtained separately from your database vendor. PeopleSoft develops, certifies and tests remote data access using the drivers provided by the DBMS vendors. Using these drivers ensures a predictable and well supported environment, with less potential for interoperability issues.

Note. These setup instructions for installing a JDBC driver on the application and batch servers are needed only if you use remote database sources for the Data Transformer feature of the Common Components. Do not install the driver otherwise.

Configuring Informix Application or Batch Servers for Remote Data Access

Before you can use remote data access with Informix on UNIX or Windows, Informix's database connectivity software (Informix Client 2.70 SDK) must be installed on the system where the application or batch server is running. If the application or batch server is running on the same machine as the Informix database server, remote data access might not work. Run the application or batch server on a machine different from the database server if you encounter a connectivity problem.

You must also install and configure the IBM Informix JDBC Driver, Version 2.21 on the same machine. The driver installation program is included as part of your Informix server software distribution — refer to your Informix JDBC Programmers Guide for details.

Important! You can install the JDBC driver anywhere. However, for proper remote data access functionality with your PeopleSoft system, you must install it under %INFORMIXDIR%/ifxjava_home.

If you're using the application server, you must manually edit the application server configuration file, psappsrv.cfg. If you're using the batch server, you must manually edit the batch server configuration file, psprps.cfg. These configuration files are located in *PS_HOME*\appserv*domain_name*\.

Under the line ";JavaVM Shared Library=" in the appropriate configuration file, add the following:

```
; RDBA Informix
Add to CLASSPATH=%INFORMIXDIR%/ifxjava_home/lib/ifxjdbc.jar
```

Configuring Oracle Application or Batch Servers for Remote Data Access

This section discusses how to:

- Prepare to configure Oracle.
- Configure Oracle 9i on UNIX.
- Configure Oracle 8 on UNIX.
- Configure Oracle 9i on Windows.
- Configure Oracle 8 on Windows.

Preparing to Configure Oracle

Before you can use Remote Data Access with Oracle, the appropriate database connectivity software must be installed on the system where the application server or batch server is running. The minimum database connectivity that needs to be installed is either Oracle Client Net9i 9.0.1.3.1 with Patch 5 Connectivity Package or Oracle Client Net8i 8.1.7 Connectivity Package. PeopleSoft recommends installing the Oracle Client Net9i package.

From the Remote Database Connection page (PeopleTools, Utilities, Administration, Remote Database Connection), you can specify an Oracle datasource as specific or with TNSNAMES. Use specific because it is simpler and faster and requires no further configuration on the part of the user. However, if TNSNAMES is desired and if the application server is being used, you must manually edit the application server configuration file, psappsrv.cfg, as described below. If the batch server is being used, you must manually edit the batch server configuration file, psprps.cfg, as well. (These configuration files live in <PS_HOME>/appserv/<DOMAIN>/ if you're on UNIX, or <PS_HOME>\appserv\<DOMAIN>\ if you're on Windows.)

Configuring Oracle 9i on UNIX

Determine the Oracle home directory and specify it in the configuration files by adding the following two lines under the ";JavaVM Shared Library=" section:

```
; RDBA Oracle 9i OCI driver
Add to CLASSPATH=%ORACLE_HOME%\jdbc\lib\classes12.jar
Add to CLASSPATH=%ORACLE_HOME%\jdbc\lib\nls_charset12.jar
```

Configuring Oracle 8 on UNIX

Determine the Oracle home directory and specify it in the configuration files by adding the following two lines under the ";JavaVM Shared Library=" section:

```
; RDBA Oracle 8 OCI driver
Add to CLASSPATH=%ORACLE_HOME%\jdbc\lib\classes12.zip
Add to CLASSPATH=%ORACLE_HOME%\jdbc\lib\nls_charset12.zip
```

Configuring Oracle 9i on Windows

Determine the Oracle 9i home directory and specify it in the configuration file. For example, if Oracle 9i's home directory is C:\Apps\DB\Oracle901, add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA Oracle 9i OCI driver
Add to CLASSPATH=C:\Apps\DB\Oracle901\jdbc\lib\classes12.jar
Add to CLASSPATH=C:\Apps\DB\Oracle901\jdbc\lib\nls_charset12.jar
```

Configuring Oracle 8 on Windows

Determine the Oracle 8 home directory and specify it in the configuration file. For example, if the Oracle 8 home directory is C:\Apps\DB\Oracle817, you should add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA Oracle 8 OCI driver
Add to CLASSPATH=C:\Apps\DB\Oracle817\jdbc\lib\classes12.zip
Add to CLASSPATH=C:\Apps\DB\Oracle817\jdbc\lib\nls_charset12.zip
```

Configuring DB2 UDB Application or Batch Servers for Remote Data Access

This section discusses how to:

- Configure DB2 UDB for Linux, UNIX and Windows.
- Configure DB2 UDB for OS/390 and z/OS.

Before you can use remote data access, you must install the appropriate database connectivity software on the system where the application server or batch server is running. In addition, you must modify the psappsrv.cfg and, if needed, the psprps.cfg configuration files.

Configuring DB2 UDB for Linux, UNIX and Windows

The minimum database connectivity software that needs to be installed is the IBM DB2 Run-Time Client. In addition, the application server configuration file, psappsrv.cfg, must be manually edited. If the batch server is being used, the batch server configuration file, psprps.cfg, must be manually edited as well. (These configuration files reside in *PS_HOME*/appserv/*domain*/, if you're on UNIX, or *PS_HOME*\appserv*domain*\ if you're on Windows.)

Note. The current product works only for the Version 7 type 2 driver as documented in this appendix. Support for the V8 Type 4 driver is not yet implemented.

To manually edit the file, determine DB2 UDB's home directory and specify it in the configuration file.

For example, if DB2 UDB's home directory is C:\Apps\DB\DB2ODBC7, you should add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA DB2
Add to CLASSPATH=C:\Apps\DB\DB2ODBC7\java12\db2java.zip
```

Or, if DB2 UDB's home directory is /mnt/db2/v7.1/java12, you should add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA DB2
Add to CLASSPATH= lmnt/db2/v7.1/java12/db2java.zip
```

Configuring DB2 UDB for OS/390 and z/OS

The database connectivity software that needs to be installed is IBM DB2 Connect. In addition, if the application server is being used, the application server configuration file, psappsrv.cfg, must be manually edited. If the batch server is being used, the batch server configuration file, psprps.cfg, must be manually edited as well. (These configuration files live in *PS_HOME*/appserv/*domain*.)

To manually edit the file, determine DB2 UDB's home directory and specify it in the configuration file. For example, if DB2 UDB's home directory is C:\Apps\DB\DB2ODBC7, add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA DB2
Add to CLASSPATH=C:\Apps\DB\DB2ODBC7\java12\db2java.zip
```

Configuring Sybase Application or Batch Servers for Remote Data Access

Before you can use remote data access with Sybase on UNIX or Windows 2000, Sybase's database connectivity software (Sybase 12.5 Client with Patch SWR 9988) must be installed on the system where the application server or batch server is running. In addition, if the application server is being used, the application server configuration file, psappsrv.cfg, must be manually edited. If the batch server is being used, the batch server configuration file, psprps.cfg, must be manually edited as well. (These configuration files live in*PS_HOME*/appserv/*domain*/, if you're on UNIX, or *PS_HOME*\appserv*domain*\ if you're on Windows.)

Under the ";JavaVM Shared Library=" section add the following lines:

```
; RDBA Sybase
Add to CLASSPATH=%SYBASE%/jConnect-5_5/classes/jconn2.jar
```

Installing and Configuring the Microsoft SQL Server JDBC Driver

Microsoft does not allow the redistribution of the JDBC driver for the Structured Query Language (SQL) server. Customers must download it from the Microsoft website, and install it into the *PS_HOME*\class directory.

To load and configure the Microsoft SQL Server JDBC Driver:

1. Download the Microsoft SQL Server 2000 Driver for JDBC from http://www.microsoft.com/sql/downloads.

Note. These drivers are not used by BEA Tuxedo or the web server, so you can ignore any comments on this site about supported versions of WebSphere or WebLogic.

2. Click on the link Windows(English)(2,096,944 bytes).
3. Save the program to disk in c:\temp or desktop.
4. Double-click the setup.exe file.
5. Accept all the defaults.

   The JDBC driver files are located in C:\program files\microsoft SQL server 2000 for JDBC\lib.
6. Copy all three files—mssbase.jar, mssqlserver.jar,msutil.jar—to *PS_HOME*\class.

APPENDIX H

Archiving Data (Deprecated)

This chapter provides overviews of PeopleSoft Data Archive Manager, archiving strategies, and archiving techniques, and discusses how to:

- Create and design archive templates.
- Work with the archives.
- Work with archive data.
- Run data archival processes.
- Run data archival reports and audits.
- Perform additional archival procedures.

Note. The Data Archive Manager documented in this chapter is a deprecated feature used in PeopleSoft 8.40 through 8.43. It has been replaced in the current release by a new Data Archive Manager, which is documented in this PeopleBook.

See Also

Chapter 3, "Using PeopleSoft Data Archive Manager," page 49

Understanding PeopleSoft Data Archive Manager

PeopleSoft applications create and maintain data. Often the data in online tables is no longer required, but you cannot simply delete the data to make room for new data. Managing the historical data is a time-consuming challenge for many database administrators. However, unless you develop an archiving strategy, databases increase to unmanageable sizes. Also, removing the historical data from online tables can improve overall performance.

PeopleSoft Data Archive Manager enables you to select the rows of data that you no longer need in the online system and move those rows into history and staging tables or into a flat file format. Maintaining the data in the system in history and staging tables keeps the data available for queries and reporting.

Also, you can archive data directly to a flat file for long-term storage if the historical data is no longer needed for reporting. You can export the data to flat files and delete it completely from the system.

Warning! PeopleSoft Data Archive Manager is a powerful tool. Improper use may result in data loss or corruption of the database. Only experienced database administrators who understand the archival process should use PeopleSoft Data Archive Manager.

Understanding Archiving Strategies

This section discusses:

- Types of strategies.
- History tables and staging tables.
- Flat files.

Types of Strategies

Determining an archiving strategy is essential for using PeopleSoft Data Archive Manager efficiently. This strategy depends on how the archived data will be used. The following table describes the two main archiving strategies.

Archiving to History Table Strategy	Archiving to Flat File Strategy
• Uses history tables for storing archived data. • Enables reporting and queries from history tables. • Requires secondary step to delete archived data from online tables. • Requires additional database space.	• Archives data directly to a flat file. • Deletes the archived data from the online tables. • Provides efficient, one-step archiving process. • Requires no additional database storage space.

The two strategies offer different approaches to archiving. If you are archiving to history tables, you must first create the history tables for temporary storage. If you are archiving to flat files, you can accomplish the task in one simple step. The system is designed to provide as much flexibility as possible. By reviewing your business requirements, you will be able to determine which strategy best fits your business needs.

Note. Before deciding which archiving strategy to use, review this entire chapter so that you are familiar with all of the archiving features.

History Tables and Staging Tables

Archiving to history tables involves using tables that you create for the sole purpose of storing archived data. You must determine whether the archived data should be stored in the history tables temporarily or on a long-term basis.

By definition, history tables are identical copies of the online tables, except they have an additional column, PSARCH_ID. The system uses this key field to denote when a piece of data was archived and to uniquely identify it. Some PeopleSoft applications deliver history tables prebuilt for use in common archiving processes. If you design a custom archiving scheme, you need to create the history tables using PeopleSoft Application Designer.

Here is a high-level-overview of the steps:

1. You move data into the history tables.

   This is known as the selection process. This enables you to query the selected data for information and delete the data from the online tables.

2. If you accidentally delete the data from the online tables, there is a process to restore the data back from the history tables.

   This rollback process is the optional second step.

3. When you no longer need to reference the data from the history tables, you can export them to flat files and delete them completely from the system.

4. If necessary, you can return the archived data back to the system using staging tables.

   Staging tables are identical to online tables.

History Table Considerations

When archiving to history tables, it is important to consider the following points:

- History tables can reside on the same database as the online tables, or they can reside in separate databases.

  If the history tables reside in a separate database, you need to set up database links using the proprietary methods for the relational database management system.

- After the archive process moves the data into the history table, the data resides in both the online tables and in the history table; you then have two options:

  - Deleting the archived data from the online tables.

  - Leaving the archived rows in the online tables such that the data exists in parallel.

Procedure to Build History Tables

Before you run the archiving process, you must first create (or build) the history tables.

You must build one history table for each table to be archived. The history table must be identical to the archive table, with an extra column PSARCH_ID.

The following example uses the record JRNL_HEADER.

To build a history table:

1. Open PeopleSoft Application Designer.
2. Open the JRNL_HEADER table.
3. Select File, Save As and name the history table with an appropriate name, such as JRNL_HEADER_HST.
4. When prompted to copy the PeopleCode associated with the table, click No.
5. Select Insert, Field and insert the PSARCH_ID field.
6. Save the record.
7. Build the table by selecting Build, Current Object.
   - Select the following build options: Create Tables and Create Indexes.
   - Select the following build execute options: Execute and Build script.
   - Click Build.

Staging Table Considerations

Staging tables are used in the archiving process as a holding area for data. When using staging tables, consider the following:

- Staging tables are exact duplicates of the online tables.

They do not contain the PSARCH_ID column.

- Staging tables are often used for reporting purposes.

Flat Files

You can also archive data directly from online tables to flat files. When you are ready to archive data and you no longer need to reference it from the online tables, you can extract the data to flat files.

The data is stored in the comma-separated values (CSV) format, which enables a portable representation of the online tables and data.

To archive to flat files:

1. Export the selected online rows to a CSV file.
2. If you need to access the archived data in the future, you can restore the data into staging tables.

Note. The data is deleted from the online tables as it is archived.

Understanding Archiving Techniques

This section discusses:

- Business requirements analysis.
- Commits.
- Performance enhancement.
- Index limitations.
- Data limitations.

Business Requirements Analysis

It is important to devise a business strategy before archiving the data. First, you must identify the tables that you want to archive. This includes identifying all of the parent and child tables associated with the tables. Failing to identify all of the related tables can cause corruption to the database. Next, you must know exactly which data to archive. It is important to recognize which rows are safe to remove from the online tables. Remember to remove only the data that is not required to maintain the day-to-day business and reporting.

Consider PeopleSoft General Ledger as an example. General Ledger contains the greatest amount of data to be archived because it is the module where the majority of reporting is required. There are two sets of data types that need to be maintained: balance information and transactional information. Balance information is retained in the ledger records. You might require balance information for online and reporting purposes to be available for a three-year period. On the other hand, transactional data is maintained in the journal header and line tables. Suppose that you require only one year of transactional data to be retained in the system for online purposes, but three years to be retained for reporting purposes.

Any data beyond the above time frames for balances and transactions can be archived and is only be accessed through reports. The data can be archived to history tables or as flat files to secondary storage devices. If data were to be archived into history tables, the data would still be available online for reporting purposes. However, you could not view it through standard PeopleSoft Internet Architecture pages without special configuration. In addition, reports would need to be modified to access the data in history tables. Archiving to secondary storage devices is generally used for long-term data retention. This option is preferred for data that is rarely retrieved, and secondary storage devices are usually used to satisfy legal requirements.

Commits

For both batch and online execution, the Archive Selection, Remove from History, Rollback, and Delete processes issue commits after each record has been processed unless a commit level has been specified otherwise.

Performance Enhancement

For better performance and increased speed during archiving processes, try dropping the indexes before inserting data from online tables into history tables.

Index Limitations

The database platform may have a limitation on the number of columns that an index can contain. Some have a restriction of 16 columns for an index. If the table that you want to archive already has 16 keys, then you cannot add another key (PSARCH_ID) to the corresponding history table.

To solve this problem, you can use the flat file archiving strategy. Alternatively, you can create the history table with the PSARCH_ID as a non-key field. For this situation, it is recommended that you either have different history tables for different archiving projects, or that you purge the history tables before executing the selection process of another archiving project.

Data Limitations

Due to platform and meta-SQL restrictions, the Data Archive Manager for PeopleTools 8.4x does not support archiving of records with LONG, IMAGE, or ATTACHMENT columns. The selection process (inserting data from the online records to the history records) will result in the loss of the long, image, or attachment columns in the history record.

Creating and Designing Archive Templates

This section discusses how to:

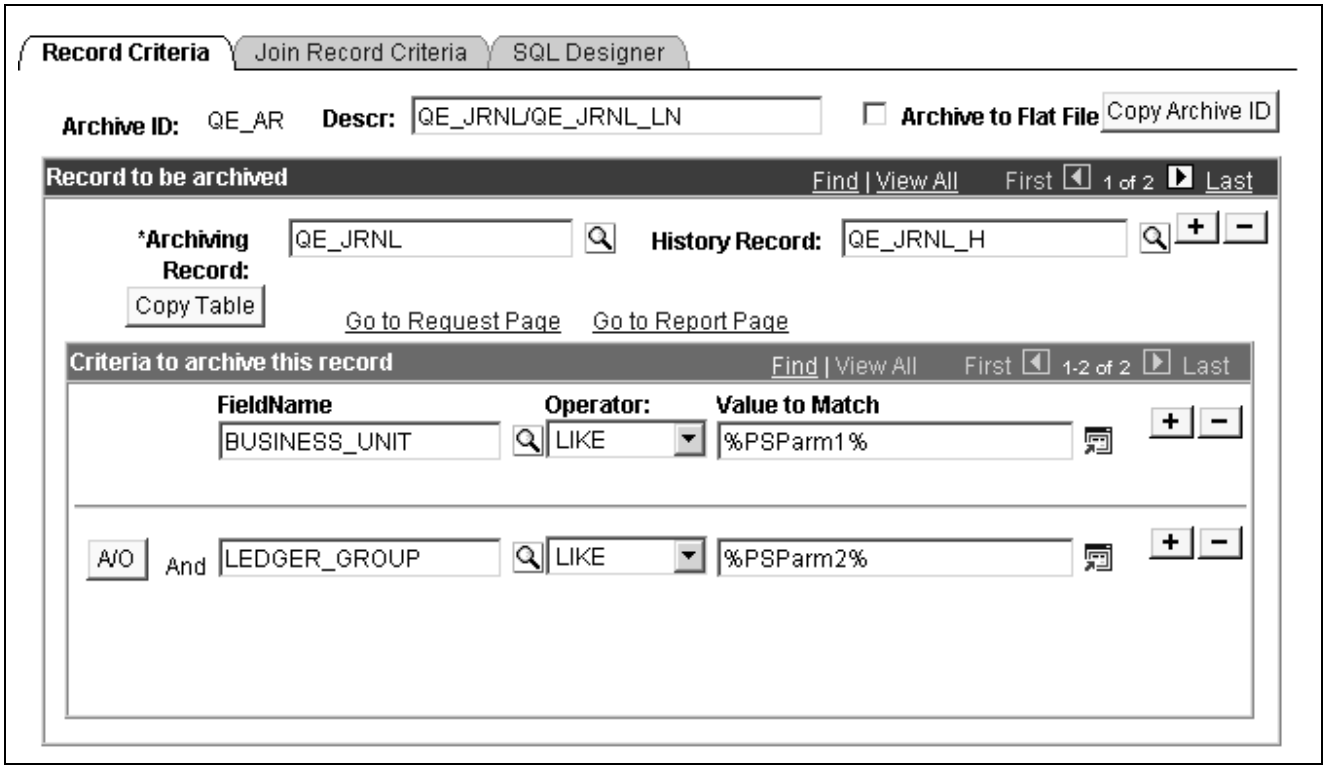
- Specify fields and archival criteria.
- Join record criteria.
- Generate and edit Structured Query Language (SQL).

Pages Used to Create and Design Archive Templates

Page Name	Object Name	Navigation	Usage
Record Criteria	ARCH_PROJ	PeopleTools, Archive Data, Archive Designer, Record Criteria	Use this page to specify fields and archival criteria.
Join Record Criteria	ARCH_OTH_CTRL	PeopleTools, Archive Data, Archive Designer, Join Record Criteria	Use this page to join the record criteria.
SQL Designer	ARCH_SQL	PeopleTools, Archive Data, Archive Designer, SQL Designer	Use this page to generate and edit the SQL that will be used to perform the archive process.

Specifying Fields and Archival Criteria

Access the Record Criteria page.



Record Criteria page

The process of archiving data begins with the creation of an archiving template, which logically groups all of the online tables that are to be archived into a single entity. You associate the online table with its history table counterpart, and you select the fields to archive and the criteria by which to archive.

Archive ID Displays the ID for a group of transactions that comprise an archive definition during the archiving process.

Description Enter a description. Use up to 30 characters to describe the archive.

Archive to Flat File Select to archive the project *directly* to a flat file without having to create history tables.

Copy Archive ID Click to copy the current archive project to a new archive ID. All tables, criteria, and other criteria are copied to the new archive ID.

Archiving Record Select the online tables to be archived. You can archive multiple online tables within one archive ID.

History Record Enter the name of the table where the archived data will be stored.

Copy Table Click to copy all criteria to a new row in the existing archive ID. This button is useful when handling multiple tables.

Go to Request Page Click to access the Archiving Process page.

Go to Report Page Click to access the Report Request page.

FieldName Enter columns in the online tables to specify archive criteria. Specifying the fields and adding the conditions is comparable to the WHERE clause in a SQL statement.

Operator Select an operator. Options are =, <>, <, >, <=, >=, LIKE, and NOT LIKE.

Value to Match Enter a column value to match against, as in 07/01/1999 or $75,000.

You can also use special parameter markers in the format of %PSPARMnn% where *nn* can be any number. For example, valid parameter markers could be %PSPARM1% or %PSPARM18%.

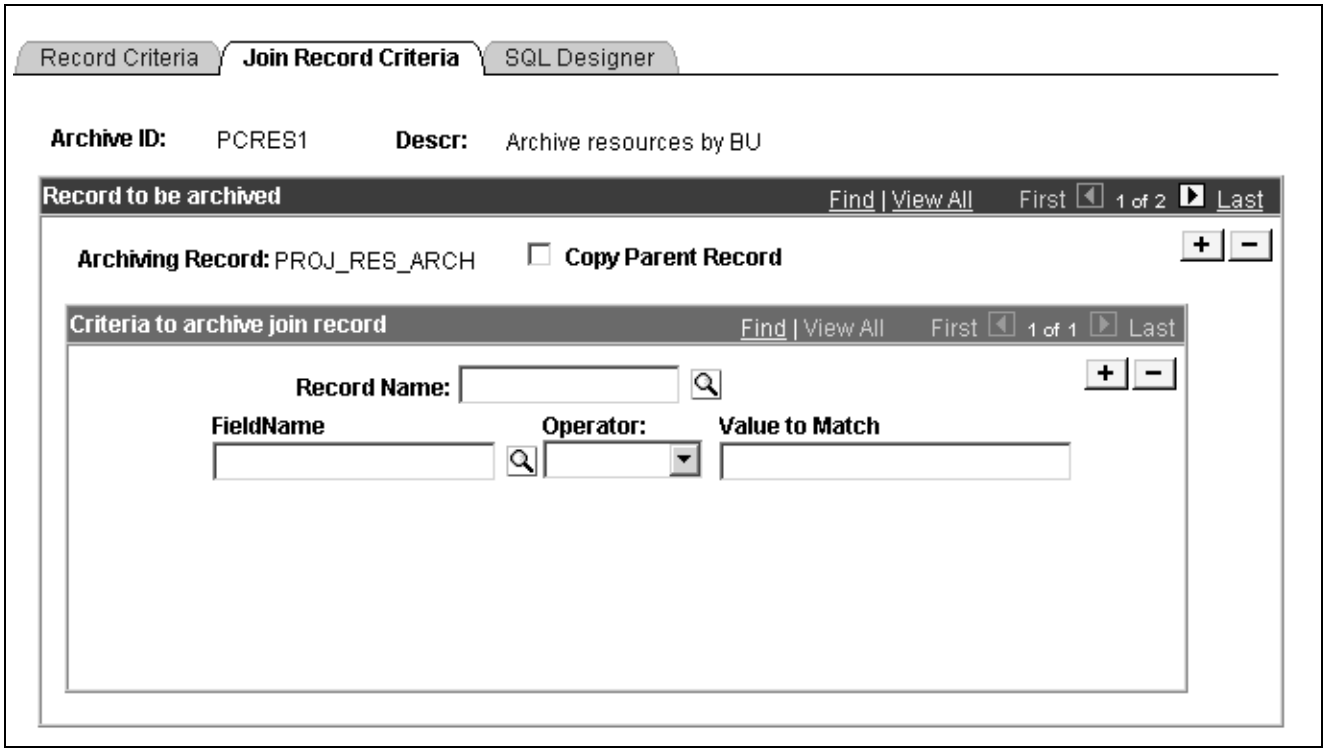
When the system generates the SQL statement, %PSPARMnn% is embedded into the SQL statement and substituted with values entered using the run control pages. For example, you can create an archive project based on a business unit and then enter the actual business unit at run time.

Note. Parameter markers are currently not implemented with DATE fields.

A/O Click to specify AND or OR. This button is only visible if you add multiple lines to the field list.

Joining Record Criteria

Access the Join Record Criteria page.



Join Record Criteria

If there are dependencies from other tables in the archiving template, such as parent-child relationships or joining against reference tables, you must include the criteria on this page. This can also be done by selecting the Copy Parent Record check box. For this to work correctly, the parent table criteria must already exist on the Record Criteria page. You can specify multiple levels, such as grand-parent-to-parent, grand-parent-to-parent-to-child, and so on.

Archiving Record Displays the table to be archived.

Copy Parent Record Select to enable the criteria that exist in the parent record on the Record Criteria page to be copied to the Join Record Criteria page. When you select this check box, an edit box appears for you to select the parent table.

Record Name Enter the name of the table to be joined. You can request multiple table joins per archiving table. The two tables must share common keys.

Field Name Enter the columns of the online tables to add to the archive criteria.

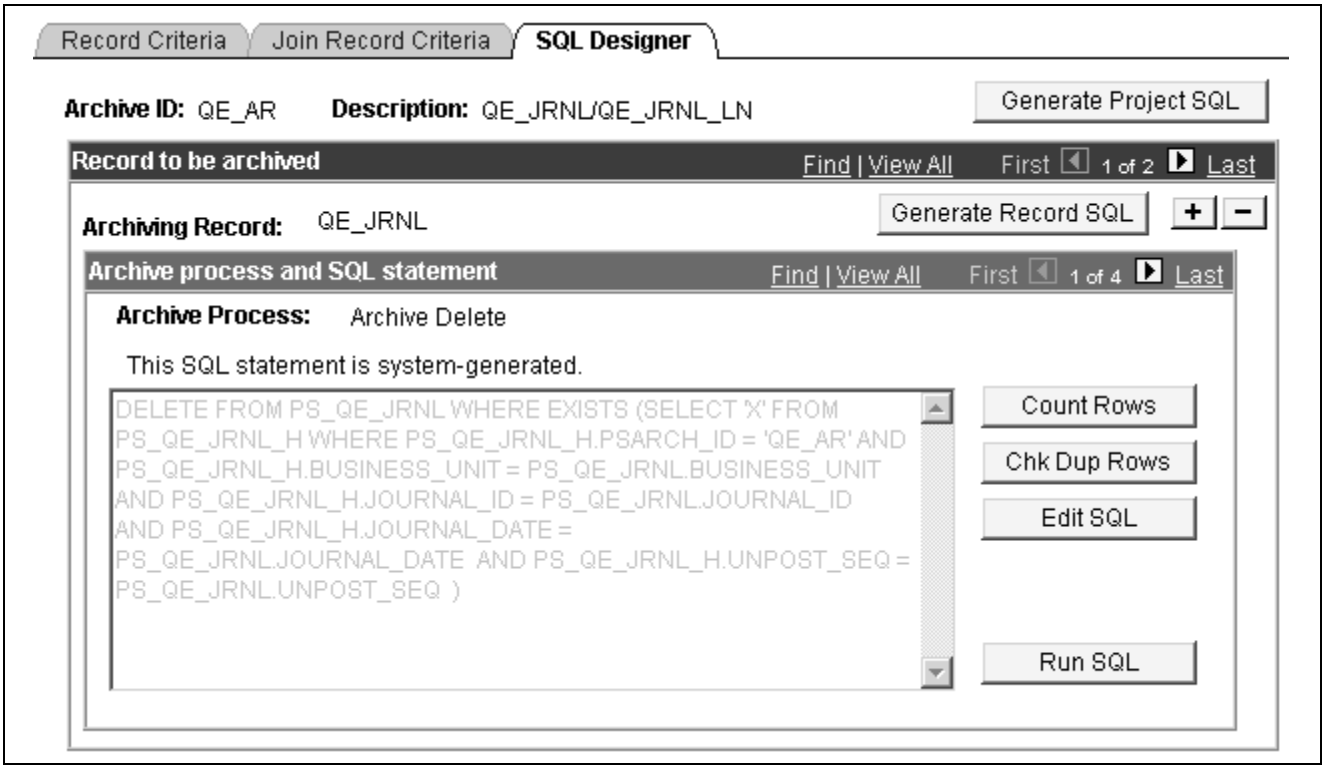
Operator Select an operator. Options are =, <>, <, >, <=, >=, LIKE, and NOT LIKE.

Value to Match Enter a column value to match.

A/O Click to select AND or OR.

Generating and Editing SQL

Access the SQL Designer page.



SQL Designer page

The SQL Designer page is useful for generating and editing the SQL that will be used to perform the archive process. In addition, you can count the number of rows that will be affected by the current archive process and check for duplicate rows that the SQL is affecting. To access this page, you must have entered basic information on the Record Criteria page and the Join Record Criteria page.

Note. The buttons that appear on this page depend on your security access privileges and the current archive setting. To set security access to the page, access the Archive Security page.

Generate Project SQL Click to create all SQL statements for the entire archive template.

Generate Record SQL Click to produce the SQL statements for the current record. The following types of SQL are created:

- Delete from the online tables (Archive Delete process).
- Remove data from history tables (Remove from History process).
- Roll back (Archive Rollback process).
- Create SELECT that moves rows from the online table to the history table (Archive Selection process).

Archive Process Displays the processes that have been selected on the Archive Data page.

Count Rows Click to view the row count that the generated SQL will affect.

Chk Dup Rows (check duplicate rows) Click to see if an incorrect join will cause duplicate rows to be archived.

Edit SQL Click to modify the generated SQL. If you edit and save the SQL, a flag is used to indicate that the SQL is user-modified and is not system-generated.

When you modify the SQL and save it, the text above the edit box indicates that the SQL has been altered from the original, system-generated SQL.

Run SQL Click to run the generated SQL. Typically, this button used by the archive developer during the development and testing of the archive. After the archive template is developed, a PeopleSoft Application Engine program runs the SQL in batch.

Working With the Archives

This section discusses how to:

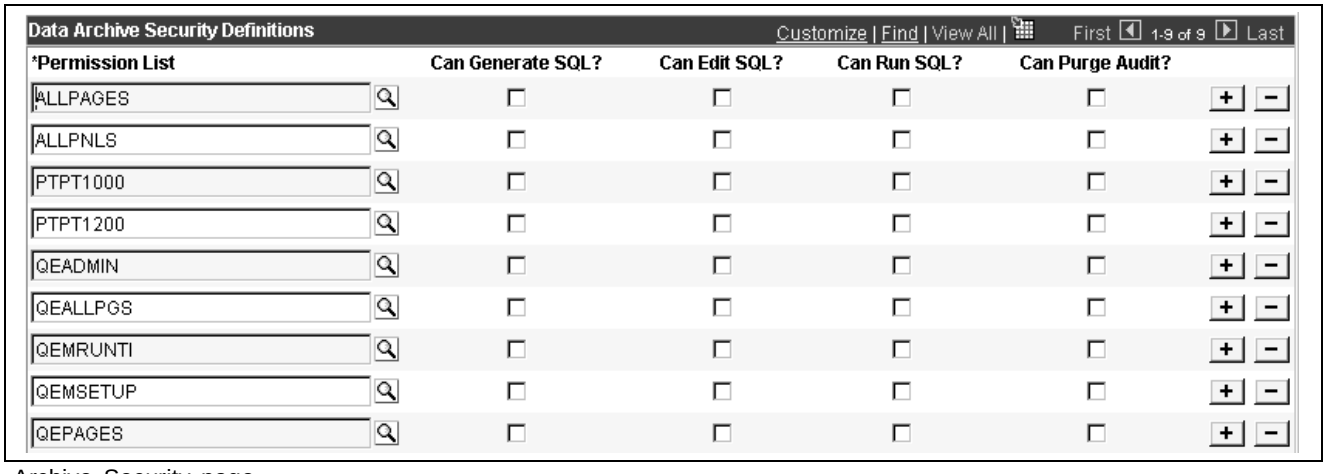
- Grant access rights.
- Administer archive projects.

Pages Used to Work with the Archives

Page Name	Object Name	Navigation	Usage
Archive Security	ARCH_SECURITY	PeopleTools, Archive Data, Archive Security	Use this page to grant access rights to the permission lists that use PeopleSoft Data Archive Manager.
Archive Utilities	ARCH_UTILS	PeopleTools, Archive Data, Archive Utilities	Use this page to perform administrative tasks associated with the archive projects.

Granting Access Rights

Access the Archive Security page.



Archive Security page

This page enables you to grant access rights to the permission lists that use PeopleSoft Data Archive Manager.

The permission lists that you add need to exist already in PeopleTools Security. The permission list must be the primary permission list for the user profile in order for the user to have access to the command button.

Permission List Select the permission lists to which you grant archive data access.

Can Generate SQL? Select to enable the user to generate SQL on the SQL Designer pages by activating the Generate SQL button.

Can Edit SQL? Select to enable the user to edit the SQL on the SQL Designer page.

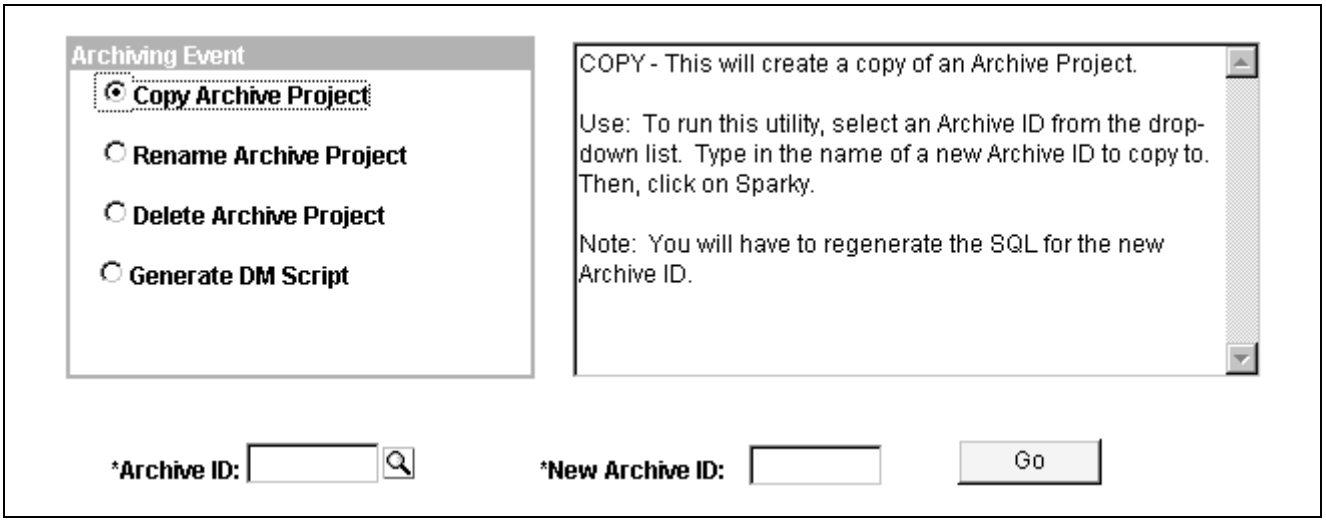
Can Run SQL? Select to enable the user to run SQL on the SQL Designer page.

Can Purge Audit? Select to enable the user to purge the audit history on the Archiving Audit page.

Note. PeopleTools delivers a process group ARCHALL that should be added to the appropriate permission list using PeopleTools Security. You cannot run any archive processes until this group is added to the permissions list.

Administering Archive Projects

Access the Archive Utilities page.



Archive Utilities page

The Archive Utilities page is used for archive project administration. The administrative operations include copying, renaming, deleting, exporting and importing.

Select the action that you want to perform. The instructions for each action appears in the text box to the right.

Copy Archive Project Select to Create a copy of the archive template that you specify in the Archive ID field and give it the name that you enter in the New Archive ID field.

Rename Archive ID Select to rename the archive according to the values that you enter in the Archive ID and New Archive ID fields.

Delete Archive ID Select to delete the archive that you specify in the Archive ID field. Once an archive is deleted, you can no longer access it.

Generate DM Script Select to generate both the import and the export PeopleSoft Data Mover scripts for the archive. You can also export all projects by selecting the corresponding check box.

Working With Archive Data

This section discusses how to:

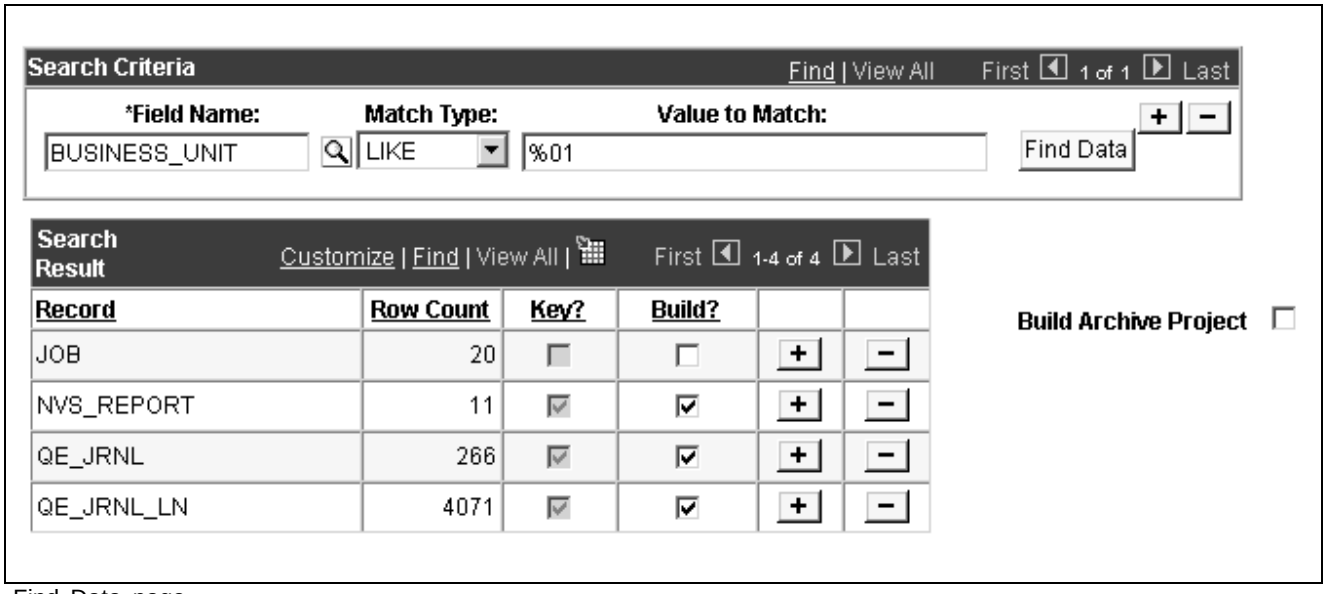
- Find data that meets your criteria.
- Create scripts to move data.
- View and edit scripts.

Pages Used to Work with Archive Data

Page Name	Object Name	Navigation	Usage
Find Data	DATA_FIND_INPUT	PeopleTools, Archive Data, Find Data	Use this page to find data in the online system that meets your criteria for your archive project.
Data Transfer Input	DATA_TRANS_INPUT	PeopleTools, Archive Data, Transfer Data, Data Transfer Input	Use this page to search for specific fields in the database and create PeopleSoft Data Mover export and import scripts to move the data between databases.
Data Transfer Output	DATA_TRANS_OUTPUT	PeopleTools, Archive Data, Transfer Data, Data Transfer Output	Use this page to export and import data by running the PeopleSoft Data Mover scripts.

Finding Data That Meets Your Criteria

Access the Find Data page.



Find Data page

This page enables you to find data in the online system that meets your criteria. Once you have located the data, you can then immediately create a new archive project.

Field Name Enter the name of the field that you want to find a match for.

Match Type Specify whether the match between the field value and the match value should be equal or *like*. Options are = and *LIKE*.

Value to Match Enter the value for the system to search for within the specified field name.

Find Data After you have entered the desired criteria, click Find Data for the system to begin searching the online data.

Record Displays the record containing the rows that meet the criteria.

Row Count Displays the number of rows in the record that meet the criteria.

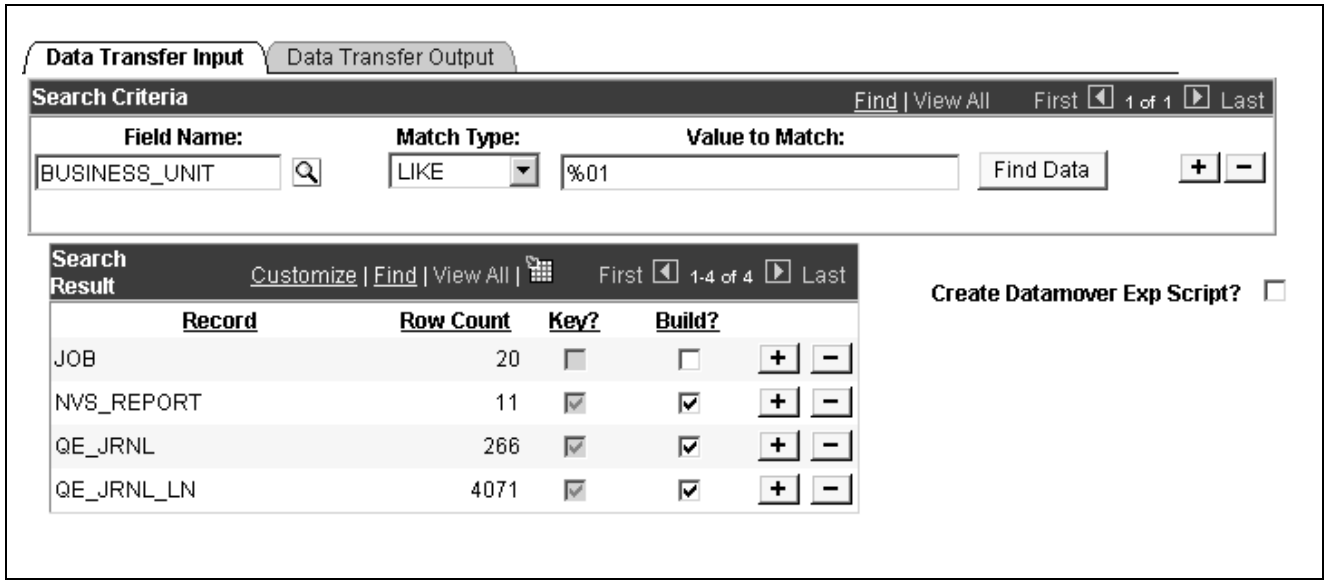
Key? Indicates whether the field is a key field in the record.

Build? Click to include the record in the generated archiving project. This check box is automatically selected if the field is a key field.

Build Archive Project Select to create a new archive project. After entering the name and description of the project, click Go to display the Archive Designer pages that enable you to create the new archive.

Creating Scripts to Move Data

Access the Data Transfer Input page.



Data Transfer Input page

The Data Transfer Input page enables you to search for specific fields in the database and create PeopleSoft Data Mover export and import scripts to move the data between databases. For example, this can be used to generate Data Mover scripts that can export data from the production database into a training or test database.

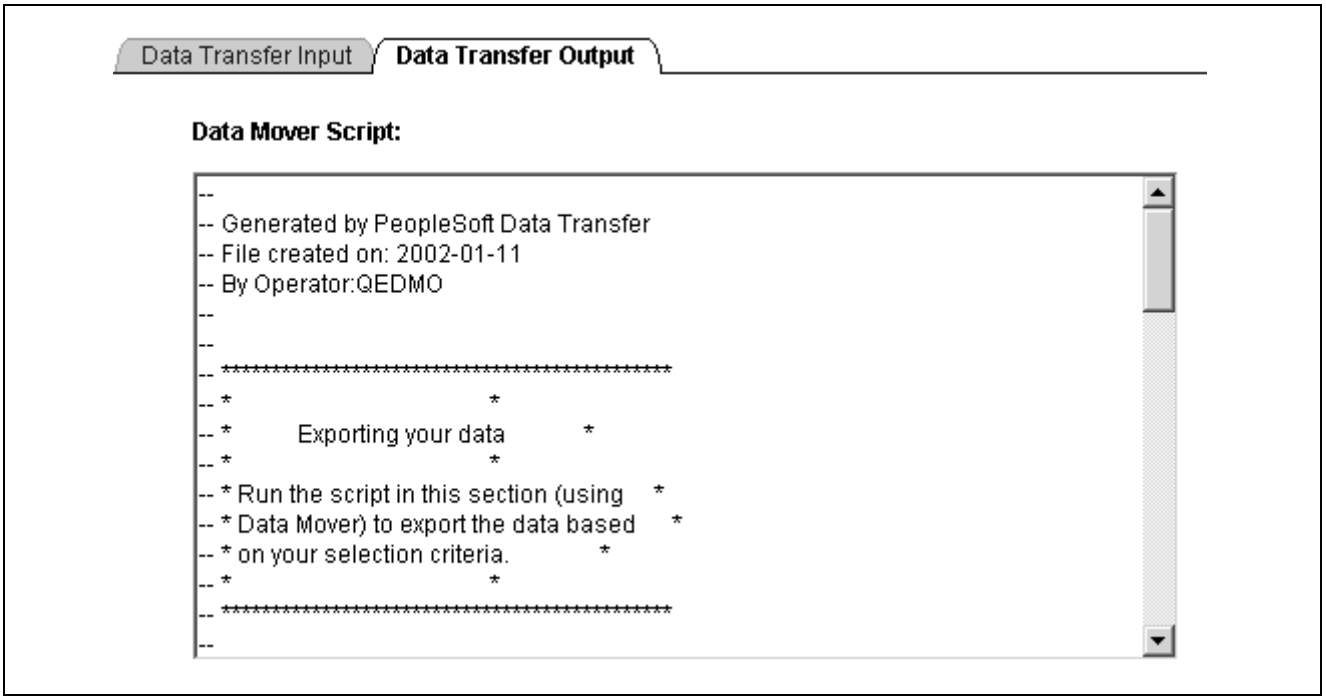
Field Name Enter the name of the field for which you want to find a match.

Match Type	Select the type of match between the field value and the match value. Options are = and *LIKE*.
Value to Match	Enter the value for the system to search for within the specified field name.
Find Data	Click to search for the values that you've specified.
Record	Displays the record containing the rows that meet the criteria.
Row Count	Displays the number of rows in the record that meet the criteria.
Key?	Indicates whether the field is a key field in the record.
Build?	Click to include this record in the generated script. This check box is automatically selected if the field is a key field.
Create Data Mover Export Script	Select to enable the system to create a PeopleSoft Data Mover export script.
Data Mover File Path	Enter the file path of the PeopleSoft Data Mover files in the generated script.
Data Mover Export File Name	Enter the export file name in the generated script.
Data Mover Import File Name	Enter the import file name in the generated script.
Delete Before Import	If this check box is selected, the generated script includes a DELETE statement for the user-specified criteria in the WHERE clause. The DELETE statement appears before the Data Mover IMPORT statement.

Note. The generated PeopleSoft Data Mover script appears in a text box on the Data Transfer Output page.

Viewing and Editing Scripts

Access the Data Transfer Output page.



Data Transfer Output page

The Data Transfer Output page contains two PeopleSoft Data Mover scripts: one to export the data and one to import the data.

To run the script using PeopleSoft Data Mover, you need to copy the generated script to a text editor (Notepad, for example) and then save it as a Data Mover script (DMS) file.

Running Data Archival Processes

This section discusses how to:

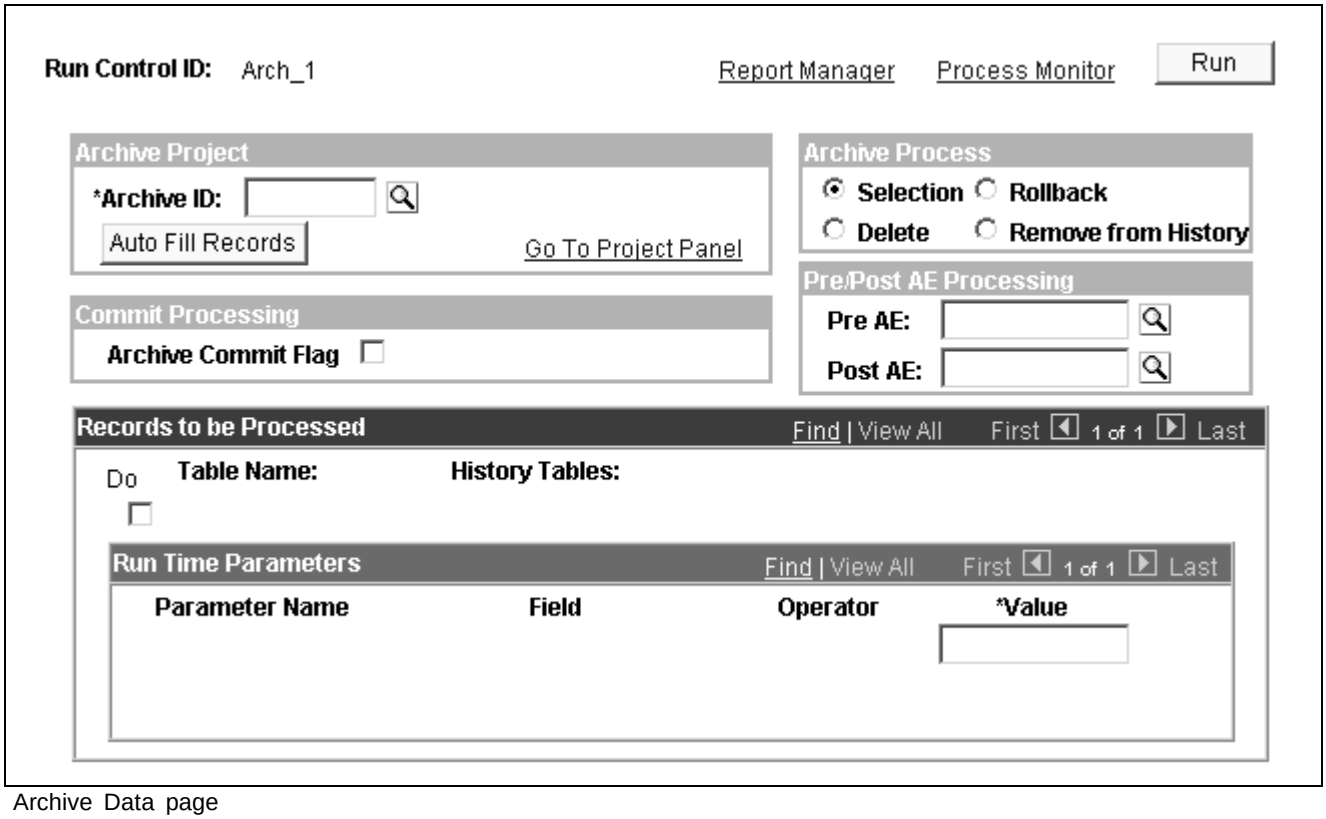
- Begin the archiving process.
- Export data from online tables to flat files.
- Export data from history tables to flat files.
- Restore archived data using staging tables.

Pages Used to Run Data Archival Processes

Page Name	Object Name	Navigation	Usage
Archive Data	ARCH_RQST	PeopleTools, Archive Data, Archive Data	Use this page to archive the selected data from your online tables.
Archive Online to Flat Files	ARCH_FLT_RQST	PeopleTools, Archive Data, Archive Online to Flat Files	Use this page to export data from online tables to flat files.
Export History to Flat Files	ARCH_HST_RQST	PeopleTools, Archive Data, Export History to Flat Files	Use this page to export data from history tables to flat files.
Import from Flat Files	ARCH_IMP_RQST	PeopleTools, Archive Data, Import from Flat Files	Use this page to restore archived data using staging tables.

Beginning the Archiving Process

Access the Archive Data page.



Archive Data page

Once you have created an archive template, you can begin the archiving process. This is when the system moves the selected data from your online tables.

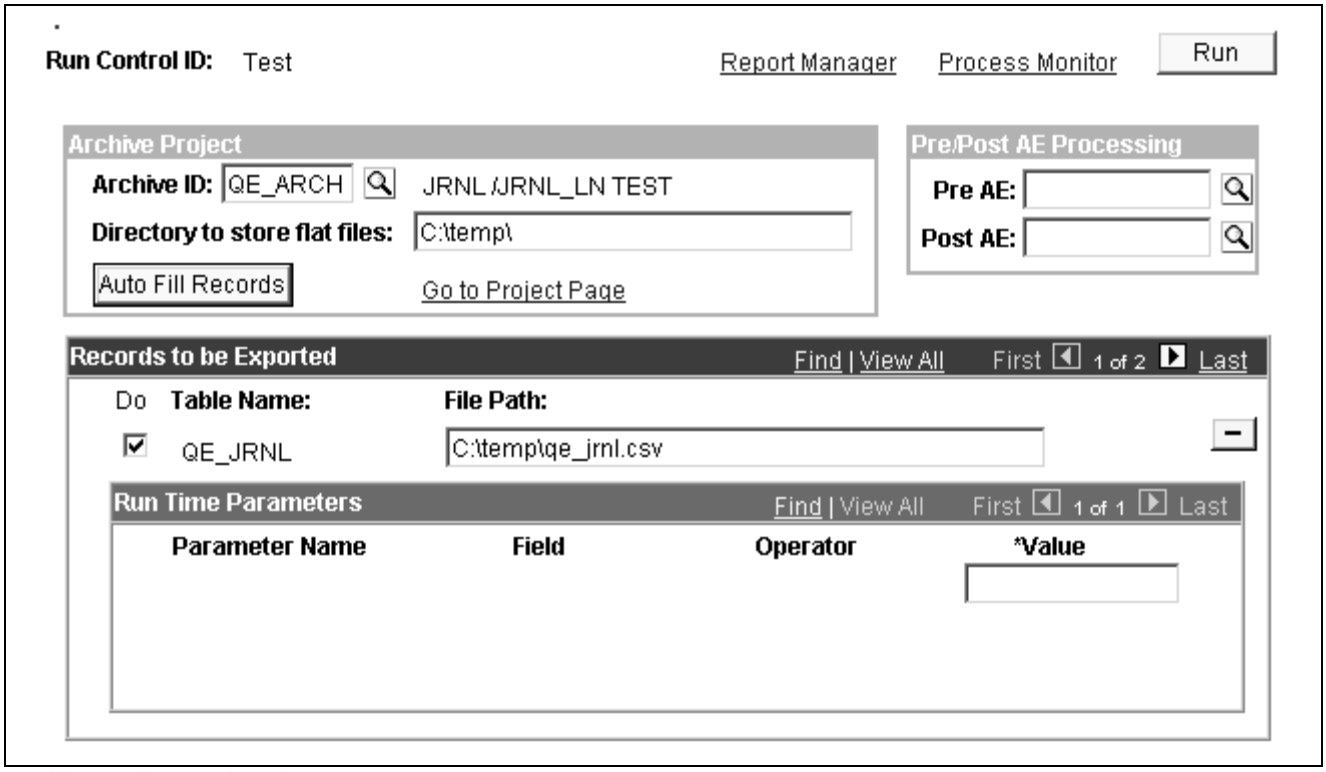
Archive ID Select an existing archive ID.

Auto Fill Records Click to display all the tables to be included in the archive project.

Go to Project Page	Click to access the Archive Designer pages.
Archive Process	Select the archive process to run. Options are: • Selection: Copy the data from the online tables to the history tables. • Rollback: Copy the data from the history tables to the online tables. • Delete: Remove the data from the online tables, but only when they have already been copied to the history tables. • Remove from History: Delete the data from the history tables.
Commit Processing	Select to enable the Commit After option, which specifies how many rows of data the system processes before issuing a database commit. Otherwise, the system issues a commit after each record has been processed.
Pre/Post AE Processing (pre/post Application Engine processing)	If you have any custom PeopleSoft Application Engine programs that you want to run against the data, either before or after archiving, specify the appropriate program here. The available fields are: • Pre AE: Select an existing Application Engine program to run before the archiving process. • Post AE: Select an existing Application Engine program to run after the archiving process.
Do	Select to indicate whether the table is processed at run time
Table Names	Displays the tables containing the data to be archived.
History Tables	Displays the tables where the system stores the archived data.
Run Time Parameters	If your archive template contains runtime parameters (%PSPARMnn%), specify a value that the system substitutes in the SQL statement at run time.

Exporting Data From Online Tables to Flat Files

Access the Archive Online to Flat Files page.



Archive Online to Flat Files page

Note. This archive process deletes the data from the online tables immediately after the system has exported it to the flat files.

Run Click to export the data to the flat files.

Archive ID Select an existing archive ID.

Directory to store flat files Enter the path for the directory in which you want to store the resulting flat files. When you click Auto Fill Records, the directory is added as a prefix to each of the resulting file names.

Auto Fill Records Once you've selected an archive ID, click to display the table names associated with the archive. In addition, you must specify the path where the data will be exported.

Go to Project Page Click to access the Archive Designer pages.

Pre/Post AE Processing (pre/post Application Engine processing) If you have any custom PeopleSoft Application Engine programs that you want to run against your data either before or after archiving, specify the appropriate program here. The available fields are:

- Pre AE: Select an existing Application Engine program to run before the archiving process.
- Post AE: Select an existing Application Engine program to run after the archiving process.

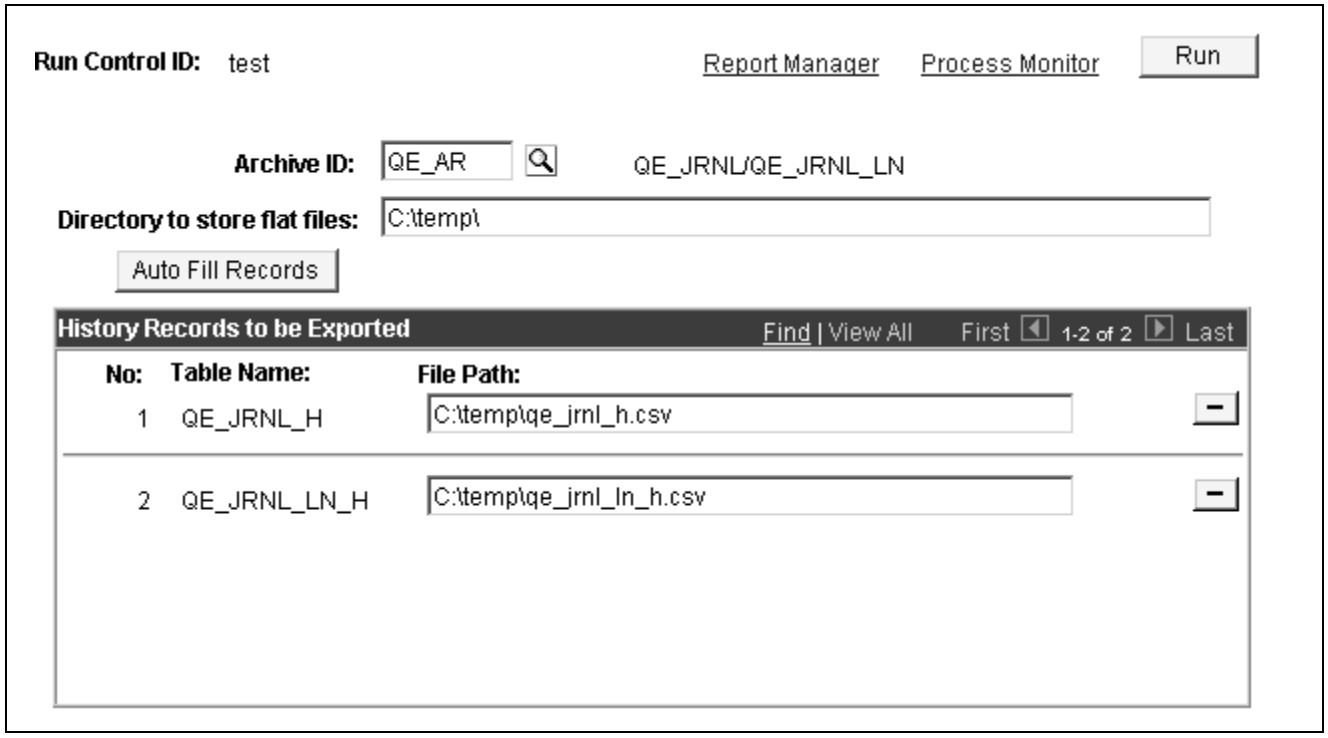
Do Select to specify whether to process this table.

File Path Displays path and file names to which the data for each table will be written. If necessary, you can manually enter the file path so that a table has unique file name and location.

Run Time Parameters If the archive process contains the runtime parameter markers (%PSPARMnn%), specify a value that the system substitutes in the SQL statement at run time.

Exporting Data From History Tables to Flat Files

Access the Export History to Flat Files page.



Export History to Flat Files page

Run Click to export the data in the history table to the designated flat file.

Archive ID Select an existing archive ID.

Directory to store flat files Enter the path for the directory in which to store the resulting flat files. When you click Auto Fill Records, the path is added as a prefix to each of the resulting the file names.

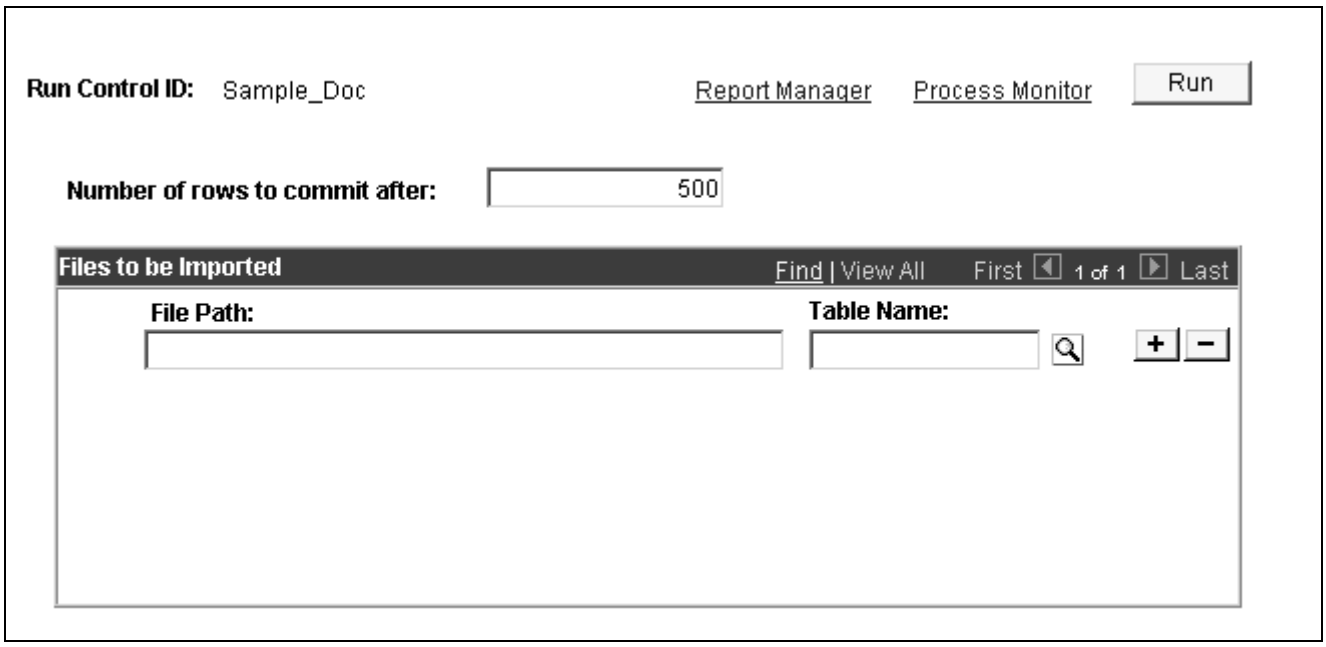
Auto Fill Records Click to display the table names. In addition, you must specify the path where the data is exported.

Table Name Displays the name of the table that is processed. This field is populated by clicking the Auto Fill Records button.

File Path Displays the path and file names to which the data for each table is written. If necessary, you can manually enter the file path so that a table has a unique file name and location.

Restoring Archived Data Using Staging Tables

Access the Import From Flat Files page.



Import From Flat Files page

Run	Click to import the data from the flat file into the designated staging table.
Number of rows to be processed before commit to DB	Enter the number of rows that the program processes before it issues a COMMIT.
File Path	Enter the path and the name of the file containing the flat file that you want to import into the database.
Table Name	Enter the staging table name where the program inserts the data from the import file.

Running Data Archival Reports and Audits

This section provides an overview of archival reports and audits and discusses how to:

- Run archive reports.
- Create an audit anquiry.
- View audit results.

Understanding Archive Reports and Audits

Before running the archive process, you should generate reports to verify the data that you are archiving before deleting it from the online tables. This report lists the archive ID definitions, which consist of the following:

- Archiving tables.

- Selection criteria.
- Dependency criteria (criteria from other tables).
- SQL that will run each of the archiving processes.

These definitions also help you create your own reports through PeopleSoft Query.

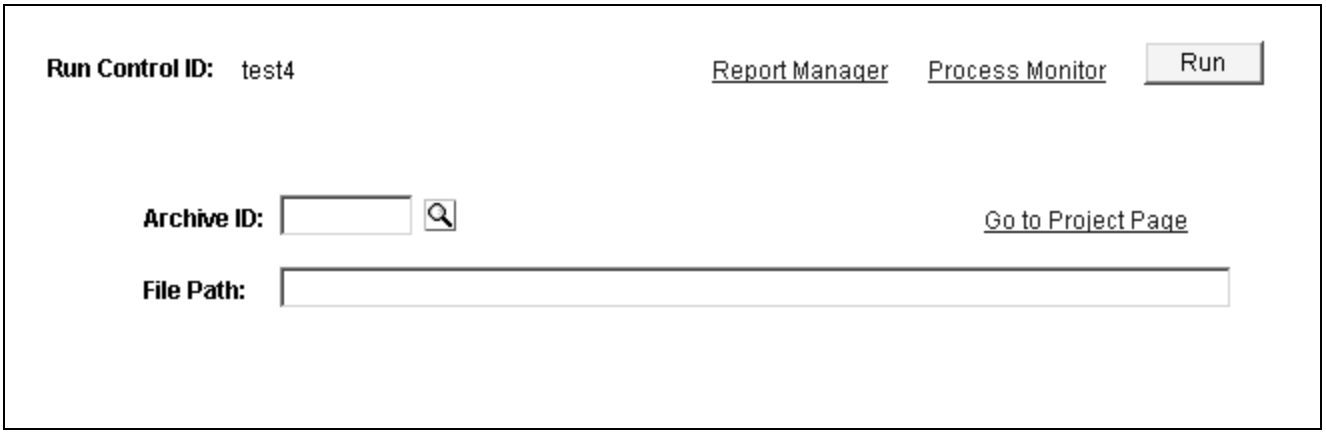
The Audit Report page generates report for a specific archive ID and places the file in the destination that you provide.

Pages Used to Run Data Archival Reports and Audits

Page Name	Object Name	Navigation	Usage
Archive Report	ARCH_RPT	PeopleTools, Archive Data, Archive Report	Use this page to run archive reports.
Audit Inquire	ARCH_AUDIT_INQ	PeopleTools, Archive Data, Audit Archiving, Audit Inquire	Use this page to view all online processes that have been executed in the Data Archiving tool without waiting for the output from the batch process. You can also delete the audit if you have the necessary access privileges.
Audit Report	ARCH_AUDIT_RPT	PeopleTools, Archive Data, Audit Report	Use this page to view all processes that have been run in the Data Archiving tool. You can also print the resulting SQL.

Running Archive Reports

Access the Archive Report page.



Archive Report page

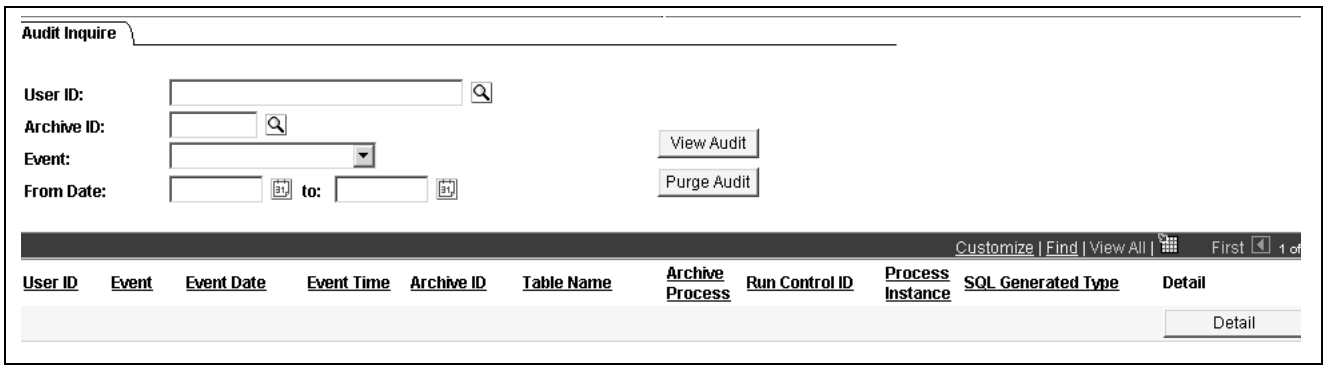
Archive ID Select the archive ID to audit.

Go to Project Page Click to access the Archive Designer pages.

File Path Enter the path where the generated report is saved on the batch server.

Creating an Audit Inquiry

Access the Audit Inquire page.



Audit Inquire page

The Audit Inquire page is useful for online viewing of all processes that have been executed in the Data Archiving tool without waiting for the output from the batch process. In addition, you can delete the audit if you have the necessary access privileges.

Add the criteria in the edit boxes, click View Audit, and view the results arranged by column.

User ID Select which user to audit.

Archive ID Select an existing archive ID.

Event Select an archiving event from the list. You can select from all processes that have been run by Data Archiving Manager.

From Date Select a start date for the audit

To Select an ending date for the audit.

View Audit Click to have the system create the audit report and display the appropriate fields on the page.

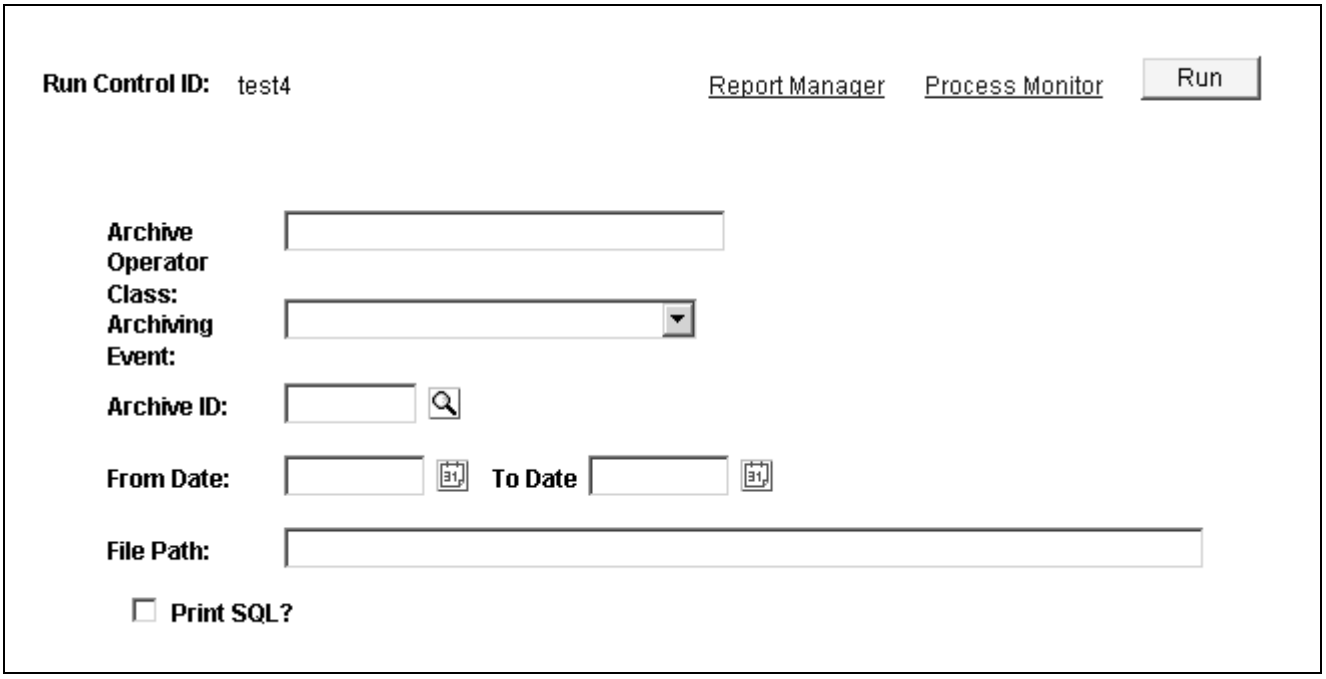
Purge Audit If you have the correct security permission set up on the Audit Security page, you can click to purge an audit.

File Path Enter the path where the generated report is saved on the batch server.

Detail Click to show the details related to the audit events . For example, if you edited the SQL using the SQL Designer page, the Details page will show the original SQL as well as the modified SQL.

Viewing Audit Results

Access the Audit Report page.



Audit Report page

The Audit Report page creates a batch output file showing all processes that have been run in the Data Archiving tool. In addition, you can print the resulting SQL.

Archive Operator Class Select which user ID to audit.

Archive ID Select an existing archive ID.

Event Select an archiving event from the list. You can select from all processes that have been run by Data Archiving Manager.

From Date Select a start date for the audit.

To Select an ending date for the audit.

Print SQL? Select to have the SQL statements for each of the archiving processes printed in the report

Performing Additional Archival Procedures

This section discusses how to:

- Archive from online tables to history tables.
- Roll back history table data.
- Archive from history tables to flat files.
- Restore archived data from flat files.
- Understand commits.
- Gain increased performance.

- Modify indexes.

Archiving from Online Tables to History Tables

Use this procedure to copy the data from the online table to the history tables using the selection process and then remove the data from the online tables using the delete process.

To complete the online-to-history-table process:

1. Create an archive project.

   Create a project (with a unique archive ID) that identifies the tables or rows that are to be archived.

2. Run archiving reports.

   Print an archive project definition report to verify the data that you have selected to archive.

3. Run archive data with the Archive Selection process.

   Based on the archive ID, run the selection process. This copies the data rows for archiving to the history tables.

4. Run archive data with the Delete process.

   This removes the data from the online tables. Do this only when you are confident that the archived data has been successfully stored in the history tables.

Rolling Back History Table Data

To roll back history table data:

1. Run archive data with the Rollback process.

   This copies the data back from the history tables to the online tables. You use this when the data has been accidentally deleted from the online tables, and it has already been archived to the history tables.

2. Run archive data with the Remove from History process.

   If the Archive Selection process has been run, and it selected the wrong set of data to be copied to history tables, run the Remove from History process to remove the misplaced data. This returns you to the state where you started before the Archive Selection process.

Archiving From History Tables to Flat Files

To archive from a history table to a flat file:

1. Run the Export History to Flat Files process.

   This process copies the data to the flat files.

2. Run archive data with the Remove from History process.

   This deletes the data from the history tables. Only run this process after you have verified that the data has been successfully archived to flat files.

Restoring Archived Data From Flat Files

To restore archived data from flat files:

1. Run the Archive Audit report for the Batch History process.

This report lists the definition of the history tables at the time the data was archived. This report provides the names of the flat files that contain the data.

2. Create staging tables.

   The staging table is a clone of the online table. The Import process imports the data from the flat files into staging tables.

3. Run the Import for Flat Files process.

4. Run the programs to populate the tables from the flat files.

Understanding Commits

For both batch and online execution, the Archive Selection, Remove from History, Rollback, and Delete Processes issue commits after each record has been processed unless a commit level has been specified otherwise.

Gaining Increased Performance

For better performance and increased speed during archiving processes, try dropping the indexes before inserting data from online tables into history tables.

Modifying Indexes

Your database platform may have a limitation to the number of columns an index can contain. Some have a restriction of 16 columns for an index. If the table that you want to archive already has 16 keys, then you cannot add another key (PSARCH_ID) to the corresponding history table.

The first option for solving this situation is to use the Flat File Archiving Strategy. The second option for solving this situation is to create the history table with the PSARCH_ID as a non-key field. For this situation, it is recommended that you either have different history tables for different archiving projects, or you have a strategy of purging the history tables prior to executing the selection process of another archiving project.

APPENDIX I

ISO Country and Currency Codes

PeopleBooks use International Organization for Standardization (ISO) country and currency codes to identify country-specific information and monetary amounts.

This appendix discusses:

- ISO country codes.
- ISO currency codes.

See Also

"About These PeopleBooks Preface," Typographical Conventions and Visual Cues

ISO Country Codes

This table lists the ISO country codes that may appear as country identifiers in PeopleBooks:

ISO Country Code	Country Name
ABW	Aruba
AFG	Afghanistan
AGO	Angola
AIA	Anguilla
ALB	Albania
AND	Andorra
ANT	Netherlands Antilles
ARE	United Arab Emirates
ARG	Argentina
ARM	Armenia
ASM	American Samoa
ATA	Antarctica

ISO Country Code	Country Name
ATF	French Southern Territories
ATG	Antigua and Barbuda
AUS	Australia
AUT	Austria
AZE	Azerbaijan
BDI	Burundi
BEL	Belgium
BEN	Benin
BFA	Burkina Faso
BGD	Bangladesh
BGR	Bulgaria
BHR	Bahrain
BHS	Bahamas
BIH	Bosnia and Herzegovina
BLR	Belarus
BLZ	Belize
BMU	Bermuda
BOL	Bolivia
BRA	Brazil
BRB	Barbados
BRN	Brunei Darussalam
BTN	Bhutan
BVT	Bouvet Island
BWA	Botswana
CAF	Central African Republic
CAN	Canada
CCK	Cocos (Keeling) Islands

ISO Country Code	Country Name
CHE	Switzerland
CHL	Chile
CHN	China
CIV	Cote D'Ivoire
CMR	Cameroon
COD	Congo, The Democratic Republic
COG	Congo
COK	Cook Islands
COL	Colombia
COM	Comoros
CPV	Cape Verde
CRI	Costa Rica
CUB	Cuba
CXR	Christmas Island
CYM	Cayman Islands
CYP	Cyprus
CZE	Czech Republic
DEU	Germany
DJI	Djibouti
DMA	Dominica
DNK	Denmark
DOM	Dominican Republic
DZA	Algeria
ECU	Ecuador
EGY	Egypt
ERI	Eritrea
ESH	Western Sahara

ISO Country Code	Country Name
ESP	Spain
EST	Estonia
ETH	Ethiopia
FIN	Finland
FJI	Fiji
FLK	Falkland Islands (Malvinas)
FRA	France
FRO	Faroe Islands
FSM	Micronesia, Federated States
GAB	Gabon
GBR	United Kingdom
GEO	Georgia
GHA	Ghana
GIB	Gibraltar
GIN	Guinea
GLP	Guadeloupe
GMB	Gambia
GNB	Guinea-Bissau
GNQ	Equatorial Guinea
GRC	Greece
GRD	Grenada
GRL	Greenland
GTM	Guatemala
GUF	French Guiana
GUM	Guam
GUY	Guyana
GXA	GXA - GP Core Country

ISO Country Code	Country Name
GXB	GXB - GP Core Country
GXC	GXC - GP Core Country
GXD	GXD - GP Core Country
HKG	Hong Kong
HMD	Heard and McDonald Islands
HND	Honduras
HRV	Croatia
HTI	Haiti
HUN	Hungary
IDN	Indonesia
IND	India
IOT	British Indian Ocean Territory
IRL	Ireland
IRN	Iran (Islamic Republic Of)
IRQ	Iraq
ISL	Iceland
ISR	Israel
ITA	Italy
JAM	Jamaica
JOR	Jordan
JPN	Japan
KAZ	Kazakstan
KEN	Kenya
KGZ	Kyrgyzstan
KHM	Cambodia
KIR	Kiribati
KNA	Saint Kitts and Nevis

ISO Country Code	Country Name
KOR	Korea, Republic of
KWT	Kuwait
LAO	Lao People’s Democratic Rep
LBN	Lebanon
LBR	Liberia
LBY	Libyan Arab Jamahiriya
LCA	Saint Lucia
LIE	Liechtenstein
LKA	Sri Lanka
LSO	Lesotho
LTU	Lithuania
LUX	Luxembourg
LVA	Latvia
MAC	Macao
MAR	Morocco
MCO	Monaco
MDA	Moldova, Republic of
MDG	Madagascar
MDV	Maldives
MEX	Mexico
MHL	Marshall Islands
MKD	Fmr Yugoslav Rep of Macedonia
MLI	Mali
MLT	Malta
MMR	Myanmar
MNG	Mongolia
MNP	Northern Mariana Islands

ISO Country Code	Country Name
MOZ	Mozambique
MRT	Mauritania
MSR	Montserrat
MTQ	Martinique
MUS	Mauritius
MWI	Malawi
MYS	Malaysia
MYT	Mayotte
NAM	Namibia
NCL	New Caledonia
NER	Niger
NFK	Norfolk Island
NGA	Nigeria
NIC	Nicaragua
NIU	Niue
NLD	Netherlands
NOR	Norway
NPL	Nepal
NRU	Nauru
NZL	New Zealand
OMN	Oman
PAK	Pakistan
PAN	Panama
PCN	Pitcairn
PER	Peru
PHL	Philippines
PLW	Palau

ISO Country Code	Country Name
PNG	Papua New Guinea
POL	Poland
PRI	Puerto Rico
PRK	Korea, Democratic People’s Rep
PRT	Portugal
PRY	Paraguay
PSE	Palestinian Territory, Occupie
PYF	French Polynesia
QAT	Qatar
REU	Reunion
ROU	Romania
RUS	Russian Federation
RWA	Rwanda
SAU	Saudi Arabia
SDN	Sudan
SEN	Senegal
SGP	Singapore
SGS	Sth Georgia & Sth Sandwich Is
SHN	Saint Helena
SJM	Svalbard and Jan Mayen
SLB	Solomon Islands
SLE	Sierra Leone
SLV	El Salvador
SMR	San Marino
SOM	Somalia
SPM	Saint Pierre and Miquelon
STP	Sao Tome and Principe

ISO Country Code	Country Name
SUR	Suriname
SVK	Slovakia
SVN	Slovenia
SWE	Sweden
SWZ	Swaziland
SYC	Seychelles
SYR	Syrian Arab Republic
TCA	Turks and Caicos Islands
TCD	Chad
TGO	Togo
THA	Thailand
TJK	Tajikistan
TKL	Tokelau
TKM	Turkmenistan
TLS	East Timor
TON	Tonga
TTO	Trinidad and Tobago
TUN	Tunisia
TUR	Turkey
TUV	Tuvalu
TWN	Taiwan, Province of China
TZA	Tanzania, United Republic of
UGA	Uganda
UKR	Ukraine
UMI	US Minor Outlying Islands
URY	Uruguay
USA	United States

ISO Country Code	Country Name
UZB	Uzbekistan
VAT	Holy See (Vatican City State)
VCT	St Vincent and the Grenadines
VEN	Venezuela
VGB	Virgin Islands (British)
VIR	Virgin Islands (U.S.)
VNM	Viet Nam
VUT	Vanuatu
WLF	Wallis and Futuna Islands
WSM	Samoa
YEM	Yemen
YUG	Yugoslavia
ZAF	South Africa
ZMB	Zambia
ZWE	Zimbabwe

ISO Currency Codes

This table lists the ISO country codes that may appear as currency identifiers in PeopleBooks:

ISO Currency Code	Description
ADP	Andorran Peseta
AED	United Arab Emirates Dirham
AFA	Afghani
AFN	Afghani
ALK	Old Lek
ALL	Lek
AMD	Armenian Dram

ISO Currency Code	Description
ANG	Netherlands Antilles Guilder
AOA	Kwanza
AOK	Kwanza
AON	New Kwanza
AOR	Kwanza Reajustado
ARA	Austral
ARP	Peso Argentino
ARS	Argentine Peso
ARY	Peso
ATS	Schilling
AUD	Australian Dollar
AWG	Aruban Guilder
AZM	Azerbaijanian Manat
BAD	Dinar
BAM	Convertible Marks
BBD	Barbados Dollar
BDT	Taka
BEC	Convertible Franc
BEF	Belgian Franc
BEL	Financial Belgian Franc
BGJ	Lev A/52
BGK	Lev A/62
BGL	Lev
BGN	Bulgarian LEV
BHD	Bahraini Dinar
BIF	Burundi Franc
BMD	Bermudian Dollar

ISO Currency Code	Description
BND	Brunei Dollar
BOB	Boliviano
BOP	Peso
BOV	Mvdol
BRB	Cruzeiro
BRC	Cruzado
BRE	Cruzeiro
BRL	Brazilian Real
BRN	New Cruzado
BRR	Brazilian Real Dollar
BSD	Bahamian Dollar
BTN	Ngultrum
BUK	N/A
BWP	Pula
BYB	Belarussian Ruble
BYR	Belarussian Ruble
BZD	Belize Dollar
CAD	Canadian Dollar
CDF	Franc Congolais
CHF	Swiss Franc
CLF	Unidades de fomento
CLP	Chilean Peso
CNX	Peoples Bank Dollar
CNY	Yuan Renminbi
COP	Colombian Peso
CRC	Costa Rican Colon
CSD	Serbia Dinar

ISO Currency Code	Description
CSJ	Krona A/53
CSK	Koruna
CUP	Cuban Peso
CVE	Cape Verde Escudo
CYP	Cyprus Pound
CZK	Czech Koruna
DEM	Deutsche Mark
DJF	Djibouti Franc
DKK	Danish Krone
DOP	Dominican Peso
DZD	Algerian Dinar
ECS	Sucre
ECV	Unidad de Valor
EEK	Kroon
EGP	Egyptian Pound
EQE	Ekwele
ERN	Nakfa
ESA	Spanish Peseta
ESB	Convertible Peseta
ESP	Spanish Peseta
ETB	Ethiopian Birr
EUR	euro
FIM	Markka
FJD	Fiji Dollar
FKP	Falklands Isl. Pound
FRF	French Franc
GBP	Pound Sterling

ISO Currency Code	Description
GEK	Georgian Coupon
GEL	Lari
GHC	Cedi
GIP	Gibraltar Pound
GMD	Dalasi
GNE	Syli
GNF	Guinea Franc
GNS	Syli
GQE	Ekwele
GRD	Drachma
GTQ	Quetzal
GWE	Guinea Escudo
GWP	Guinea-Bissau Peso
GYD	Guyana Dollar
HKD	Hong Kong Dollar
HNL	Lempira
HRD	Dinar
HRK	Kuna
HTG	Gourde
HUF	Forint
IDR	Rupiah
IEP	Irish Pound
ILP	Pound
ILR	Old Shekel
ILS	New Israeli Sheqel
INR	Indian Rupee
IQD	Iraqi Dinar

ISO Currency Code	Description
IRR	Iranian Rial
ISJ	Old Krona
ISK	Iceland Krona
ITL	Italian Lira
JMD	Jamaican Dollar
JOD	Jordanian Dinar
JPY	Yen
KES	Kenyan Shilling
KGS	Som
KHR	Riel
KMF	Comoro Franc
KPW	North Korean Won
KRW	Won
KWD	Kuwaiti Dinar
KYD	Cayman Islands dollar
KZT	Tenge
LAJ	Kip Pot Pol
LAK	Kip
LBP	Lebanese Pound
LKR	Sri Lanka Rupee
LRD	Liberian Dollar
LSL	Loti
LSM	Maloti
LTL	Lithuanian Litas
LTT	Talonas
LUC	Convertib Francl
LUF	Luxembourg Franc

ISO Currency Code	Description
LUL	Financial Franc
LVL	Latvian Lats
LVR	Latvian Ruble
LYD	Libyan Dinar
MAD	Moroccan Dirham
MAF	Mali Franc
MDL	Moldovan Leu
MGF	Malagasy Franc
MKD	Denar
MLF	Mali Franc
MMK	Kyat
MNT	Tugrik
MOP	Pataca
MRO	Ouguiya
MTL	Maltese Lira
MTP	Maltese Pound
MUR	Mauritius Rupee
MVQ	Maldive Rupee
MVR	Rufiyaa
MWK	Malawian Kwacha
MXN	Mexican Peso
MXP	Mexican Peso
MXV	Mexican UDI
MYR	Malaysian Ringgit
MZE	Mozambique Escudo
MZM	Metical
NAD	Namibia Dollar

ISO Currency Code	Description
NGN	Naira
NIC	Cordoba
NIO	Cordoba Oro
NLG	Netherlands Guilder
NOK	Norwegian Krone
NPR	Nepalese Rupee
NZD	New Zealand Dollar
OMR	Rial Omani
PAB	Balboa
PEI	Inti
PEN	Nuevo Sol
PES	Sol
PGK	Kina
PHP	Philippine Peso
PKR	Pakistan Rupee
PLN	Zloty
PLZ	Zloty
PTE	Portuguese Escudo
PYG	Guarani
QAR	Qatari Rial
ROK	Leu A/52
ROL	Leu
RUB	Russian Ruble
RUR	Russian Federation Rouble
RWF	Rwanda Franc
SAR	Saudi Riyal
SBD	Solomon Islands

ISO Currency Code	Description
SCR	Seychelles Rupee
SDD	Sudanese Dinar
SDP	Sudanese Pound
SEK	Swedish Krona
SGD	Singapore Dollar
SHP	St Helena Pound
SIT	Tolar
SKK	Slovak Koruna
SLL	Leone
SOS	Somali Shilling
SRG	Surinam Guilder
STD	Dobra
SUR	Rouble
SVC	El Salvador Colon
SYP	Syrian Pound
SZL	Lilangeni
THB	Baht
TJR	Tajik Ruble
TJS	Somoni
TMM	Manat
TND	Tunisian Dinar
TOP	Pa’anga
TPE	Timor Escudo
TRL	Turkish Lira
TTD	Trinidad Dollar
TWD	New Taiwan Dollar
TZS	Tanzanian Shilling

ISO Currency Code	Description
UAH	Hryvnia
UAK	Karbovanet
UGS	Uganda Shilling
UGW	Old Shilling
UGX	Uganda Shilling
USD	US Dollar
USN	US Dollar (Next day)
USS	US Dollar (Same day)
UYN	Old Uruguay Peso
UYP	Uruguayan Peso
UYU	Peso Uruguayo
UZS	Uzbekistan Sum
VEB	Bolivar
VNC	Old Dong
VND	Dong
VUV	Vatu
WST	Tala
XAF	CFA Franc BEAC
XAG	Silver
XAU	GOLD
XBA	European Composite Unit
XBB	European Monetary Unit
XBC	European Unit of Account 9
XBD	European Unit of Account 17
XCD	East Caribbean Dollar
XDR	SDR
XEU	EU Currency (E.C.U)

ISO Currency Code	Description
XFO	Gold-Franc
XFU	UIC-Franc
XOF	CFA Franc BCEAO
XPD	Palladium
XPF	CFP Franc
XPT	Platinum
XTS	For Testing Purposes
XXX	Non Currency Transaction
YDD	Yemeni Din
YER	Yemeni Rial
YUD	New Yugoslavian Dinar
YUM	New Dinar
YUN	Yugoslavian Dinar
ZAL	Financial Rand
ZAR	Rand
ZMK	Zambian Kwacha
ZRN	New Zaire
ZRZ	Zaire
ZWC	Rhodesian Dollar
ZWD	Zimbabwe Dollar

Glossary of PeopleSoft 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.

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 Human Resources, PeopleSoft Benefits Administration, PeopleSoft Stock Administration, and the COBRA Administration feature of the Base Benefits business process.

action template In PeopleSoft 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 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.

agreement In PeopleSoft 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 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.

AR specialist Abbreviation for *receivables specialist.* In PeopleSoft Receivables, an individual in who tracks and resolves deductions and disputed items.

arbitration plan In PeopleSoft Enterprise Pricer, defines how price rules are to be applied to the base price when the transaction is priced.

assessment rule In PeopleSoft 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 Directory Interface, relates the data that makes up an entry in the directory information tree.

authentication server A server that is set up to verify users of the system.

base time period In PeopleSoft Business Planning, the lowest level time period in a calendar.

benchmark job In PeopleSoft Workforce Analytics, a benchmark job is a job code for which there is corresponding salary survey data from published, third-party sources.

book In PeopleSoft 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."

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 event In PeopleSoft Receivables, defines the processing characteristics for the Receivable Update process for a draft activity.

	In PeopleSoft 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 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 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.
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 Catalog Management, translates values from the catalog source data to the format of the company’s catalog.
catalog partner	In PeopleSoft 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.
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 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 Promotions Management, enables you to view financial data (such as planned, incurred, and actual amounts) that is related to funds and trade promotions.
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.*
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.
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, PeopleSoft maintains a set of collections (one per language code) for each search index object.

collection rule In PeopleSoft 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.

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.

condition In PeopleSoft 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. 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.

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 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 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.

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 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.

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 Supply Chain Management, identifies the method by which goods are shipped to their destinations (such as truck, air, rail, and so on). 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.

directory information tree In PeopleSoft Directory Interface, the representation of a directory's hierarchical structure.

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 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 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.

elimination set In PeopleSoft General Ledger, a related group of intercompany accounts that is processed during consolidations.

entry event In PeopleSoft 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 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.

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 Human Resources, also refers to an incident that affects benefits eligibility.

event propagation process In PeopleSoft 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.

	Sales Incentive Management uses this mechanism to implement splits, roll-ups, and so on. Event propagation determines who receives the credit.
exception	In PeopleSoft Receivables, an item that either is a deduction or is in dispute.
exclusive pricing	In PeopleSoft 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 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.
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.
fund	In PeopleSoft 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.
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.
group	In PeopleSoft Billing and Receivables, a posting entity that comprises one or more transactions (items, deposits, payments, transfers, matches, or write-offs). In PeopleSoft 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 Time and Labor, for example, employees are placed in groups for time reporting purposes.
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, user interaction objects, and so on.
incentive rule	In PeopleSoft 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 Promotions Management, to become liable for a promotional payment. In other words, you owe that amount to a customer for promotional activities.
item	In PeopleSoft Inventory, a tangible commodity that is stored in a business unit (shipped from a warehouse). In PeopleSoft 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 Receivables, an individual receivable. An item can be an invoice, a credit memo, a debit memo, a write-off, or an adjustment.
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.

LDIF file Abbreviation for *Lightweight Directory Access Protocol (LDAP) Data Interchange Format file.* Contains discrepancies between PeopleSoft 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.

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.

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.

load In PeopleSoft Inventory, identifies a group of goods that are shipped together. Load management is a feature of PeopleSoft Inventory that is used to track the weight, the volume, and the destination of a shipment.

local functionality In PeopleSoft 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 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.
match group	In PeopleSoft 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 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.
meta-SQL	Meta-SQL constructs expand into platform-specific Structured Query Language (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 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 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.
node-oriented tree	A tree that is based on a detail structure, but the detail values are not used.
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 and non-PeopleSoft 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 Payables, a set of rules that define the criteria by which it should select scheduled payments for payment creation.
pending item	In PeopleSoft 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.

PeopleCode PeopleCode is a proprietary language, executed by the PeopleSoft application processor. PeopleCode generates results based upon existing data or user actions. By using business interlink objects, external services are available to all PeopleSoft applications wherever PeopleCode can be executed.

PeopleCode event An action that a user takes upon an object, usually a record field, that is referenced within a PeopleSoft page.

PeopleSoft 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.

plan In PeopleSoft 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 Supply Planning, a set of data (business units, items, supplies, and demands) constituting the inputs and outputs of a supply plan.

portal registry In PeopleSoft 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.

price list In PeopleSoft Enterprise Pricer, 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 In PeopleSoft Enterprise Pricer, defines 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 condition In PeopleSoft Enterprise Pricer, selects the price-by fields, the values for the price-by fields, and the operator that determines how the price-by fields are related to the transaction.

price rule key In PeopleSoft Enterprise Pricer, defines the fields that are available to define price rule conditions (which are used to match a transaction) on the price rule.

process category In PeopleSoft Process Scheduler, processes that are grouped for server load balancing and prioritization.

process group In PeopleSoft 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.

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 category In PeopleSoft Enterprise Incentive Management, indicates an application in the Enterprise Incentive Management suite of products. Each transaction in the PeopleSoft Enterprise Incentive Management system is associated with a product category.

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 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 Project Costing, an individual transaction line that represents a cost, time, budget, or other transaction row.

promotion In PeopleSoft Promotions Management, a trade promotion, which is typically funded from trade dollars and used by consumer products manufacturers to increase sales volume.

publishing In PeopleSoft Enterprise Incentive Management, a stage in processing that makes incentive-related results available to participants.

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 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 Order Management, Billing, or Receivables, where it is assumed that you are always recording only output VAT, or in PeopleSoft 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*.

reference data In PeopleSoft Sales Incentive Management, system objects that represent the sales organization, such as territories, participants, products, customers, channels, and so on.

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 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.

REN server Abbreviation for *real-time event notification server* in PeopleSoft MultiChannel Framework.

requester In PeopleSoft eSettlements, an individual who requests goods or services and whose ID appears on the various procurement pages that reference purchase orders.

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.

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.

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.

serial genealogy In PeopleSoft Manufacturing, the ability to track the composition of a specific, serial-controlled item.

serial in production In PeopleSoft Manufacturing, enables the tracing of serial information for manufactured items. This is maintained in the Item Master record.

session 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 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.

single signon With single signon, users can, after being authenticated by a PeopleSoft application server, access a second PeopleSoft application server without entering a user ID or password.

source transaction In commitment control, any transaction generated in a PeopleSoft 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.

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.

statutory account Account required by a regulatory authority for recording and reporting financial results. In PeopleSoft, this is equivalent to the Alternate Account (ALTACCT) ChartField.

step In PeopleSoft Sales Incentive Management, a collection of sections in a plan. Each step corresponds to a step in the job run.

storage level In PeopleSoft 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 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.

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 Receivables, an activity that defines how the system generates accounting entries for the general ledger.

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.

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, you use templates to build a page by combining HTML from a number of sources. For a PeopleSoft portal, all templates must be registered in the portal registry, and each content reference must be assigned a template.

territory In PeopleSoft Sales Incentive Management, hierarchical relationships of business objects, including regions, products, customers, industries, and participants.

TimeSpan A relative period, such as year-to-date or current period, that can be used in various PeopleSoft General Ledger functions and reports when a rolling time frame, rather

	than a specific date, is required. TimeSpans can also be used with flexible formulas in PeopleSoft Projects.
trace usage	In PeopleSoft 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 systems that displays the relationship between all accounting units (for example, corporate divisions, projects, reporting groups, account numbers) and determines roll-up hierarchies.
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 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.
user interaction object	In PeopleSoft Sales Incentive Management, used to define the reporting components and reports that a participant can access in his or her context. All 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 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 data warehouse that consists of predefined ETL maps, data warehouse tools, and DataMart definitions.

work order In PeopleSoft 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.

worksheet A way of presenting data through a PeopleSoft 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 schema An XML definition that standardizes the representation of application messages, component interfaces, or business interlinks.

yield by operation In PeopleSoft 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.

Index

B

C

D

E

F

G

H

I

Q

R

S

T

U