ORACLE

PEOPLESOFT

PeopleTools 8.54: Data Management

November 2016

ORACLE

PeopleTools 8.54: Data Management
CDSKU

License Restrictions Warranty/Consequential Damages Disclaimer

Warranty Disclaimer

Restricted Rights Notice

Hazardous Applications Notice

Trademark Notice

Third Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Contents

Preface ..... **xv**
Understanding the PeopleSoft Online Help and PeopleBooks xv
PeopleSoft Hosted Documentation xv
Locally Installed Help xv
Downloadable PeopleBook PDF Files xv
Common Help Documentation xvi
Field and Control Definitions xvi
Typographical Conventions xvi
ISO Country and Currency Codes xvii
Region and Industry Identifiers xvii
Using and Managing the PeopleSoft Online Help xviii
Contact Us xviii
Follow Us xviii
Chapter 1: Getting Started with Data Management **19**
Data Management Overview 19
PeopleSoft Data Mover 19
PeopleSoft Data Archive Manager 19
Data Integrity and Auditing 20
Diagnostics Framework 21
Database Platform Considerations 21
Data Management Implementation 22
Chapter 2: Using PeopleSoft Data Mover **23**
Understanding PeopleSoft Data Mover 23
PeopleSoft Data Mover Overview 23
PeopleSoft Data Mover Environment 23
PeopleSoft Data Mover Operating Modes 24
Understanding Data Mover Scripts 24
Data Mover Script Commands 25
Supported SQL Commands 26
Data Mover Commands Compared to SQL Commands 28
PeopleSoft Data Mover COMMIT Statements 29
Using the Development Environment 29
Signing In to the Development Environment 29
Navigating the Data Mover Window 29
Creating and Running PeopleSoft Data Mover Scripts 30
Understanding Command Types 31
Understanding Syntax Rules 31
Creating and Editing Scripts 33
Preparing to Run Export Scripts 34
Running Scripts 34
Using the Database Setup Utility 35
Accessing the Database Setup Utility 36
Working with the Database Setup Utility 36
Checking the Generated Script 38
Using the PeopleSoft Data Mover Command-Line Interface 38
Understanding the PeopleSoft Data Mover Command-Line Interface 39
Setting Up UNIX to Run PeopleSoft Data Mover 39

Setting Up Tracing ... 40
Running Data Mover Scripts from the Command Line ... 41
Using PeopleSoft Data Mover Commands ... 44
CHANGE_ACCESS_PASSWORD ... 45
CREATE_TEMP_TABLE ... 45
CREATE_TRIGGER ... 45
ENCRYPT_PASSWORD ... 46
EXPORT ... 46
IMPORT ... 47
REM, REMARK, and – – ... 49
RENAME ... 49
REPLACE_ALL ... 52
REPLACE_DATA ... 52
REPLACE_VIEW ... 53
RUN ... 53
SET ... 53
SET BASE_LANGUAGE ... 54
SET IGNORE_ERRORS ... 54
SET COMMIT ... 55
SWAP_BASE_LANGUAGE ... 55
Using PeopleSoft Data Mover Command Modifiers ... 56
AS ... 56
IGNORE_DUPS ... 57
UPDATE_DUPS ... 58
WHERE ... 58
Using SET Parameters ... 59
COMMIT ... 59
CREATE_INDEX_BEFORE_DATA ... 60
DBSPACE ... 60
DDL ... 61
EXECUTE_SQL ... 62
EXTRACT ... 62
IGNORE_DUPS ... 62
INPUT ... 63
INSERT_DATA_ONCE ... 63
LOG ... 64
NO DATA ... 64
NO INDEX ... 65
NO RECORD ... 65
NO SPACE ... 65
NO TRACE ... 66
NO VIEW ... 66
OUTPUT ... 66
SIZING SET ... 67
SPACE ... 67
START ... 68
STATISTICS ... 68
UNICODE ... 68
VERSION ... 69
Using Script Examples ... 69
Exporting Databases ... 69

Building Databases....70
Recreating All Views....70
Importing with REPLACE_ALL with a Commit Level....70
Combining SQL Commands and IMPORT....70
Chapter 3: Using PeopleSoft Data Archive Manager....73
Understanding PeopleSoft Data Archive Manager....73
Understanding Archiving Strategy....74
Archiving Strategy....74
History Tables....75
Understanding Archiving Techniques....76
Business Requirements Analysis....76
Commits....77
Enhancing Performance....77
Index Considerations....77
Data Limitations....77
Accessing the Data Archive Manager Homepage....78
Managing Archive Objects....79
Understanding the Base Table and Non-base Tables....79
Managing Archive Objects....79
Defining Archive and Restore Queries....79
Managing Archive Templates....80
Managing Archive Templates....80
Managing Archive Jobs....82
Defining Archive Jobs....83
Viewing Details....85
Defining Archive Query Binds....86
Auditing Archive Processes....86
Audit Archiving....86
Chapter 4: Ensuring Data Integrity....89
Understanding Data Integrity Tools....89
Running SQL Alter....89
Understanding Table and Column Audits....90
Running DDDAUDIT....91
DDDAUDIT Queries....91
Running SYSAUDIT....92
Understanding How to Run SYSAUDIT....93
Application Data Set Integrity....95
Application Engine Integrity....96
BI Publisher Integrity....100
Clear List Integrity....101
Connected Query Integrity....102
EDI Manager Integrity....104
Feeds Integrity....106
Field Integrity....110
File Processing Integrity....110
Integration Broker Integrity....110
Menu Integrity....116
Optimization Integrity....117
Page Integrity....119
PeopleCode Integrity....121
Pivot Grid Integrity....125

PSLOCK Version Integrity....126
Process Scheduler....126
Query Integrity....129
Record Integrity....132
Related Language Integrity....135
Security Integrity....139
SQL Integrity....145
Style Sheet Integrity....147
Tree Integrity....148
Translate Integrity....155
Search Integrity....155
Related Content Integrity....157
Chapter 5: Employing Database Level Auditing....161
Understanding Database Level Auditing....161
Creating Audit Record Definitions....162
Working With Auditing Triggers....164
Defining Auditing Triggers....165
Creating and Running the Auditing Triggers Script....166
Deleting Auditing Triggers....167
Viewing Audit Information....167
Creating Queries to View Audit Records Details....168
Creating an Access Group....169
Listing All Audit Records in PS_AUDIT_JOB....169
Listing All Audit Records for a Specified User ID....170
Listing All Audit Records Containing an Invalid OPRID....171
Listing All Audit Records for a Specified Time Period....172
Using Microsoft SQL Server Trigger Information....173
Using Microsoft SQL Server Trigger Syntax....173
Using Microsoft SQL Server to Capture Text/Image Columns....175
Administering Microsoft SQL Server Trigger Maintenance....176
Using DB2 UDB for z/OS Trigger Information....177
Understanding DB2 z/OS Trigger Information....177
DB2 z/OS Trigger Syntax....177
DB2 z/OS Trigger Maintenance....178
Using DB2 UDB for Linux, Unix, and Windows (LUW) Trigger Information....179
DB2 LUW Trigger Syntax....179
DB2 LUW Trigger Maintenance....180
Using Oracle Trigger Information....181
Using Oracle Trigger Syntax....181
Maintaining Oracle Triggers....183
Using Sybase Trigger Information....184
Using Sybase Trigger Syntax....185
Using Sybase Trigger Maintenance....185
Chapter 6: Working With The Diagnostics Framework....187
Understanding Diagnostics Framework....187
What Is the Diagnostics Framework?....187
Diagnostics Framework Benefits....187
Diagnostics Framework Architecture....188
Setting Up Security for Diagnostics Framework....189
Understanding Security for Diagnostics Framework....189
Granting Access to the Diagnostics Framework Pages....190

Granting Access to the WEBLIB_PTDIAG Web Library....190
Running Diagnostics....191
Launching Diagnostic Plug-Ins....191
Providing Additional Information For Diagnostic Plug-ins....192
Obtaining Diagnostic Results....193
Importing Post-Release Plug-Ins....196
Chapter 7: Developing Diagnostic Plug-Ins....197
Understanding Diagnostic Plug-In Development....197
Developing Diagnostic Plug-Ins....198
Creating the Diagnostic Application Package....198
Creating the Diagnostic Application Classes....198
Implementing the Diagnostic PeopleCode....198
Registering the Diagnostic Plug-In....200
Sharing Diagnostic Definitions....200
Working With The Delivered PT_DIAGNOSTIC Application Package....201
PTDiagnostics Application Class....202
PTDiagnostics Class Methods....202
GetDiagnosticInfo....202
GetDynamicPrompt....203
GetUserInputByKey....204
InsertData....204
InsertQuestion....205
IsPlugIn....206
SetProperty....207
PTDiagnostics Class Properties....207
hasRowset....208
Purpose....208
Where....208
Diagnostic Plug-In Examples....209
Example: Rowset-Based Output....209
Example: String-Based Output....210
Example: Number-Based Output....210
Example: Prompting for Global Information Input....211
Example: Prompting for Global and Class-Level Information Input....212
Example: Joining Two Records....213
Example: Handling Constructor Failure....214
Example: Handling InsertData Method Failure....215
Example: Handling Dynamic Prompting Failure....216
Appendix A: Administering PeopleSoft Databases on Microsoft SQL Server....219
Server Options....219
Delivered Configuration....219
Access ID....219
Service Packs and Quick Fix Enhancements (QFE)....220
Required Database Configuration....220
ANSI Nullability....220
Working with Functional Indexes....220
Database Collation Settings....221
Implementing Transparent Data Encryption....221
Understanding Transparent Data Encryption....221
Enabling Transparent Data Encryption....222
Microsoft SQL Server Feature Considerations....223

Recovery Model ... 223
Nested Triggers ... 223
Auto Create Statistics and Auto Update Statistics ... 223
Automatic File Growth ... 224
Autoshrink ... 224
Read Committed Snapshot Isolation ... 224
File Management ... 225
Tempdb ... 225
Trace Flags ... 226
Database Monitoring ... 227
Appendix B: Administering PeopleSoft Databases on DB2 UDB for z/OS ... 229
Understanding DB2 UDB for z/OS Administration ... 229
Database Considerations ... 229
Concurrency ... 230
Monitoring Batch Programs ... 231
Understanding Batch Program Monitoring Tools ... 231
Enabling DB2 CLI/ODBC Trace ... 232
Enabling the PTPSQLRT Mainframe Statistics Report ... 233
Enabling Dynamic Explains ... 236
Enabling Parallelism ... 237
Enabling PeopleSoft SQL Trace ... 238
Enabling SQR Monitoring ... 240
Associating PeopleSoft Users, Modules, and Actions with DB2 UDB Threads ... 242
Running COBOL ... 244
Understanding COBOL API and Meta SQL ... 244
Running COBOL Outside of Process Scheduler ... 245
Disabling Persistent Cursors ... 245
Administering SQR for z/OS ... 246
Understanding SQR on z/OS ... 246
Running SQRs Outside of Process Scheduler ... 246
Specifying Input and Output Files ... 247
Printing SQRs ... 248
Updating Statistics ... 249
Understanding %UpdateStats ... 249
Setting Up the IBM System Stored Procedure: DSNUTILS ... 249
Installing the Database Following the Enhanced Installation Path ... 250
Updating System Tables with Database and Tablespace Information ... 250
Activating %UpdateStats ... 251
Setting the Number of Temporary Tables ... 253
Creating Temporary Tables ... 253
Working with Alters on DB2 z/OS ... 254
Understanding Alters on DB2 z/OS ... 254
Advisory Reorg Pending and Rebuild Pending Status ... 255
Determining Whether A Tablespace or Index Is In A Pending Status ... 255
Tablespace Versioning ... 256
Determining When to Recycle Tablespace Version Numbers ... 256
Working with DB2 Tablespace Versioning and PeopleSoft Upgrades ... 258
Appendix C: Administering PeopleSoft Databases on DB2 UDB for Linux, UNIX, and Windows ... 261
Understanding Administration on DB2 UDB for Linux, UNIX, and Windows ... 261
Instances ... 261

Understanding Instances....262
SYSADM Authority and Security....262
Instances and Connectivity....263
Other Considerations....263
Configuration Parameters....264
Definition of Configuration Parameters....264
Useful Configuration Commands....264
Parameters Overview....265
Tablespaces....266
DDL Scripts....267
Using the PeopleSoft DMS Tablespace DDL....267
DMS Tablespaces: Cooked or Raw....267
System Catalog Tablespace and Other Initial Tablespaces....268
Optimizing Table Space Capacity With Auto-Resize....268
Temporary Table Creation....271
Client Database Catalog....272
Meta-SQL %TruncateTable()....272
Handling Errors....273
DB2 UDB for Linux, UNIX, and Windows Administration....273
Updating Statistics....273
Performing Queries on a Windows Workstation....274
Object Restrictions....274
Administrative Tools....274
Connectivity Using ODBC/CLI....274
Checklists and Troubleshooting....276
Connectivity Checklist....276
Diagnosing Transaction Hangs....277
DB2DIAG.LOG....278
ODBC Trace....278
db2trc....278
DB2 UDB Help Facility....278
Monitoring Module and Action Information....278
Appendix D: Administering PeopleSoft Databases on Informix....281
Database Terminology....281
Working With Dbspaces....281
Dbspace Strategy....281
Dbspace Sizing....282
Database Server Directory Structure....284
Troubleshooting Model....284
Appendix E: Administering PeopleSoft Databases on Oracle....287
Working With Oracle Connectivity....287
Oracle Net Services....287
PeopleSoft Servers and the Oracle Connection String....287
Open Cursors....289
Monitoring PeopleSoft Database Connections....290
Understanding PeopleSoft Database Connections....290
Enabling Database Connection Monitoring....291
Tracking PeopleSoft Database Connections by PeopleSoft User ID....291
Monitoring PeopleSoft MODULE and ACTION Information....301
Exposing PeopleSoft User Information Through the CLIENT_IDENTIFIER Column....302
Converting Descending Indexes....303

Setting the Number of Temporary Tables....304
Using Locally Managed Tablespaces....304
Maintaining Partition Definitions....305
Understanding Partition Management....305
Partitioning Terminology....306
Establishing Partitioning Definitions....310
Applying and Maintaining Partitioning DDL....315
Migrating Partitioning....317
Using Pluggable Databases....319
Understanding Pluggable Databases....319
Implementing Pluggable Databases....321
Related Documentation....321
Using Materialized Views....321
Understanding Materialized Views....321
Defining Materialized Views....322
Converting An Existing View to a Materialized View....324
Maintaining Materialized Views....324
Audits for Materialized Views....326
Guidelines....326
Refresh Mode/Method Recommendations....326
Using Materialized Views with Oracle Golden Gate or Oracle Active Data Guard....327
Improving Process Performance with Global Temporary Tables....328
Working With Oracle Consumer Groups....328
Reviewing PeopleSoft Resource Groups....329
Determining Where to Implement a Consumer Group....332
Creating an Oracle Resource Plan and Consumer Groups....332
Example: Creating PeopleSoft Resource Plan and Consumer Groups SQL Script....334
Mapping PeopleSoft Resource Groups to Oracle Consumer Groups....335
Implementing Oracle Transparent Data Encryption....335
Understanding Transparent Data Encryption....336
Determining Fields to Encrypt....336
Managing the Oracle Wallet....337
Setting the Encryption Algorithm....338
Encrypting Fields....339
Managing Fields Encrypted for TDE....340
Protecting and Managing PeopleSoft Applications with Database Vault....341
Understanding Oracle Database Vault....341
Restricting Access For the Access ID....342
Restricting Access For PSFTDBA ID....342
Using Multiple Alternate Access IDs....343
Working With Oracle Security Features....345
Understanding Default Profiles....345
Encountering Issues Related to Oracle Security....346
Oracle Security Configuration Options....348
Working With Oracle Transparent Application Failover....350
End-User System Behavior With TAF Configured....351
Batch System Behavior With TAF Configured....351
Implementing Oracle Active Data Guard....352
Understanding Active Data Guard Within PeopleSoft....352
Installing and Configuring Oracle Active Data Guard....356
Creating the Secondary Access ID....356

Creating the Domain Boot User ... 357
Creating Synonyms and Database Links ... 357
Configuring Domains ... 359
Configuring Read-Only Components ... 359
Configuring Read-Only Processes ... 360
Disabling Mostly-Read-Only Attributed Features ... 361
Implementing Oracle GoldenGate for PeopleSoft Off-Load Reporting ... 362
Understanding GoldenGate Within PeopleSoft ... 362
Installing Oracle GoldenGate for a PeopleSoft Implementation ... 367
Creating Subdirectories for Primary and Standby GoldenGate Installations ... 368
Configuring PeopleSoft Databases for Oracle GoldenGate ... 370
Generating PeopleSoft Parameter File Input ... 374
Creating Oracle GoldenGate Parameter Files for the Primary Database ... 375
Creating Oracle GoldenGate Parameter Files for the Standby Database ... 381
Creating Database Links and Remote Synonyms ... 382
Configuring Oracle GoldenGate for PeopleSoft ... 385
Configuring PeopleSoft to Work with GoldenGate ... 386
Setting Up the PeopleSoft Installation with Oracle RAC ... 386
Understanding the PeopleSoft Installation on Oracle RAC ... 387
Setting Up Prerequisites ... 387
Creating the Database ... 387
Creating Raw Devices ... 388
Editing the CREATEDB.SQL Script ... 388
Editing the UTLSPACE.SQL Script ... 389
Editing the XXDDL Script ... 390
Creating Initialization Files ... 390
Configuring Database Security ... 390
Configuring the Tnsnames and Listener Files ... 391
Configuring the Server Domains ... 392
Working with Oracle Fine Grained Auditing ... 393
Working with Oracle SecureFiles ... 394
Implementing the Oracle Database File System ... 395
Understanding Oracle DBFS ... 395
Installing and Configuring DBFS ... 396
Implementing Oracle DBFS on your PeopleSoft System ... 396
Using Oracle Data Masking ... 399
Appendix F: Administering PeopleSoft Databases on Sybase ... 401
Required Configuration ... 401
Server Options ... 401
Database options ... 402
Trace Options ... 402
Trace Flags in PeopleSoft Tools ... 402
Sybase API-Specific Tracing ... 403
Other Considerations ... 403
Database Monitoring ... 404
Device Management ... 404
Caches ... 404
Segments ... 404
Tempdb ... 405
Network Packet Size ... 405
Updating Statistics ... 406

Appendix G: Configuring Remote Data Access..... 407
Understanding Remote Data Access..... 407
Configuring Application Servers and Process Scheduler Servers for Remote Data Access for Informix..... 407
Configuring Application Servers or Process Scheduler Servers for Remote Data Access for Oracle..... 408
Preparing to Configure Oracle Remote Data Access..... 408
Configuring Oracle Connectivity for Remote Data Access on UNIX..... 408
Configuring Oracle Connectivity for Remote Data Access on Windows..... 408
Configuring Application Servers or Process Scheduler Servers for Remote Data Access with DB2 UDB..... 409
Configuring Remote Data Access for DB2 UDB for Linux, UNIX and Windows..... 409
Configuring Remote Data Access for DB2 UDB for z/OS..... 409
Configuring Application or Process Scheduler Servers for Remote Data Access with Sybase..... 410
Installing and Configuring the Microsoft SQL Server JDBC Driver..... 410
Appendix H: Archive Data Tool (Deprecated in PeopleTools 8.44)..... 411
Creating and Designing Archive Templates..... 411
Specifying Fields and Archival Criteria..... 411
Joining Record Criteria..... 413
Generating and Editing SQL..... 415
Working With the Archives..... 416
Granting Access Rights..... 416
Administering Archive Projects..... 417
Working With Archive Data..... 418
Finding Data That Meets Your Criteria..... 419
Creating Scripts to Move Data..... 420
Viewing and Editing Scripts..... 421
Running Data Archival Processes..... 422
Beginning the Archiving Process..... 422
Exporting Data From Online Tables to Flat Files..... 424
Exporting Data From History Tables to Flat Files..... 426
Restoring Archived Data Using Staging Tables..... 427
Running Data Archival Reports and Audits..... 428
Understanding Archive Reports and Audits..... 428
Running Archive Reports..... 428
Creating an Audit Inquiry..... 429
Viewing Audit Results..... 430
Appendix I: Mass Change..... 433
Understanding Mass Change..... 433
Defining Types..... 434
Generating SQL..... 437
Defining Templates..... 438
Selecting Prompt Tables..... 439
Configuring Date and Datetime Formatting..... 440
Building Mass Change Definitions..... 440
Creating Groups..... 442
Executing Mass Change Definitions..... 443
Executing Online..... 443
Executing in the Background..... 443
Performing Mass Changes in PeopleSoft Asset Management..... 444
Downloading and Uploading Data with Mass Change..... 444

Preface

Understanding the PeopleSoft Online Help and PeopleBooks

The PeopleSoft Online Help is a website that enables you to view all help content for PeopleSoft Applications and PeopleTools. The help provides standard navigation and full-text searching, as well as context-sensitive online help for PeopleSoft users.

PeopleSoft Hosted Documentation

You access the PeopleSoft Online Help on Oracle's PeopleSoft Hosted Documentation website, which enables you to access the full help website and context-sensitive help directly from an Oracle hosted server. The hosted documentation is updated on a regular schedule, ensuring that you have access to the most current documentation. This reduces the need to view separate documentation posts for application maintenance on My Oracle Support, because that documentation is now incorporated into the hosted website content. The Hosted Documentation website is available in English only.

Note: Only the most current release of hosted documentation is updated regularly. After a new release is posted, previous releases remain available but are no longer updated.

Locally Installed Help

If your organization has firewall restrictions that prevent you from using the Hosted Documentation website, you can install the PeopleSoft Online Help locally. If you install the help locally, you have more control over which documents users can access and you can include links to your organization's custom documentation on help pages.

In addition, if you locally install the PeopleSoft Online Help, you can use any search engine for full-text searching. Your installation documentation includes instructions about how to set up Oracle Secure Enterprise Search for full-text searching.

See *PeopleTools Installation* for your database platform, "Installing PeopleSoft Online Help." If you do not use Secure Enterprise Search, see the documentation for your chosen search engine.

Note: Before users can access the search engine on a locally installed help website, you must enable the Search portlet and link. Click the Help link on any page in the PeopleSoft Online Help for instructions.

Downloadable PeopleBook PDF Files

You can access downloadable PDF versions of the help content in the traditional PeopleBook format. The content in the PeopleBook PDFs is the same as the content in the PeopleSoft Online Help, but it has a different structure and it does not include the interactive navigation features that are available in the online help.

Common Help Documentation

Common help documentation contains information that applies to multiple applications. The two main types of common help are:

- Application Fundamentals
- Using PeopleSoft Applications

Most product families provide a set of application fundamentals help topics that discuss essential information about the setup and design of your system. This information applies to many or all applications in the PeopleSoft product family. Whether you are implementing a single application, some combination of applications within the product family, or the entire product family, you should be familiar with the contents of the appropriate application fundamentals help. They provide the starting points for fundamental implementation tasks.

In addition, the *PeopleTools: Applications User's Guide* introduces you to the various elements of the PeopleSoft Pure Internet Architecture. It also explains how to use the navigational hierarchy, components, and pages to perform basic functions as you navigate through the system. While your application or implementation may differ, the topics in this user's guide provide general information about using PeopleSoft Applications.

Field and Control Definitions

PeopleSoft documentation includes definitions for most fields and controls that appear on application pages. These definitions describe how to use a field or control, where populated values come from, the effects of selecting certain values, and so on. If a field or control is not defined, then it either requires no additional explanation or is documented in a common elements section earlier in the documentation. For example, the Date field rarely requires additional explanation and may not be defined in the documentation for some pages.

Typographical Conventions

The following table describes the typographical conventions that are used in the online help.

Typographical Convention	***Description***
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.
. . . (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 (|).
[ ] (square brackets)	Indicate optional items in PeopleCode syntax.

Typographical Convention	*Description*
& (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.
⇒	This continuation character has been inserted at the end of a line of code that has been wrapped at the page margin. The code should be viewed or entered as a single, continuous line of code without the continuation character.

ISO Country and Currency Codes

PeopleSoft Online Help topics use International Organization for Standardization (ISO) country and currency codes to identify country-specific information and monetary amounts.

ISO country codes may appear as country identifiers, and ISO currency codes may appear as currency identifiers in your PeopleSoft documentation. Reference to an ISO country code in your documentation does not imply that your application includes every ISO country code. The following example is a country-specific heading: "(FRA) Hiring an Employee."

The PeopleSoft Currency Code table (CURRENCY_CD_TBL) contains sample currency code data. The Currency Code table is based on ISO Standard 4217, "Codes for the representation of currencies," and also relies on ISO country codes in the Country table (COUNTRY_TBL). The navigation to the pages where you maintain currency code and country information depends on which PeopleSoft applications you are using. To access the pages for maintaining the Currency Code and Country tables, consult the online help for your applications for more information.

Region and Industry Identifiers

Information that applies only to a specific 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 region-specific heading: "(Latin America) Setting Up Depreciation"

Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in the PeopleSoft Online Help:

- 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 the PeopleSoft Online Help:

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

Using and Managing the PeopleSoft Online Help

Click the Help link in the universal navigation header of any page in the PeopleSoft Online Help to see information on the following topics:

- What's new in the PeopleSoft Online Help.
- PeopleSoft Online Help acessibility.
- Accessing, navigating, and searching the PeopleSoft Online Help.
- Managing a locally installed PeopleSoft Online Help website.

Contact Us

Send us your suggestions Please include release numbers for the PeopleTools and applications that you are using.

Follow Us

Get the latest PeopleSoft updates on Facebook.

Follow PeopleSoft on Twitter@PeopleSoft_Info.

Chapter 1

Getting Started with Data Management

Data Management Overview

This section discusses:

- PeopleSoft Data Mover.
- PeopleSoft Data Archive Manager.
- Data integrity and auditing.
- Diagnostics 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 Understanding PeopleSoft Data Mover.

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: PeopleSoft Data Archive Manager replaces a deprecated feature (Archive Data) used in PeopleSoft 8.40 through 8.43.

Related Links

Understanding PeopleSoft Data Archive Manager

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 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 Understanding Data Integrity Tools.

Trigger-Based Database Level Auditing

PeopleSoft provides trigger-based auditing functionality as an alternative to the record-based auditing that 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 Understanding Database Level Auditing.

Diagnostics Framework

PeopleSoft provides a framework for defining and retrieving application data diagnostics within the PeopleSoft Internet Architecture (PIA) environment. Diagnostics 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 Diagnostics Framework, you can perform diagnostic tests on your system with minimal instructions from the PeopleSoft Support team. 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 Importing Post-Release Plug-Ins.

See Understanding Diagnostic Plug-In Development.

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, there are some differences in the way you administer them. This documentation includes topics that provide specific guidelines for administering the following supported platforms:

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

Note: *DB2 UDB for z/OS* is the official IBM name for the DBMS. For the sake of brevity, this documentation sometimes refers to DB2 UDB for z/OS as *DB2 z/OS,* and it sometimes refers to DB2 UDB for Linux, UNIX, and Windows as *DB2 LUW.*

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 the product documentation for *PeopleTools Installation* for your database platform and product line.

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

   See the product documentation for *PeopleTools: 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 PeopleSoft product documentation.

Chapter 2

Using PeopleSoft Data Mover

Understanding PeopleSoft Data Mover

This section discusses:

- PeopleSoft Data Mover overview.
- PeopleSoft Data Mover environment.
- PeopleSoft Data Mover operating modes.

Note: PeopleSoft supports a number of versions of UNIX and Linux in addition to Microsoft Windows. Throughout this documentation, we make reference to operating system configuration requirements. Where necessary, the documentation refers to specific operating systems by name. However, for simplicity the word UNIX refers to all UNIX-like operating systems, including Linux.

PeopleSoft Data Mover Overview

PeopleSoft Data Mover enables you to:

- 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.

Note: Data in PeopleSoft databases generally can't be transferred directly between major releases using PeopleSoft Data Mover. For example, you can't import data from a PeopleTools 7.x database into a PeopleTools 8.x database.

PeopleSoft Data Mover Environment

There are two ways to run PeopleSoft Data Mover:

- Using the Data Mover development environment.

This a graphical user interface (GUI), which runs only in Microsoft Windows. Use the Data Mover shortcut in the PeopleSoft program group. Select Start, Programs, *your_PSFT_program_group*, Data Mover.

- Using the Data Mover command-line interface.

  The command-line interface is intended mainly for UNIX servers. You run PeopleSoft Data Mover from a console in Microsoft Windows and from a telnet session in UNIX.

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

You set Data Mover environment variables in PeopleSoft Configuration Manager on Windows and in the psconfig.sh for UNIX.

Related Links

Setting Up UNIX to Run PeopleSoft Data Mover

PeopleSoft Data Mover Operating Modes

Operating modes determine how you are connected to the database. PeopleSoft Data Mover operating 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 script commands are not valid: EXPORT and RENAME.

Understanding Data Mover Scripts

This section discusses:

- Data Mover script commands.
- Supported SQL commands.
- Data Mover commands compared to SQL commands.
- Data Mover COMMIT statements.

Data Mover Script Commands

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

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. It then does the following: 1. creates the table. 2. creates any triggers. 3. inserts the data. 4. creates indexes.
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.

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

Related Links

Using PeopleSoft Data Mover Commands
Using PeopleSoft Data Mover Command Modifiers
Using SET Parameters

Supported SQL Commands

With PeopleSoft Data Mover, you can use supported SQL commands in scripts on any supported database platform. Except as noted in the following discussion regarding standard SQL commands, you can use all of the supported SQL commands with the following 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
- TRUNCATE

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.

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;
```

Expressing Dates and Time in SQL Used in Data Mover

When you need to express dates and time in Data Mover SQL statements, use PeopleSoft meta-SQL date and time constructs, such as %CURRENTDATEOUT, %CURRENTTIMEOUT, %CURRENTDATETIMEOUT, %DATEIN, %TIMEIN, and so on.

See "Understanding Meta-SQL" (PeopleTools 8.54: PeopleCode Language Reference).

Truncating Tables

Occasionally, it is necessary to delete all the rows in a table from Data Mover. Data Mover supports the use of the TRUNCATE command. When used, Data Mover resolves the command to the current database syntax using PeopleTools meta-SQL constructs.

For best performance, using the TRUNCATE command is recommended, rather than performing a mass DELETE operation, which can encounter performance issues where large tables and triggers are involved.

See "%TruncateTable" (PeopleTools 8.54: PeopleCode Language Reference).

Data Mover Commands Compared to SQL Commands

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 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
Delete *all* rows	DDL	TRUNCATE	None
Insert rows.	DML	INSERT STORE	IMPORT REPLACE_ALL REPLACE_DATA
Delete rows.	DML	DELETE 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 ROLLBACK	SET (when used with COMMIT or NO COMMIT)

Function	*Command Type*	*Supported SQL Commands*	*Data Mover Commands*
Control or run other PeopleSoft Data Mover commands.	Environment	None	SET RUN
Denote an explanatory statement.	Comment	None	REM REMARK --

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, PTS920HR, and so on).
- Database type (for example, Oracle, DB2, 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 Edit 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.

Related Links

Using SET Parameters
NO TRACE
PeopleSoft Data Mover Operating Modes

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.

Related Links

Understanding Data Mover Scripts

Understanding Syntax Rules

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 SYAUD01.SQR and fix any errors.
5. Use the SQL Alter function in 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 Application Designer to create all records, using the *If table exists ... Never recreate* option.

Related Links

System and Server Administration
Understanding Data Integrity Tools
Application Designer Developer's Guide

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 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 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

This section discusses how to:

- Access the Database Setup utility.
- Work with the Database Setup utility.

- Check the generated script.

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.

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.

Working with the Database Setup Utility

This section discusses the dialog boxes 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 set. 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 the Remove 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, takes the place of PeopleSoft user IDs for the sign-in process.

Table Owner

(DB2 UDB for 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 z/OS) Enter the storage group where the index spaces are created.

Table Space Storage Group (DB2 UDB for 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.

Related Links

CREATE_TEMP_TABLE
SWAP_BASE_LANGUAGE

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 the environment variable $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 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

Data Mover environment variables for UNIX are stored in the psconfig.sh script. 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.

Statement	*Description*
`$DM_HOME=$HOME/PS_DM`	Data Mover output, log file and trace file path. This environment setting needs to point to a writable location in case of a secure, read-only PS_HOME environment. The default setting is $HOME/PS_DM
`PS_DM_DATA_IN=$PS_HOME/data`	Specifies the directory where PeopleSoft Data Mover searches for input data (.DAT) files. The default setting is $PS_HOME/data.
`PS_DM_DATA_OUT=$DM_HOME/data`	Specifies the directory where PeopleSoft Data Mover writes the output data (.DAT) files. The default setting is $DM_HOME/data.
`PS_DM_SCRIPT=`*`script_path`*`;export PS_DM_SCRIPT`	$PS_DM_SCRIPT specifies the location of the PeopleSoft Data Mover script files. The default setting is $PS_HOME/scripts.
`PS_DM_LOG=`*`log_path`*`;export PS_DM_LOG`	$PS_DM_LOG specifies the location of PeopleSoft Data Mover log files. The default is $DM_HOME/log, as in as in $HOME/PS_DM/log.

Note: $DM_HOME/log is an environment variable for UNIX specifying a location to which Data Mover has write access in the case of a read-only PS_HOME configuration.

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

Note: If you have installed your PeopleSoft application outside of PS_HOME, make sure to consider the PS_APP_HOME location.

See Setting Up Tracing.

Setting Up Tracing

To enable tracing for PeopleSoft Data Mover, you must set the PS_SERVER_CFG environment variable to point to the Data Mover configuration file, which contains parameters for tracing and character set:

- In Windows, you can set PS_SERVER_CFG in the System control panel, or issue the following command in a batch file or at a command prompt:

  ```
  set PS_SERVER_CFG=%PS_HOME%\setup\psdmtx.cfg
  ```

- In UNIX, edit psconfig.sh to include the following statement:

  ```
  PS_SERVER_CFG=$PS_HOME/setup/psdmtx.cfg;export PS_SERVER_CFG
  ```

To configure tracing for PeopleSoft Data Mover, you must edit the psdmtx.cfg file to specify the appropriate tracing behavior. Use the *TraceSql* bitfield parameter to set the level of the SQL trace by

adding together the numeric 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
```

After running PeopleSoft Data Mover, look for the generated trace log file in *PS_CFG_HOME*\log \APPSRV.LOG.

For UNIX, the TraceFile environment variable enables you to configure the trace file path and name. The system only uses the TraceFile value when TraceSql is set to a value greater than 0. The default value of TraceFile is DM_HOME/datamover.trc. For example,

```
TraceFile=%DM_HOME%/datamover.trc
```

Running Data Mover Scripts from the Command Line

The PeopleSoft Data Mover command line program is located as follows:

- Microsoft Windows: *PS_HOME*\bin\client\winx86
- UNIX: *PS_HOME*/bin

At a command prompt, change to the program directory and issue the **psdmtx** command with the appropriate parameters.

Standard Command Line Syntax

Use the following standard syntax to run most Data Mover scripts:

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

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.

The `-CS server` parameter is required only for the Informix and Sybase database platforms.

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 (\).

To display a listing of all the command-line parameters and their arguments at the command prompt, enter:

```
psdmtx /help
```

Standard Command Line Parameters

The following table lists the standard command-line parameters and arguments for running the **psdmtx** command:

Parameter	*Argument*	*Example*
–CT	Specify the database type. Valid values are DB2ODBC, DB2UNIX, INFORMIX, MICROSFT, ORACLE, and SYBASE. **Note:** Notice the spelling of MICROSFT. DB2ODBC is the database type for DB2 z/OS.	`-CT ORACLE`
–CS	(Optional) 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 JPHAM2`
–CP	Specify the user password for the PeopleSoft user ID you specified.	`-CP MYPASS`
–CI	(Optional) 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	(Optional) 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`
–I	(Optional) Specify the Process Scheduler process instance.	`-I %%INSTANCE%%`

Parameter	*Argument*	*Example*
	Note: This parameter is required only if you're running PeopleSoft Data Mover from PeopleSoft Process Scheduler. You generally enter the predefined meta-string %%INSTANCE%% in the process type definition, and PeopleSoft Process Scheduler inserts the correct value at runtime.	
–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.	**psdmtx** /help

Note: When running scripts, be sure to consider your PS_APP_HOME location if you've installed your PeopleSoft application outside of PS_HOME.

Following is an example of a standard **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.

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

```
psdmtx c:\dms\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 = number
```

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 {* | recordname}
```

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 conditions];
```

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 *;
```

IMPORT

Syntax

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

Description

The IMPORT command:

- creates database spaces.
- creates nonexisting tables and indexes.
- appends non-duplicate rows to records.
- 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 table name.

Using * with AS or WHERE is allowed in cases where the imported data file consists *only* of a single table. For example,

```
IMPORT * AS table_name
```


If the data file consists of the more than one table when using * AS or WHERE, the system returns the following error message:

Error: The Input File *file_name* contains *number_of_records* records.

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.*

See WHERE.

Records defined using IMPORT can have a maximum of 500 columns with multiple long columns. The number of long columns allowed is determined by the limitations for long columns set by the database platform. Refer to your database vendor documentation for restrictions on the number of long columns allowed for the platform.

There are two variations of IMPORT that you can use:

- REPLACE_ALL
- 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

PeopleSoft Data Mover 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 Oracle 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.

Related Links

REPLACE_ALL
REPLACE_DATA

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 a definition in the PeopleSoft tables. To write the record and field change to the system tables, you must use Application Designer to modify the affected definitions.

To rename a record field, 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 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 selected by default).
   - 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.

Note: For subrecord field renames *only,* data is not automatically migrated from the old field to the new field. You will need to migrate this data manually.

To remove the old field from the tables:

1. In 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. It then:

1. creates the table.
2. creates any triggers.
3. inserts data.
4. creates indexes.

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 defaults 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 Using SET Parameters.

SET BASE_LANGUAGE

Syntax

```
SET BASE_LANGUAGE current_language_code; SWAP_BASE_LANGUAGE recordname;
```

Description

Use only when there is an error with any of the tables after the SWAP_BASE_LANGUAGE *new_language_code* command.

Note: Never run SET BASE_LANGUAGE *current_language_code* and SWAP_BASE_LANGUAGE *recordname* commands before SWAP_BASE_LANGUAGE *new_language_code*.

SET IGNORE_ERRORS

Syntax

```
SET IGNORE_ERRORS; SWAP_BASE_LANGUAGE language_code;
```

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 DUT;
```

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

```
SET IGNORE_ERRORS;
SWAP_BASE_LANGUAGE JPN;
```

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 *recordname* command

SET COMMIT

Syntax

```
Set COMMIT level;
```

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 2;
SWAP BASE_LANGUAGE FRA;
```

or

```
Set COMMIT 2;
SET BASE_LANGUAGE ENG;
SWAP_BASE_LANGUAGE MY_RECORD;
```

SWAP_BASE_LANGUAGE

Syntax

```
SWAP_BASE_LANGUAGE new_language_code;
```

or

```
SET BASE_LANGUAGE current_language_code; SWAP_BASE_LANGUAGE recordname;
```

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_code* and SWAP_BASE_LANGUAGE *recordname*) is used only when there is an error with any of the tables after the SWAP_BASE_LANGUAGE *new_language_code* command has been run.

Note: Never run a combination of SET BASE_LANGUAGE *current_language_code* and SWAP_BASE_LANGUAGE *recordname* command before SWAP_BASE_LANGUAGE *new_language_code*.

Example

To swap English for Canadian French, enter the following:

```
SWAP_BASE_LANGUAGE CFR
```

CFR is the new language code.

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.

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 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 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 IMPORT.

Parameters

IMPORT.

WHERE

Syntax

```
Data Mover Command {record | *} WHEREcondition(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

Retrieves a partial set of rows from the data source. The data source could be a data file or the database. The syntax and conditions of a Data Mover WHERE clause are similar to a WHERE clause in SQL. For EXPORT, you can write the WHERE clause with comparison operands in-line or as bind variables. For Import the WHERE clause only supports comparison operators as bind variables. Parentheses are not support in IMPORT.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

IMPORT

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_spacename} AS {new_dbname.new_spacename};
```

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 z/OS. You use this command in place of the SPACE command used on other platforms.

Parameters

IMPORT and 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_spacename AS new_dbname.new_spacename
```

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

```
SET DBSPACE *.old_spacename AS *.new_spacename
```


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_spacename

SET DBSPACE *.* AS *.new_spacename

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

IMPORT and REPLACE_ALL.

Example

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

```
SET DDL RECORD   *  INPUT dbname   AS ps910dg0;
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 IMPORT.

See IGNORE_DUPS.

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 Edit Profile, Common tab.
- If the input directory setting is blank (not set) on the Edit Profile, 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 *DM_HOME*\log.

The system uses the PeopleSoft Data Mover log directory specified on the Edit Profile, 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.

By default, the trace file is written to *DM_HOME*\trace. The default trace file name is datamover.trc.

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 an output file is not created. If you do not want to specify an output 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 Edit Profiles, 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 output file to the default output directory.

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 System and Server Administration.

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 Databases

Description

This example shows how to build a 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

Understanding PeopleSoft Data Archive Manager

Note: PeopleSoft supports a number of versions of UNIX and Linux in addition to Microsoft Windows. Throughout this documentation, we make reference to operating system configuration requirements. Where necessary, the documentation refers to specific operating systems by name. However, for simplicity the word UNIX refers to all UNIX-like operating systems, including Linux.

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 and restoration 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 elements:

- 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 the 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 the query to use.

- Restore query definition.

  You define restore jobs to restore any subset of archived data to production tables. As with archive jobs, you must first define the job information, and then run the job using the process scheduler.

- 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 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.

Building 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 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 considerations.
- 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.

Enhancing Performance

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

Index Considerations

Index considerations include:

- Database differences.
- Non-unique indexes.

Platform Considerations

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.

Non-Unique Indexes

The SQL generated by the Data Archive Manager assumes that index keys identify unique rows. Therefore, the base table of the base object must have unique indexes.

Data Limitations

For Oracle databases only, due to platform and meta-SQL restrictions, Data Archive Manager does not support archiving of records with LONG, IMAGE, or ATTACHMENT columns if you have not performed a data type switch. If you have performed a data type switch, there are no limitations. 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.

Note: This potential limitation applies *only* to Oracle databases. No other databases are affected.

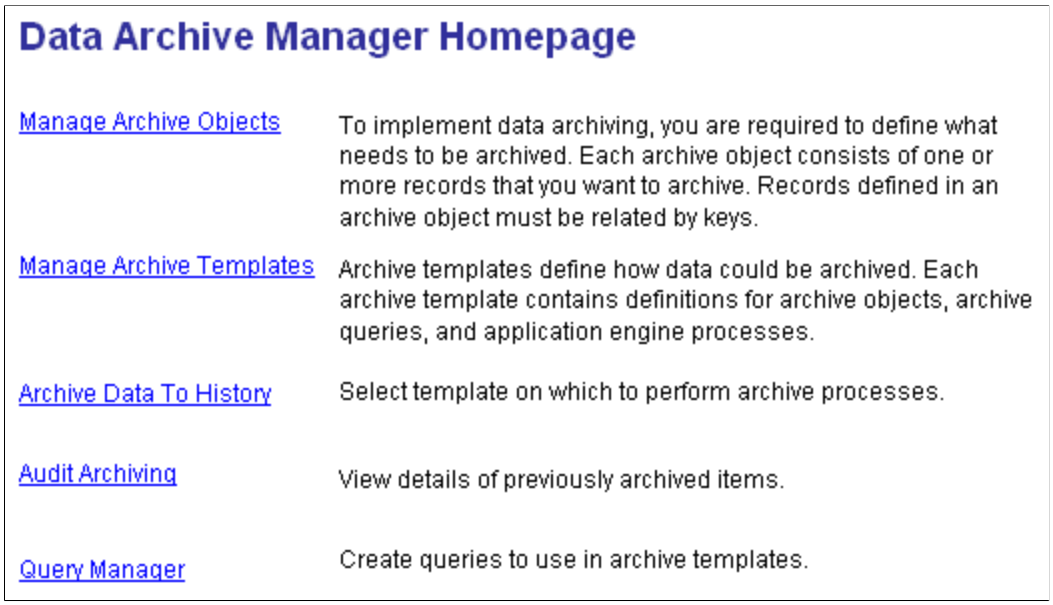
Accessing the Data Archive Manager Homepage

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 Query Manager.

Select PeopleTools, Data Archive Manager, Homepage to access the PeopleSoft Data Archive Manager homepage.

Image: Data Archive Manager Homepage

This example illustrates the fields and controls on the 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 Managing Archive Objects.

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 Managing Archive Templates.

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 Managing Archive Templates.

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

See Auditing Archive Processes.

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 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.

Managing Archive Objects

Access the Manage Archive Objects page (PeopleTools, Data Archive Manager, Manage Archive Objects).

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 and Restore 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 or *Restore* type.

For an 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.

A restore query is a type of archive query that is based on the history table rather than the online table.

See Query

Managing Archive Templates

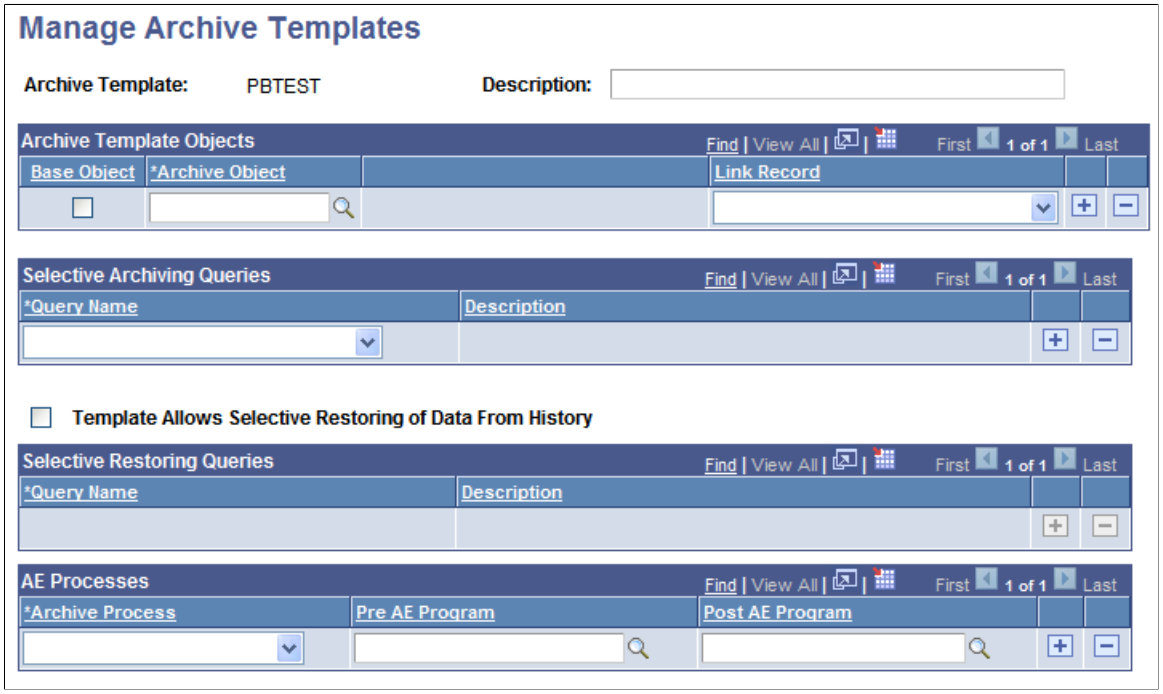
This section discusses how to manage archive templates.

Managing Archive Templates

Access the Manage Archive Templates page (PeopleTools, Data Archive Manager, Manage Archive Templates).

Image: Manage Archive Templates page

This example illustrates the fields and controls on the Manage Archive Templates page.



Archive Template Objects

Base Object Select this check box 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.

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

Description Displays the description of the query.

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* or *Restore* can be defined in the Archive Template. You can insert multiple archive or restore queries into the template.

Description Displays the description of the query.

Queries Run on Restore 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* or *Restore* can be defined in the Archive Template. You can insert multiple archive or restore 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 Application Engine (AE) programs to run for each of the archiving processes. For example, you can define an Application Engine 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 Application Engine 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.

Managing Archive Jobs

This section discusses how to:

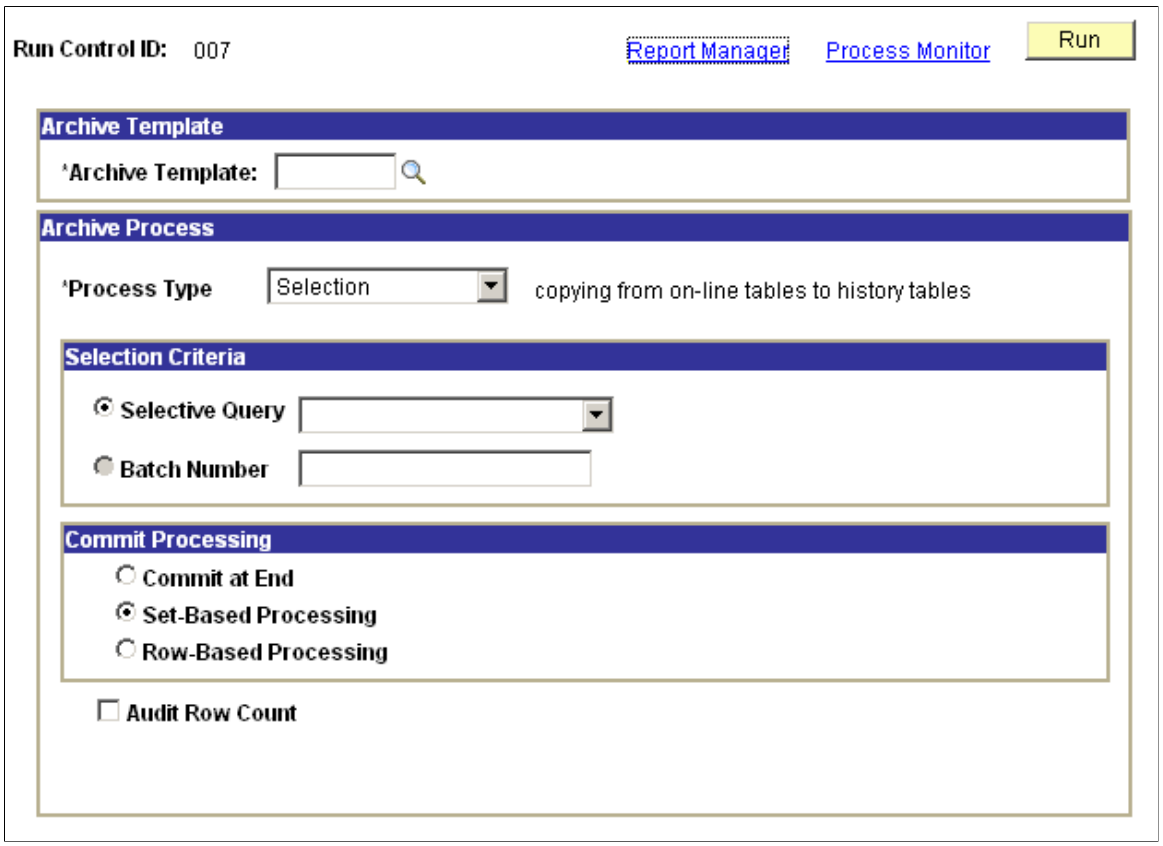
- Define archive jobs.
- View details.
- Define archive query binds.

Defining Archive Jobs

Access the Archive Data To History page (PeopleTools, Data Archive Manager, Archive Data to History).

Image: Manage Archive Jobs page

This example illustrates the fields and controls on the Manage Archive Jobs page.



Archive Template Select the archive template to use for this batch job. Choosing a restore or an archive template is the way you choose the type of job to run.

Run Click to run this batch job after defining the archive process 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 Viewing Details.

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 commits by table, 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 will perform a rollback of everything that has not committed.

For commits by row, 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.

Process Type

Select an option:

- Select Delete to delete data from transaction tables. Data rows will be deleted from the transaction tables only if they've already been archived in the history tables.
- Select Rollback to copy data from history tables back to transaction tables.

  Important! History rows have the same keys as their corresponding transaction rows, so attempting to copy them to the transaction tables will fail with a duplicate key error if the transaction rows still exist. Before running a rollback process for a given archive job, you must first run a delete process to delete the transaction rows for the same job, so that the history rows can be successfully copied into the transaction tables.
- Select Remove from History to delete data from the history tables.

Selective Query

Specify the archive or restore query defined within the archive template to use at run time. 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 Defining Archive Query Binds.

Batch Number

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.

Batch Number

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. This number is displayed in the Number of Rows field on the Audit Archiving page.

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

Commit at End 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.

Commit by Table 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.

Commit by Row 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 Control Details page (PeopleTools, Data Archive Manager, Archive Data to History, View Details).

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 (PeopleTools, Data Archive Manager, Archive Data to History, Define Binds).

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:

Audit Archiving

Access the Audit Archiving page (PeopleTools, Data Archive Manager, Audit Archiving).

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. **Note:** This field displays valid information only if you selected Audit Row Count on the Archive Data to History page.
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

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.

It is good practice to run and read the audit reports, which include SYSAUDIT, DDDAUDIT, and ALTER audit, after making changes such as:

- Applying patches.
- Applying bundles.
- Performing database upgrades.

Running the audits helps you to make sure that the tables are internally and externally synchronized.

Furthermore, it is recommended that you schedule regular maintenance runs of these audits, for example weekly, enabling you to discover and resolve any data integrity issues in a timely manner.

Running SQL Alter

The primary purpose of the 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 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 OK to 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 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 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 queries that check tables, views, indexes, and triggers.

DDDAUDIT.SQR. is located in *PS_HOME*\sqr.

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.

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 Application Designer and not found in the database.	Use Application Designer to create the index.
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 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 Application Designer.
TABLE-4	Tablespace is not defined for the SQL table in Application Designer.	If you're using or migrating to a relational database management system that uses table spaces, you should use Application Designer to assign table spaces to these tables.

Query	*If Rows are Returned?*	*Resolution*
TABLE-5	Table contains more than 500 fields.	Use 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 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 Application Designer.
TRIGGER-1	Trigger defined in the Application Designer and not found in the database.	Delete the definition if it is not needed. Otherwise, use 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 Data Set integrity
- Application Engine integrity.
- Clear list integrity.
- EDI Manager integrity.
- Field integrity.
- Feeds integrity.
- Integration Broker integrity.
- Menu integrity.
- Optimization integrity.
- Page integrity.
- PeopleCode integrity.
- Process Scheduler.
- Query integrity.
- Record integrity.
- Related language integrity.

- Security integrity.
- SQL integrity.
- Tree integrity.
- Translate integrity.
- PSLOCK integrity.
- BI Publisher integrity.

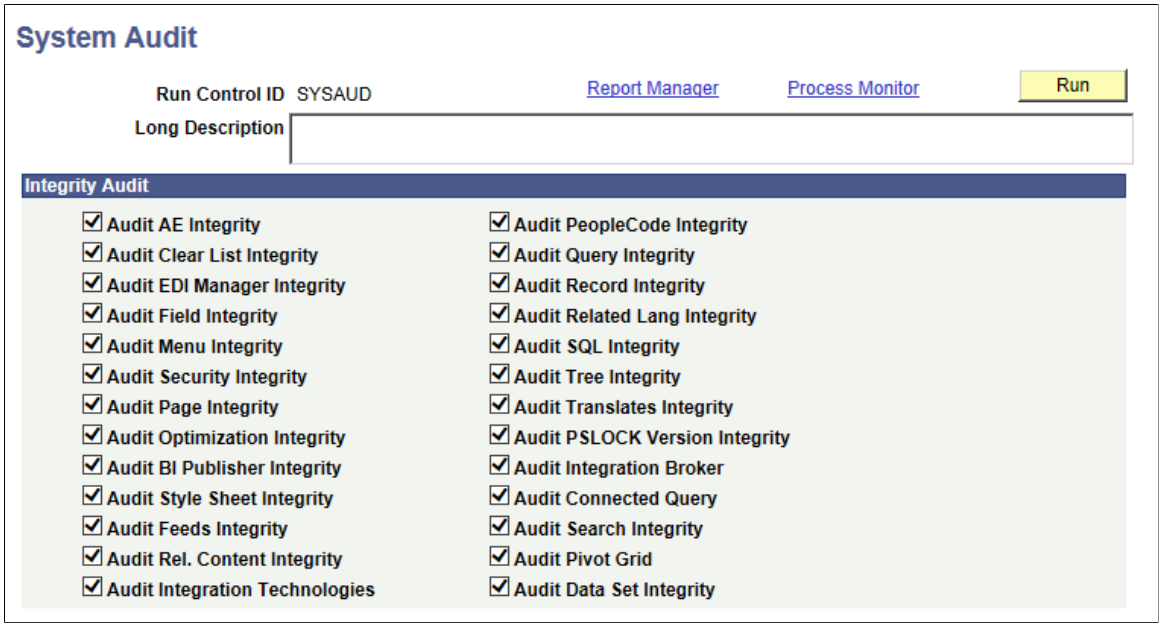
Understanding How to Run SYSAUDIT

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. Select the appropriate check boxes to run the audits that you want.

Image: System Audit page

This example illustrates the fields and controls on the 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 Application Designer fields.

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

Audit Security Integrity	Audits the AUTH*, OPRDF* components for PeopleTools Security.
Audit Page Integrity	Audits the PNL* component for Application Designer pages.
Audit Optimization Integrity	Audits the definitions for Optimization Engine.
Audit BI Publisher Integrity	Audits the referential integrity of the tables of the definitions that are associated with BI Publisher.
Audit Style Sheet Integrity	Audits the referential integrity of the tables of the definitions that are associated with style sheets.
Audit Feeds Integrity	Audits the referential integrity of the tables of the definitions that are associated with feeds.
Audit Rel. Content Integrity	Audits the referential integrity of the tables and definitions associated with the Related Content.
Audit Integration Technologies	Audits integration technologies other than those used for PeopleSoft Integration Broker, including file processing URL definitions and Ren Server SSL configuration.
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 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 PSLOCK Version Integrity	Audits the VERSN* component.
Audit Integration Broker	Runs a collection of audits on the Integration Broker configuration.
Audit Connected Query Integrity	Audits the referential integrity of the tables of the definitions that are associated with connected query.
Audit Search Integrity	Audits the referential integrity of the tables and definitions associated with the PeopleSoft Search Framework and PeopleSoft Search.
Audit Pivot Grid	Audits the definitions for pivot grids.

Audit Data Set Integrity Audits the definitions for application data sets.

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.

Accessing SYSAUDIT Output

When you run SYSAUDIT, you can specify the type and format of the output on the Process Scheduler Request page, as you can with any Process Scheduler request. By default, the results are written to the configured report repository as an Adobe Acrobat PDF file called SYSAUDIT_*runctrl_ID*.pdf, where *runctrl_ID* is the run control ID you specified for the audit.

The tables in the following sections list 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 these tables are arranged alphabetically, and are not necessarily in the order in which they appear in the output.

Related Links

"Scheduling Process Requests" (PeopleTools 8.54: Process Scheduler)

Application Data Set Integrity

The following table describes the audit queries and resolutions for the application data sets (ADS):

Query	*Description*	*Resolution*
SYSADS-01	This audit is run against the PSADSDEFNITEM table to check if all the record definitions references in a data set definition exist.	Remove the missing record from the data set, or create the missing record.
SYSADS-02	This audit is run against PSADSDEFNITEM table to check if the parent record definitions references in a data set definition exists.	Remove the missing record from the data set, or create the missing record.

Query	Description	Resolution
SYSADS-03	This audit is run against PSADSDEFN table to check if all the specified extension application classes for a data set exist. While copying a new data set from a source database if the application package and its PeopleCode are not copied then the extension application classes for the data set may become missing.	Copy the associated application package and its PeopleCode together with the dataset.
SYSADS-04	This audit is run against PSADSDEFN table to check if the component from which a data set is derived exists. While copying a component-derived data set if you miss copying the component then orphaned data sets will arise.	Copy the flagged component together with the derived data set.

Application Engine Integrity

The following table describes the audit queries and resolutions for this area:

Query	Description	Resolution
AE-01	This audit lists the AE programs without any sections.	If the affected program is delivered by PeopleSoft and is not modified, contact My Oracle Support. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. 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.

Query	*Description*	*Resolution*
AE-02	This audit lists the AE sections without AE programs.	If the affected program is delivered by PeopleSoft and is not modified, contact My Oracle Support. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. If the affected program is a customization, it is not possible to recover the missing program. Restore it from a backup if needed. Run SysAECleanUp.dms to remove any orphans remaining after you have followed the steps above.
AE-03	This audit lists the AE state records without AE programs.	If the affected record is delivered by PeopleSoft, contact My Oracle Support. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. Otherwise, ignore the warnings or restore the program from a backup. It is not possible to recover the missing program.
AE-04	This audit lists the AE state records without record definitions.	If the affected record is delivered by PeopleSoft, contact My Oracle Support. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. Otherwise, use Application Designer to remove invalid records from the program definition or create record definitions.
AE-05	This audit lists the AE section details without base section definitions.	If the affected program is delivered by PeopleSoft and is not modified, contact My Oracle Support. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. Otherwise, ignore the warnings or restore the program from a backup. It is not possible to recover the missing sections.

Query	*Description*	*Resolution*
AE-06	This audit lists the AE steps without sections.	If the affected program is delivered by PeopleSoft and is not modified, contact My Oracle Support. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. If the affected program is a customization, it is not possible to recover the missing program. Restore it from a backup if needed. Run SysAECleanUp.dms to remove any orphans remaining after you follow the steps above.
AE-07	This audit lists the AE Call Section actions referring to nonexistent sections	If the affected program is delivered by PeopleSoft and is not modified, contact My Oracle Support. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. 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	This audit lists the AE Log Message actions without an AE step.	If the affected record is delivered by PeopleSoft, contact My Oracle Support. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. If the affected program is a customization, it is not possible to recover the missing program; restore it from a backup if needed. Run SysAECleanUp.dms to remove any orphans remaining after you follow the steps above.

Query	*Description*	*Resolution*
AE-09	This audit lists the AE actions without an AE step.	If the affected record was delivered by PeopleSoft, contact My Oracle Support. If the affected program was converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. If the affected program is a customization, it is not possible to recover the missing program; restore it from a backup if needed. Run SysAECleanUp.dms to remove any orphans remaining after you follow the steps above.
AE-10	This audit lists the AE temp tables that are attached to invalid AE programs.	If the affected temp table was delivered by PeopleSoft, contact My Oracle Support. If the affected program is converted as part of an upgrade, this may be a symptom that the conversion failed. Contact My Oracle Support. Otherwise, ignore the warnings or restore the program from a backup. It is not possible to recover the missing programs.
AE-11	This audit lists the orphaned AE PeopleCode.	Because of platform issues and Structured Query Report (SQR), this check may not be included in the audit report. Run SysAECleanUp.dms to clean up these orphans.
AE-12	This audit lists the orphaned AE SQL objects.	Because of platform issues and SQR, this check may not be included in the audit report. Run SysAECleanUp.dms to clean up these orphans.
AE-13	This audit verifies that PS_ AEONLINEINST contains the correct number of rows.	Run ps_aeonlineinst.dms. If you do not have the Data Mover script, contact My Oracle Support.
AE-14	This audit verifies that PS_ AEINSTANCENBR contains the correct number of rows.	Run ps_aeinstancenbr.dms. If you do not have the Data Mover script, contact My Oracle Support.
AE-15	This audit verifies that PS_ AELOCKMGR contains a row .	Resolution may vary based on implementation. Contact My Oracle Support.

Note: Locate the Data Mover scripts in the PS_HOME\scripts directory unless otherwise noted.

BI Publisher Integrity

The following table describes the audit queries and resolutions for this area:

Query	*Description*	*Resolution*
BIP-01	Query Data Source does not exist in Query Definition table.	Remove the data source and all report definitions using the data source.
BIP-02	Data Source Definition used by a Report Definition but does not exist in Data Source Definition table.	If this error is observed after the upgrade copy process, try copying the data source object from source database. If this is not the case, remove the report definition.
BIP-03	Template Definition used by a Report Definition but does not exist in Template Definition table.	If this error is observed after the upgrade copy process, try copying the missing template definition from the source database. If this is not the case, remove the report definition.
BIP-04	Template Definition not associated with any Report Definition.	Remove the template object.
BIP-05	Template Definition associated with more than one Report Definition.	Delete all the report definitions using the template object and recreate them again from the user interface.
BIP-06	Sub-Template Definition associated with any Report Definition.	Delete all the report definitions using the template object and recreate them again from the user inferface.
BIP-07	Template File does not exist in file table.	If this error is observed after the upgrade copy process, try copying the missing file definition from the source database. If this is not the case, the template definition should be deleted and recreated.
BIP-08	PDF Map File does not exist in file table.	If this error is observed after the upgrade copy process, try copying the missing file definition from the source database. If this is not the case, the template definition should be deleted and recreated.
BIP-09	XLIFF File does not exist in file table.	If this error is observed after the upgrade copy process, try copying the missing file definition from the source database. If this is not the case, the template definition should be deleted and recreated.
BIP-10	File definition not used by template file definition.	Run Application Engine program PSXPCLEAN to delete orphan file definitions.

Query	Description	Resolution
BIP-11	File data not referenced by file definition.	Run Application Engine program PSXPCLEAN to delete orphan file definitions.
BIP-12	File definitions not referenced by file data.	Run Application Engine program PSXPCLEAN to delete orphan file definitions.

Clear List Integrity

The following table describes the audit queries 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.
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.

Query	Description	Resolution
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.
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.

Connected Query Integrity

The following table describes the audit queries and resolutions for this area:

Query	Description	Resolution
SysConqrs-01	Identifies Connected Query definitions in the PSCONQRSMAP table that do not exist in the PSCONQRSDEFN table.	Delete the invalid definition from the PSCONQRSMAP table and the PSCONQRSFLDREL table (if it exists).

Query	*Description*	*Resolution*
SysConqrs-02	Identifies Connected Query definitions in the PSCONQRSDEFN table that do not exist in the PSCONQRSMAP table.	Delete the invalid definition using Connected Query Manager.
SysConqrs-03	Identifies Connected Query definitions in the PSCONQRSFLDREL table that do not exist in the PSCONQRSMAP table.	Delete the invalid definition using Connected Query Manager or delete invalid definition directly from the PSCONQRSFLDREL table and the PSCONQRSDEFN table (if it exists).
SysConqrs-04	Identifies Connected Query definitions being used in the PSCONQRSRUNCNTR table that do not exist in the PSCONQRSDEFN table.	Delete the invalid definition from the PSCONQRSRUNCNTR table and the PSCONQRSRUNPRM table (if it exists).
SysConqrs-05	Identifies Connected Query definitions being used in the PSCONQRSRUNPRM table that do not exist in the PSCONQRSRUNCNTR table.	Delete the invalid definition from the PSCONQRSRUNPRM table.
SysConqrs-06	Verifies query definition usage according to the following rules: • Query definitions being used in PSCONQRSMAP table should exist in PSQRYDEFN table. • Only public query definitions should be used for public Connected Query definitions. • Private query definitions can be used for private Connected Query definitions only if they have the same owner.	If the query referenced in a Connected Query definition does not exist, delete it or open it in the Connected Query Manager and correct query selection and file mapping. If a public Connected Query uses a private query, delete it or open it in the Connected Query Manager and correct query selection and file mapping. You can also save an invalid Connected Query as a private query (with the same owner) with a new name and delete the original invalid public query. After deleting the original Connected Query, save the new private Connected Query using the original name.
SysConqrs-07	Verifies Connected Query structure (PSCONQRSMAP table), ensuring: • No duplicate parents exist for child queries. • No duplicate combinations of parent and child queries exist for a Connected Query.	Delete the invalid definition using Connected Query Manager.
SysConqrs-08	Verifies Connected Query structure (PSCONQRSMAP table), ensuring a parent(root) query is defined for all Connected Query definitions.	Delete the invalid definition using Connected Query Manager.
SysConqrs-09	Verifies Connected Query structure (PSCONQRSMAP table), checking if any parent query exists as a child query.	Delete the invalid definition using Connected Query Manager.

Query	Description	Resolution
SysConqrs-10	Verifies Connected Query structure (PSCONQRSMAP table), ensuring that the number of child queries should be equal to the maximum value of the SEQNUM field.	Open the Connected Query and re-map member query fields. If the Connected Query can't be opened, it should be deleted using Connected Query Manager.

EDI Manager Integrity

The following table describes the audit queries 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.
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.

Query	*Description*	*Resolution*
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.
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.

Query	Description	Resolution
SYSECMGR-1	Inbound work record field does not exist in the type definitions in PSDBFIELD	Select, PeopleTools, EDI Manager, Setup Trading Partners, Remove EDI Manager Objects. In the Delete EC Map field, delete all maps *except* the following: • AUDIT • BCWCB • CL_APP_V4 • CL_CHNG_V4 • CL_EFT_V4 • CL_ORIG_V4 • OUAC-LAW-A • OUAC-LAW-U • OUAC-PT • OUAC-TEA-A • OUAC-TEA-U • OUAC_UAS_A • OUAC_UAS_U • TS130_MAP • TS189_MAP

Feeds Integrity

The following table describes the audit queries and resolutions for this area:

Query	Description	Resolution
FEED-01	Feed referencing a nonexistent feed data type.	Remove the feed referencing the nonexistent feed data type from the feed definition table.
FEED-02	Feed data type Integration Broker operations table referencing a nonexistent feed data type.	Remove the Integration Broker operations referencing the nonexistent feed data type from the feed data type Integration Broker operations table. Run the following SQL: `DELETE FROM PSFP_DTYPE_IBSO WHERE PTFP_DATATYPE_ ID='<DATATYPEID>'`

Query	*Description*	*Resolution*
FEED-03	Default feed attributes table referencing a nonexistent feed data type.	Remove the default feed attributes referencing the nonexistent feed data type from the default feed attributes table. Run the following SQL: `DELETE FROM PSFP_DTYPE_ATTR WHERE PTFP_DATATYPE_ID='<DATATYPEID>'`
FEED-04	Feed attributes table referencing nonexistent feed.	Remove the feed attributes referencing the nonexistent feed from the feed attributes table. Run the following SQL: `DELETE FROM PSFP_ATTRS WHERE PTFP_FEED_ID='<FEEDID>'`
FEED-05	Feed data source settings table referencing a nonexistent feed.	Remove the data source settings referencing the nonexistent feed from the feed data source settings table. Run the following SQL: `DELETE FROM PSFP_SETTINGS WHERE PTFP_FEED_ID='<FEEDID>'`
FEED-06	Feed data source parameters table referencing a nonexistent feed.	Remove the data source parameters referencing the nonexistent feed from the feed data source parameters table. Run the following SQL: `DELETE FROM PSFP_PARMS WHERE PTFP_FEED_ID='<FEEDID>'`
FEED-07	Feed security table referencing a nonexistent feed.	Remove the feed security referencing the nonexistent feed from the feed security table. Run the following SQL: `DELETE FROM PSFP_SECURITY WHERE PTFP_FEED_ID='<FEEDID>'`
FEED-08	User specified data source parameter table referencing a nonexistent feed.	Remove the user specified data source parameters referencing the nonexistent feed from the user specified data source parameter table. Run the following SQL: `DELETE FROM PSFP_PVALS WHERE PTFP_FEED_ID='<FEEDID>'`

Query	*Description*	*Resolution*
FEED-09	Admin personalization table referencing a nonexistent feed.	Remove admin personalization data referencing the nonexistent feed from the admin personalization table. Run the following SQL: `DELETE FROM PSFP_ADMN_PREF WHERE PTFP_FEED_ID='<FEEDID>'`
FEED-10	User personalization table referencing a nonexistent feed.	Remove user personalization data referencing the nonexistent feed from the user personalization table Run the following SQL: `DELETE FROM PSFP_USER_PREF WHERE PTFP_FEED_ID='<FEEDID>'`

FEED-11		Remove the invalid category association in the feed. Run the following SQL, depending on your database platform.
		
		Oracle:
		
		`UPDATE PSFP_FEED SET PTFP_CATEGORY_ID = ' ' WHERE PTFP_FEED_ID='<FEEDID>'`
		
		Non-Oracle:
		
		`UPDATE PSFP_FEED SET PTFP_CATEGORY_ID = '' WHERE PTFP_FEED_ID='<FEEDID>'`
		
		Note: There is a space within the quotes for an Oracle database, and no space within quotes for a non-Oracle database.

GENERICFEED-01	Integration Broker Generic Feed referencing a nonexistent Integration Broker service operation.	Remove the Integration Broker Generic Feed that references a nonexistent Integration Broker service operation.
WORKLISTFEED-01	Worklist feed referencing a nonexistent activity.	Remove the worklist feed that references a nonexistent activity.
WORKLISTFEED-02	Worklist feed referencing a nonexistent business process.	Remove the worklist feed referencing a nonexistent business process.
WORKLISTFEED-03	Worklist feed referencing a nonexistent event.	Remove the worklist feed referencing a nonexistent event.
WORKLISTFEED-04	Worklist feed referencing a nonexistent worklist name.	Remove the worklist feed referencing a nonexistent worklist name.

Query	*Description*	*Resolution*
WORKLISTFEED-05	Worklist feed referencing a nonexistent "From" user.	Remove the worklist feed referencing a nonexistent user.
QUERYFEED-01	Query feed definition referencing a nonexistent query.	Remove the query feed referencing the nonexistent query.
QUERYFEED-02	Query feed parameter definition referencing a nonexistent query bind.	Remove the query feed parameter referencing a nonexistent query bind. Run the following SQL: `DELETE FROM PSFP_PARMS WHERE PTFP_FEED_ ID='<FEEDID>' AND PTFT_FIELD_NAME = '<FIELDNAME>'`
QUERYFEED-03	Query feed entry element maps to a template that referencing a nonexistent query result column.	Remove query feed entry template referencing a nonexistent query result column. Run the following SQL: `DELETE FROM PSFP_ATTRS WHERE PTFP_FEED_ ID='<FEEDID>' AND PTFT_FIELD_NAME = '<QRYFLDNAME>'`

To remove a feed shown in these audits:

- FEED-01
- GENERICFEED-01
- WORKLISTFEED-01
- WORKLISTFEED-02
- WORKLISTFEED-03
- WORKLISTFEED-04
- WORKLISTFEED-05
- QUERYFEED-01

Run the following SQL:

```
DELETE FROM PSFP_ADMN_PREF WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_USER_PREF WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_ATTRS WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_ATTRS_LANG WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_FEED WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_FEED_LANG WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_PARMS WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_PARMS_LANG WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_PUB_SITES WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_PVALS WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_PVALS_LANG WHERE PTFP_FEED_ID IN ('<FEEDID>');
```

```
DELETE FROM PSFP_SECURITY WHERE PTFP_FEED_ID IN ('<FEEDID>');
DELETE FROM PSFP_SETTINGS WHERE PTFP_FEED_ID IN ('<FEEDID>');
```

Field Integrity

The following table describes the audit queries and resolutions for this area:

Query	*Description*	*Resolution*
FIELD-3	This query lists invalid default fields.	Modify the default value in record field properties.
FIELD-4	This query lists fields that are used in record definitions but do not exist in PSDBFIELD.	Define the field in Application Designer.
FIELD-5	This query lists fields that have multiple default field labels in PSDBFLDLABL.	Open the field, select the default label, and save.
FIELD-06	This query lists deleted fields that have orphaned field labels in PSDBFLDLABL.	Run this SQL: `DELETE FROM PSDBFLDLABL WHERE FIELDNAME NOT IN (SELECT FIELDNAME FROM PSDBFIELD)`

File Processing Integrity

The following table describes the audit queries and resolutions for this area:

Query	*Description*	*Resolution*
FILEPROC-01	Applies to file processing URLs which use HTTPS and FTPS protocols. The system audit utility lists all the URLs that do not have a `KEYSTOREPASSWORD` property defined for them. File processing uses the password for PKCS12 files used internally. For further information on `KEYSTOREPASSWORD` property, see "URL Maintenance" (PeopleTools 8.54: System and Server Administration).	Use the `KEYSTOREPASSWORD` property for all FTPS and HTTPS URLs. It ensures that PKCS12 certificates used internally are password protected.

Integration Broker Integrity

The following table describes the audit queries and resolutions for this area:

Query	*Description*	*Resolution*
IBRK-01	Message parts referencing a message/ version that does not exist.	Most likely caused by moving a container message in a project and not including all the part messages. Always make sure that when moving a container message in a project, every part message is also included if it doesn't already exist in the target database.
IBRK-02	Rowset based messages referencing records that do not exist.	Most likely caused by moving a message in a project and not including records referenced in the message. Always make sure that when moving a rowset-based message in a project, every record that is not in the target database is also included in the project.
IBRK-03	Message parts must reference message/ versions that are defined as part messages.	Most likely caused by moving a container message in a project and not including records referenced in the message. By definition, only part messages can be found in a container message. Always make sure that when moving a container message in a project, every part message contained is also in the project, unless the part message is already in the target database defined as a part message.
IBRK-04	Service operations need at least one version, the default.	Service operations cannot exist without a service operation version. These are separate managed objects and can therefore be added individually into a project. However, care must be taken when moving service operations in projects to always include the default version if it doesn't already exist in the target database.
IBRK-05	Service references service operation(s) that do not exist	This audit reports any services that contain service operations that don't exist. Typically, this is caused when moving a service from one database to another by including the service in the project but not including all related service operations. Care must be taken when moving services to always include the associated service operations in the project, unless the same service already exists in the target database.

Query	*Description*	*Resolution*
IBRK-06	Service operation versions must have a valid Service operation	Service operations and service operation versions are separately managed objects. When moving service operation versions in a project, be sure to include the service operation with the same name unless the service operation already exists in the target database.
IBRK-07	Handlers must have a valid service operation.	Service operation handlers and service operations are separately managed objects. When moving handlers, be sure to include the related service operation unless the operation already exists in the target database.
IBRK-08	Routings must reference a valid service operation version. **Note:** Routing IB_ADMIN_ROUTING is excluded because it is a dummy routing used during upgrade to 8.48.	Service operation routings and service operation versions are separately managed objects. When moving routings, be sure to include the related service operation version unless the operation version already exists in the target database.
IBRK-09	Routings must reference valid nodes	Service operation routings and nodes are separately managed objects. When moving routings, be sure to include any related Node that doesn't exist in the target database.
IBRK-10	Routings must reference valid service operation handlers	Service operation routings and service operation Handlers are separately managed objects. When moving routings, be sure to include the related service operation Handlers that appear in the routing component unless the Handlers already exist in the target database.
IBRK-11	Routing parameters must reference valid transform application engine programs	Service operation routings reference application engine transform programs. When moving routings, be sure to include any referenced Application Engine programs unless the programs that appear in the routing parameter component already exist in the target database.
IBRK-12	Routing parameters must reference valid message/version combinations.	Service operation routings reference messages. When moving routings, be sure to include any referenced messages unless the messages that appear in the routing parameter component already exist in the target database.

Query	*Description*	*Resolution*
IBRK-13	Service operation versions need to reference valid messages	Service operation versions reference messages. When moving versions in a project, be sure to include any referenced messages unless the messages that appear in the service operation or service operation versions component already exist in the target database.
IBRK-14	Service operation versions need to reference valid queues	Service operation versions reference Integration Broker queues. When moving versions in a project, be sure to include any referenced queues unless the queues that appear in the service operation or service operation versions component already exist in the target database.
IBRK-15	Service operation versions need to reference valid transform programs	Service operation versions reference application engine transform programs. When moving versions in a project, be sure to include any referenced application engine programs unless the programs that appear in the service operation versions component already exist in the target database.
IBRK-16	Service operation versions with validation turned on require each -- message to have a schema defined	Service operations versions with validation turned on (see the Service Operations component), require all referenced messages to have schemas defined. When moving service operation versions that have validation turned on, include all referenced messages unless the referenced messages already exist in the target database with valid schemas. Also note that when moving messages in projects the related schemas are not brought along. These need to be moved via Data Mover scripts. (PSIBMSGSCHEMA_IMP.DMS and PSIBMSGSCHEMA_EXP.DMS)
IBRK-17	Component interface handlers should reference valid component interfaces.	Service operation handlers that are of type CI reference component interfaces. When moving service operation handlers, be sure to also include any referenced component interfaces in the project unless the CIs already exist in the target database.

Query	*Description*	*Resolution*
IBRK-18	Application class handlers should reference valid application packages	Service operation handlers that are of type Application Class reference application classes. When moving service operation Handlers, be sure to also include any referenced application classes in the project unless they already exist in the target database. Care must be taken to make sure that the referenced application class PeopleCode is also included in the project.
IBRK-19	Part messages need to have a schema defined.	When moving messages in projects, the related schemas are not brought along. The schemas need to be moved using Data Mover scripts. (see PSIBMSGSCHEMA_IMP.DMS and PSIBMSGSCHEMA_EXP.DMS). For part messages, having a schema is mandatory.
IBRK-20	Container messages need to have a schema defined.	When moving messages in projects, the related schemas are not brought along. These need to be moved using Data Mover scripts. (see PSIBMSGSCHEMA _IMP.DMS and PSIBMSGSCHEMA _EXP.DMS). For container messages, having a schema is mandatory.
IBRK-21	Operations with duplicate routings.	This audit reports service operations with multiple active routings where the sender and receiver node are identical. There should never be multiple active routings where the sender and receiver node are identical for a service operation. Delete or inactivate one of the offending routings for each service operation.

Query	*Description*	*Resolution*

IBRK-22	Operations with duplicate ANY routings	This audit reports service operations with multiple active routings where the sender node is set to *ANY*. There should never be multiple active routings where the sender node is set to *ANY* for a service operation.
		
		Delete or inactivate one of the offending routings for each service operation.
		
		You can change the status of one of the duplicate routings as follows:
		
		`UPDATE PSIBRTNGDEFN`
		`SET EFF_STATUS = 'I'`
		`WHERE ROUTINGDEFNNAME =`
		` <NAME>`
		`AND EFFDT = <EFFDT>`
		
		To delete a duplicate routing, select PeopleTools, Integration Broker, System Utilities, Service Administration. On the Routings tab, find the routing to remove and remove it. Alternatively, you can run the following set of SQL statements:
		
		`DELETE FROM PSIBRTNGDEFN`
		`WHERE ROUTINGDEFNNAME =`
		` <NAME>`
		`AND EFFDT = <EFFDT>`
		
		`DELETE FROM PSIBRTNGSUBDEFN`
		`WHERE ROUTINGDEFNNAME =`
		` <NAME>`
		`AND EFFDT = <EFFDT>`
		
		`DELETE FROM PSRTNGDFNPARM`
		`WHERE ROUTINGDEFNNAME =`
		` <NAME>`
		`AND EFFDT = <EFFDT>`
		
		`DELETE FROM PSRTNGDFNCONPRP`
		`WHERE ROUTINGDEFNNAME =`
		` <NAME>`
		`AND EFFDT = <EFFDT>`

IBRK-23	Operation with no Service Relationship	There should never be operations referencing services that don't exist. Typically, this is caused when importing a project containing a service operation that belongs to a service that does not exist in the target database. Always make sure that the services that a service operation belongs to exist in the target database or are included in the import project. All service operations must belong to at least one service.

Query	Description	Resolution
IBRK-24	DMS Handler referencing invalid message	This audit identifies handlers of type Data Mover with a missing message. Certain Data Mover handlers require an associated message. When moving Data Mover handlers in a project, all related messages should also be included. To resolve issues, locate the message(s) in the database that was used to create the handler. Import that message into the target database (where you ran SYSAUDIT). If you can't locate the message, recreate the handler in the source database and then be sure to include both the handler and the related message in the project, and import the project.
IBRK-25	IB PSOPERATIONAC referencing a handlername that doesn't exist.	Service operation handler definitions refer to Application Classes that contain the Peoplecode logic that gets triggered when requests come in. This audit identifies service operation handlers that have a defined Application Class name, but the Application Class doesn't exist in the database. To resolve issues, either copy the Application Package/Application Class referred to by the handler definition, or delete the service operation using the following SQL statement: `DELETE FROM PSOPERATIONAC WHERE IB_OPERATIONNAME = <NAME>`
IBRK-26	IB Nodes with missing gateway definition.	You must specify the integration gateway that a node uses for integrations. On the Nodes-Connectors page, define the integration gateway to use for integrations in the Gateway ID field.

Menu Integrity

The following table describes the audit queries 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';`

Query	Description	Resolution
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-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.

Optimization Integrity

The following table describes the audit queries 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 = 'recordname';`
		`DELETE FROM PSOPTFIELD WHERE RECNAME ='recordname';`

OPTZN-02	Optimization delete records that do not have matching definitions.	In 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 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 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.

Query	*Description*	*Resolution*
OPTZN-06	Optimization base record defn has the trigger flag set but has no delete record name, or vice versa.	Using 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-07	Optimization records that need to have trigger flag set and do not.	Using 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 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-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 FROM PSOPTSYNC`
		`WHERE RECNAME NOT IN`
		`(SELECT RECNAME`
		`FROM PSOPTREC)`

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 Application Designer. Inspect the offending transaction parameter and make sure that the default value contains a valid record name.

Query	*Description*	*Resolution*
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.
OPTZN-17	This query identifies readable base records in an analytic type that don't have an optimization delete record specified.	In Application Designer, either specify a delete record for the analytic type record, or clear the Readable check box for the analytic type record.
OPTZN-18	This query identifies base records in an analytic type that have an optimization delete record specified, but aren't readable.	In Application Designer, either select the Readable check box for the analytic type record, or don't specify an optimization delete record for the analytic type record.
OPTZN-19	This query identifies base records and their associated delete records in an analytic type that don't have all fields in the same order.	In Application Designer, change the field order of one of the records to match the field order of the other record.
OPTZN-21	This query identifies fields in main records used in an analytic model that aren't selected in the analytic type associated with that model.	In Application Designer, either specify an appropriate record in the analytic model, or select the appropriate corresponding fields in the analytic type definition.
OPTZN-22	This query identifies fields in aggregate records used in an analytic model that aren't selected in the analytic type associated with the model.	In Application Designer, either specify an appropriate record in the analytic model, or select the appropriate corresponding fields in the analytic type definition.

Page Integrity

The following table describes the audit queries 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 Application 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 Application Designer to change each of these components to reference an access detail page that exists.
PAGE -10	A component's specified search record has no corresponding row in the RecordDefinition table.	Use Application 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 Application Designer to change each of these components to reference an add search record that exists.
PAGE-12	There is a discrepancy between MAXPNLFLDID on the page definition table (PSPNLDEFN) and PNLFLDID on the page field table (PSPNLFIELD).	Use your SQL Editor to set PSPNLDEFN.MAXPNLFLDID equal to the highest PSPNLFIELD.PNLFLDID value for the page.

PeopleCode Integrity

The following table describes the audit queries 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 Application Designer in two-tier mode in order to compile all the PeopleCode. Select the Compile all and Save all PeopleCode option in the Application Designer.. See also, "Finding References to Application Packages and Classes" (PeopleTools 8.54: PeopleCode Developer’s Guide).
PEOPLECODE-4	PeopleCode contains invalid FILELAYOUT References.	Open the PeopleCode program in Application Designer and correct the invalid reference.
PEOPLECODE-5	PeopleCode reference to an invalid record or field.	Open the PeopleCode program in Application Designer and correct the invalid reference.
PEOPLECODE-6	PeopleCode reference to an invalid field.	Open the PeopleCode program in Application Designer and correct the invalid field name.

Query	*Description*	*Resolution*

PEOPLECODE-7	There is orphaned Application Package PeopleCode.	Run the following SQL:
		
		`DELETE FROM PSPCMPROG`
		`WHERE OBJECTID1 = 104`
		`AND OBJECTID2 = 107`
		`AND OBJECTVALUE2 NOT IN`
		`(SELECT APPCLASSID`
		`FROM PSAPPCLASSDEFN P`
		`WHERE P.PACKAGEROOT =`
		`OBJECTVALUE1 AND`
		`P.APPCLASSID =`
		`OBJECTVALUE2)`

PEOPLECODE-8	There is orphaned Application Package PeopleCode.	Run the following SQL:
		
		`DELETE FROM PSPCMPROG`
		`WHERE OBJECTID1 = 104`
		`AND OBJECTID2 = 105`
		`AND OBJECTID3 = 107`
		`AND OBJECTVALUE3 NOT IN`
		`(SELECT APPCLASSID`
		`FROM PSAPPCLASSDEFN P`
		`WHERE P.PACKAGEROOT =`
		`OBJECTVALUE1 AND`
		`P.QUALIFYPATH =`
		`OBJECTVALUE2 AND`
		`P.APPCLASSID =`
		`OBJECTVALUE3)`

Query	*Description*	*Resolution*
PEOPLECODE-9	There is orphaned Application Package PeopleCode.	For Microsoft SQL Server, run the following SQL: `DELETE FROM PSPCMPROG WHERE OBJECTID1 = 104 AND OBJECTID2 = 105 AND OBJECTID3 = 106 AND OBJECTID4 = 107 AND OBJECTVALUE4 NOT IN (SELECT APPCLASSID FROM PSAPPCLASSDEFN F WHERE F.PACKAGEROOT = OBJECTVALUE1 AND F.QUALIFYPATH = RTRIM(OBJECTVALUE2)+ ':'+ RTRIM(OBJECTVALUE3) AND F.APPCLASSID = OBJECTVALUE4)`  For Informix, run the following SQL:  `DELETE FROM PSPCMPROG WHERE OBJECTID1 = 104 AND OBJECTID2 = 105 AND OBJECTID3 = 106 AND OBJECTID4 = 107 AND OBJECTVALUE4 NOT IN (SELECT APPCLASSID FROM PSAPPCLASSDEFN F WHERE F.PACKAGEROOT = OBJECTVALUE1 AND QUALIFYPATH = TRIM(P.OBJECTVALUE2) || ':' || TRIM(P.OBJECTVALUE3) AND F.APPCLASSID = OBJECTVALUE4)`  For all other DB platforms  `DELETE FROM PSPCMPROG WHERE OBJECTID1 = 104 AND OBJECTID2 = 105 AND OBJECTID3 = 106 AND OBJECTID4 = 107 AND OBJECTVALUE4 NOT IN (SELECT APPCLASSID FROM PSAPPCLASSDEFN F WHERE F.PACKAGEROOT = OBJECTVALUE1 AND QUALIFYPATH = RTRIM(P.OBJECTVALUE2) || ':' || RTRIM(P.OBJECTVALUE3) AND F.APPCLASSID = OBJECTVALUE4)`
PEOPLECODE-10	Lists PeopleCode programs that contain directives and that have not been compiled after a PeopleTools upgrade.	Compile the PeopleCode programs in Application Designer.

Pivot Grid Integrity

The following table describes the audit queries and resolutions for this area:

Query	*Description*	*Resolution*
PIVOT-01	Pivot Grid Model referencing a nonexistent data type.	This should generally not happen and is an upgrade issue. In such cases, use Pivot Grid Administration to delete the model.
PIVOT-02	Pivot Grid definitions not having a valid Pivot Grid Model.	This indicates a partially saved, unusable model. In such cases, use Pivot Grid Administration to delete the model.
PIVOT-03	Pivot Grid Definitions not having a default view.	This indicates a partially saved, unusable model. In such cases, use Pivot Grid Administration to delete the model.
PIVOT-04	Pivot Grid Definitions not having default grid options.	This indicates a partially saved, unusable model. In such cases, use Pivot Grid Administration to delete the model.
PIVOT-05	Pivot Grid Definitions not having default chart options.	This indicates a partially saved, unusable model. In such cases, use Pivot Grid Administration to delete the model.
PIVOT-06	Pivot Grid Model based on a nonexistent PSQuery.	This can happen if the PSQuery associated with the Pivot Grid Model is deleted. In such cases, use Pivot Grid Administration to delete the model.
PIVOT-07	PSQuery field name in PSQRYFIELD table does not match field defined in PSPGMODEL table.	This can happen if the field in PSQuery is modified. In such cases, use Pivot Grid Administration to delete the model.
PIVOT-08	PSQuery field headings in PSQRYFIELD table do not match column name defined in PSPGMODEL table.	This can happen if the field in PSQuery is modified. In such cases, using the Pivot Grid Wizard re-select all the relevant columns and save the Model again.
PIVOT-09	PSQuery prompt name in PSPGQRYPROMPT table for Pivot Grid does not match the bind field defined in the PSQRYBIND table.	This can happen if the PSQuery prompts are modified. In such cases, use Pivot Grid Administration to delete the model.
PIVOT-10	PSQuery bind name in PSQRYBIND does not match prompt defined in PSPGQRYPROMPT table for Pivot Grid.	This can happen if the PSQuery prompts are modified. In such cases, mark the Model as Invalid by unchecking the “Valid” flag for the Model. Then delete the model using the Delete_All_ InvalidModels.dms script.
PIVOT-11	PSQuery bind variable heading in PSQRYBIND does not match prompt name defined in PSPGQRYPROMPT table for Pivot Grid.	This can happen if the PSQuery prompts are modified. In such cases, use Pivot Grid Administration to delete the model.

Query	Description	Resolution
PIVOT-12	PSQuery bind variable prompt records in PSQRYBIND table do not match prompt record defined in PSPGQRYPROMPT table for Pivot Grid.	This can happen if the PSQuery prompts are modified. In such cases, use Pivot Grid Administration to delete the model.
PIVOT-13	Pagelets referencing non-existent pivot grid model.	Update the pagelet to reference a valid pivot grid model.
PIVOT-14	Pagelets referencing non-existent pivot grid view.	Update the pagelet to reference a valid pivot grid view.

See, "Deleting Pivot Grid Models" (PeopleTools 8.54: Pivot Grid) for more information on deleting Pivot Grid Models.

PSLOCK Version Integrity

The following table describes the audit queries 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.

Process Scheduler

The following table describes the audit queries 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:** The PRRSCHED-01 query 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)*.

Query	*Description*	*Resolution*
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 product documentation for PeopleSoft Process Scheduler.
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.
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.

Query	*Description*	*Resolution*
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.
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.
PRCSSCHED-18	There are queued processes in tables used by Process Scheduler (PSPRCSPARMS and PSPRCSRQST) containing a DBNAME different than the current database name.	This situation can occur when a database has been renamed. To resolve the database name, shut down the Process Scheduler server(s), and run the MGRPRCSTBL Application Engine program from the command line. See "Running Application Engine Programs" (PeopleTools 8.54: Application Engine).

Query	Description	Resolution
PRCSSCHED-19	Process definitions, where a process definition specifies a run time parameter record/field that does not exist.	For the listed processes, select PeopleTools, Process Scheduler, Processes, Runtime Parameters, and perform one of the following: • select the correct runtime parameter record/field. • remove the incorrect parameters.

Query Integrity

The following table describes the audit queries 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:
		`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	*Description*	*Resolution*
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-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 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-22	This audit identifies queries that 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	*Description*	*Resolution*

QUERY-23	This audit identifies queries that reference non-existent database records. The query definitions are corrupt.	This is an internal PeopleSoft audit. Contact My Oracle Support for resolution.
		
		Run the following SQL:
		
		`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	This audit identifies queries that were created with the name UNTITLED.	Queries should not be saved as UNTITLED. Either rename or delete these queries.
QUERY-25	This audit identifies queries that were created with blank query names.	You must either rename or delete these queries.
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.

QUERY-27	This audit identifies the queries that reference query translate fields that have been incorrectly modified to a non-translate type. When run in Microsoft Windows Query Manager, queries that reference these fields might return an inaccurate zero result set because the query metadata is corrupt.	Run the following SQL:
		
		`UPDATE PSQRYFIELD`
		`SET XLATTYPE = 1`
		`WHERE FIELDNAME IN`
		`(<list of fieldnames`
		`from the audit>)`

Record Integrity

The following table describes the audit queries 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';`

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 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 Application Designer to open the definition and fix the invalid fields.

RECORD-9	Invalid record definitions in record group definitions.	Run the following SQL:
		
		`DELETE FROM PS_REC_GROUP_`
		`REC`
		`WHERE RECNAME NOT IN`
		`(SELECT DISTINCT RECNAME`
		`FROM PSRECDEFN)`

Query	Description	Resolution
RECORD-11	Record definitions that contain more than two Long character field types.	**Note:** The use of LONG data types can introduce performance and storage overhead on database platforms. The judicious use of multiple LOB data types in a single record definition is recommended. Although PeopleTools does not prevent the use of multiple LONG fields in a single record definition for either DB2 zOS or DB2 LUW, assigning multiple LONGs to a single record definition is discouraged. For DB2 LUW, LONGs are implemented as varchar(n) / vargraphic(n) or Large Object (CLOB / DBCLOB / BLOB) data types depending on the field length and whether the data is in character or binary format. Assigning multiple LONGs to a single record definition can cause performance overhead, especially if the LONGs have been implemented as LOBs. For DB2 zOS, LONGs are implemented as LONG VARCHAR / VARGRAPHIC data types. DB2 determines the actual length of a LONG VARCHAR / VARGRAPHIC data type when the table is created by summing the internal lengths of all of the non-LONG VARCHAR / VARGRAPHIC columns in the table, and subtracting that summed length value from the page size (PeopleTools assigns tables that contain LONGs to a 32KB page size). DB2 then reserves the remainder of the row space in the table for the LONG VARCHAR / LONG VARGRAPHIC column. If there are multiple LONG VARCHAR / VARGRAPHIC columns in the table, each must equally divide this remaining space in the row. This is why PeopleTools always accomplishes alters to tables that contain LONGs through the Alter By Table Rename method rather than the Alter In Place method. At alter time, if new columns are subsequently added to the table, or if existing column definitions are changed such that the sum of the lengths of the non-LONG columns increases, the amount of row space for the LONG VARCHAR / LONG VARGRAPHIC columns is reduced. For this reason, please use the PeopleTools LONG data type judiciously for DB2 zOS and note

Query	*Description*	*Resolution*
		that placing multiple LONGs in a single record definition is highly discouraged. For the Oracle platform, the judicious use of multiple LOB data types in a single record definition is recommended. The use of multiple LONG fields in a single record definition is supported. Multiple LONG support is made possible with the adoption of the CLOB (Character Large Object) and BLOB (Binary Large Object) Oracle LOB data types. While technically possible, you should exercise caution when using multiple LOBs in a single record definition because of potential performance overhead which might occur based on the placement and actual size of the LOB.
RECORD-12	Record definitions with a blank or null record name value.	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 <> ' '`

RECORD-14	The Field Number field in the RecordField table has an invalid value.	If the reported record was delivered by PeopleSoft or generated as part of an upgrade, contact My Oracle Support. Run the following SQL to determine which fields need to be updated:
		
		`SELECT FIELDCOUNT FROM PSRECDEFN WHERE RECNAME='X';`
		
		`SELECT FIELDNUM FROM PSRECFIELD WHERE RECNAME='X' ORDER BY FIELDNUM;`
		
		Note: The PSRECFIELD.FIELDNUM values should be numbered sequentially 1 through PSRECDEFN.FIELDCOUNT. If any value is skipped then renumber the FIELDNUM values accordingly.

Related Language Integrity

The following table describes the audit queries 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)`
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)`

Query	*Description*	*Resolution*

SYSLANG-05	Foreign Language translate fields that are found in the XLATTABLE table without related base language translate fields.	Either delete the offending entries using SQL, or use Application Designer to add the equivalent entries in the base language of the database. To delete the offending entries using your platform query tool, first, perform a SELECT to verify the rows indicated in the report. Then, delete the selected rows.
		
		`SELECT A.LANGUAGE_CD, A. FIELDNAME, A.FIELDVALUE, A.EFFDT FROM PSXLATITEMLANG A WHERE NOT EXISTS (SELECT 'X' FROM PSXLATITEM B WHERE A.FIELDNAME = B. FIELDNAME AND A.FIELDVALUE = B. FIELDVALUE AND A.EFFDT = B.EFFDT);`
		
		`DELETE FROM PSXLATITEMLANG A WHERE NOT EXISTS (SELECT 'X' FROM PSXLATITEM B WHERE A.FIELDNAME = B. FIELDNAME AND A.FIELDVALUE = B. FIELDVALUE AND A.EFFDT = B.EFFDT);`

SYSLANG-07	Related Language records which are not valid records.	In Application Designer, delete the specified Related Language Records. Or, use your platform query tool to delete the related language reference for each record that is listed. First, perform a SELECT to verify the rows indicated in the report. Then, update the selected rows.
		
		`SELECT A.RECNAME, A. RELLANGRECNAME, A.OBJECTOWNERID FROM PSRECDEFN A WHERE A.RELLANGRECNAME <> ' ' AND A.RELLANGRECNAME NOT IN (SELECT B.RECNAME FROM PSRECDEFN B) ORDER BY A.RECNAME;`
		
		`UPDATE PSRECDEFN SET RELLANGRECNAME = ' ' WHERE RELLANGRECNAME <> ' ' AND RELLANGRECNAME NOT IN (SELECT B.RECNAME FROM PSRECDEFN B);`

SYSLANG-08	The displayed Related Language Records are effective-dated but do not have an EFFDT field defined.	In Application Designer, add EFFDT to the specified related language table.

Query	*Description*	*Resolution*

SYSLANG-09	The displayed Related Language records point to another Related Language record	In Application Designer, delete the related language reference for each record that is listed. Or, use your platform query tool to delete the related language reference for each record that is listed. First, perform a SELECT to verify the rows indicated in the report. Then, update the selected rows.
		
		`SELECT A.RECNAME, A. RELLANGRECNAME, B.RELLANGRECNAME, A. OBJECTOWNERID FROM PSRECDEFN A, PSRECDEFN B WHERE A.RELLANGRECNAME <> ' ' AND A.RELLANGRECNAME = B. RECNAME AND B.RELLANGRECNAME <>' ' ORDER BY A.RECNAME;`
		
		`UPDATE PSRECDEFN SET RELLANGRECNAME=' ' WHERE RECNAME IN (SELECT A.RELLANGRECNAME FROM PSRECDEFN A, PSRECDEFN B WHERE A.RELLANGRECNAME <> ' ' AND A.RELLANGRECNAME = B. RECNAME AND B.RELLANGRECNAME <>' ');`

SYSLANG-13	Identify related language records that have more than one base record defined.	In Application Designer, remove the related language table link to one of the base records.

SYSLANG-15	The displayed Related Language fields in the PSDBFLDLABLLANG table are orphaned.	Using the platform query tool, first, perform a SELECT to verify the rows indicated in the report. Then, delete the selected rows.
		
		This is sample SQL. Adjust accordingly for your platform:
		
		`SELECT A.FIELDNAME, A. LANGUAGE_CD FROM PSDBFLDLABLLANG A WHERE NOT EXISTS (SELECT 'X' FROM PSDBFIELD B WHERE A.FIELDNAME = B.FIELDNAME) ORDER BY 1,2`
		
		`DELETE FROM PSDBFLDLABLLANG A WHERE NOT EXISTS (SELECT 'X' FROM PSDBFIELD B WHERE A.FIELDNAME = B.FIELDNAME)`

Query	Description	Resolution

SYSLANG-16	Invalid language code found in PSOPRDEFN.	Using your SQL query tool, issue a SELECT to verify the rows indicated in the report. Update the affected rows as shown in the following sample SQL.
		
		`SELECT OPRID, LANGUAGE_CD`
		`FROM PSOPRDEFN`
		`WHERE OPRID = 'VP1'`
		
		`UPDATE PSOPRDEFN`
		`SET LANGUAGE_CD = 'ENG'`
		`WHERE OPRID = 'VP1'`

Security Integrity

The following table describes the audit queries 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'`

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'`
		
		When the SEC-3 exception does not list the permission list, check the CLASSID of the permission list that are named as a BLANK and delete them using the following script:.
		
		`SELECT DISTINCT A.CLASSID FROM PSAUTHPRCS A`
		`WHERE NOT EXISTS(SELECT 'X'FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A.CLASSID);`
		
		`Delete FROM PSAUTHPRCS A`
		`WHERE NOT EXISTS(SELECT 'X'FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A.CLASSID);`

Query	*Description*	*Resolution*

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:	If this is a Permission List that should exist, recreate it through PeopleTools Security. Otherwise, Delete the Permission List from the Role.
		
		`DELETE FROM PSROLECLASS`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A.CLASSID)`

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.

SEC-17	Primary Permission List specified for user does not exist in the Permission List Definition table: (Verifies the existence of Permission Lists Assigned to User Profiles Primary Permission List.)	If this is a Permission List that should exist, recreate it through PeopleTools Security. Otherwise, Delete the Primary Permission List value for the User Profiles via the following SQL:
		
		`UPDATE PSOPRDEFN A SET A.OPRCLASS = ' '`
		`WHERE A.OPRCLASS <> ' ' AND`
		`NOT EXISTS (SELECT 'X' FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A.OPRCLASS)`

SEC-18	User named in a User-Role relationship does not exist in the User Definition table: (Verifies the existence of Users owning Roles.)	If this is a User that should exist, recreate it through PeopleTools Security. Otherwise, Delete the User from the Role.
		
		`DELETE FROM PSROLEUSER`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSOPRDEFN B`
		`WHERE B.OPRID = A.ROLEUSER)`

SEC-19	User named in a User-Role relationship does not exist in the User Definition table: (Verifies the existence of Users owning Roles.)	If this is a Role that should exist, recreate it through PeopleTools Security. Otherwise, Delete the Role from the User.
		
		`DELETE FROM PSROLEUSER`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSROLEDEFN B`
		`WHERE B.ROLENAME = A.ROLENAME)`

Query	*Description*	*Resolution*

SEC-20	Role named in a Role-Permission List relationship does not exist in the Role Definition table: (Verifies existence of Roles owning Permission Lists.)	If this is a Role that should exist, recreate it through PeopleTools Security. Otherwise, Delete the Role From the Permission List.
		
		`DELETE FROM PSROLECLASS`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSROLEDEFN B`
		`WHERE B.ROLENAME = A.ROLENAME)`

SEC-24	Row Level Security Permission List specified for user does not exist in the Permission List Definition table:
	
	(Verifies the existence of Permission Lists Assigned to User Profiles Row Security Permission List.)
		If this is a permission list that should exist, recreate it through PeopleTools Security. Otherwise, Delete the Row Security Permission List value for the User Profiles via the following SQL:
		
		`UPDATE PSOPRDEFN A SET A.ROWSECCLASS = ' '`
		`WHERE A.ROWSECCLASS <> ' '`
		`AND NOT EXISTS(SELECT 'X' FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A.ROWSECCLASS)`

SEC-25	Process Profile Permission List specified for user does not exist in the Permission List Definition table: (Verifies the existence of Permission Lists Assigned to User Profiles Process Profile Permission List.)	If this is a permission list that should exist, recreate it through PeopleTools Security. Otherwise, delete the Process Profile Permission List value for the User Profiles via the following SQL:
		
		`UPDATE PSOPRDEFN A SET A.PRCSPRFCLS = ' '`
		`WHERE A.PRCSPRFCLS <> ' ' AND NOT EXISTS`
		`(SELECT 'X' FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A.PRCSPRFCLS)`

SEC-26	Navigator Homepage Permission List specified for user does not exist in the Permission List Definition table: (Verifies the existence of Permission Lists Assigned to User Profiles Navigator Homepage Permission List.)	If this is a permission list that should exist, recreate it through PeopleTools Security. Otherwise, Delete the Navigator Homepage Permission List value for the User Profiles via the following SQL:
		
		`UPDATE PSOPRDEFN A SET A.DEFAULTNAVHP = ' '`
		`WHERE A.DEFAULTNAVHP <> ' ' AND NOT EXISTS`
		`(SELECT 'X' FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A. DEFAULTNAVHP)`

SEC-27	Access Profile specified for user does not exist in the Access Profile table:	If this is an Access Profile that should exist, recreate it through Application Designer/Data Mover and or your DBA. Otherwise, manually change the value to a valid Access Profile or blank via Security. It is not advised to blank out the Access Profile unless you intend to disable the User Profile. If so, you can delete the Access Profile value for the User Profiles via the following SQL:
		
		`DELETE FROM PS_SCRTY_SRCHGRP A`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A.CLASSID)`

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 NOT (PSAUTHITEM.MENUNAME = 'ADS_DESIGNER'`
		`AND PSAUTHITEM.BARNAME = 'TOOLS_DATASETS')`
		`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`
		`)`
		`)`
		`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`

Query	Description	Resolution
		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)))))
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)

Query	Description	Resolution
		) 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);
SEC-30	Missing users in the PS_ ROLEXLATOPR table.	Every User that is defined in the PSOPRDEFN table should have a corresponding Role User entry in the PS_ROLEXLATOPR table.
SEC-31	Verify that the user definition table PSOPRDEFN has an entry corresponding to each user assigned to a role.	The role users returned by the audit do not have corresponding user IDs in the PSOPRDEFN table. That is, the user ID's don't exist. These role users should be removed from the PS_ROLEXLATOPR table. Run the following SQL: DELETE FROM PS_ROLEXLATOPR A WHERE NOT EXISTS (SELECT 'X' FROM PSOPRDEFN B WHERE B.OPRID = A.OPRID)
SEC-32	Verify that no inactive roles exist in the PSROLEDEFN table.	The roles returned by the audit need to be fixed by either deleting them or making them active. To remove a role, use the following SQL: DELETE FROM PSROLEDEFN WHERE ROLESTATUS <> 'A' Or, use the Delete Roles page. Select PeopleTools, Security, Permissions & Roles, Delete Roles. To activate inactive roles, use the following SQL: UPDATE PSROLEDEFN SET ROLESTATUS = 'A' WHERE ROLESTATUS <>'A'

Query	Description	Resolution

SEC-34	Incomplete permission list: orphan service operation.	Use the Integration Broker interface to open the service operation listed in the audit. On the Web Service Access page, remove any invalid permission lists.
		
		Or, submit the following SQL:
		
		`DELETE FROM PSAUTHWS`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSCLASSDEFN WHERE PSAUTHWS.CLASSID =`
		`PSCLASSDEFN.CLASSID);`

SEC-35	Incomplete Permission List: Orphan Search Groups: (Verifies the existence of Permission Lists owning Search Groups.)	If this is a Permission List that should exist, recreate it through PeopleTools Security. Otherwise, Delete the Permission List value for a Search Group via the following SQL:
		
		`DELETE FROM PS_SCRTY_SRCHGRP A`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A.CLASSID)`

SEC-36	Permission List references a non-existent Search Group: (Verifies the existence of Search Groups Assigned to Permission Lists.)	Use the Integration Broker interface to open the service operation listed in the audit. On the Web Service Access page, remove any invalid permission lists. Or, submit the following SQL:
		
		`DELETE FROM PS_SCRTY_SRCHGRP A`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSPTSF_SRCCAT B`
		`WHERE B.PTSF_ISGBLSRCH = 'Y'`
		`AND B.PTSF_SRCCAT_NAME = A.PTSF_SRCHGRP_NAME)`

SQL Integrity

The following table describes the audit queries 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)`

Query	*Description*	*Resolution*
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 My Oracle Support 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 My Oracle Support to make sure that it is not a symptom of a larger problem, such as a corrupted AE application.
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.

Query	Description	Resolution

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)`

Style Sheet Integrity

The following table describes the audit queries and resolutions for this area:

Query	Description	Resolution

STYLESHEET-1	Orphaned free form style sheet data.	Issue the following SQL to remove the orphaned rows:
		
		`DELETE * FROM PSCONTDEFN`
		`WHERE CONTTYPE = 9 AND`
		`CONTNAME = %1;`
		
		`DELETE * FROM PSCONTENT`
		`WHERE CONTTYPE = 9 AND`
		`CONTNAME = %1;`
		
		`DELETE * FROM`
		`PSCONTDEFNLANG`
		`WHERE CONTTYPE = 9 AND`
		`CONTNAME = %1;`
		
		`DELETE * FROM PSCONTENTLANG`
		`WHERE CONTTYPE = 9 AND`
		`CONTNAME = %1;`

STYLESHEET-2	Orphaned free form style sheet definitions.	Use Application Designer to delete the free form style sheet(s).

Query	Description	Resolution
STYLESHEET-3	Parent style sheet not found.	Open the affected style sheet(s) in Application Designer. Remove the reference to the nonexistent parent in the Parent Style Sheet dropdown on the Style Sheet Properties dialog box.
STYLESHEET-04	Sub Style Sheet not found.	Use Application Designer to delete references to any missing sub style sheets.

Tree Integrity

The following table describes the audit queries 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.
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.

Query	*Description*	*Resolution*
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.

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 = 'tree_`
		`name'`
		` AND SETID = 'setid'`
		` AND EFFDT = 'effdt';`
		
		`UPDATE PSTREEDEFN`
		`SET LEVEL_COUNT = $count`
		`WHERE TREE_NAME = 'tree_`
		`name'`
		` AND SETID = 'setid'`
		` AND EFFDT = 'effdt';`

Query	Description	Resolution

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 = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt' AND TREE_BRANCH = 'tree_branch_name';`

		For branched trees, use following SQL for each branch in the tree.
		
		Note: For trees with branches, the UPDATE also uses a different TABLE NAME and there are two UPDATE statements.
		
		`SELECT COUNT(*) FROM PSTREENODE WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt' AND TREE_BRANCH = 'tree_branch_name';`
		
		`UPDATE PSTREEBRANCH SET NODE_COUNT = $count WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt' AND TREE_BRANCH = 'tree_branch_name'`
		
		`UPDATE PSTREEDEFN SET NODE_COUNT = 0, WHERE TREE_NAME = 'tree_name' AND SETID = 'setid' AND EFFDT = 'effdt' AND TREE_BRANCH = 'tree_branch_name'`

Query	*Description*	*Resolution*

		For trees without branches, do not include the "TREE_BRANCH=" lines. For example:
		
		`SELECT COUNT(*)`
		`FROM PSTREENODE`
		`WHERE TREE_NAME = 'tree_name'`
		`AND SETID = 'setid'`
		`AND EFFDT = 'effdt'`
		
		`SELECT COUNT(*)`
		`FROM PSTREENODE`
		`WHERE TREE_NAME = 'tree_name'`
		`AND SETID = 'setid'`
		`AND EFFDT = 'effdt'`
		
		`UPDATE PSTREEDEFN`
		`SET NODE_COUNT = 0,`
		`WHERE TREE_NAME = 'tree_name'`
		`AND SETID = 'setid'`
		`AND EFFDT = 'effdt'`

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'`
		`AND TREE_BRANCH =`
		`'tree_branch_name';`
		
		`UPDATE PSTREEDEFN`
		`SET LEAF_COUNT = $count`
		`WHERE TREE_NAME = 'tree_name'`
		`AND SETID = 'setid'`
		`AND EFFDT = 'effdt'`
		`AND 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.

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);`

Query	*Description*	*Resolution*
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.	Run this SQL: `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" topic in the PeopleSoft Tree Manager documentation 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" topic in the PeopleSoft Tree Manager documentation 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, run the following SQL:

```
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, run the following SQL (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, run the following SQL (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'

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), run the following SQL:

```
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 describes the audit queries and resolutions for this area:

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

Search Integrity

The following table describes the audit queries and resolutions for the PeopleSoft Search Framework:

Query	*Description*	*Resolution*
PTSF-01	Search definition mapped in a search category is missing.	If the definitions are delivered by Oracle and not modified, contact My Oracle Support. If the definitions are customized or the issue appears after a project migration or upgrade, please verify the related definitions are included in the project. Also check, People Tools, Search Framework, Designer, Search Definition
PTSF-02	There is no search category created for a search definition.	If the definitions are delivered by Oracle and not modified, contact My Oracle Support. If the definitions are customized or the issue appears after a project migration or upgrade, please verify the related definitions are included in the project. Also check, PeopleTools, Search Framework, Designer, Search Categories

Query	*Description*	*Resolution*
PTSF-03	Search category does not contain a search definition with the same name as search category	This report is provided for informational purposes. It is not necessary for a category to have a definition with the same name. However the reverse case is a mandate. A search category can contain any number of search definitions which are deployed by its own categories.
PTSF-04	Source name in search definition is not available in Query or Connected Query definitions.	If the definitions are delivered by Oracle and not modified, contact My Oracle Support. If the definitions are customized or the issue appears after a project migration or upgrade, please verify the related definitions are included in the project. Also check Reporting Tools, Query, Query Manager
PTSF-05	Delete query mapped to a search definition is missing in query definition	If the definitions are delivered by Oracle and not modified, contact My Oracle Support. If the definitions are customized or the issue appears after a project migration or upgrade, please verify the related definitions are included in the project. Also check Reporting Tools, Query, Query Manager
PTSF-06	Pre/Post processing AE mapped to a search definition is missing in Application Engine library	If the definitions are delivered by Oracle and not modified, contact My Oracle Support. If the definitions are customized or the issue appears after a project migration or upgrade, please verify the related definitions are included in the project.
PTSF-07	Security Application Class specified in a search definition is missing in Application Class library.	If the definitions are delivered by Oracle and not modified, contact My Oracle Support. If the definitions are customized or the issue appears after a project migration or upgrade, please verify the related definitions are included in the project.
PTSF-08	Indexed field in a search definition does not exist in the query fields list	If the definitions are delivered by Oracle and not modified, contact My Oracle Support. If the definitions are customized or the issue appears after a project migration or upgrade, please verify the related definitions are included in the project.

Query	*Description*	*Resolution*
PTSF-09	Attribute selected in a search definition is missing in the search attribute list	If the definitions are delivered by Oracle and not modified, contact My Oracle Support. If the definitions are customized or the issue appears after a project migration or upgrade, please verify the related definitions are included in the project.
PTSF-10	Selected attribute display name does not exists in the field label definition	If the definitions are delivered by Oracle and not modified, contact My Oracle Support. If the definitions are customized or the issue appears after a project migration or upgrade, please verify the related definitions are included in the project.
PTSF-11	Attribute in the search attribute list is not referenced by any search definition	Use your SQL Editor to delete the unused attributes from record PSPTSF_ATTRS.

Related Content Integrity

The following table describes the audit queries and resolutions for the PeopleSoft Related Content:

Query	*Description*	*Resolution*
RELCONTENT-01	This audit verifies that the PSVERSION table contains the entries for the Related Content managed objects.	Run the VERSION Application Engine program to reset the PSVERSION and PSLOCK entry.
RELCONTENT-02	This audit verifies that the PSLOCK table contains the entries for the Related Content managed objects.	Run the VERSION Application Engine program to reset the PSVERSION and PSLOCK entry.
RELCONTENT-03	This audit lists all the missing service definitions required by the service configuration.	Use the Application Designer to import the missing Related Content Definitions into the target database.
RELCONTENT-04	This audit lists all the missing menu entries required by service configuration.	Sign on to PeopleTools using the browser, and make a change on the Manage Service Configuration page add missing menu metadata.

RELCONTENT-05	This audit lists all mapped fields that are not part of any service configuration.	Delete orphaned mapped fields with this SQL:
		
		`DELETE FROM PSPTCS_MAPFLDS A`
		`WHERE A.PTCS_SERVICEID NOT IN`
		`(SELECT B.PTCS_SERVICEID`
		`FROM PSPTCSSRVCONF B)`

Query	*Description*	*Resolution*

RELCONTENT-06	This audit lists all the root folders that are not part of any menu.	Delete orphan folders:
		
		`DELETE FROM PSPTCS_MNUFLDRS A`
		`WHERE A.PTCS_FOLDERID = A. PTCS_PRNT_FOLDERID`
		`AND NOT EXISTS`
		`(SELECT B.PTCS_FOLDERID FROM PSPTCS_MNULINKS B`
		`WHERE A.PORTAL_NAME = B. PORTAL_NAME`
		`AND A.PORTAL_OBJNAME = B. PORTAL_OBJNAME`
		`AND A.PTCS_MENUID = B.PTCS_ MENUID`
		`AND A.PTCS_FOLDERID = B. PTCS_FOLDERID)`

RELCONTENT-07	This audit lists all the menus with no folders.	Delete empty menus using this SQL:
		
		`DELETE FROM PSPTCS_MENU_TBL A`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSPTCS_ MNUFLDRS B`
		`WHERE A.PORTAL_NAME = B. PORTAL_NAME`
		`AND A.PORTAL_OBJNAME = B. PORTAL_OBJNAME`
		`AND A.PTCS_MENUID = B.PTCS_ MENUID)`

RELCONTENT-08	This audit lists all the missing application packages of service definitions.	Re-import the missing application packages using Application Designer.

RELCONTENT-09	This audit lists all the orphaned links.	Delete the orphaned links using this SQL:
		
		`DELETE FROM PSPTCS_MNULINKS A`
		`WHERE A.PTCS_FOLDERID NOT IN`
		`(SELECT B.PTCS_FOLDERID`
		`FROM PSPTCS_MNUFLDRS B`
		`WHERE A.PORTAL_NAME = B. PORTAL_NAME`
		`AND A.PORTAL_OBJNAME = B. PORTAL_OBJNAME`
		`AND A.PTCS_MENUID = B.PTCS_ MENUID)`

RELCONTENT-10	This audit lists all the orphaned folders.	Delete folders using this SQL:
		
		`DELETE FROM PSPTCS_MNUFLDRS A`
		`WHERE A.PTCS_MENUID NOT IN`
		`(SELECT B.PTCS_MENUID`
		`FROM PSPTCS_MENU_TBL B`
		`WHERE A.PORTAL_NAME = B. PORTAL_NAME`
		`AND A.PORTAL_OBJNAME = B. PORTAL_OBJNAME)`

Query	*Description*	*Resolution*
RELCONTENT-11	This audit lists all menu entries pointing to nonexistent service configurations.	Delete menu entries, using this SQL: `DELETE FROM PSPTCS_MNULINKS A WHERE NOT EXISTS (SELECT 'X' FROM PSPTCSSRVCONF B WHERE A.PORTAL_NAME = B. PORTAL_NAME AND A.PORTAL_OBJNAME = B. PORTAL_OBJNAME AND A.PTCS_SERVICEID = B. PTCS_SERVICEID AND A.PTCS_INSTANCEID = B. PTCS_INSTANCEID)`
RELCONTENT-12	This audit lists missing application package PeopleCode for service definitions.	Use the Application Designer to re-import the missing application packages.
RELCONTENT-13	This audit lists all the menu links that are referring to service definitions that are not present in the database. This scenario occurs: • If service definitions are deleted but not the services using these service definitions. • While importing from a database project the service definition entries are missed out. In this case, importing the missing service definitions will resolve the issue.	Use the RCF_SERVICE_DEFINITIONS application data set to move a new service definition which is not included in a project. Delete the orphaned menu links and folders using the following SQL: `DELETE FROM PSPTCSSRVCONF A WHERE A.PORTAL_NAME = <PORTAL_NAME> AND A.PTCS_SERVICEID = <SERVICE _DEFN_ID> DELETE FROM PSPTCS_MNULINKS A WHERE A.PORTAL_NAME = <PORTAL_NAME> AND A.PTCS_SERVICEID = <SERVICE_DEFN_ID>`
RELCONTENT-14	This audit lists the menu links and folders which have no association to any menu layout because the related content framework objects were not deleted and exist with inconsistencies in the database. The audit may also lists menu links and folders if the corresponding menu layouts are not copied to the project in the database.	Move the related Menu Layouts using the RCF_MENU_LAYOUTS application data sets. These Menu links do not appear on the Contextual menu. Delete the menu links and folders which have no association with any menu layout, using the following SQL: `DELETE FROM PSPTCS_MNULINKS A WHERE A.PORTAL_NAME = <PORTAL_NAME> AND A.PORTAL_OBJNAME = <PORTAL_OBJNAME> AND A.PTCS_MENUID = <MENUID>`

Query	*Description*	*Resolution*
RELCONTENT-15	This audit lists all the Related Content Services that are not attached to any Content Reference. This can occur if the parent content reference is deleted from the Structure and Content page or a new content reference is created but not copied to the target database.	Find and delete orphaned related content data using the following SQL: `SELECT * FROM PSPTCS_SRVCFG AWHERE A.PORTAL_NAME <> '_PTCS_PTPG' AND NOT EXISTS (SELECT 'X' FROM PSPRSMDEFNWHEREPORTAL_REFTYPE ='C' ANDPORTAL_NAME = A.PORTAL_NAME ANDPORTAL_OBJNAME = A.PORTAL_OBJNAME)` If a new content reference is not copied to the target database then move the content reference using correct mechanism.

Chapter 5

Employing Database Level Auditing

Understanding Database Level Auditing

PeopleSoft provides trigger-based auditing functionality as an alternative to the record-based auditing that 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 EnableDBMonitoring domain parameter set in PSADMIN to retrieve that information.

Note: If you implement trigger-based auditing, be aware that there is an unavoidable amount of additional overhead associated with auditing, which can effect the system's overall performance. EnableDBMonitoring isn't supported for Informix.

The elements 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 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 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 in Application Designer and build the 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 all edit and key attributes from the newly saved record. Add to the top of the audit record the following 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 for the Microsoft SQL Server requirement of not including ntext, text columns in the trigger syntax, as well as Oracle's requirement to exclude the LONG data type 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).

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,
   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.

Working With Auditing Triggers

This section discusses how to:

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

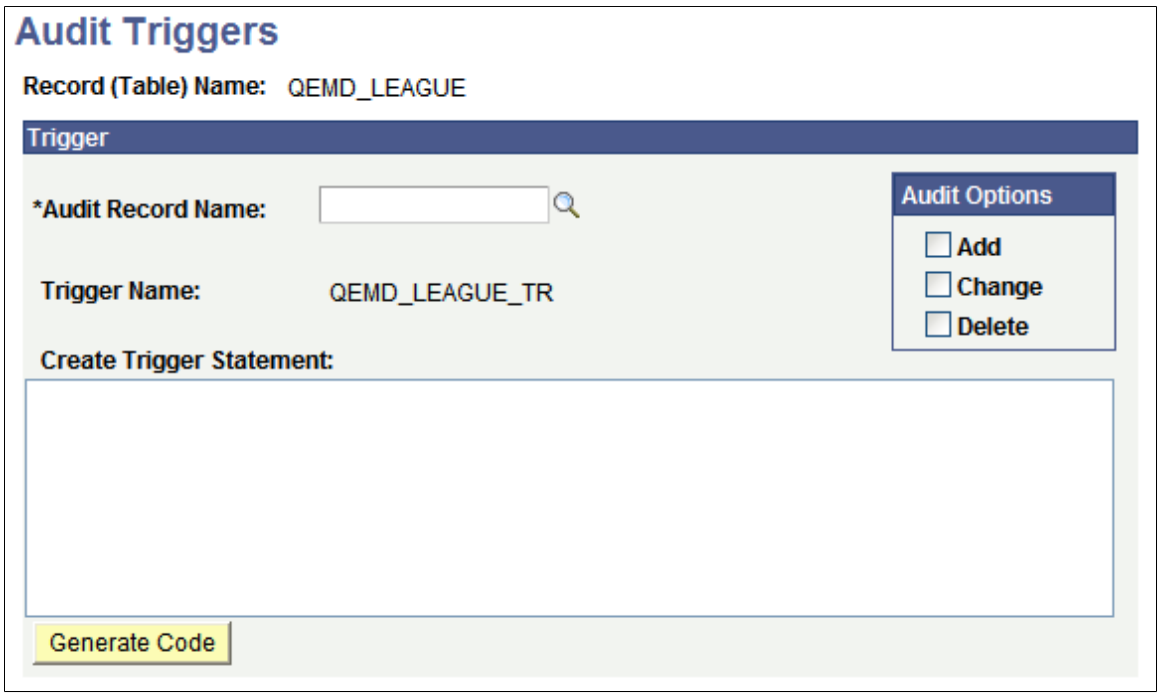
A trigger is a database level object that the system initiates based on a specified event occurring on a table. Most RDBMS platforms support a form of database triggers.

Defining Auditing Triggers

Access the Audit Triggers page PeopleTools, Utilities, Audit, Update Database Level Auditing.

Image: Audit Triggers page

This example illustrates the fields and controls on the Audit Triggers page.



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. You must use separate audit records for each record to be audited when using triggers to audit multiple records

Important! The DDL for these database audit triggers can only be properly created using the Audit Triggers page. Do not use any other means to create the DDL for these triggers.

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 create the triggers.

Access the Run Audtrgs page (PeopleTools, Utilities, Audit, Perform Database Level Audit).

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, Audit, Perform Database Level Audit.

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_D⇒
T
----------- -------------------------- ---------- ----------- ------------ -------⇒
------------------
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 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 Query Access Manager that contains all audit records. This makes it easier to access the audit records under PeopleSoft Query:

Image: Query Access Manager

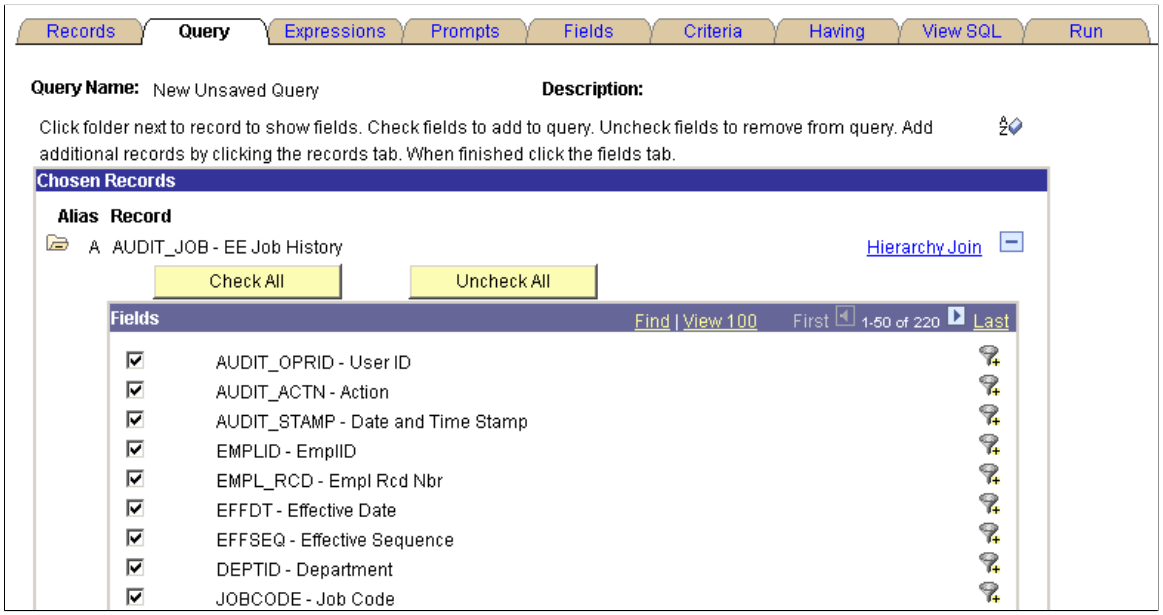
This example illustrates the fields and controls on the Query Access Manager.

Listing All Audit Records in PS_AUDIT_JOB

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

Image: Query page

This example illustrates the fields and controls on the Query page.

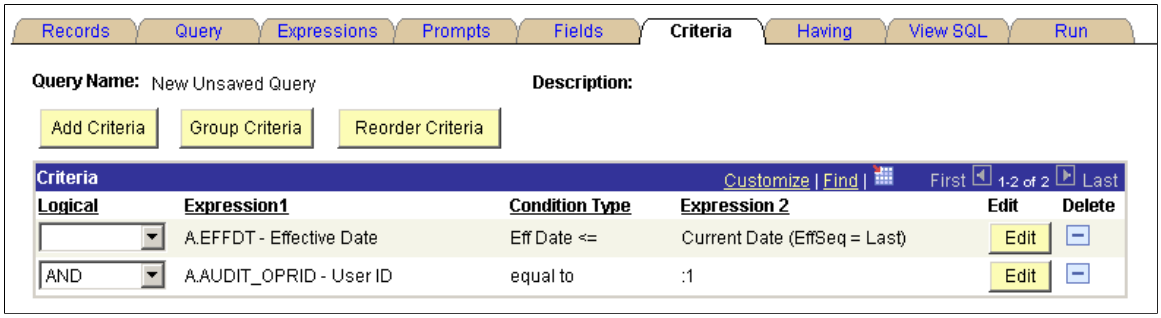


Listing All Audit Records for a Specified User ID

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

Image: Criteria page

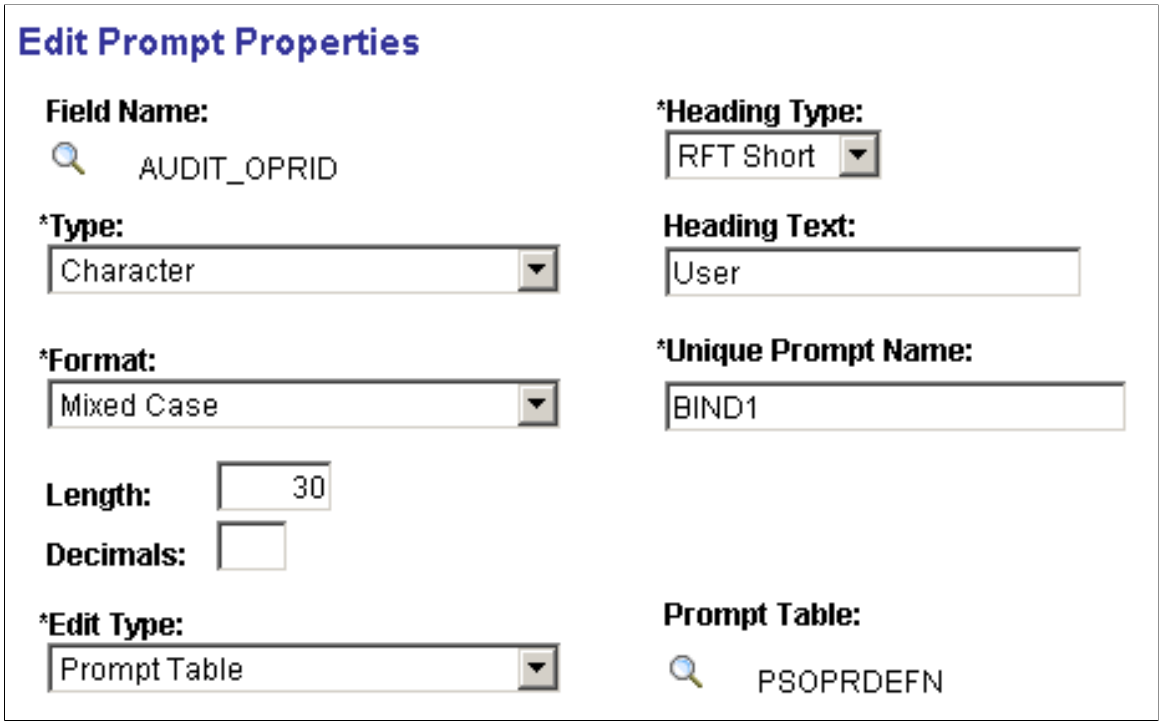
This example illustrates the fields and controls on the Criteria page.



The example shows the prompt for properties the AUDIT_OPRID field:

Image: Edit Prompt Properties page

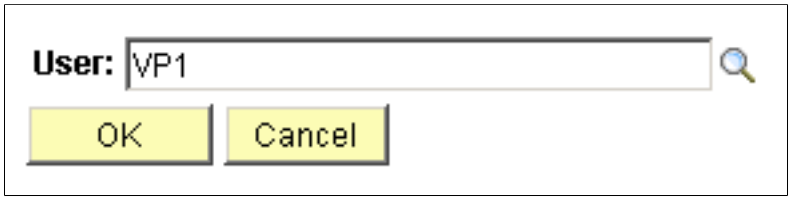
This example illustrates the fields and controls on the Edit Prompt Properties page.



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:*

Image: Prompt field

This example illustrates a prompt for User ID against the PSOPRDEFN table.

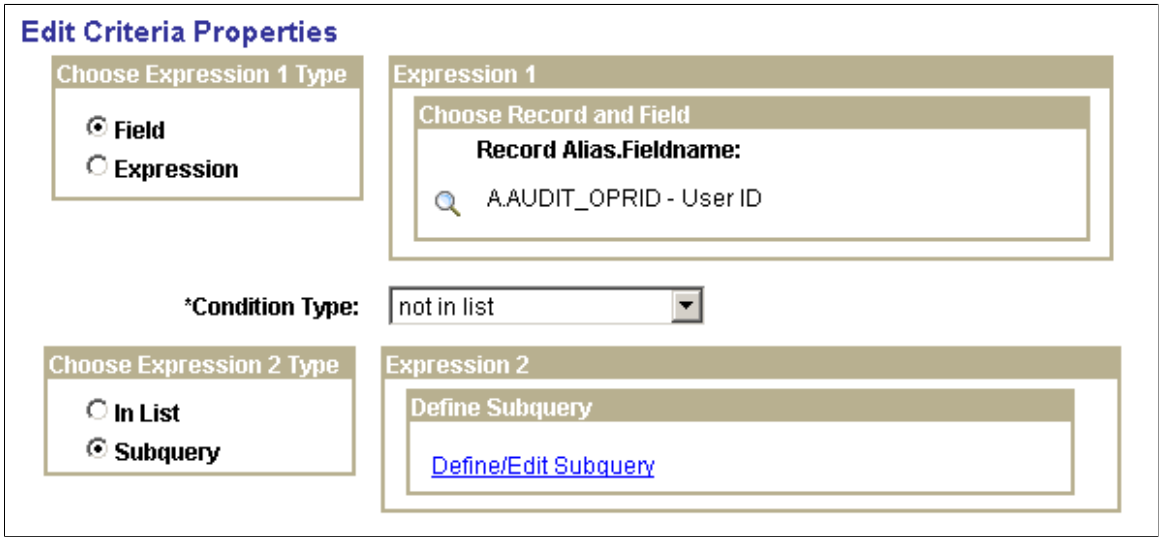


Listing All Audit Records Containing an Invalid OPRID

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

Image: Edit Criteria Properties page

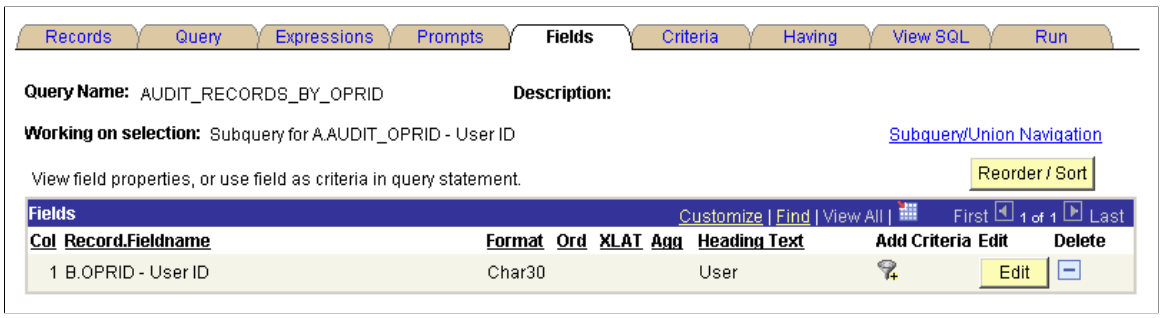
This example illustrates the fields and controls on the Edit Criteria Properties page.



Click the Define/Edit Subquery link and select the OPRID field:

Image: Fields page

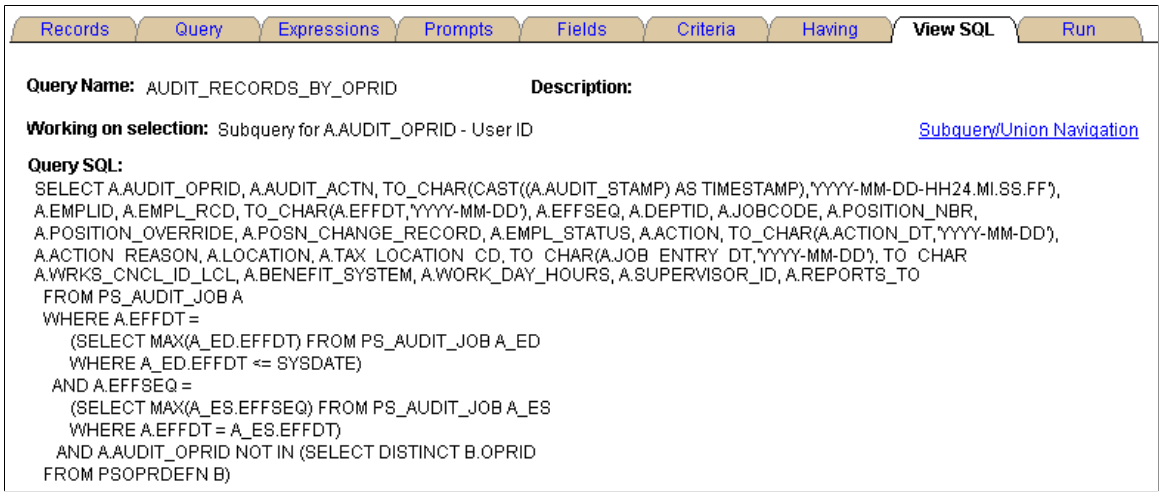
This example illustrates the fields and controls on the Fields page.



The subquery selects distinct User ID from PSOPRDEFN.

Image: View SQL page

This example shows the SQL for the query.

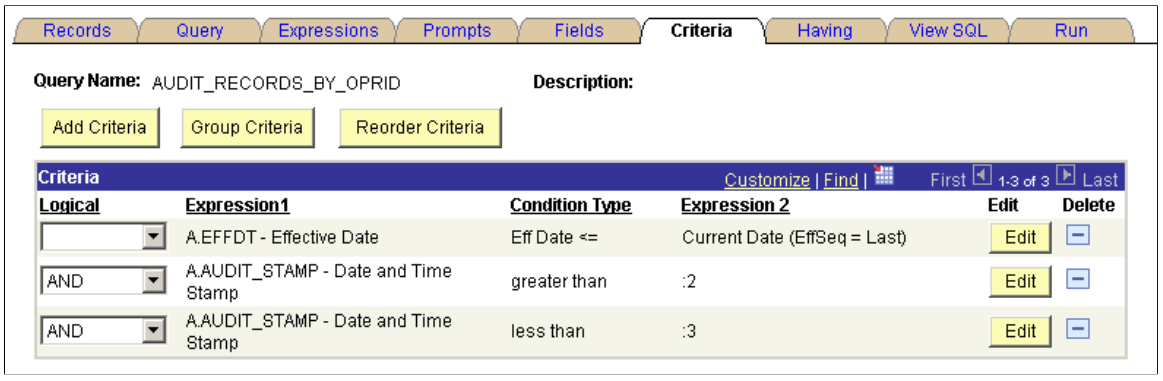


Listing All Audit Records for a Specified Time Period

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

Image: Criteria page

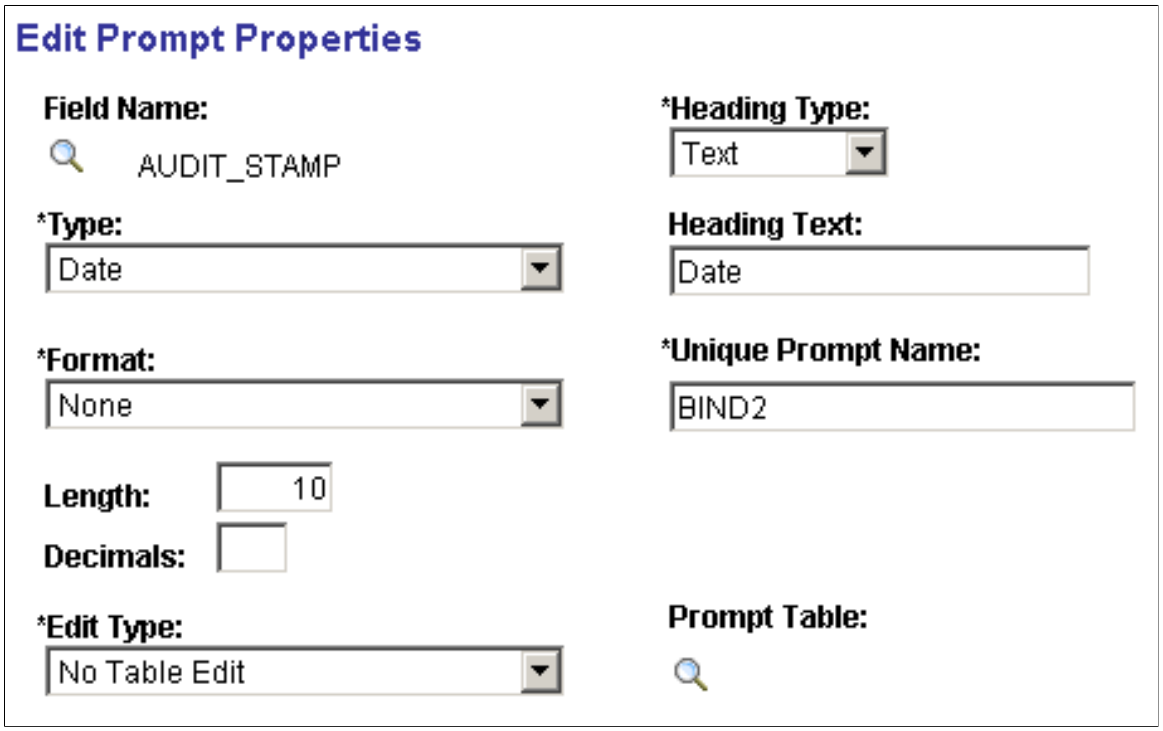
This example illustrates the fields and controls on the Criteria page.



Set the prompt properties to follow this example.

Image: Edit Prompt Properties page

This example illustrates the fields and controls on the Edit Prompt Properties page.



Change the AUDIT_STAMP field type to *Date* to enable the user to take advantage of the calendar control as a prompt mechanism.

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

[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 Microsoft SQL Server, you will have to alter the trigger scripts that are generated manually. 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

[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 BWHERE 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 BWHERE 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 ...
```

This alters the definition of a trigger that is created previously by the CREATE TRIGGER statement.

See the full definition in SQL Server Books Online.

See SQL Server Books Online for the full example.

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 re-enabled.

- 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 z/OS Trigger Information

This section provides an overview of DB2 z/OS trigger information and discusses:

- 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, and commands, involved with DB2 z/OS triggers.

Before the Trigger Audit can be implemented on DB2 z/OS, the trigger statement must be defined.

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. The following SQL is a sample of the trigger syntax.

```
--For INSERT operation:
CREATE TRIGGER PSO112 AFTER  INSERT ON PS_ABSENCE_HIST
 REFERENCING NEW AS C_ROW
 FOR EACH ROW MODE DB2SQL
 INSERT INTO PS_AUDIT_ABSENCE
 VALUES (COALESCE(NULLIF(CURRENT CLIENT_USERID,''),USER),
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);

--For DELETE operation
CREATE TRIGGER PSP112 AFTER DELETE ON PS_ABSENCE_HIST
```



```
 REFERENCING OLD AS C_ROW
 FOR EACH ROW MODE DB2SQL
 INSERT INTO PS_AUDIT_ABSENCE
 VALUES (COALESCE(NULLIF(CURRENT CLIENT_USERID,''),USER),
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);

--For UPDATE operation
CREATE TRIGGER PSQ112 AFTER UPDATE ON PS_ABSENCE_HIST
 REFERENCING NEW AS C_ROW
 FOR EACH ROW  MODE DB2SQL
 INSERT INTO PS_AUDIT_ABSENCE
 VALUES (COALESCE(NULLIF(CURRENT CLIENT_USERID,''),USER),
CURRENT TIMESTAMP,'N',
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 PSR112 AFTER UPDATE ON PS_ABSENCE_HIST
 REFERENCING OLD AS C_ROW
 FOR EACH ROW MODE DB2SQL
 INSERT INTO PS_AUDIT_ABSENCE
 VALUES (COALESCE(NULLIF(CURRENT CLIENT_USERID,''),USER),
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 might 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

You can't alter a trigger. You must drop it and recreate it.

See *DB2 for z/OS SQL Reference.*

See *DB2 for z/OS Application Programming and SQL Guide.*

Using DB2 UDB for Linux, Unix, and Windows (LUW) Trigger Information

This section discusses :

- DB2 LUW trigger syntax.
- DB2 LUW trigger maintenance.

DB2 LUW 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 following SQL is a sample of the trigger syntax:

```
CREATE TRIGGER PSO1
  AFTER  INSERT ON PS_ABSENCE_HIST
  REFERENCING NEW AS C_ROW
  FOR EACH ROW MODE DB2SQL
  INSERT INTO PS_AUDIT_ABSENCE_HIST
  VALUES (COALESCE(NULLIF(CURRENT CLIENT_USERID,''),USER),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 PSP1 AFTER DELETE ON PS_ABSENCE_HIST
  REFERENCING OLD AS C_ROW
  FOR EACH ROW MODE DB2SQL
  INSERT INTO PS_AUDIT_ABSENCE_HIST
  VALUES (COALESCE(NULLIF(CURRENT CLIENT_USERID,''),USER),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 PSQ1
  AFTER UPDATE ON PS_ABSENCE_HIST
  REFERENCING NEW AS C_ROW
  FOR EACH ROW  MODE DB2SQL
  INSERT INTO PS_AUDIT_ABSENCE_HIST
  VALUES (COALESCE(NULLIF(CURRENT CLIENT_USERID,''),USER),CURRENT TIMESTAMP,'N',
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 PSR1 AFTER UPDATE ON PS_ABSENCE_HIST
  REFERENCING OLD AS C_ROW
  FOR EACH ROW MODE DB2SQL
  INSERT INTO PS_AUDIT_ABSENCE_HIST
  VALUES (COALESCE(NULLIF(CURRENT CLIENT_USERID,''),USER),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 LUW Trigger Maintenance

These commands might be useful for administering triggers.

List All Triggers in a Database

To list all triggers:

```
SELECT trigname, trigevent, tabname FROM syscat.triggers
```

List the Trigger Definition

To list the trigger definition:

```
SELECT trigname, text FROM syscat.triggers
```

To Remove a Trigger

To remove a trigger:

```
DROP trigger TRIGGERNAME
```

See *DB2 Universal Database for Linux, Unix, and Windows SQL Reference.*

See *DB2 Universal Database for Linux, Unix, and Windows 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
```

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=trigger_name;
```

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=trigger_name;
```

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

Working With The Diagnostics Framework

Understanding Diagnostics Framework

This section discusses:

- What is the Diagnostics Framework?
- Diagnostics Framework benefits.
- Diagnostics Framework architecture.

What Is the Diagnostics Framework?

The Diagnostics Framework is a set of delivered classes and plug-ins that enable you to capture detailed information for troubleshooting PeopleSoft application issues.

A diagnostic plug-in is an application package, which is developed following the diagnostic plug-in standard. An application package is a container for application classes or other application packages. Developers use the Application Designer to create application packages that are treated as diagnostic plug-ins by the framework.

PeopleTools delivers Diagnostics Framework base classes in an application package called PT_DIAGNOSTICS, which consists of a collection of application classes as well methods which can be used to develop application packages for the Diagnostics Framework. After implementing and registering the application package within the Diagnostics Framework, the application package becomes a diagnostic plug-in.

Diagnostics Framework Benefits

The Diagnostics Framework enables you to define and retrieve application data diagnostics from a PeopleSoft database within the PeopleSoft Pure Internet Architecture (PIA) environment. With this diagnostic information, you can:

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

Using Diagnostics Framework, you can perform diagnostic tests on your system with minimal instructions from the Oracle support. These tests answer application-specific questions to help development and user support teams diagnose and troubleshoot 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 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.

Note: Diagnostics Framework is not designed to be a reporting tool, such as Query or Crystal Reports. Diagnostics 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.

Diagnostics Framework Architecture

Diagnostics 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.

Note: Oracle support might give you additional plug-ins to diagnose specific problems. These plug-ins are implemented differently from the plug-ins that you develop.

Diagnostics Framework is installed automatically when you install PeopleTools. Use standard PeopleSoft security administration to grant access to the user interface.

See Setting Up Security for Diagnostics Framework.

Application Classes

By definition, each application class is responsible for asking one diagnostic question. Each class has a method that is called by Diagnostics Framework. This method, in turn, gathers the information and calls methods in extended base classes that return the information to Diagnostics Framework

Diagnostic Plug-ins

Application packages that are used in the Diagnostics Framework are referred to as diagnostic application packages. Each is 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, Diagnostics Framework executes programs within these plug-ins and then returns the information from those programs in an HTML or XML document. These plug-ins can supply a consistent method of gathering relevant diagnostic information from your system.

These are the categories of diagnostic plug-ins:

Diagnostic Plug-In Type	*Description*
Delivered	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	Post-release diagnostic plug-ins that Oracle support might send to you for specific diagnostic purposes. You import these plug-ins to Diagnostics Framework using Application Designer.
Custom	Custom diagnostic plug-ins that you develop. You must register custom plug-ins before you can use them. **Note:** If you want to register existing plug-ins from other Oracle Enterprise PeopleSoft releases, you must ensure that they're defined according to the guidelines used for custom plug-ins in the current release.

All registered plug-ins appear in the Diagnostics 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. An Oracle support contact may ask you to run a particular plug-in, depending on the problem that you are reporting.

Related Links

"Understanding Application Classes" (PeopleTools 8.54: PeopleCode API Reference)
Understanding Diagnostic Plug-In Development

Setting Up Security for Diagnostics Framework

This section provides an overview of security for Diagnostics Framework and discusses how to:

- Grant access to the Diagnostics Framework pages.
- Grant access to the WEBLIB_PTDIAG web library.

Understanding Security for Diagnostics Framework

To gather information using Diagnostics Framework, you must have access to:

- The Diagnostics Framework pages.
- The WEBLIB_PTDIAG web library.

You use the PeopleTools Security pages to select or create a permission list to which you can add the necessary permissions for these elements. That permission list should ultimately be assigned through a role to the users who will run diagnostics.

Related Links

Security Administration

Granting Access to the Diagnostics Framework Pages

To run diagnostics, you must have access to these Diagnostics Framework pages:

- PT_DIAG_PLUGIN
- PT_DIAG_FRAME_REG

To set up security access to these pages:

1. Select PeopleTools, Security, Permission & Roles, Permission Lists.
2. Select or define a permission list to which you want to add Diagnostics Framework permissions.

   See "Understanding Permission Lists" (PeopleTools 8.54: Security Administration).
3. Access the Pages page for the selected permission list.

   This page lists the menus to which this permission list has some degree of access.
4. If the PT_DIAGNOSTICS menu isn't listed on this page, add a new row and select it.
5. Click the Edit Components link for the PT_DIAGNOSTICS menu.

   The Component Permissions page appears.
6. Click the Edit Pages link for the PT_DIAG_LAUNCH component.

   The Page Permissions page appears.
7. Click Select All to grant full access to the PT_DIAG_PLUGIN page, then click OK to return to the Component Permissions page.
8. Click the Edit Pages link for the PT_DIAG_FRAME_REG component.

   The Page Permissions page appears.
9. Click Select All to grant full access to the PT_DIAG_FRAME_REG page, then click OK to return to the Component Permissions page.
10. Click OK, then save the permission list.

Granting Access to the WEBLIB_PTDIAG Web Library

To set up security access to WEBLIB_PTDIAG:

1. Access the Web Libraries page for the permission list for which you've already granted access to the Diagnostics Framework pages.

This page lists the web libraries to which the permission list has some degree of access.

2. If the WEBLIB_PTDIAG web library isn't listed on this page, add a new row and select it.
3. Click the Edit link for the WEBLIB_PTDIAG web library.

   The Weblib Permissions page appears.
4. Click Full Access (All) to grant full access to the WEBLIB_PTDIAG web library.
5. Click OK, then save the permission list.

Running Diagnostics

This section discusses how to:

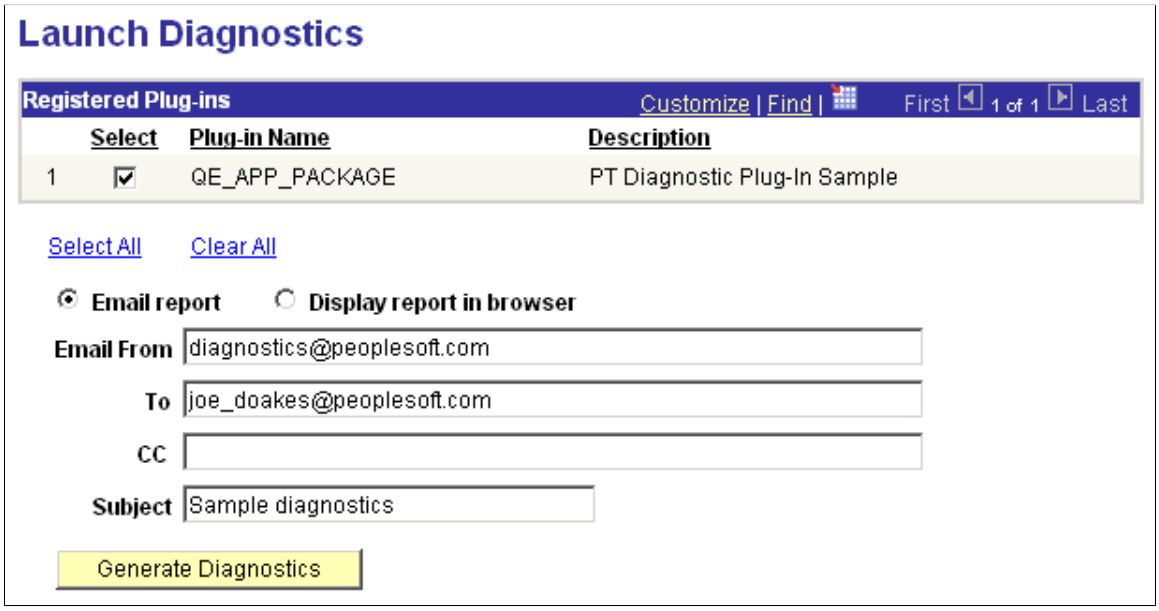
- Launch diagnostic plug-ins.
- Provide additional information.
- Obtain diagnostic results.

Launching Diagnostic Plug-Ins

Access the Launch Diagnostics page (Application Diagnostics, Launch Diagnostics).

Image: Launch Diagnostics page

This example illustrates the fields and controls on the 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.

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 "SMTP Settings" (PeopleTools 8.54: System and Server Administration).

Display report in browser Select to display the generated HTML diagnostic report in a new browser window.

Generate Diagnostics Click to launch the selected diagnostics, and either display or email the resulting report.

Note: You cannot select both Display report in browser and Email report simultaneously.

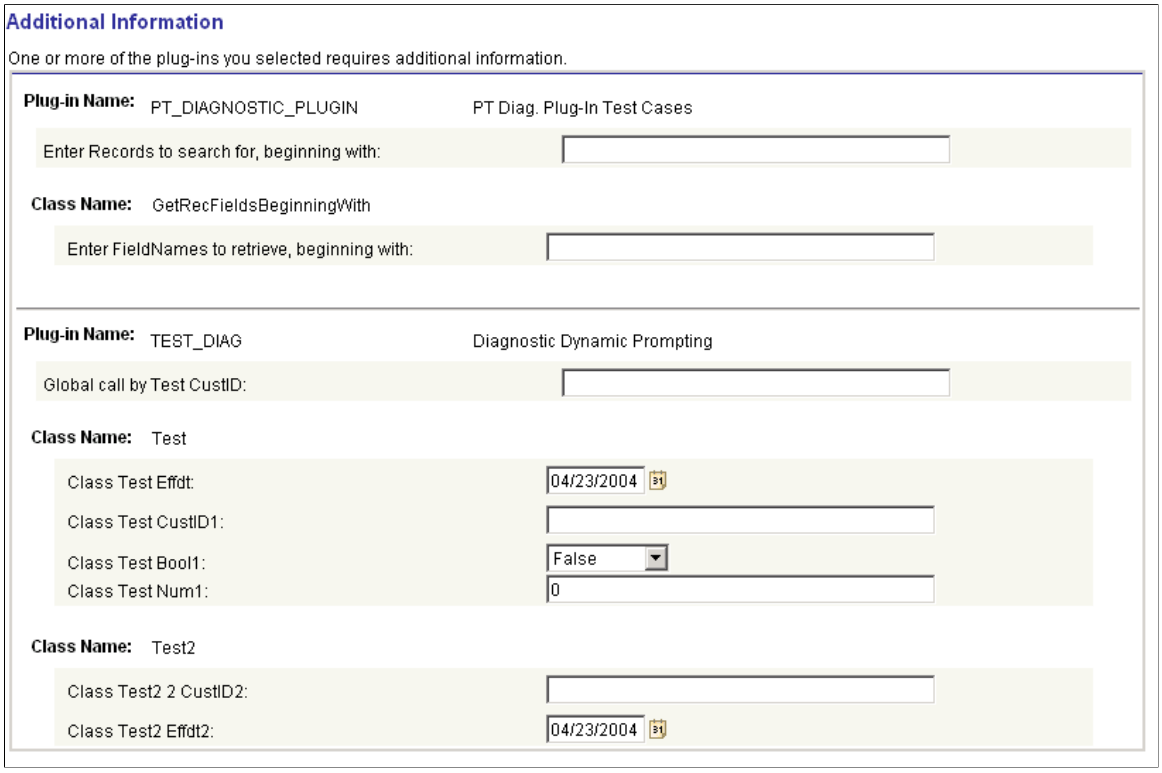
Providing Additional Information For Diagnostic Plug-ins

Access the Additional Information page (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 prompt you dynamically for relevant parameters. The Additional Information page enables you to enter the required parameters.

Image: Additional Information page

This example illustrates the fields and controls on the 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.

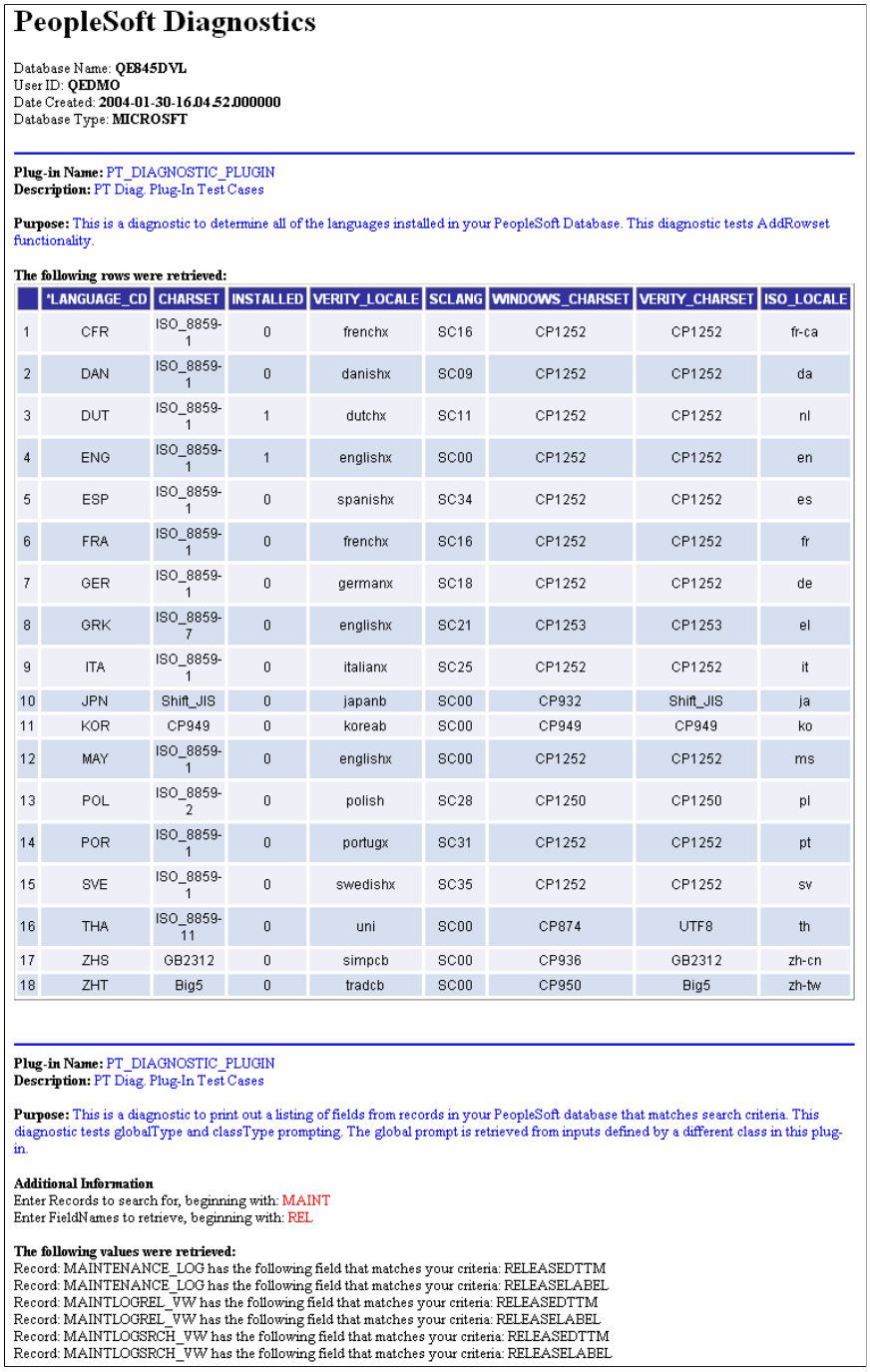
Obtaining Diagnostic Results

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.

Image: PeopleSoft Diagnostics page in HTML format

This example illustrates 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

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 sent (email) 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 Oracle Support team may provide additional plug-ins. You import these plug-ins into your database using the Copy Project from File option in 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 Oracle Support or move a file between databases, use the Copy Project to File option in Application Designer.

See "Copying Projects" (PeopleTools 8.54: Application Designer Lifecycle Management Guide).

Plug-ins that you import are registered automatically and become available immediately on the Launch Diagnostics page.

Chapter 7

Developing Diagnostic Plug-Ins

Understanding Diagnostic Plug-In Development

A diagnostic plug-in is an application package developed while adhering to the diagnostic plug-in standard. Developers use the application packages editor in Application Designer to create the application packages that are treated as diagnostic plug-ins by the framework.

PeopleTools delivers Diagnostics Framework base classes in an application package called PT_DIAGNOSTICS. To create your own diagnostic plug-in, a new application package needs to be created and extended from the application package PT_DIAGNOSTICS. The new application package can have multiple numbers of application classes which should be extended from the base classes in PTDiagnostics, delivered in the PT_DIAGNOSTICS application package.

The sub classes can call the base class methods to collect the diagnostic information and return the same information to the Diagnostics Framework. Each of the application classes within the diagnostic plug-in focuses one diagnostic area and can return different information, depending on the state of the application and the nature of the 'question'.

The application class also can contain an optional public method, called GetDynamicPrompt, to prompt users for additional information.

The following types of data can be retrieved and displayed in the Diagnostics Framework:

- String
- Date
- Number
- Boolean
- Rowset

Note: You can also define your own private methods within the application class, which you can call only within the class.

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:

- must be instantiated only by Diagnostics Framework. They can't be called from any other location, including other PeopleCode programs.
- must contain three mandatory methods that are recognized and used by Diagnostics Framework.

Note: Developing custom diagnostic plug-ins requires a working knowledge of PeopleCode and application classes.

Related Links

"Understanding Application Classes" (PeopleTools 8.54: PeopleCode API Reference)

Developing Diagnostic Plug-Ins

This section discusses how to:

- Create the diagnostic application package.
- Create the diagnostic application classes.
- Implement the diagnostic PeopleCode.
- Register the diagnostic plug-in.
- Share diagnostic plug-ins

Note: Except for registering the Diagnostic Plug-in, which is performed using a PIA page, you complete all of these development steps using Application Designer.

Creating the Diagnostic Application Package

To create a diagnostic application package: Open the application package in Application Designer, Save the package.

1. In Application Designer select File, New, Application Package.
2. Save and name the package.

See "Understanding Application Classes" (PeopleTools 8.54: PeopleCode API Reference).

Creating the Diagnostic Application Classes

In the application package you've created, create a new application class, and save the class.

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.

See "Understanding Application Classes" (PeopleTools 8.54: PeopleCode API Reference).

Implementing the Diagnostic PeopleCode

To implement the diagnostic PeopleCode:

1. Open the PeopleCode editor (View, PeopleCode).
2. Import the PTDiagnostic package, by entering:

```
import PT_DIAGNOSTICS:*;
```


3. This imports all the classes in the PT_DIAGNOSTICS application package.
4. Define the class, using the same procedure as your would for any PeopleCode program. For example,

   ```
   class <class name> extends PTDiagnostic:PTDiagnostic
   ```

   This makes the class a diagnostic application class.

   The class name mentioned in the PeopleCode should be the same as the class name which is under the application package.

   All the classes defined in the diagnostic plug-in application package should be extended from the class PTDiagnostic:PTDiagnostic.
5. Define the following mandatory methods:

Method	***Description***

`Method <class name>`	This is the constructor of the class.
	
	Inside the method it is mandatory to have`%Super = create PT_DIAGNOSTICS:PTDiagnostics()`.
	
	More description of `%Super` resides in the PTDiagnostics application class.
	
	If you want to display the rowset in the browser, then you have to set the `hasRowset` property to True, otherwise make it False. For example,
	
	`&status = %Super.SetProperty (%This, "hasRowset", "Boolean", False);`
	
	If you want to call additional information from the user during the execution and use it as the search criteria, then set the `Where` property to true, otherwise make it False. For example,
	
	`&status = %Super.SetProperty (%This, "Where", "Boolean", True);`

`Method GetDiagnosticInfo`	This is the method that gets called when you launch the diagnostic plug-in. To display any output in the browser you have to call `%Super.Insertdata(Data type, <String to name the display>, <variable name>)`.

Method	*Description*
`Method IsPlugIn`	The purpose of this method is to identify the application package as a diagnostic plug-in. If this method is not present, then the system does not recognize it as a diagnostic plug-in during the registration process. This method should appear at the end of the diagnostic application class, and it should be an empty method. The definition of this should be : `Method IsPlugIn end-method;`

Registering the Diagnostic Plug-In

Before a diagnostics application package can be used, you have to register it using the Register Diagnostics page. Once the registration is complete, the application package becomes a diagnostic plug-in.

To register a diagnostic plug-in:

1. Access the Register Diagnostics page by selecting Application Diagnostics, Register Diagnostics.
2. Add a row, if needed.
3. In the Plug-In Name edit box, enter the name of the application package, or use the lookup prompt to select it.

   Note: When using the lookup prompt, the system displays *all* application packages not only diagnostic plug-in packages. When developing custom diagnostic plug-ins, using a naming convention can be helpful to refine the search.

4. Click Save.

Note: If you have not defined the IsPlugIn method or if it is not at the end of the class, an error message appears. The system only registers application packages containing all the required elements of a diagnostic plug-in.

Sharing Diagnostic Definitions

Once the registration is complete, the application package can be selected as a *Diagnostic Plug-In* in Application Designer. Inserting diagnostic plug-ins into projects, enables them to be shared, copied to or compared between databases, and exported to and imported from files. Sharing diagnostic plug-ins would be necessary for plug-ins that need to be sent to Oracle support staff for their review, for example, or to other developers at your site.

The project should contain the:

- Diagnostic plug-in
- application package

- application package PeopleCode

To insert a plug-in into a project:

1. In Application Designer, select Insert, Definitions into Project.
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.

Once the diagnostic plug-in definitions have been inserted into a project, you can share a diagnostic plug-in. Check the Upgrade tab to see these definition types: Application Packages, Diagnostic Plug-Ins, and PeopleCode. You cannot see these with the Development tab selected.

To share a diagnostic plug-in:

1. Open the project containing the diagnostic definitions in Application Designer.
2. Select Tools, Copy Project, to File.
3. Share the generated XML file with the interested parties.

Related Links

"Understanding Projects" (PeopleTools 8.54: Application Designer Developer's Guide)

Working With The Delivered PT_DIAGNOSTIC Application Package

PeopleTools delivers the PT_DIAGNOSTIC application package as part of the Diagnostics Framework. This package is the base package and its classes are base classes for the diagnostic plug-in.

To define a new plug-in, create a new application package, containing one or more application classes that imports the PT_DIAGNOSTIC application package.

When working with the PT_DIAGNOSTIC application package, make sure you understand:

- PTDiagnostics Application Class
- PTDiagnostics Class Methods
- PTDiagnostics Class Properties

These elements are described in the following sections.

PTDiagnostics Application Class

The PTDiagnostics application class is part of the PT_DIAGNOSTICS application package. It establishes the basic framework for developing the diagnostic plug-ins. The PTDiagnostics application class contains methods and properties that you can extend to develop your diagnostic plug-ins. 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 into your program, using an import statement. The application package PT_DIAGNOSTICS contains the PTDiagnostics class. The import statement should be 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.

In the constructor of the application class which extends the PTDiagnostics class you need to instantiate the PTDiagnostics class. The extended application classes collect the information whenever required and pass it to the super class, which is the PTDiagnostics class. You instantiate this class as:

```
%Super = create PT_DIAGNOSTICS:PTDiagnostics()
```

PTDiagnostics Class Methods

This section discusses the diagnostic methods for the PTDiagnostics PeopleCode class. The methods are listed in alphabetical order.

GetDiagnosticInfo

Description

Use this required public method to define the code that retrieves diagnostic information and returns it to Diagnostics Framework for presentation to the user. This method is invoked by Diagnostics 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 Diagnostics 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.

Considerations for GetDiagnosticInfo include:

- 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.
- 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.

Example

Following is an example of GetDiagnosticInfo that passes rowset data as output:

```
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;
```

GetDynamicPrompt

Description

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, Diagnostics Framework ignores the GetDynamicPrompt method. Within the GetDynamicPrompt method, use the base class InsertQuestion method to define the questions used to prompt the users.

Example

```
method GetDynamicPrompt
   Local boolean &status;

   &status = %Super.InsertQuestion("Recs", "Enter Records to search for, beginning
    with: ", "String", True);
end-method;
```

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.

InsertData

Syntax

InsertData(*propFormat*, *propDescr*, *&data*)

Description

The InsertData method passes data to Diagnostics 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 Diagnostics 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.

InsertQuestion

Syntax

```
InsertQuestion(sKeyID, sQuestion, sType,  GblBool)
```

Description

The InsertQuestion method passes a question to Diagnostics 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.

Parameter	*Description*
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.

IsPlugIn

Description

This required public method is invoked by Diagnostics Framework when you register the plug-in. It verifies that the class is part of a diagnostic plug-in. If it is not present, the system will not register the plug-in.

IsPlugIn should be:

- at the end of the Diagnostic Application Class.
- an empty method.

Example

```
method IsPlugIn
end-method
```

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 Diagnostics 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 Diagnostics 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.

Related Links

InsertData
SetProperty

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.

Related Links

SetProperty

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.

Related Links

SetProperty
GetDynamicPrompt

Diagnostic Plug-In Examples

The following are examples of typical actions found in diagnostic plug-ins. Examples include:

- Rowset-Based Output.
- String-based output.
- Number-based output.
- Prompting for global information input.
- Prompting for global and class-level information input.
- Joining two records.
- Handling constructor failure.
- Handling InsertData method failure.
- Handling dynamic prompting failure.

Example: Rowset-Based Output

The following example demonstrates how to retrieve record output and display it. In this case, the plug-in retrieves the list of languages from 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;
```

Example: String-Based Output

This example demonstrates how to retrieve string-based output and display it. In this case, the plug-in retrieves the 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;
```

Example: Number-Based Output

This example demonstrates how to retrieve a Number type output and display it. In this case, we retrieve the number of rows from the PSRECDEFN based on different conditions.

```
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;
```

Example: Prompting for Global Information Input

This example demonstrates how to use global prompting. This example includes the use of these constructs:

- GetDynamicPrompt (): This function prompts for the input.
- InsertQuestion (): Inserts a question.
- GetUserInputByKey (): Gathers the input.

In this example, the plug-in retrieves the number of rows from the PSRECDEFN based on different records. The search record is usually retrieved from the user during runtime.

```
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;
```

Example: Prompting for Global and Class-Level Information Input

The following example demonstrates the use of global and class-level prompts.

```
import PT_DIAGNOSTICS:*;

class GetRecFieldsBeginningWith extends PT_DIAGNOSTICS:PTDiagnostics
   /* Constructor */
   method GetRecFieldsBeginningWith();
```



```
   /* Public Method */
   method GetDiagnosticInfo();
   method GetDynamicPrompt();
   method IsPlugIn();
private

end-class;

method GetRecFieldsBeginningWith;
   Local boolean &status;
   %Super = create PT_DIAGNOSTICS: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 the global prompt*/
   &status = %Super.InsertQuestion("Recs", "Enter RecordNames to retrieve,
 beginning with:", "String", True);

   /* 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;
```

If you only need to define the class level prompt, use only the second InsertQuestion() method in the GetDynamicPrompt() method and one GetUserInputByKey().

If the message is not required during the dynamic input, then pass an empty string in the second parameter of the InsertQuestion() method.

Example: Joining Two Records

This example demonstrates the join between two records. This join can be done by creating a view also.

In this example, the plug-in retrieves the objects from the PSLOCK and PSVERSION tables where versions of the objects don't match.

```
import PT_DIAGNOSTICS:*;

class MatchVersions extends PT_DIAGNOSTICS:PTDiagnostics
   /* Constructor */
   method MatchVersions();

   /* Public Method */
   method GetDiagnosticInfo();
   method IsPlugIn();

private

end-class;

method MatchVersions;
   Local boolean &status = False;
   %Super = create PT_DIAGNOSTICS:PTDiagnostics();
   &status = %Super.SetProperty(%This, "Purpose", "String", "This is to retrieve
 the objects whose versions doesnot match in PSLOCK and PSVERSIONS.");
end-method;

method GetDiagnosticInfo
   Local boolean &status;
   Local Rowset &rs1;
   Local Rowset &rs2;
   Local integer &i, &j;
   Local Row &ro1;

   &rs1 = CreateRowset(Record.PSVERSION);
   &rs1.Fill();

   &rs2 = CreateRowset(Record.PSLOCK);
   &rs2.Fill();

   For &i = 1 To &rs1.RowCount
      For &j = 1 To &rs2.RowCount
         If (&rs1.GetRow(&i).GetRecord(Record.PSVERSION).OBJECTTYPENAME.Value =
 &rs2.GetRow(&j).GetRecord(Record.PSLOCK).OBJECTTYPENAME.Value) And
             (&rs1.GetRow(&i).GetRecord(Record.PSVERSION).VERSION.Value <>
 &rs2.GetRow(&j).GetRecord(Record.PSLOCK).VERSION.Value) Then
           &status = %Super.InsertData("String", "OBJECTTYPENAME: ", &rs1.GetRow
(&i).GetRecord(Record.PSVERSION).OBJECTTYPENAME.Value | " PSVERSION.VERSION: "
 | &rs1.GetRow(&i).GetRecord(Record.PSVERSION).VERSION.Value | " PSLOCK.VERSION: "
 | &rs2.GetRow(&j).GetRecord(Record.PSLOCK).VERSION.Value);
         End-If;
      End-For;
   End-For;


end-method;

method IsPlugIn
end-method;
```

Example: 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;
```

Example: 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;
```

Example: 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

Server Options

This section discusses:

- Delivered configuration.
- Access ID.
- Service Packs and QFE.

Delivered Configuration

The PeopleSoft server configuration parameters are initially set to Microsoft SQL Server defaults. It's a good practice to review the parameters and modify them to your site requirements if necessary. Use the file *PS_HOME*\scripts\spconfig.sql on your database server to keep track of your changes. This file is used by the database configuration wizard when installing a PeopleSoft database.

Note: Don’t use "priority boost" when running additional applications like PeopleSoft Process Scheduler on your database server machine.

Access ID

The user ID used as an ACCESSID is not required to be a member of the SQL Server "sysadmin" server role. This restricts the activities of this user ID, which enhances overall application security.

The PeopleSoft ACCESSID is a member of the following fixed database roles:

- db_datareader
- db_datawriter
- db_ddladmin

Additionally, it is necessary to grant ALTER TRACE permissions to the ACCCESSID to take full advantage of the tracing capabilities available in PeopleTools.

Note: Keep in mind that utilizing these roles for the PeopleSoft ACCESSID login, restricts the ability to run administrative tasks not specific to PeopleSoft applications, such as creating backups and restoring them, defining new server logins, modifying server settings, creating and dropping databases, and so on.

Service Packs and Quick Fix Enhancements (QFE)

PeopleSoft always runs certifications on the latest SQL Server service packs as they become available. Service packs contain large number of improvements and have been tested extensively by Microsoft.

A QFE is a fix intended to solve a specific problem that's usually documented in a Microsoft Knowledge Base (KB) article. PeopleSoft doesn't run certification tests for any particular SQL Server QFE, but considers them to be supported when they're recommended by Microsoft to solve specific problems. However, to install a QFE, PeopleSoft recommends appropriate testing before applying it to a production environment. It's important to take into consideration that a QFE is an enhancement targeted to solve a specific problem. “Secondary effects” as a result of its installation can be determined only with proper testing.

PeopleSoft does not distribute SQL Server QFE software; please contact Microsoft to determine if a QFE is required, and for instructions on how 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 using Query Analyzer:

```
EXEC sp_dboption databasename, 'ansi null default', true
```

Working with Functional Indexes

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 limit, which is a maximum of 16 key columns for an index. What makes a functional index special is that it's required only when the number of key columns exceeds the SQL Server limit.

PeopleSoft implements the functional index by creating an index over a computed column. The computed column MSSCONCATCOL is the sum of all the key columns required to keep uniqueness.

In order to create indexes on computed columns, SQL Server 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.

Note: Both *Quoted Identifier* and *Arithabort* are explicitly set during installation automatically by the Database Configuration Wizard or when running the script, createdb_2005.sql, at the database installation.

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. This collation was selected for being compatible with the binary sort order used on previous versions of SQL Server.

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.

Consult your PeopleTools installation guide and the application installation manual for further details on the collation configuration required for your database server.

For environments running English-only databases and languages covered by the Latin1 character set (such as Western European languages), PeopleSoft recommends the collation delivered as default in the PeopleSoft installation scripts. The database collation is set when running the creatdb_2005.sql script at installation time. The script runs automatically when you use the database configuration wizard. It is a requirement to run the script when installing the database manually.

See the product documentation for *PeopleTools Installation for Microsoft SQL Server*.

Implementing Transparent Data Encryption

This section provides an overview and discusses how to enable Transparent Data Encryption (TDE).

Important! PeopleTools has not introduced any functionality for TDE with respect to Microsoft SQL Server; PeopleTools only supports the use of it. You should always refer to your Microsoft Documentation for any issues with respect to TDE for Microsoft SQL Server.

Understanding Transparent Data Encryption

PeopleTools supports the use of Transparent Data Encryption (TDE) if you are running your database on Microsoft SQL Server 2008 (or higher). TDE provides enhanced encryption and decryption of both data files and log files through the use of database encryption files (DEK). This enables your organization to comply with numerous privacy laws, regulations, and guidelines that are required in certain industries. When implementing TDE for Microsoft SQL Server, you can apply the following AES or 3DES encryption algorithms without making any changes to your existing applications.

- AES_128
- AES_192

- AES_256
- TRIPLE_DES_3KEY

When specifying the desired encryption algorithm, make sure to enter it exactly as it appears in the list above.

While there will always be some overhead associated with any encryption processing, the performance impact introduced with TDE is minimal.

Important! Make sure you have read and fully understand all of the Microsoft documentation related to this feature before you implement it. This PeopleTools documentation outlines PeopleTools-specific items and is not intended to replace any existing Microsoft documentation. For example, make sure you are aware of the usage recommendations and restrictions described in the Microsoft documentation as they apply also to your PeopleSoft application databases.

Enabling Transparent Data Encryption

To enable TDE:

1. Create a master key.
2. Create or obtain a certificate protected by the master key.
3. Create a database encryption key and protect it by the certificate.
4. Set the database to use encryption.

The following example illustrates encrypting and decrypting the TDEPT85X database using a certificate installed on the server named PeopleToolsEncryptCert.

```
USE master;
GO
/* Create Master Key Using a strong password. */
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<UseStrongPasswordHere>';
go
/* Create a PeopleTools Encryption Certificate. *?/
CREATE CERTIFICATE PeopleToolsEncryptCert
WITH SUBJECT = 'PeopleTools Encrypt Certificate';
go
USE TDEPT85X;
GO
/* Create Database Encryption Key Using PeopleTools Encryption Certificate. */
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE PeopleToolsEncryptCert;
GO
/* Enable Database Encryption. */
ALTER DATABASE TDEPT85X
SET ENCRYPTION ON;
GO
```

Note: When implementing TDE, *all* files and filegroups in the database are encrypted. If any filegroups in a database are marked *read only*, the database encryption operation will fail.

Microsoft SQL Server Feature Considerations

This section discusses:

- Recovery model.
- Nested triggers.
- Auto create statistics and auto update statistics.
- Automatic file growth.
- Autoshrink.
- Read Committed Snapshot Isolation.
- File management.
- Tempdb.
- Trace flags.
- Database monitoring.

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.

Nested Triggers

Some PeopleSoft applications take advantage of database triggers. Make sure that the *nested triggers* option is enabled for the database server hosting the PeopleSoft databases. You can use *sp_dboption* or Enterprise Manager to enable this option on the server.

Auto Create Statistics and Auto Update Statistics

Microsoft SQL Server enables you to create statistics and update them automatically. It's recommended that you leave this feature enabled for PeopleSoft applications.

However, sometimes you should disable these features for a particular table. For example, if you want to modify the sample size used to create the statistics, you need to do so manually.

Another example is when the data varies considerably, and the statistics that are created are not accurate. For this you might want to disable auto create statistics and auto update statistics manually, and adjust the statistics as needed.

In general, auto create statistics and auto update statistics should be enabled for most of the tables in your database unless you need to disable the feature for specific reasons.

Automatic File Growth

Microsoft SQL Server enables you to let a database file grow automatically when it's full. PeopleSoft recommends that you leave this feature enabled, however, it should be used with caution. When the database server is in the process of increasing the size of a data file, all other activities in the server stop, which can cause server performance problems. Ideally, in a well-tuned environment this won't occur — properly sizing the data files eliminates the performance problem.

When installing PeopleSoft applications using the Database Configuration Wizard, you have the option to let the data files grow until there's no more space on the storage devices. When installing the database manually, it's necessary to manually review and modify the file *PS_HOME*\scripts\createdb_2005.sql. It includes the following lines that the database administrator should review and update with appropriate values:

```
-- ALTER DATABASE <DBNAME> MODIFY FILE (NAME = <DATANAME>, MAXSIZE = UNLIMITED)
-- go
-- ALTER DATABASE <DBNAME> MODIFY FILE (NAME = <LOGDATANAME>, MAXSIZE = UNLIMITED)
-- go
```

Autoshrink

For PeopleSoft databases, make sure the *autoshrink* option is disabled. In very specific scenarios it will be necessary to "shrink" a database file. This should be done with caution; in general, it's a better practice to do it manually.

Read Committed Snapshot Isolation

PeopleSoft applications use a "pessimistic" implementation of the READ COMMITTED isolation level. SQL Server supports optimistic concurrency control with its implementation of the READ COMMITTED isolation level, called READ COMMITTED SNAPSHOT.

Optimistic concurrency control has these benefits:

- The overhead required for managing locks is minimized.
- Data modification operations cannot be blocked by read operations.

Disabling Read Committed Snapshot Isolation

Under normal circumstances, this feature should always remain enabled. You can disable it if a critical problem is identified.

Before disabling the feature:

- Make sure there are no open transactions. This means that you must close down the application server and the process scheduler.
- If there is a risk that users are still connected with open transactions, then change the database to single user mode before continuing.

The command to disable the feature is:

```
ALTER DATABASE dbname
     SET READ_COMMITTED_SNAPSHOT  OFF
```


The command to change the database to single user mode is:

```
ALTER DATABASE dbname
     SET SINGLE_USER  ON
```

File Management

PeopleSoft recommends the use of separate physical disks for the Microsoft SQL Server data files. Ideally, databases like master, tempdb, and application databases should be on separate disks, as should the operating system paging file (in case you run some additional applications other than the database software). As a general rule, the more spindles the better; always choose more smaller-size disks over fewer larger-size disks. If you don't have separate physical disks for each of the datafiles, you should at least place your tempdb, data, and log files on separate physical devices. Make sure that your log device is using its own disk controller and is not accessed by any other device.

Note: You should always consider disk fault tolerance when deciding how you want the database server configured.

Using Filegroups

Microsoft SQL Server 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.

For more information, see the product documentation for *Microsoft SQL Server*and *Microsoft Windows*.

Tempdb

PeopleSoft heavily uses the tempdb database. Consider moving tempdb to its own set of disks or disk array. The size of tempdb should be adjusted to be approximately 15% to 20% of the total size of your PeopleSoft database.

Another good practice is to distribute tempdb into several data files of the same size. As a guideline, you might want to have one file for each processor assigned for SQL Server. If possible, spread these data files on a high performance disk array.

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
```

Trace Flags

When reporting problems to PeopleSoft 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, including:

- if the statement was recompiled.
- if the statement was using an old query plan.
- the time it took to execute.
- the time between executions.
- if the SQL was parametrized.

Once you find the SQL with problems, you can use the SQL Server profiler to reproduce this outside of your PeopleSoft application.

Note: Keep in mind that tracing could affect performance considerably, and you won't be able to reproduce some problems with tracing enabled.

Related Links

"Understanding PSADMIN" (PeopleTools 8.54: System and Server Administration)

"Understanding PeopleSoft Configuration Manager" (PeopleTools 8.54: System and Server Administration)

"Using the PSADMIN Utility to Configure Process Scheduler Tuxedo Servers" (PeopleTools 8.54: Process Scheduler)

Database Monitoring

Available through the configuration files for the Process Scheduler and the Application Server, the activation of the EnableDBMonitoring option 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 selected. Modify the scripts according to your needs.

```
--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>
```

Appendix B

Administering PeopleSoft Databases on DB2 UDB for z/OS

Understanding DB2 UDB for z/OS Administration

This section discusses:

- Database considerations.
- Concurrency.

Note: *DB2 UDB for z/OS* is the official IBM name for the DBMS.
For the sake of brevity, these topics sometimes refer to DB2 UDB for z/OS as *DB2 z/OS.*

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

For PeopleTools, the use of Cursor With Hold (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.
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.

Utility	*Description*
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 directly in the DB2CLI.INI directly or using the Client Configuration Assistant.

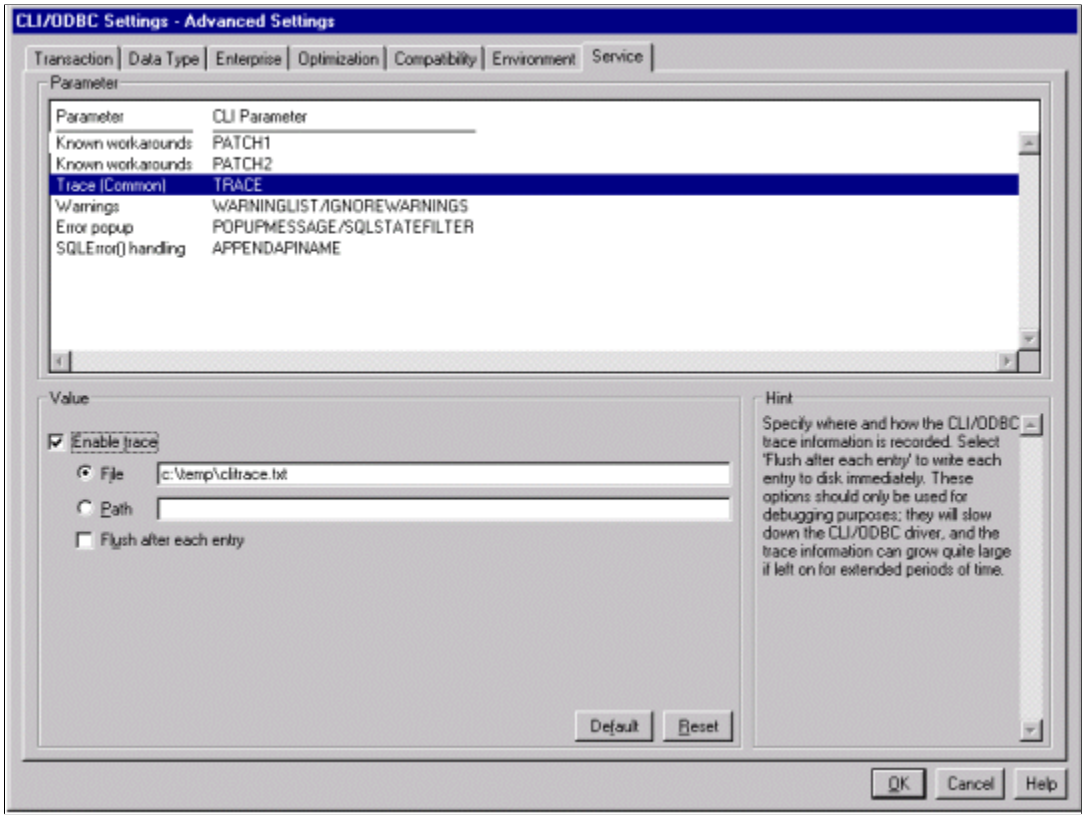
When updating the DB2CLI.INI directly, these are the recommended settings for enabling Trace:

```
[COMMON]
TRACEFLUSH=1
TRACEPATHNAME=C:\TEMP\DB2TRACE\
 (or use TRACEFILENAME=C:\TEMP\filename to direct to file)
TRACECOMM=1
TRACE=1   (trace=0 turns the trace OFF)
TRACEREFRESHINTERVAL=60
```


Or you can enable the trace, with the same options, using the Service tab in Client Configuration Assistant.

Image: DB2 UDB Client Configuration Assistant

This example illustrates the fields and controls on the Service tab in Client Configuration Assistant for DB2 UDB.



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 z/OS Server

To enable the statistics report on the DB2 UDB for 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 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). 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 Users, Modules, and Actions with DB2 UDB Threads

Users log into a PeopleSoft Application with their individual PeopleSoft Operator Ids; however, these ids do not physically exist in the zSeries server's security facility. PeopleTools Security validates the authenticity of the PeopleSoft Operator ID, and then connects to DB2 for z/OS through the Application Server using a single ID referred to as the PeopleSoft Access ID.

See *PeopleTools Installation for DB2 for z/OS* for more information

To enable PeopleSoft DBAs and systems administrators to distinguish one application user from another, PeopleTools provides a mechanism to associate PeopleSoft Operator Ids with distributed DB2 threads. PeopleTools uses the following CLI Connection Attributes to pass the PeopleSoft operator ID, client workstation name, application name, PIA Component Name and current HTML page (PIA Module Action), or Application Engine Program Name, Section, Step, Type (AE Module Action) to DB2 for z/OS.

CLI Connection Attribute	***Description***
SQL_ATTR_INFO_USERID	Populated with PeopleSoft User ID.
SQL_ATTR_INFO_WRKSTNNAME	Populated with: • Client workstation name for two-tier connection. • Web server name for PIA transactions. • Application server name for three-tier connections.
SQL_ATTR_INFO_APPLNAME	Populated with: • Application server domain ID (not the domain name) for three-tier connections and PIA connections. • Module name (such as pside.exe or psqed.exe) for two-tier connections.
SQL_ATTR_INFO_ACCTSTR	Module-Action data for distributed PIA transactions and Application Engine programs.

Note: Ensure that EnableDBMonitoring has been activated for the application server domain (see the Database Options section of the domain configuration file), and that DB2 Monitor Trace Class 1 is enabled for the DB2 subsystem.

Note: This does not apply to USS-based processing. This applies to distributed PeopleSoft connections only (as in, PIA transactions and two-tier connections connecting through DB2 Connect).

The following output from the –DISPLAY THREAD command shows a two-tier Application Designer connection. User QEADMIN is connected through Application Designer (pside.exe), from client workstation SCL34150:

```
NAME      ST  A   REQ  ID                AUTHID   PLAN            ASID   TOKEN
SERVER   R2       0       pside.exe      CERT001   DISTSERV    007D   14772
```



```
 V437-WORKSTATION=SCL34150, USERID=QEADMIN,
      APPLICATION NAME=pside.exe
 V442-CRTKN=10.204.56.216.50770.11071904055
 V445-GACC38D8.C652.C816EA9282DD=14772 ACCESSING DATA FOR
  ::10.204.56.216
```

The following output from the –DISPLAY THREAD command shows an n-tier PIA transaction. User ID QEADMIN is running a transaction through web server dhcp-pleasanton2-4 using application server domain PEOPLESOFT.

```
NAME       ST A   REQ    ID                          AUTHID     PLAN          ASID TO⇒
KEN
SERVER   RA *   16099  PSAPPSRV.exe  CERT001    DISTSERV  007D 14756
 V437-WORKSTATION=dhcp-pleasanton2-4, USERID=QEADMIN,
      APPLICATION NAME=PEOPLESOFT
  V442-CRTKN=10.204.56.216.50588.11071903542
  V445-GACC38D8.C59C.C816E7F87993=14756 ACCESSING DATA FOR
   ::10.204.56.216
```

The Module-Action data seeded by PeopleTools using the SQL_ATTR_INFO_ACCTSTR CLI connection attribute can be found in DDF accounting trace records in the QMDASUFX field of the QMDASQLI DSECT, which is mapped by the DSNDQMDA macro. The DSNDQMDA mapping macro is contained in prefix.SDSNMACS, which is shipped with DB2 for z/OS.

Some DB2 monitoring products, such as Omegamon for DB2 z/OS, will allow you to view an online snapshot of these trace records, or to format and print the output. It is also possible to write a custom application to collect and format accounting trace records using the Instrumentation Facility Interface (IFI) if you do not have such a monitoring product.

See the appropriate IBM documentation for details for using IFI

The following examples are online snapshots of the PeopleSoft Module-Action data using IBM's Omegamon for DB2 z/OS. Omegamon displays the QMDASUFX, area, which contains the Module-Action data populated by PeopleTools, under the label DDCS Accounting Suffix.

The following is the N-tier PIA Transaction –DISPLAY THREAD output:

```
DSNV401I  -5K DISPLAY THREAD REPORT FOLLOWS -
 DSNV402I  -5K ACTIVE THREADS

 NAME     ST   A   REQ     ID                            AUTHID     PLAN         ASID⇒
 TOKEN
 SERVER  RA  *    16099   PSAPPSRV.exe    CERT001    DISTSERV 007D    14756
  V437-WORKSTATION=dhcp-pleasanton2-4, USERID=QEADMIN,       APPLICATION NAME=PEOPL⇒
ESOFT
V442-CRTKN=10.204.56.216.50588.11071903542
V445-GACC38D8.C59C.C816E7F87993=14756 ACCESSING DATA FOR   ::10.204.56.216
```

The following is the Omegamon For DB2 z/OS Requestor Correlation Data – Active Thread Identification:

```
                    Requester Correlation Data
 Command ===> ____________________________________________________

 Product ID . . . . . . . . . . . . . : CLIENT/SERVER
 Product Version  . . . . . . . . . . : V9 R7 M3
 Client Platform  . . . . . . . . . . : NT
 Client Application Name  . . . . . . : PEOPLESOFT
 Client Authid  . . . . . . . . . . . : QEADMIN

 DDCS Accounting Suffix
 ORCLPS01SCPERSONALDICT SCPERSONALDICTLANG ORCLPS52
```


Where the PIA component Name (Module) is SCPERSONALDICT and the current PIA page (Action) is SCPERSONALDICTLANG

The following is a two-tier distributed Application Engine Program –DISPLAY THREAD output:

```
NAME     ST A   REQ   ID              AUTHID   PLAN            ASID  TOKEN
SERVER  RA *  17664  psae.exe     CERT001  DISTSERV   007D  14835
  V437-WORKSTATION=SCL34150, USERID=QEADMIN,
       APPLICATION NAME=psae.exe
  V442-CRTKN=10.204.56.216.52097.11071906585
  V445-GACC38D8.CB81.C817113C2D33=14835 ACCESSING DATA FOR
   ::10.204.56.216
```

The following is Omegamon For DB2 z/OS Requestor Correlation Data – Active Thread Identification:

```
                        Requester Correlation Data
 Command ===> ____________________________________________________________

 Product ID . . . . . . . . . . . . : CLIENT/SERVER
 Product Version  . . . . . . . . . : V9 R7 M3
 Client Platform  . . . . . . . . . : NT
 Client Application Name  . . . . . : psae.exe
 Client Authid  . . . . . . . . . . : QEADMIN

 DDCS Accounting Suffix
 ORCLPS02PSAE.AEMINITEST.5540 AEMINITEST.MAIN.Step02.P ORCLPS52
```

The Application Engine program name is AEMINITEST (Module), and the current Section, Step, Type (Action) is MAIN, Step02, Type P (PeopleCode).

The format of the PeopleTools Module-Action trace data is as follows:

```
Prefix:  will always be ORCLPS
Record Type: 01 denotes PIA transaction data, OR,
Record Type: 02 denotes Application Engine data
PIA Transaction Module-Action: PIA Component Name (Module) followed by
Current HTML page (Action) AE Program Module-Action:
PSAE.<AE program name.unique process instance number>
<AE Program Name.Section.Step.Type (Action)>
```

The Suffix will always be ORCLPS with current PeopleTools release identifier, such as 52, which denotes release 8.52.

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 (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 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.

Be aware that some application processes are designed to run only through Process Scheduler.

SQR for z/OS is delivered with the PeopleTools installation. The PeopleTools DB2 UDB for 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.

Allow at least 12 cylinders of 3390 DASD or equivalent disk space to complete the installation.

Related Links

Process Scheduler

Running SQRs Outside of Process Scheduler

These run time parameters are supplied to SQRs initiated by the Process Scheduler.

- PROCESS_INSTANCE
- OPRID
- RUN_CNTL_ID (required by specific applications only, such as Financials)

When an SQR request is 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 PeopleSoft application documentation 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 product documentation.

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.

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.

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.

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.

Note: The system overrides the FILEPREFIX and FILESUFFIX variables with the value specified in setfile.sqc, when running the SQR reports through the Process Scheduler with an output type of *Web.* The FILEPREFIX variable will be substituted with the log_output folder path of the Process Scheduler domain (<DOMAIN>/log_output/SQR_<process_name>_<prcs_instance>/). The FILESUFFIX variable will be substituted with a blank value. When there is a requirement to generate a static file or provide a static file input to the SQR report for an output type of *Web,* these variables should be redefined in the SQR file itself (rather than in the SQC file) to any suitable path and extension.

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: The %UpdateStats meta-SQL is enabled by default for both z/OS and Windows Process Scheduler servers..

See *PeopleTools Installation for DB2 for z/OS*.

Note: A document from IBM entitled “Enabling the DSNUTILS Stored Procedure for DB2 for OS/390” is available on My Oracle Support. This document outlines the minimum requirements to run Workload Manager in goal mode which is required by the DSNUTILS stored procedure.

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 combines multiple tables into a single tablespace. The “Enhanced” installation path has segregated the PeopleSoft temporary tables into separate tablespaces.

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 used for 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. Database and tablespace name values are 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.

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 z/OS.*

Activating %UpdateStats

%UpdateStats can be disabled by setting the DbFlags application server domain parameter.

This parameter has two values that apply to %UpdateStats:

- *0* – enable %UpdateStats.
- *1* – disable %UpdateStats.

%UpdateStats is enabled by default for Windows and z/OS Process Scheduler servers.

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 Configure a Process Scheduler Server or Edit a Process Scheduler Configuration File.
3. If you select the first option, set the value for DbFlags to *0* to enable or *1* to disable %UpdateStats.
4. If you select the second option, 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 Installation for DB2 for 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.

Creating Temporary Tables

For each temporary table you define, a base table structure and a number of its instances are created in the database as ordinary tables with ordinary table structures. The number of temporary table instances is determined by the value of the Temp Table Instances setting in PeopleTools Options added to the number of PeopleSoft Application Engine temporary tables. These temporary tables are used as work tables that hold transient data, and because they are real tables, they are permanent structures in the database, remaining until an explicit DROP TABLE command is executed against them.

The nature of a temporary table means that the amount of data that each temporary work table holds varies significantly after each use. Therefore, when RUNSTATS are executed against them, there is a good chance that the statistics captured may not apply and will negatively influence the DB2 optimizer access path selection the next time you use the temporary work table.

Each record of the type Temporary Table is defined as a VOLATILE table in DB2 (beginning with version 8). This definition takes advantage of the DB2 optimizer's enhanced capability to formulate efficient index access paths for those tables that hold volatile data, without relying on current table statistics.

Example: VOLATILE Used in CREATE TABLE DDL

This example shows the VOLATILE parameter in the CREATE TABLE DDL for the base temp table and its instances.

```
CREATE TABLE FSDMOA.PS_AEEXT_TAO (PROCESS_INSTANCE DECIMAL(10) NOT
 NULL,
   AE_INT_1 SMALLINT NOT NULL,
   AE_APPLID CHAR(12) NOT NULL,
   AE_SECTION CHAR(8) NOT NULL,
   AE_STEP CHAR(8) NOT NULL) VOLATILE IN FSDMOA.PTAPPE;
```

Working with Alters on DB2 z/OS

This section provides an overview and discusses:

- Advisory reorg pending and rebuild pending status.
- Determining whether a tablespace or index is in a pending status.
- Tablespace versioning.
- Determining when to recycle tablespace version numbers.
- Working with DB2 tablespace versioning and PeopleSoft upgrades.

Understanding Alters on DB2 z/OS

PeopleTools Data Administration tools support native DB2 zOS alter syntax (known as DB2 Online Schema Evolution) for altering tables. For example:

```
ALTER TABLE <table name> ALTER COLUMN <column name> SET DATA TYPE <new definition>
```

Use of this Alter syntax provides the capability to make structural changes to a table without the requirement to drop and recreate it. The data remains available for both inquiry and update processing. This means that when possible, PeopleTools use "Alter In Place" through this native alter syntax, rather than "Alter By Table Rename" for specific additional use cases.

As a result of this Alter In Place syntax, existing data rows are not immediately reformatted at the time that the alter is committed to DB2. Instead, as a result of the use of the ALTER TABLE <table name> ALTER COLUMN <column name> SET DATA TYPE <new definition> syntax, DB2 does the following:

- Places the corresponding tablespace or indices in either an Advisory Reorg Pending (AREO*), or Rebuild Pending (RBDP) status depending on the nature of the change (alter) made to a particular table.
- Creates a new version of the tablespace which reflects the format of the desired change. Despite the pending status, the data continues to be available for inquiry and update processing, and rows

are subsequently materialized in the post-alter format (the current tablespace version) as they are retrieved.

Advisory Reorg Pending and Rebuild Pending Status

The following types of changes will cause tablespaces and indices to be placed in Advisory Reorg Pending (AREO*) status:

DB2 z/OS version	*Changes*
9.1 (and later)	• Changes made to the length of character or vargraphic (Unicode) columns will cause the tablespace that contains the table to be placed in AREO* • Changes between compatible numeric data types will cause the tablespace that contains the table to be placed in AREO* • Altering a table to add a new column if the tablespace that contains the table is currently at version 0 (see tablespace versioning below), or, adding a new column and issuing DML (insert, update, or delete) across the commit scope (also see APAR PK54341 for more details).

The following types of changes will cause indices to be placed in Rebuild Pending (RBDP) status:

DB2 z/OS version	*Changes*
9.1 and later	Changes between compatible numeric data types will place any index that contains the affected column in RBDP.

Determining Whether A Tablespace or Index Is In A Pending Status

To find objects in a pending status, run the display database command using the DB2 Interactive Command Processor (DB2I), DSN session under TSO, or a z/OS console session. As an example, the following command displays all tablespaces in the Advisory Reorg Pending (AREO*) status and all indices in the Rebuild Pending status for a database called Q51802R1:

```
-DIS DB(Q51802R1) SPACENAM(*) ADVISORY(AREO*) RESTRICT(RBDP)

DSNT360I  -5A **********************************
DSNT361I  -5A *  DISPLAY DATABASE SUMMARY
              *    RESTRICTED ADVISORY
DSNT360I  -5A **********************************
DSNT362I  -5A     DATABASE = Q51802R1  STATUS = RW
                  DBD LENGTH = 2996486
DSNT397I  -5A
NAME     TYPE PART  STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ----- ----------------- -------- -------- -------- -----
PTTBLZZ  TS         RW,AREO*
IDX1R9   IX         RW,RBDP
 ******* DISPLAY OF DATABASE Q51802R1 ENDED      **********************
```

Tablespace PTTBLZZ is in Read, Write and Advisory Reorg Pending status.

Index IDX1R9 is in Read, Write and Rebuild Pending status.

Tablespace Versioning

In most circumstances, a committed alter will cause DB2 to create a new version of the tablespace, which reflects the format of the desired alter. Rows are subsequently materialized in the post-alter format as they are retrieved. DB2 can store up to a maximum of 256 active versions of a tablespace, numbered 0 to 255. Version 0 indicates a tablespace that has never been altered, and version 0 is never reused. All rows are formally converted to the format determined by the latest tablespace version when the tablespace is reorganized.

When a tablespace reaches the maximum number of versions, it is important to note that subsequent attempts to alter any table contained in the tablespace will fail with SQL Code -4702. At this point, the DB2 Reorg Tablespace and Modify Recovery utilities must be executed to:

- Reformat the data as dictated by the latest version of the tablespace (by running the Reorg Tablespace utility).
- Recycle the version numbers so that DB2 can reuse all version numbers other than the active version of the tablespace (by running the Modify Recovery utility).

Note: The Reorg Tablespace utility with Index(All) option also removes the AREO* and RBDP statuses.

Important! When executing a significant number of Alters In Place (such as during the Upgrade Alter Without Deletes step or when applying a Maintenance Pack), the potential exists to create the maximum number of tablespace versions, particularly when running against shared tablespaces.

To mitigate the risk of encountering the -4702 SQLCode, PeopleTools Development recommends that you query the DB2 zOS Catalog to determine if there are any tablespaces in your PeopleSoft database that are close to the maximum version limit, and then execute the Reorg and Modify Recovery utilities accordingly to recycle version numbers before beginning any step that executes a significant number of Alters In Place.

Determining When to Recycle Tablespace Version Numbers

To determine which tablespaces may require recycling of version numbers prior to beginning the Alter Without Deletes step, review the OLDEST_VERSION and CURRENT_VERSION columns of SYSIBM.SYSTABLESPACE for each tablespace in your PeopleSoft database as described in this section.

Use the following query as a guide to list the oldest and current version numbers for all PeopleSoft tablespaces in your environment:

```
SELECT NAME, DBNAME, OLDEST_VERSION, CURRENT_VERSION FROM
SYSIBM.SYSTABLESPACE
WHERE CREATOR = '<authid of the owner of the tablespace>'
AND CURRENT_VERSION > 0;
```

Recycling tablespace version numbers is mandatory when all version numbers are currently in use. All tablespace versions are currently in use when one of the following conditions is true:

- The value of the CURRENT_VERSION column is 255 (SYSIBM.SYSTABLESPACE).
- The value of the CURRENT_VERSION column is one less than the value of the OLDEST_VERSION column.

Example: The value of the CURRENT_VERSION column is 255 (SYSIBM.SYSTABLESPACE)

All versions for tablespace PTTLRG0M are now in use, and the version numbers must be recycled.

```
DBNAME       NAME           OLDEST_VERSION  CURRENT_VERSION
------------ -------------  --------------  ---------------
CEBC0003     PTTLRG0M                    0              255
```

No subsequent Alters to objects in this tablespace will be allowed, and any attempt to do so will result in a SQLCode of -4702.

After executing the Reorg Tablespace and Modify Recovery utilities, CURRENT_VERSION and OLDEST_VERSION are equal:

```
DBNAME     NAME       OLDEST_VERSION    CURRENT_VERSION
---------- ---------  ----------------  ----------------
CEBC0003   PTTLRG0M               255               255
```

The following is the result of another Alter committed against an object in tablespace PTTLRG0M after the Reorg Tablespace and Modify Recovery utilities were executed to recycle the version numbers:

```
DBNAME     NAME       OLDEST_VERSION    CURRENT_VERSION
---------- ---------  ----------------  ----------------
CEBC0003   PTTLRG0M               255                 1
```

The value of CURRENT_VERSION for tablespace PTTLRG0M will now continue to increment from 1 to 254 as Alters In Place are committed. When the value of CURRENT_VERSION reaches 254, version numbers must again be recycled.

Example: The Value of the CURRENT_VERSION Column Is One Less Than The Value of the OLDEST_VERSION Column

For the following example, assume that shared tablespace PTAMSG01 was at its initial version (version 0), and that several Alters In Place were committed against multiple tables contained therein:

Initial version (version 0) of PTAMSG01:

```
DBNAME     NAME        OLDEST_VERSION  CURRENT_VERSION
---------- ----------  --------------  ---------------
CEBC0003   PTAMSG01                 0                0
```

After 108 committed Alters In Place to various tables within PTAMSG01:

```
DBNAME     NAME        OLDEST_VERSION  CURRENT_VERSION
---------- ----------  --------------  ---------------
CEBC0003   PTAMSG01                 0              108
```

Now assume that the Reorg Tablespace and Modify Recovery utilities were executed against tablespace PTAMSG01 when the CURRENT_VERSION was 108--prior to reaching the maximum of 255.

As a result of executing these utilities, the values for OLDEST_VERSION and CURRENT_VERSION were both set to 108:

```
DBNAME       NAME       OLDEST_VERSION CURRENT_VERSION
------------ ---------  --------------  --------------
CEBC0003     PTAMSG01              108             108
```

The value of CURRENT_VERSION for this tablespace then continued to increment from 108 to 255--and then onward to 107 (below).

At a value of 107, the value of CURRENT_VERSION was one less than the OLDEST_VERSION and execution of the Reorg and Modify Recovery utilities was mandatory.

```
DBNAME       NAME       OLDEST_VERSION   CURRENT_VERSION
------------ ---------  ---------------  ---------------
CEBC0003     PTAMSG01               108              107
```

Any attempt to alter another table in tablespace PTAMSG01 while at a CURRENT_VERSION of 107 would have resulted in SQLCode -4702 because the value of CURRENT_VERSION was one less than the value of OLDEST_VERSION.

Working with DB2 Tablespace Versioning and PeopleSoft Upgrades

This section covers these key issues related to DB2 tablespace versioning and PeopleSoft upgrades:

- Avoiding SQL code -4702.
- Ensuring optimal performance for Data Conversion steps.

Avoiding SQL Code -4702

An Alter Without Deletes script executes a significant number of alters. Although the PeopleTools alter processing for DB2 zOS was designed to prevent DB2 from creating an excessive number of tablespace versions by carefully controlling the manner in which table alters are committed per tablespace, it is possible that DB2 may still create the maximum number of tablespace versions when running the Alter Without Deletes script if there are shared tablespaces already close to the maximum 255 version numbers.

To mitigate the possibility of the Alter Without Deletes script stopping due to SQL code -4702, run the following query prior to the Alter Without Deletes step, and run the Reorg Tablespace and Modify Recovery utilities accordingly for any tablespaces that may be close to the maximum allowed version number (either the CURRENT_VERSION is equal to 255, or CURRENT_VERSION is one less than OLDEST_VERSION, as previously explained).

```
SELECT NAME, DBNAME, OLDEST_VERSION, CURRENT_VERSION FROM
SYSIBM.SYSTABLESPACE
WHERE CREATOR = '<authid of the owner of the tablespace>'
AND CURRENT_VERSION > 0;
```

Then continue with the Alter Without Deletes script as documented.

If you run the Alter Without Deletes script manually (outside of the Change Assistant) using a tool, such as the DB2 Command Line Processor, Command Editor, SPUFI, and so on, disable the auto-commit feature. Change Assistant disables auto-commit when it invokes the Command Line Processor.

Ensuring Optimal Performance For Data Conversion Steps

While the PeopleTools alter processing for DB2 zOS was designed to prevent DB2 from creating excessive tablespace versions, you should still expect some shared tablespaces in your environment to become multi-versioned as a result of the normal execution of the Alter Without Deletes step. Until you reorganize, you may notice:

- performance of dynamic SQL statements executed against a tablespace with multiple versions may suffer.
- any indices in RBDP may be ignored by the DB2 optimizer.

To ensure optimal performance, we strongly recommend that you run the Reorg Tablespace (Index All) utility and the Modify Recovery utility to reformat data rows into the format described by the most current tablespace version for any tablespaces with several versions, after the Alter steps (Alter With/ Without Deletes) have completed and prior to beginning the Upgrade Data Conversion steps.

For more details regarding the use of the DB2 zOS Reorg Tablespace and Modify Recovery utilities, refer to your IBM DB2 documentation.

See *IBM DB2 UDB for z/OS Utility Guide and Reference*

Appendix C

Administering PeopleSoft Databases on DB2 UDB for Linux, UNIX, and Windows

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 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 displays 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 need to decompress it using the *db2insthtml* command under the DB2 UDB-installed directory. On AIX, it is similar to /usr/lpp/db2_xx_01/doc/db2insthtml. On Windows, the above steps are not needed because all the HTML files are uncompressed during normal DB2 UDB installation.

For more information, see your *DB2 Administration Guide* and *DB2 System Monitor Guide and Reference*.

Note: Oracle supports a number of versions of UNIX and Linux. Throughout this documentation, the word UNIX refers to all UNIX-like operating systems, including Linux.
For the sake of brevity, this documentation sometimes refers to DB2 UDB for Linux, UNIX, and Windows as DB2 LUW.

Instances

This section provides an overview of instances and discusses:

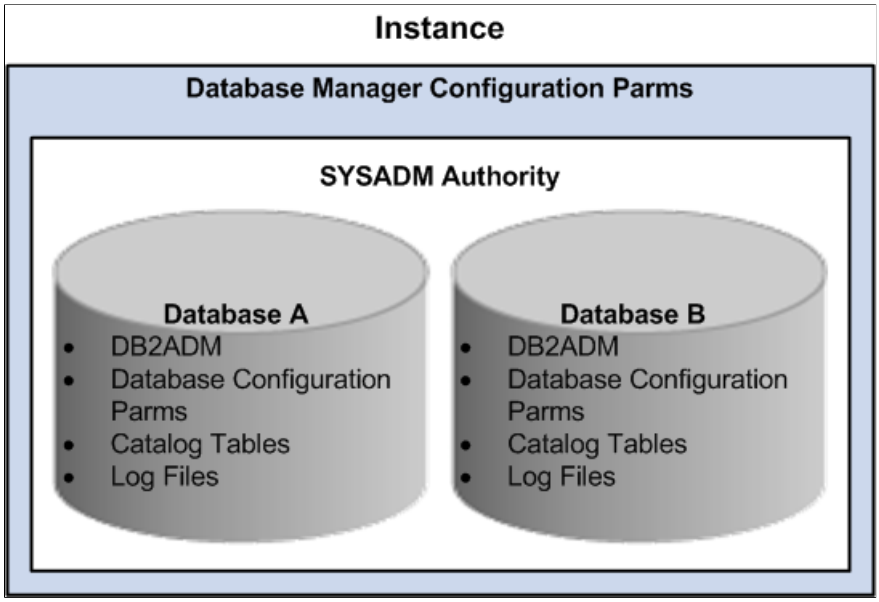
- SYSADM authority and security.
- Instances and Connectivity.
- Other considerations.

Understanding Instances

Operating System administrators (for UNIX logged in as root, for Windows 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.

Image: DB2 LUW instance housing a collection of databases sharing the same engine and database manager parameters yet with each database having its own DB2ADM, specific database configuration, catalog tables, and log files

The following graphic shows a conceptual view of a DB2 LUW instance. 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.



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 create all the PeopleSoft non-production 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 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 must assign different TCP/IP port numbers for the respective DB2 UDB instances.

To register the pair of TCP/IP ports, you edit the services file. Both the UNIX Server (/etc/services) and Windows (\windows\systemXX\drivers\etc\services) service files must specify the following:

```
db2dudb 50000/tcp#DB2 Client Application Enabler-Dev
db2pudb 50010/tcp#DB2 Client Application Enabler-Prod
```

Note: The names db2dudb and db2pudb are user-defined.

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:

- 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. The Control Center is a Java application or can be run as a Java applet within a browser. To start the Control Center, enter the command *db2cc* at a command prompt.

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.

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 HCM 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 the product documentation for *PeopleTools Installation for DB2 for Linux, UNIX, and Windows* for more information.

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)
```

On AIX, the COOKED File System refers to the Journal File System (JFS). On Windows, 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*.

Optimizing Table Space Capacity With Auto-Resize

DB2 LUW offers the “AUTORESIZE” option for use with DMS table spaces so that the database system can automatically manage the allocation of additional space when a previous limit has been reached. PeopleTools supports the use of AUTORESIZE.

The AUTORESIZE option enables database administrators to create table spaces with an ample yet reasonable initial size and then specify the increment by which the system enlarges the table space when necessary. AUTORESIZE is transparent to any application connected to the database, and because it automatically manages table space size based on the specified configuration settings, database administrators do not need to enlarge table spaces manually on an ad hoc basis.

Note: AUTORESIZE is only available for table spaces within database-managed spaces (DMS). However, PeopleTools, as a standard, only creates DMS table spaces, so this restriction does not apply within the PeopleTools context. PeopleTools does not create system-managed spaces (SMS).

Enabling AUTORESIZE

The SQL parameters for enabling and configuring auto-resize are:

Parameter	*Description*
AUTORESIZE	YES | NO Indicates whether auto-resize should be enabled for a table space. Disabling auto-resize is not recommended.

Parameter	*Description*
INCREASESIZE	K | M | G Specify the size of the increments by which the system should increase the table space size. Use an integer value in either kilobytes (K), megabytes (M), or gigabytes (G). **Note:** This value can also be expressed in terms of a percentage by which to increase the table space size. However, expressing the increase size value in terms of a percentage is discouraged, unless you have detailed knowledge of how DB2 calculates the percentage. If you do not set the percentage to a value that optimally increases table space size based on the current size and the amount of data typically inserted in your system, you may incur undesired amounts of wasted space.
MAXSIZE	K | M | G | NONE Specify a maximum size that the table space can reach. Use an integer value in either kilobytes (K), megabytes (M), or gigabytes (G). NONE indicates that the table space can grow to the limit imposed by the file system.

How you enable AUTORESIZE depends on the status of your current implementation, as in, whether you are creating or upgrading a database, or just working with an existing database.

If you are creating a new PeopleSoft database or performing an upgrade on a PeopleSoft database, you use the Database Creation Wizard and the delivered DDL scripts to enable auto-resize as per the instructions in your PeopleSoft upgrade and/or installation documentation.

The DDL of the provided scripts and those created using the Database Creation Wizard is similar to the following CREATE TABLESPACE example:

```
CREATE TABLESPACE PTTLRG MANAGED BY DATABASE USING
(
 FILE '/data1/psdb2/ptdbname/PTTLRG.DBF'  10 M
) EXTENTSIZE 16 PREFETCHSIZE 48 DROPPED TABLE RECOVERY OFF
AUTORESIZE YES INCREASESIZE 10 M MAXSIZE NONE ;
```

In this example, the system creates the table space PTTLRG with an initial size of 10 Mb, with the AUTORESIZE option on, specifying that the database system will increase the table space size by 10 Mb each time a limit is reached.

Note: If you have already determined the appropriate initial size, increase size, and maximum size for table spaces at your site, edit the delivered scripts to reflect those values.

If you are working with an existing PeopleSoft database and not performing any database creation or upgrade tasks on the entire database, you can submit a SQL ALTER command to update any existing table spaces for which you want to enable this feature.

The ALTER syntax is:

```
ALTER TABLESPACE <name> AUTORESIZE YES INCREASESIZE <size> MAXSIZE <size> ;
```

Disabling Auto-Resize

By default, AUTORESIZE is enabled for all new and upgraded PeopleSoft databases. Because of its convenience and because the PeopleSoft system assumes AUTORESIZE is on, disabling AUTORESIZEis not recommended.

To disable the auto-resize option for a table space, issue a SQL ALTER statement using the following syntax:

```
ALTER TABLESPACE <name> AUTORESIZE NO ;
```

Determining Increase Size

To determine the appropriate increase size value for your table spaces, Oracle recommends first categorizing your table spaces into the following categories: small, medium, and large. These categories reflect the growth potential of the table space. For example, a small table space is one that is not expected to increase at the same rate or reach the same size as a large table space.

The following table provides some suggested increase sizes by category.

Category	***Suggested Increase Size Range***
Small	10 M – 100 M
Medium	300 M – 500 M
Large	700 M – 1000 M

Confirming that AUTORESIZE is Enabled

To confirm that AUTORESIZEis enabled and working as desired, use the DB2 table space monitor snapshot output. For example, assume you create the table space PSHRDATA with AUTORESIZEon. PSHRDATA, historically, is a table space that typically reached maximum space if not properly monitored and adjusted by database administrators. In this example, PSHRDATA is created with an initial space allocation of 1500 8k pages, using the following SQL:

```
CREATE TABLESPACE  PSHRDATA  PAGESIZE 8K MANAGED BY DATABASE USING
(
 FILE '/data1/psdb2/ptdbname/PTTREEIDX.DBF'  1500
) EXTENTSIZE 16 PREFETCHSIZE 48 BUFFERPOOL PSUBUFPOOL DROPPED TABLE RECOVERY OFF
AUTORESIZE YES INCREASESIZE 10 M MAXSIZE NONE ;
```

After a period of time in which you can assume large amounts of transactional data has been inserted into your database, connect to the database and show the detail report on the table spaces. Use the **list tablespaces show detail** command.

```
db2 connect to <db-name>
db2 list tablespaces show detail
```

A section of the output would appear similar to the following:

```
.......
Tablespace ID                        = 45
```



```
 Name                                 = PSHRDATA
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 2768
 Useable pages                        = 2752
 Used pages                           = 2000
 Free pages                           = 752
 High water mark (pages)              = 2000
 Page size (bytes)                    = 8192
 Extent size (pages)                  = 16
 Prefetch size (pages)                = 48
 Number of containers                 = 1
......
```

The current footprint (Total Pages) of PSHRDATA is now 2768 8k pages, well over the initial allocation. This clearly indicates that DB2 detected a request for additional free pages beyond the initial space allocation for PSHRDATA and automatically extended the table space.

Monitoring Table Space Size Allocation

While the AUTORESIZEoption greatly reduces the amount of manual intervention, database administrators should continue to monitor the size and growth of the following elements of your database to ensure that you are optimizing space usage:

- File system
- Table space container

Temporary Table Creation

For each temporary table you define, a base table structure and a number of its instances are created in the database as ordinary tables with ordinary table structures. The number of temporary table instances is determined by the value of the Temp Table Instances setting in PeopleTools options Utilities, Administration, PeopleTools Options added to the number of PeopleSoft Application Engine temporary tables. These temporary tables are used as work tables that hold transient data, and because they are real tables, they are permanent structures in the database and remain until an explicit **drop table** command is executed against them.

The nature of a temporary table means that the amount of data that each temporary work table holds varies significantly after each use. Therefore, when RUNSTATS are executed against them, there is a good chance that the statistics captured may not apply and will negatively influence the DB2 optimizer access path selection the next time you use the temporary work table.

Each record of the type Temporary Table is defined as a `VOLATILE` table in DB2 (beginning with version 8). This definition takes advantage of the DB2 optimizer's enhanced capability to formulate efficient index access paths for those tables that hold volatile data, without relying on current table statistics. Additionally, because each temporary work table can only be assigned to a single process, the temporary work table is defined with the `LOCKSIZE TABLE` attribute to reduce the number of lock resources to be managed by DB2. Both the `VOLATILE` and`LOCKSIZE TABLE` attributes are be implemented using `ALTER` table statements.

Example: ALTER Statement Using VOLATILE and LOCKSIZE TABLE

This example shows the additional DDL for ALTER statements generated for temporary tables that implements the VOLATILE andLOCKSIZE TABLE attributes.

```
CREATE TABLE PS_AC_CSTSEQ_TAO1 (
PROCESS_INSTANCE DECIMAL(10) NOT NULL,
    DEPOSIT_BU CHAR(5) NOT NULL,
    DEPOSIT_ID CHAR(15) NOT NULL,
    PAYMENT_SEQ_NUM INTEGER NOT NULL,
    BUSINESS_UNIT CHAR(5) NOT NULL,
    CUST_ID CHAR(15) NOT NULL,
    ID_SEQ_NUM INTEGER NOT NULL)
IN ARWORK INDEX IN ARWORKIDX
NOT LOGGED INITIALLY;

ALTER TABLE PS_AC_CSTSEQ_TAO1 VOLATILE;
ALTER TABLE PS_AC_CSTSEQ_TAO1 LOCKSIZE TABLE;
```

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()

This section discusses DB2 LUW and %TruncateTable().

Note: Beginning with PeopleTools 8.52 when installed on DB2 LUW 9.7, whenever it is operationally applicable, PeopleTools translates %TruncateTable() metaSQL to the native TRUNCATE TABLE command introduced in DB2 LUW 9.7. However, if an earlier version of DB2 LUW (less than 9.7) is installed, PeopleTools reverts to the previous method of using a DB2 utility as documented in this section.

Prior to DB2 LUW 9.7, there's no SQL implementation of a Truncate Table command, such as the one found in Oracle. PeopleTools 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 negative effect of bufferpool flushing comes 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, you can convert %TruncateTable into a SQL DELETE FROM clause.

To enable this conversion there is a setting (DbFlags) in PSPRCS.CFG and PSAPPSRV.CFG (or PSADMIN). 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=2 will enable the workaround.

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

If you are running %TruncateTable on Windows, 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 SHRLEVELCHANGE
```


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 the product documentation for *PeopleTools Installation for DB2 for Linux, UNIX, and Windows* for more information.

Performing Queries on a Windows Workstation

Query capability on Windows workstations can be accomplished using 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 retrieved for later use from the Script Center.
- Third-party vendor tools such as Business Objects, Information Advantage, and so forth.

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 Java applet using a standard browser.

Connectivity Using ODBC/CLI

A PeopleTools development environment (two-tier client) establishes connectivity through these technology layers:

- PeopleTools layer.
- Microsoft ODBC layer.

- IBM ODBC Driver layer.

The following table describes the events occurring within each layer during a connectivity request.

Technology Layer	***Description of Events***
PeopleTools	A PeopleTools application, such as Application Designer, Issues a connect to database request (SQL Connect). The PeopleTools utility PSODBC.DLL processes the request and formats the SQL request in an ODBC-compliant format and invokes the ODBC SQLConnect function. Information passed includes: • Database name. • User ID/Table Owner • Password
Microsoft ODBC	In the Microsoft ODBC Layer, ODBC.DLL reads the registry entry for the ODBC checking for the data source name (in this case, the database name). It finds this entry and loads the associated vendor driver, such as \WINDOWS \SYSTEM32\DB2CLI.DLL. **Note:** By reading ODBC.INI, ODBC.DLL determines which of several possible vendor ODBC-compliant drivers to load. In this case, it loads the IBM driver, DB2CLI.DLL.
IBM ODBC Driver	In the IBM ODBC Driver Layer, DB2CLI.DLL: • reads the Windows Environment Variable (DB2PATH) to obtain the path for the db2cae executable. • reads *db2 connect install dir*\db2\sqldbdir\sqldbdir and *db2 connect install dir*\db2\sqlnodir\sqlnodir to obtain database directory and node directory information. • formats and submits the connect request to the database server.

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 the PeopleSoft signon dialog box, the database name must be specified in upper case.
- On the PeopleSoft signon dialog box, the user/password is case-sensitive (examine table PSOPRDEFN).
- Are the ConnectID and ConnectPSWD specified properly in the PeopleTools Configuration Manager?
- Is the DB2 LUW database running? To check it:

  ```
  Database Server, type "db2 connect to database-name"
  ```

- Does the PS.PSDBOWNER table 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 requires that the user ID, which is cataloging databases and nodes, be an Administrator (not just a user with administrative authorization). The Administrator's user ID must not exceed 8 characters.

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. Allow statistics to compile.
4. Issue the following statement(s):

   ```
   db2 "get  snapshot for database on hr910dmo" > snapsht1.dbx
   ```
5. Wait a minute to allow additional statistics to compile.
6. Issue the following statement:

   ```
   db2 "get  snapshot  for  database on hr910dmo" > 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 parameter 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.

Monitoring Module and Action Information

For select PeopleTools "modules", the system captures the Module identifier and stores it in the DB2 ACCOUNTING field, which you can query as part of your typical performance monitoring. This can help you to associate transactions with a particular module when monitoring or troubleshooting.

PeopleTools populates the ACCOUNTING field as follows:

PeopleSoft	*Module*	*Action*
PIA transactions	PeopleSoft application component	PeopleSoft application page within the component.
Integration Broker	Service operation	PeopleCode event.
Application Engine	'PSAE'	Program name, section, step, and step type.

You can use the GET SNAPSHOT command to view samples of the information passed per module type.

Appendix D

Administering PeopleSoft Databases on Informix

Database Terminology

PeopleSoft uses the following technical naming conventions for Informix 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.

Working With Dbspaces

This section discusses:

- Dbspace strategy.
- Dbspace sizing.

Dbspace Strategy

The following are some dbspace strategies to consider:

- 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 the dbspace for indexes on a separate drive.

Dbspace Sizing

This section provides an overview and discusses:

- Creating dbspaces.
- Setting Buffer Pools.
- Setting the PHYSBUFF Parameter.

Understanding Dbspace Size

Informix Dynamic Server enables you to configure the page size of standard or temporary dbspace a with non-default page size. By default, PeopleTools DDL scripts set dbspaces to 4KB regardless of the default page size of the operating system.

Advantages of having larger page sizes include:

- Dbspaces of larger page sizes can support larger index key sizes.
- More items can fit in a larger index page and there will be a decreased number of levels in a btree index, which reduces index scan times.
- Increased access efficiency because with a larger page, larger numbers of rows get transferred to/from shared memory/disk, which means fewer I/O operations for data and indices.
- Longer rows can fit into a single page reduces data access times due to the decreased number of pages read per row.
- Pages large enough to fit "oversized" rows eliminate the overhead of access time for remainder pages.
- Checkpoint time is typically reduced with larger pages.

Creating Dbspaces

PeopleTools delivers two scripts, ptddl.sh and createtblspace.sh, located in <PS_HOME>/scripts/unix directory. The ptddl.sh script is executed by the Database Creation Wizard to create dbspaces required by a PeopleTools Demo database. The createtblspace.sh script is executed when the database needs to be created using only two dbspaces, as in pstable1 and psindex1.

The –k <page_size> option of the onspaces command sets the dbspace size. The default page size is set to 4K, however, you can set the dbspace to a custom value for your site's requirements (6K, 8K , 16K, and so on).

Example ptddl.sh:

```
...
onspaces -c -d <DBSPACE> -k 4 -p </path>/<DBSPACE>.dbf -o 0 -s 30000
...
onspaces -c -d PSIMAGE -k 4 -p </path>/PSIMAGE.dbf -o 0 -s 150000
...
onspaces -c -d PSIMGR -k 4 -p </path>/PSIMGR.dbf -o 0 -s 5000
...
onspaces -c -d PTWORK -k 4 -p </path>/PTWORK.dbf -o 0 -s 5000
```

Example createtblspace.sh:

```
...
echo "Creating dbspace pstable1..."
```



```
$INFORMIXDIR/bin/onspaces -c -d pstable1 -k 4  -p $1/pstable1.dbf -o 0 -s $2
echo "Creating dbspace psindex1..."
$INFORMIXDIR/bin/onspaces -c -d psindex1 -k 4 -p $1/psindex1.dbf -o 0 -s $3
echo "Running a Level 0 Archive..."
$INFORMIXDIR/bin/ontape -s -L 0
```

Note: You can define a different page size for temporary tables, so that they have a separate buffer pool.

Note: Rootdbs and other critical dbspaces, logdbs and physdbs, are created with the page size specified by the operating system.

Setting Buffer Pools

When creating a dbspace of non-default page size it requires a corresponding BUFFERPOOL configuration parameter in the ONCONFIG file. If it is not defined, the database server automatically adds an entry in the ONCONFIG file, which would be based on default value of BUFFERPOOL.

For example, if the onspaces command creates a new dbspace with a 4 KB page size, the database server takes the values of buffers, lrus, lru_min_dirty, and lru_max_dirty from the BUFFERPOOL default line, unless there already is a BUFFERPOOL entry for that page size.

The following example shows how the database server would automatically add a BUFFERPOOL entry for 4 KB based on the default values:

```
BUFFERPOOL
default,buffers=1000,lrus=8,lru_min_dirty=50.000000,lru_max_dirty=60.000000
BUFFERPOOL
size=2K,buffers=5000,lrus=8,lru_min_dirty=50.000000,lru_max_dirty=60.000000
BUFFERPOOL
size=4K,buffers=1000,lrus=8,lru_min_dirty=50.000000,lru_max_dirty=60.000000
```

Buffer pools can also be added using the onparams command. For example:

```
onparams -b -g 4 -n 3000 -r 2 -x 2.0 -m 1.0
```

This example adds 3000 buffers of 4KB each with 2 LRUS with a maximum dirty of 2% and minimum dirty of 1%. When using the onparams utility to add a buffer pool or when adding a dbspace with a different page size with the onspaces utility, the information specified is automatically appended to the ONCONFIG file and new values are specified using the BUFFERPOOL keyword.

You cannot change the values by editing the onconfig file. If you need to resize or delete an existing buffer pool, you must restart the database server and then run onparams again.

Buffer pools that are added while the database server is running go into virtual memory, not into resident memory. Only those buffer pool entries that are specified in the ONCONFIG file at startup go into resident memory, depending on the availability of the memory you are using.

Setting the PHYSBUFF Parameter

It is recommended to set the PHYSBUFF configuration parameter to at least 128 kilobytes. If the database server is configured to use RTO_SERVER_RESTART, set the PHYSBUFF configuration parameter to at least 512 kilobytes. Setting PHYSBUFF to a lower value could impact transaction performance and will result in a performance warning during server initialization.

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 three 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 the *Administrator's Guide for Informix Dynamic Server* for more information.

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

Working With Oracle Connectivity

This section discusses:

- Oracle Net Services
- PeopleSoft servers and the Oracle connection string.
- Open cursors.

Oracle Net Services

Oracle Net Services (Oracle Database 11g or higher) offers peer-to-peer connectivity and a multi-protocol interchange (MPIC). The product is installed as multiple elements including:

- Transparent Network Substrate (TNS).
- Oracle Protocol Adapter.
- Multi-protocol interchange (MPIC).

Oracle Net Services uses the configuration files SQLNET.ORA and TNSNAMES.ORA, which can be created using a system editor, or with the Oracle Net Configuration Assistant.

PeopleSoft Servers and the Oracle Connection String

The format of the Oracle connect string used to connect to the database is *userid*/*password@service_name* for all PeopleSoft processes, including online, batch, and application server processes.

This makes setup and configuration easy for platform configurations that can support PeopleSoft batch server processes or application server processes. However, performance for the batch processes and application server processes on a server that also functions as the database server is slightly degraded, due to the overhead involved in routing through SQL*NET.

PeopleSoft provides a configuration parameter, UseLocalOracleDB, for you to indicate which connect string to use. You set the parameter while configuring the application server or the Process Scheduler in the Database Options section.

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]:
```

Following are descriptions of the Database Options parameters:

Parameter	***Description***
UseLocalOracleDB	Indicates if the PeopleSoft database that you are connecting to is in a Local Oracle SID. The default is *0*, meaning that the database you are connecting to is remote. The resulting connect string is in the following format: userid/password@service_name. If you set this to *1*, then the system used 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 running the application server or Process Scheduler servers. This enables Oracle to clean up any orphaned database 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 to which you want the PeopleSoft processes to connect. 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 three-tier connections. This feature is covered later in this documentation. See Monitoring PeopleSoft 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 the system 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 the system 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 operating system environment variable thus overriding the current ORACLE_SID environment variable.

Related Links

PeopleCode Language Reference

Open Cursors

The minimum number of OPEN_CURSORS required for PeopleSoft applications on an Oracle database is 1000.

Monitoring PeopleSoft Database Connections

This section provides an overview of PeopleSoft database connections and discusses how to:

- Enable database connection monitoring.
- Track PeopleSoft database connections by PeopleSoft User ID.
- Monitor MODULE and ACTION.

Understanding PeopleSoft Database Connections

PeopleTools provides the ability to monitor connections to the database server from Windows workstations (two-tier and three-tier connections) and browser connections. Some possible uses of monitoring database connections include system-wide troubleshooting, performance monitoring, "chargeback" accounting, and security audits for your system.

Administrators can obtain specific information regarding the user and the associated transaction when a user is connected to the database. A PeopleSoft system has many clients and user sessions connecting to one application server (directly, or indirectly through the web server), with only the application server maintaining connections to the database server.

Suppose one of your users has executed an extremely inefficient query that severely impacts the rest of the system. A Database Administrator would want to identify that user and take appropriate action. However, without an ability to monitor users you would probably have to terminate the physical connection, which would mean dropping the connection between the application server and the database server, which could potentially affect hundreds of users.

However, while only the application server maintains the actual database connection, PeopleTools records various information associated with each user connection so that client information can be monitored. Monitoring client information enables an administrator to collect information from two-tier connections, three-tier connections, and browser connections alike.

Associated with each connection and transaction is the following set of user information:

- PeopleSoft user ID
- OSUserName
- MachineName
- AppServerDomainName
- ProgramExecutable

This information allows the system to associate activity on the database server with a particular workstation and user. This information is stored in the CLIENT_INFO column of the V$SESSION dynamic view.

Administrators are also often interested in compiling performance metrics based on the system usage per application. For this type of monitoring the PeopleSoft system populates the MODULE and ACTION fields of the V$SESSION dynamic view.

Oracle products, such as Oracle Enterprise Manager and Oracle Audit Vault use information stored in the CLIENT_IDENTIFIER column of V$SESSION.

Enabling Database Connection Monitoring

Database monitoring is always enabled for:

- COBOL programs.
- SQR programs.
- Processes run through Process Scheduler.
- Two-tier Windows workstation connections.

For connections handled by the application server (browser and three-tier Windows connections) the PeopleSoft systems administrator has the option to enable this feature by setting the EnableDBMonitoring parameter to '1' in PSADMIN or the application server configuration file (PSAPPSRV.CFG).

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 Windows client connections.
- Application server process connections.
- Three-tier Windows client connections.
- Browser (PIA) connections.
- Process Scheduler connections.
- SQR connections.
- COBOL connections
- Windows 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 per connection type. The following sections describe the information returned for various scenarios.

Legend

ID	*Description*
JZARATE (uppercase)	NETWORK login ID for Windows workstation.
JZARATE123199	Windows client MACHINENAME.
TMJONES (uppercase)	NETWORK login ID for Windows workstation.
TMJONES110299	Windows client MACHINENAME.
JRSMITH (uppercase)	NETWORK login ID for Windows workstation.
JRSMITH031198	Windows client MACHINENAME.
PREILLY (uppercase)	NETWORK login ID for Windows workstation.
PREILLY060499	Windows client MACHINENAME.
PT844P01	PeopleSoft schema (PS SYSADM ID or Access ID).
PT81	Tuxedo domain name.
PTDMO, VP1, and PS	PeopleSoft user IDs used to signon to the database from the various clients.
oracle (lower case)	Owner ID of all of the Oracle processes.
certora (lowercase)	UNIX login ID of the PeopleSoft administrator starting 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 are used to monitor the client info for 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 is 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 application server processes connecting to the database as user ID 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 Connections – Windows Workstations

For the three-tier connections, you can retrieve the following client information:

```
"%oprid%,%osusername%,%machinename%,%tuxedo_domain%,%executable%,""
```

When the three-tier workstation is connected, then you should see the application server process that is executing the transaction for the client. For example, the PSAPPSRV server process handles the majority of the requests. 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 workstation JRSMITH031198, signing on as PSOFT with a user ID of 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.
```

Browser Connections – (PIA)

For browser connections (PIA connections), you can retrieve the following client information:

```
"%oprid%,%osusername%,%machinename%,%tuxedo_domain%,%executable%,""
```

When the user is connected, you should see the application server process that is executing the transaction for the browser. For example, the PSAPPSRV handles the large queries executed by user connections. Let's assume for this example that the PSAPPSRV is processing the current client request.

Adding to what was previously displayed, this is a 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 windows connection and a PIA browser connection.

```
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 with a user ID of 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,,psp⇒
rcsrv,
15395    certora PT844P01 PSAPPSRV@st-sun01 (TNS V1-V3) PTDMO,,PREILLY060499,PT81,P⇒
SAPSRV,
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 an SQR report run from the workstation JZARATE123199, submitted from the user ID 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,,psp⇒
rcsrv,
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 program PTPTEDIT, run from the workstation JZARATE123199, submitted from PeopleSoft user ID 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,,psp⇒
rcsrv,
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.
```


Windows Workstation and Browser Connections Multithreading Through the Application Server

For multithreaded connections, you can retrieve the following client information:

```
"%oprid%,%osusername%,%machinename%,%tuxedo_domain%,%executable%,""
```

The application server multithreads incoming three-tier Windows or PIA browser connections through the application server processes already connected to the database. The next several examples illustrate a continual changing of the monitoring information displayed through the application server "thread" based on user activity and incoming requests.

Adding to what was previously displayed, accessing the database again from the three-tier Windows workstation JRSMITH031198 reflects a change in the user ID VP1 and client machine name for the both application server processes.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO                ⇒

-------- ------- -------- ------------------------------ --------------------------⇒
----------
15387    TMJONES PT844P01 pside.exe                      PS,TMJONES,TMJONES110299,,s⇒
ide.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,,psp⇒
rcsrv,
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 browser on machine PREILLY060499 illustrates a change in the user ID PTDMO and client machine name 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,,psp⇒
rcsrv,
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 Windows workstation JRSMITH031198 reflects a change in the user ID VP1 and client machine name 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,,psp⇒
rcsrv,
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, the following illustrates accessing the database from the three-tier Windows workstation JRSMITH031198 executing a functional process that requires use of all of the application server processes. This is reflected in the change in the user ID VP1, and client machine name for all of the application server processes.

Note: Because the SQR program has completed running, there is no SQR information 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,,psp⇒
rcsrv,
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,,psp⇒
rcsrv,
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.
```

Monitoring PeopleSoft MODULE and ACTION Information

In addition to the CLIENT_INFO field, PeopleTools also populates the MODULE and ACTION fields of the V$SESSION and V$SQL dynamic views. This provides increased monitoring capabilities if you use Oracle performance monitoring utilities, including:

- Oracle Enterprise Manager
- Oracle Database Resource Manager
- Oracle Automatic Workload Repository

Note: You must set the EnableAEMonitoring configuration setting to 1 to populate the MODULE and ACTION fields in V$SESSION and V$SQL views. By default, EnableAEMonitoring is set to 0 (disabled). To change the EnableAEMonitoring setting for an Application Server domain or a Process Scheduler domain, use the PSADMIN utility or manually modify the PSPRCS.CFG configuration file or the PSAPPSRV.CFG configuration file, then restart the respective servers.

By monitoring MODULE and ACTION values you can:

- Provide more specific PeopleSoft information for several Oracle performance monitoring tools.
- View and analyze performance and system resource usage for selected PeopleSoft application modules.
- Write custom SQL to aggregate PeopleSoft performance and system usage information based on the MODULE, ACTION, and CLIENT_INFO values.

Depending on the type of connection, or the PeopleTools feature being used, the system populates the MODULE and ACTION fields with the information described in the following table.

PeopleSoft Technology	***MODULE Value***	***ACTION Value***
application server (browser connections)	PeopleSoft component name	PeopleSoft page name
Integration Broker	service operation name	PeopleCode event
Application Engine	'PSAE'.<AE program name>.<PID>	Application Engine program name, section name, step, and type.

Each SQL statement in V$SQL has a MODULE field populated based on the MODULE field of the session that first submitted the SQL. You can write additional SQL to obtain valuable performance information aggregated based on the values of these fields.

Keeping in mind that the usage of the MODULE and ACTION values is intended mainly to be used within the context of Oracle performance monitoring utilities, to become familiar with the type of information provided you can issue SQL queries, such as the following samples:

```
select module, action, client_info from v$session;
```


or

```
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,
   substr(s.module,1,48) module,
   substr(s.action,1,32) action
from v$session s, v$process p
where s.paddr=p.addr
and s.osuser is not null
order by s.osuser;
```

Exposing PeopleSoft User Information Through the CLIENT_IDENTIFIER Column

Additional monitoring information was included enhancing the availability of PeopleSoft user information in Oracle products like Oracle Audit Vault and Oracle Enterprise Manager. PeopleSoft user ID information is also stored in the CLIENT_IDENTIFIER column of the V$SESSION table.

The CLIENT_IDENTIFIER column contains only the user ID, whereas the CLIENT_INFO column also contains the user ID value, but it is typically accompanied by other user information, like machine name for example. In some cases, a monitoring application may only need the user ID information. To get this information from the CLIENT_INFO column would require programmatic transformation and parsing of the CLIENT_INFO string. Displaying only the user ID in the CLIENT_IDENTIFIER column, simplifies the retrieval of the user ID by products like Oracle Audit Vault and Oracle Enterprise Manager. No further transformation or parsing of the string is required.

Database administrators can also retrieve the information directly from the database with queries similar to the following:

```
SQL> select module, client_identifier, client_info from v$session where module = 'p⇒
side.exe';

MODULE           CLIENT_IDENTIFIER       CLIENT_INFO
---------------  ----------------------  ------------------------------
pside.exe        QEDMO                   QEDMO,bng2,BENG-PC,,pside.exe,
```

The example above displays the User ID information in both the CLIENT_INFO and the CLIENT_IDENTIFIER columns. The latter can be used by Oracle Audit Vault. The user ID information can be retrieved from the following connection types:

- two-tier connections.
- three-tier connections.
- programs run through Process Scheduler.

The following sections provide sample queries and results.

Example: Working with CLIENT_IDENTIFIER Information and Three-Tier Connections

The following query displays the user ID information associated with a three-tier connections:

```
SQL> select module, client_identifier, client_info from v$session where client_iden⇒
tifier like 'QEDMO%;
```



```
MODULE                                           CLIENT_IDENTIFIER           CLIENT_IN⇒
FO
------------------------------------------------ --------------------------- ---------⇒
----------------------
PSAPPSRV@sp-lnx07.peoplesoft.com (TNS V1-V3)  QEDMO                       QEDMO,,10⇒
.138.230.162,FS850U02,PSAPPSRV,
```

In this example the user ID information is available under CLIENT_INTO and CLIENT_IDENTIFIER, however CLIENT_IDENTIFIER only stores the user ID information while CLIENT_INFO stores other information, like the client connection details.

Example: Working with CLIENT_IDENTIFIER Information and Process Scheduler

It is also possible to monitor the user ID for programs running through Process Scheduler. In the following example an Application Engine program ran through Process Scheduler using the user ID QESS. By running the following query, it is possible to display the user ID information.

```
SELECT module, client_identifier, client_info
FROM v$session
WHERE client_identifier like 'QESS%';

MODULE          CLIENT_IDENTIFIER               CLIENT_INFO
--------------  ------------------------------  ----------------------------------⇒
--
PSAE            QESS                            QESS,,sp-lnx07.peoplesoft.com,,PSAE⇒
SRV,
```

Converting Descending Indexes

As of PeopleTools release 8.54, to optimize performance, descending indexes are no longer supported. If you are upgrading from an earlier version of PeopleTools, you must convert any descending indexes to ascending indexes by dropping descending indexes and recreating them as ascending indexes.

The following scripts are provided in <*PS_HOME*>\SCRIPTS\ORA\ to accomplish this task:

- postupgcreatedescindexes.sql
- postupgdropdescindexes.sql

To convert descending indexes to ascending indexes, complete these steps:

1. Connect as AccessId, run postupgdropdescindexes.sql
2. Connect as AccessId, run postupgcreatedescindexes.sql
3. Connect as AccessId, run psdropdescindexes.sql
4. Connect as SYSDBA and run alter system set "_ignore_desc_in_index"=true;
5. Connect as AccessId, run pscreatedescindexes.sql
6. Connect as SYSDBA and run alter system set "_ignore_desc_in_index"=false;

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 page plus the allocations specified for each individual Application Engine program.

Related Links

Application Engine

Using Locally Managed Tablespaces

PeopleSoft supports the latest Oracle locally managed tablespace (LMT) syntax to control segment space allocation. A Locally Managed Tablespace (LMT) is a tablespace that manages its own extents maintaining a bitmap in each data file to keep track of the free or used status of blocks in that data file. Each bit in the bitmap corresponds to a block or a group of blocks. When the extents are allocated or freed for reuse, Oracle changes the bitmap values to show the new status of the blocks. These changes do not generate rollback information because they do not update tables in the data dictionary (except for tablespace quota information), unlike the default method of Dictionary - Managed Tablespaces.

Benefits of using LMTs include:

- Locally managed tablespaces do not record free space in the data dictionary, it reduces contention on these tables.
- Local management of extents automatically tracks adjacent free space, eliminating the need to coalesce free extents.
- Avoids recursive space management operations, which can occur in dictionary-managed tablespaces if consuming or releasing space in an extent results in another operation that consumes or releases space in a rollback segment or data dictionary table.
- Sizes of extents that are managed locally can be determined automatically by the system. Alternatively, all extents can have the same size in a locally managed tablespace.

- Changes to the extent bitmaps do not generate rollback information because they do not update tables in the data dictionary (except for special cases such as tablespace quota information).
- Reduced fragmentation No coalescing required.

Specifically, the following scripts have been modified to use this syntax: UTLSPACE.SQL, PTUPGDDL.SQL, and xxDDL.SQL. (Where 'xx' is the product code).

For example:

```
CREATE TABLESPACE PSINDEX DATAFILE '/u04/oradata/<SID>/psindex.dbf' SIZE 64M
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
SEGMENT SPACE MANAGEMENT AUTO;
```

The following guidelines intend to help you determine which tables to migrate to the appropriate ‘LARGE’ tablespaces based on table size during the move to production. If you change tablespace assignments, you first need to run SETASPACE.SQR to synchronize the PeopleSoft metadata with the changes made to the Oracle catalog with respect to any new table space assignments. Also we recommend that you use LMTs with AUTOALLOCATE and ASSM for all objects.

The following is an example of a large tablespace:

```
CREATE TABLESPACE PSLARGE DATAFILE '/u04/oradata/<SID>/pslarge.dbf' SIZE 64M
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
SEGMENT SPACE MANAGEMENT AUTO;
```

The following is an example of a non-large tablespace:

```
CREATE TABLESPACE PSSMALL DATAFILE '/u04/oradata/<SID>/pssmall.dbf' SIZE 64M
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
SEGMENT SPACE MANAGEMENT AUTO;
```

Maintaining Partition Definitions

This section provides an overview of partition management, defines partitioning terms, and describes how to:

- Establish partitioning definitions.
- Apply and Maintain partitioning DDL.
- Migrate partitioning.

Understanding Partition Management

For the Oracle platform, PeopleTools provides several pages, dialog boxes, and record definitions that enable you to establish and maintain table and index partition definitions within your PeopleSoft database . Partitioning subdivides tables and indexes into smaller pieces, enabling the database to access and manage the partitioned objects at a finer level of granularity. This provides more efficiency for administration with faster backups, for example, and better transaction performance, with queries being able to isolate the relevant data more quickly through partitions. SQL performance using partitioned tables or indexes may improve by several orders of magnitude.

Partitioning is transparent to Peoplesoft applications; no changes are required to underlying APIs to utilize partitioning.

Prerequisites

Tablespaces are required for partitioning definitions, and must be established before you can implement partitioning. If your PeopleSoft application includes partition definitions, the scripts for the required tablespaces are delivered, and must be run after installation. See your application documentation for more details.

References

For additional information about partitioning, see the following Oracle documentation on the Oracle Technology Network website:

Oracle Partitioning Overview.

Partitioning Terminology

The following table defines partitioning-related terms, and describes the types of partitioning that are supported.

Partitioning

Partitioning enables you to decompose very large tables and indexes into smaller and more manageable pieces called partitions. Each partition is an independent object with its own name and optionally its own storage characteristics. From the perspective of an application, only one schema object exists. DML statements require no modification to access partitioned tables. Partitioning is useful for many different types of database applications, particularly those that manage large volumes of data. Benefits include:

- Increased availability

  The unavailability of a partition does not entail the unavailability of the object. The query optimizer automatically removes unreferenced partitions from the query plan so queries are not affected when the partitions are unavailable.

- Easier administration of schema objects

  A partitioned object has pieces that can be managed either collectively or individually. DDL statements can manipulate partitions rather than entire tables or indexes. Thus, you can break up resource-intensive tasks such as rebuilding an index or table.

- Reduced contention for shared resources in OLTP systems

  In some OLTP systems, partitions can decrease contention for a shared resource. For example, DML is distributed over many segments rather than one segment.

- Enhanced query performance in data warehouses

In a data warehouse, partitioning can speed processing of ad hoc queries. For example, a sales table containing a million rows can be partitioned by quarter.

Partition Key

The partition key consists of one or more columns that determine the partition where each row is stored. Oracle automatically directs insert, update, and delete operations to the appropriate partition with the partitioning key. Each row in a partitioned table is unambiguously assigned to a single partition.

Partitioned Tables and Indexes

You can partition tables and indexes. Partitioning helps to support very large tables and indexes by enabling you to divide the tables and indexes into smaller and more manageable pieces called partitions. SQL queries and DML statements do not have to be modified to access partitioned tables and indexes. Partitioning is transparent to the application.

Hash Partitioning

In hash partitioning, the database maps rows to partitions based on a hashing algorithm that the database applies to the user-specified partitioning key. The destination of a row is determined by the internal hash function applied to the row by the database. The hashing algorithm is designed to evenly distribute rows across devices so that each partition contains about the same number of rows. Hash partitioning is useful for dividing large tables to increase manageability. Instead of one large table to manage, you have several smaller pieces. The loss of a single hash partition does not affect the remaining partitions and can be recovered independently. Hash partitioning is also useful in OLTP systems with high update contention. For example, a segment is divided into several pieces, each of which is updated, instead of a single segment that experiences contention.

List Partitioning

In list partitioning, the database uses a list of discrete values as the partition key for each partition. You can use list partitioning to control how individual rows map to specific partitions. By using lists, you can group and organize related sets of data when the key used to identify them is not conveniently ordered.

Range Partitioning

In range partitioning, the database maps rows to partitions based on ranges of values of the partitioning key. Range partitioning is the most common type of partitioning and is often used with dates.

Composite Partitioning/Sub Partitioning

Composite partitioning is a partitioning technique that combines some of the other partitioning methods. A table is initially partitioned by the first data distribution method and then each partition is sub-partitioned by the second data distribution method. All sub partitions for a given partition represent a logical subset of the data.

Composite partitioning supports historical operations, such as adding new range partitions, but also provides higher degrees of potential partition pruning and finer granularity of data placement through sub partitioning.

The following composite partitions are supported in Oracle:

- Range-hash partitioning - introduced in Oracle 8i
- Range-list partitioning - introduced in Oracle 9i
- Range-range partitioning - introduced in Oracle 11g
- List-range partitioning was introduced in Oracle 11g
- List-hash partitioning was introduced in Oracle 11g
- List-list partitioning was introduced in Oracle 11g
- Interval-range partitioning was introduced in Oracle 11g
- Interval-list partitioning was introduced in Oracle 11g
- Interval-hash partitioning was introduced in Oracle 11g

Partition Pruning

After partitions are defined, certain operations become more efficient. For example, for some queries, the database can generate query results by accessing only a subset of partitions, rather than the entire table. This technique (called partition pruning) can provide order-of-magnitude gains in improved performance.

Partition pruning is the simplest and also the most substantial means to improve performance using partitioning. Partition pruning can often improve query performance by several orders of magnitude. For example, suppose an application contains an Orders table containing a historical record of orders, and that this table has been partitioned by week. A query requesting orders for a single week would only access a single partition of the Orders table. If the Orders table had 2 years of historical data, then this query would access one partition instead of 104 partitions. This query could potentially execute 100 times faster simply because of partition pruning.

Partition pruning works with all of Oracle performance features. Oracle uses partition pruning with any indexing or join technique, or parallel access method. In addition, data management operations can take place at the partition level, rather than on the entire table. This results in reduced times for operations such as data loads; index creation and rebuilding; and backup and recovery.

Partition-Wise Joins

Partition-wise joins reduce query response time by minimizing the amount of data exchanged among parallel execution servers when joins execute in parallel. Partition-wise joins can be

applied when two tables are being joined and both tables are partitioned on the join key, or when a reference partitioned table is joined with its parent table. Partition-wise joins break a large join into smaller joins that occur between each of the partitions, completing the overall join in less time. This significantly reduces response time and improves the use of both CPU and memory resources for serial and parallel execution. In Oracle Real Application Clusters (Oracle RAC) environments, partition-wise joins also avoid or at least limit the data traffic over the interconnect, which is the key to achieving good scalability for massive join operations.

Partitioned Indexes

A partitioned index is an index that, like a partitioned table, has been decomposed into smaller and more manageable pieces. Just like partitioned tables, partitioned indexes improve manageability, availability, performance, and scalability. They can either be partitioned independently (global indexes) or automatically linked to a table's partitioning method (local indexes). In general, global indexes should be used for OLTP applications and local indexes for data warehousing or decision support systems (DSS) applications.

Local Partitioned Indexes

A local index is an index on a partitioned table that is coupled with the underlying partitioned table, 'inheriting' the partitioning strategy from the table. Consequently, each partition of a local index corresponds to one - and only one - partition of the underlying table. The coupling enables optimized partition maintenance; for example, when a table partition is dropped, Oracle simply has to drop the corresponding index partition as well. No costly index maintenance is required. Local indexes are most common in data warehousing environments.

Local partitioned indexes are easier to manage than other types of partitioned indexes. They also offer greater availability and are common in DSS environments. The reason for this is equipartitioning: each partition of a local index is associated with exactly one partition of the table. This functionality enables Oracle to automatically keep the index partitions synchronized with the table partitions, and makes each table-index pair independent. Any actions that make one partition's data invalid or unavailable only affect a single partition.

Global Partitioned Indexes

A global partitioned index is an index on a partitioned or non-partitioned table that is partitioned using a different partitioning-key or partitioning strategy than the table. Global-partitioned indexes can be partitioned using range or hash partitioning and are uncoupled from the underlying table. For example, a table could be range-partitioned by month and have twelve partitions, while an index on that table could be range-partitioned using a different partitioning key and have a different number of partitions. Global partitioned indexes are more common for OLTP than for data warehousing environments.

	Global range partitioned indexes are flexible in that the degree of partitioning and the partitioning key are independent from the table's partitioning method. The highest partition of a global index must have a partition bound; all of whose values are MAXVALUE. This ensures that all rows in the underlying table can be represented in the index. Global prefixed indexes can be unique or nonunique. Global hash partitioned indexes improve performance by spreading out contention when the index is monotonically growing. In other words, most of the index insertions occur only on the right edge of an index.
Global Non-Partitioned Indexes	A global non-partitioned index is essentially identical to an index on a non-partitioned table. The index structure is not partitioned and uncoupled from the underlying table. In data warehousing environments, the most common usage of global non-partitioned indexes is to enforce primary key constraints. OLTP environments on the other hand mostly rely on global non-partitioned indexes.

Limitations

Partitioning is supported only for alter tables; it is not supported for temporary tables.

The following partitioning types are not supported:

- Interval partitioning
- Reference partitioning
- Virtual column based partitioning
- System partitioning

The following compositve/sub-partitioning types are not supported:

- List-Hash
- Range-Hash
- Interval-Range
- Interval-List
- Interval-Interval

Establishing Partitioning Definitions

To establish partitioning definitions, use the Partitioning page (PPMU_GEN_DDL_PG).

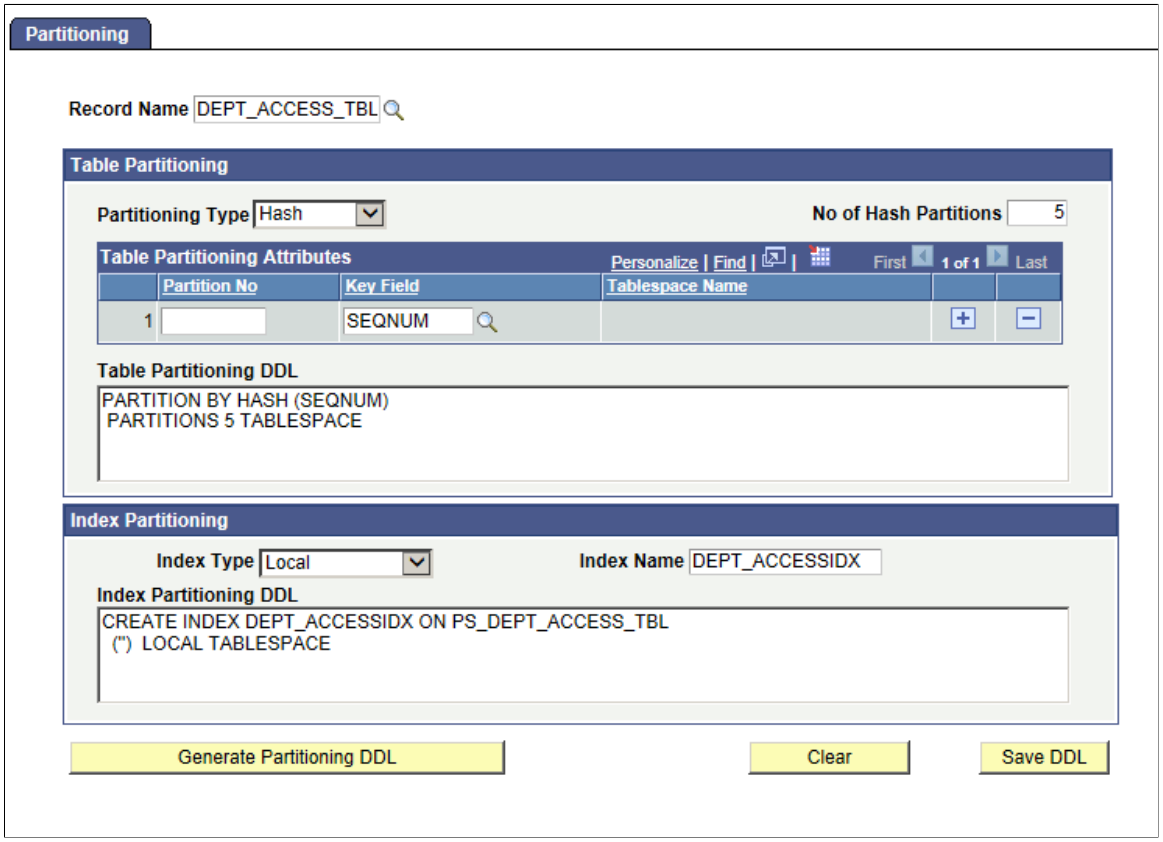
This page enables you to define partitioning parameters and generates the DDL scripts to create partitioned tables and indexes.

Navigation

PeopleTools, Utilities, Partitioning Utilities

Image: Partitioning Page

This example illustrates the fields and controls on the Partitioning page.



Complete the fields on this page to define table and/or index partitioning for a record, generate the partitioning DDL, revise the DDL (if required), and save the DDL.

Record Name Select the table record for which to specify partitioning definitions.

Table Partitioning

Complete the fields within the Table Partitioning group box to define table partitioning.

Hash, list, and range partitioning each have their own specific requirements:

- For hash partitioning: specify the number of hash partitions and the tablespace for each partition.
- For list partitioning: specify the partition name, the partition value, and the tablespace for each partition.
- For range partitioning: specify the partition name, the partition criteria, and the tablespace for each partition.

Partitioning Type

Select the partitioning type. Options are:

- *Hash*
- *List*
- *Range*

For definitions of these partitioning types, see Partitioning Terminology

Sub Partitioning Required

Select this check box if the table requires composite/sub partitioning. When you select this option, the sub-partitioning grid becomes available. This option is not available if the partitioning type is *Hash*.

No of Hash Partitions

Enter the number of hash partitions to generate. This option is available only when the partitioning type is set to hash.

Table Partitioning Attributes

For each table partition, complete the following fields (the available fields differ depending on the partitioning type):

For hash partitions: Partition No (partition number), Key Field, TableSpace Name.

For list partitions: Partition No, Key Field, Key Value, Partition Name, TableSpace Name.

For range: Partition No, Key Field, Key Value, Partition Name, TableSpace Name

Sub Partitioning Type

Select the sub partitioning type. Options are: *Hash*, *List*, and *Range*. The available options differ depending on the selected partitioning type. Only these sub-partitioning types are supported:

- Range-Range
- Range-List
- List-List
- List-Range

Templatize Sub Partition

A display-only indicator that the system uses the sub partition definition as a sub partition template for each and every partition definition.

Sub Partitioning Attributes

If the Sub -Partitioning check box is selected, use the fields within this grid to specify sub-partitioning parameters.

For each sub-partition, complete the following fields:

Partition No, Key Field, Key Value, Partition Name, TableSpace Name.

Table Partitioning DDL This edit box displays the database definition language (DDL) for the table partitioning options specified. Initially, this field is blank. The system populates this field when you click the Generate Partitioning DDL button. If this page has been previously used to define table partitioning for the record, then this box will display the currently defined DDL.

Index Partitioning

Complete the fields within the Index Partitioning group box to define index partitioning.

Note: Index partitioning is optional; a partitioned table does not require a partitioned index. If you change a table to a partitioned table, its existing indexes are generated as is with no change to the syntax. These "regular" indexes on a partitioned table are known as global non-partitioned indexes.

Index Type Select the index type. Options are:

- *Local*.

  Local partitioned indexes do not require any additional attributes; Oracle automatically keeps the index partitions synchronized with the table partitions.

- *Global*.

  Global-partitioned indexes can be partitioned using range or hash partitioning and are uncoupled from the underlying table.

For definitions of these index types, see Partitioning Terminology

Index Name Specify the index name. To avoid clashes, avoid using the regular PeopleSoft index naming convention for the index name.

Partitioning Type Select the index partitioning type. Options are:

- *Hash*
- *List*
- *Range*

Index Partitioning Attributes For global partitioning index types only. Define the index partitioning attributes using the fields in this grid. The available fields differ depending on the index partitioning type.

For hash partitions: Partition No (partition number), Key Field, TableSpace Name

For list partitions: Partition No, Key Field, Key Value, Partition Name, TableSpace Name

For range: Partition No, Key Field, Key Value, Partition Name, TableSpace Name

No of Hash Partitions Enter the number of hash partitions to generate. This option is available only when the index partitioning type is set to hash.

Index Partitioning DDL This edit box displays the database definition language (DDL) for the index partitioning options specified. Initially, this field is blank. The system populates this field when you click the Generate Partitioning DDL button. Initially, this field is blank. The system populates this field when you click the Generate Partitioning DDL button. If this page has been previously used to define index partitioning for the record, then this box will display the currently defined DDL.

If you create a partitioning index *with keys that are identical* to an existing global non-partitioned index, in order for the partitioning index to be used, you must disable generation of the previously defined global non-partitioned index using Application Designer by completing these steps:

1. In Application Designer, open the record.
2. Select Tools, Data Administration, Indexes.
3. Double-click the index name that has keys identical to the partitioned index.

   The Edit Index dialog box opens.
4. Set the Platform radio button to Some.
5. Deselect the Oracle check box.
6. Click OK.

DDL Actions

Generate Partitioning DDL Click to populate the Table Partitioning DDL and Index Partitioning DDL fields.

Clear Click to clear all values in the page.

Save DDL Click to save the DDL to the PeopleTools metadata tables.

The DDL is written to these tables:

The table partitioning DDL is stored in PS_PTTBLPARTDDL.

The index partitioning DDL is stored in PS_PTIDXPARTDDL

Only the Table and Index Partitioning DDL is stored. The Table/Index Partitioning attributes are not captured, to prevent synchronization issues with the system catalog.

To apply the partition, in Application Designer, use the Maintain Partitioning DDL dialog to apply the partitioning and build (

alter) the record. At that point the DDL from the definition is applied, and updates the Oracle database system catalog.

Applying and Maintaining Partitioning DDL

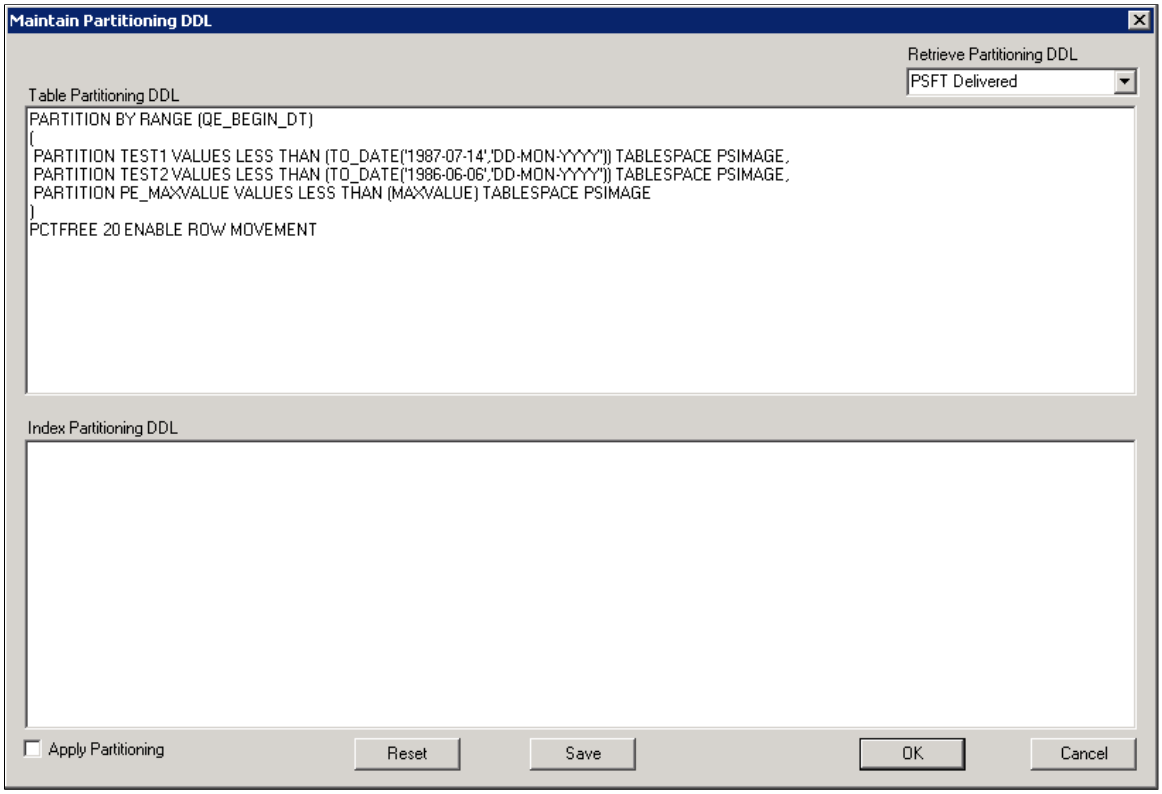
In PeopleSoft Application Designer, you review and apply partition DDL using the Maintain Partitioning DDL dialog. You can review DDL from the PeopleTools metadata tables or from the system catalog tables, and apply it. If partitioning DDL has not been defined, you can enter it directly here.

Navigation

Tools, Data Administration, Partitioning

Image: Maintain Partitioning DDL Dialog Box

This example illustrates the Maintain Partitioning DDL dialog box.



Warning! PeopleTools performs no validation on the SQL that you enter in the table or index partitioning DDL edit boxes. It is your responsibility to ensure that the DDL SQL is correct.

Retrieve Partitioning DDL Select the source of the partitioning DDL to view. Options are:

PSFT Delivered: Choose this option to view DDL from the PeopleTools Metadata tables. This is the DDL that is generated by the partitioning attributes specified on the Establishing Partitioning Definitions (PPMU_GEN_DDL_PG) in PeopleSoft Internet Architecture. The table and index partitioning

DDL is retrieved from the PS_PTTBLPARTDDL and PS_PTIDXPARTDDL tables, respectively.

System Catalog Choose this option to view DDL from the Oracle database system catalog table. This enables you to review and apply any customized partitioning that is currently defined on your Oracle database instead of the PeopleTools partitioning DDL.

Table Partitioning DDL This edit box contains the table partitioning DDL from the PeopleTools Metadata table or the system catalog, depending on the option selected in Retrieve Partitioning DDL. If no table partitioning is currently defined, then the edit box will be blank.

You can review and modify the DDL, or enter the DDL if the table has none defined.

Index Partitioning DDL This edit box contains the index partitioning DDL from the PeopleTools Metadata table or the system catalog, depending on the option selected in Retrieve Partitioning DDL. If no index partitioning is currently defined, then the edit box will be blank.

You can review and modify the DDL, or enter the DDL if the table has none defined.

Apply Partitioning Select this check box to specify that the system apply the partitioning DDL to the CREATE/ALTER TABLE DDL while altering/creating the table for this record.

The value set for Apply Partitioning (Y if selected, N if deselected) is stored in the AUXFLAGMASK field of the PSRECDEFN table against the record name and it is used during the build process to determine if partitioning needs to be applied or not.

You must build the record definition to update the PeopleTools and system catalog tables. During the build process this is reset to N.

Reset Click to clear the contents of the Table Partitioning DDL and index Partitioning DDL edit boxes.

Save Click to store the partitioning DDL in the PeopleTools metadata table.

OK Click to save and exit the dialog box.

Cancel Click to exit the dialog box without making any changes.

Note: The partitioning DDL is not automatically updated for subsequent table customizations.

Migrating Partitioning

Record and index partitioning is not migrated as part of the IDE project. If you want to migrate the partitioning metadata along with the record, you will need to complete the following tasks:

1. Create an IDE project containing the record or records on the source database.
2. Create a Data Migration project containing the partitioning metadata on the source database and copy the project to file.
3. Copy the IDE project to the target database.
4. Load the Data Migration Project on the target database.
5. Optionally, you can run a compare on the project.
6. Copy the Data Migration project to the target database.
7. In Application Designer on the target database, open the project containing the partitioning and alter the records.

To create the Data Migration project on the Source Database:

1. In PIA for the source database, select PeopleTools, Lifecycle Tools, Migrate Data, Data Migration Workbench.
2. Click the Add a New Value link.
3. Enter a project name and description.
4. Select PTTBLIDXPART as the Data Set Name.
5. The Insert Data Content page will open.
6. Enter the criteria for the record or records containing portioning that you want to migrate and click Search.

   For search options, refer to "Defining Data Migration Project" (PeopleTools 8.54: Data Migration Workbench).
7. Select the records that you want to migrate from the Search results.
8. Click the Insert and Return button to insert the selected items.
9. Click OK on the message that the instances were inserted into the project.
10. Click Save
11. Click the Copy to File button.

    Before you can Copy to File, the Project Repository must be defined. Define the same project repository on both the source and target database.

    See "Managing Data Migration Project File Locations" (PeopleTools 8.54: Data Migration Workbench).

To load the Data Migration Project on the Target Database:

1. In PIA for the source database, select PeopleTools, Lifecycle Tools, Migrate Data, Data Migration Workbench.
2. Click the Load Project From File link.
3. Select the file to load and click Load.
4. If you want to compare the file:
   a. Click the Compare button.
   b. Click Run on the Compare From File page.
   c. Click OK on the Process Scheduler Request page.
   d. Click OK again to return to the Project.
   e. Click Refresh, when the compare has completed, the compare results will be displayed.

      For details on viewing compare reports see "Viewing Compare Reports" (PeopleTools 8.54: Data Migration Workbench).
5. Click Submit for Copy.
6. Click Run on the Copy From File page.
7. Click OK on the Process Scheduler Request page.
8. Click OK to return to the project.
9. Click Refresh to verify the copy completed successfully.

To Alter the records in Application Designer:

Note: The tablespace for the partitioned records must exist on the target database.

---

1. Open the project in Application Designer on the target database.
2. Open the record.
3. Select Tools, Data Administration, Partitioning.
4. Select Apply Partitioning and click Save.
5. Click OK.
6. If you partitioned on indexes, select Tools, Data Administration, Indexes.
7. Select the Index and click Edit Index DLL.
8. Select Some for platform and deselect Oracle if it is selected.
9. Click OK twice.
10. Save the record.
11. Repeat steps 2 through 10 for each record in the project that contains partitioning.

12. Select Build, Project.
13. Select Alter Tables.
14. Click the Settings button and go to the Alter tab.
15. Select Alter even if no changes and Alter by Table Rename.
16. Set your logging and script options.
17. Click OK.
18. Click Build.
19. Use your SQL tool to view the script and run it.

Using Pluggable Databases

This topic provides an overview of pluggable databases, discusses how to implement pluggable databases, and provides references to related documentation that includes detailed information on their implementation and use.

Note: The information provided in this topic is a high-level overview. For detailed information about pluggable databases and the multitenant architecture, please see your Oracle Database documentation, on the Oracle Technology Network at http://www.oracle.com/technetwork/index.html.

Understanding Pluggable Databases

Beginning with Oracle Database 12c, Oracle introduced a multitenant architecture, which enables Oracle databases to function as container databases (CDBs) that include zero, one, or more customer-created pluggable databases (PDBs). PeopleTools supports this architecture, whereby you can define PeopleSoft databases as pluggable databases during the installation process.

Mutlitennant Architecture Overview

A container is a collection of schemas, objects, and related structures in a (CDB) that appears logically to an application as a separate database. Within a CDB, each container has a unique ID and name.

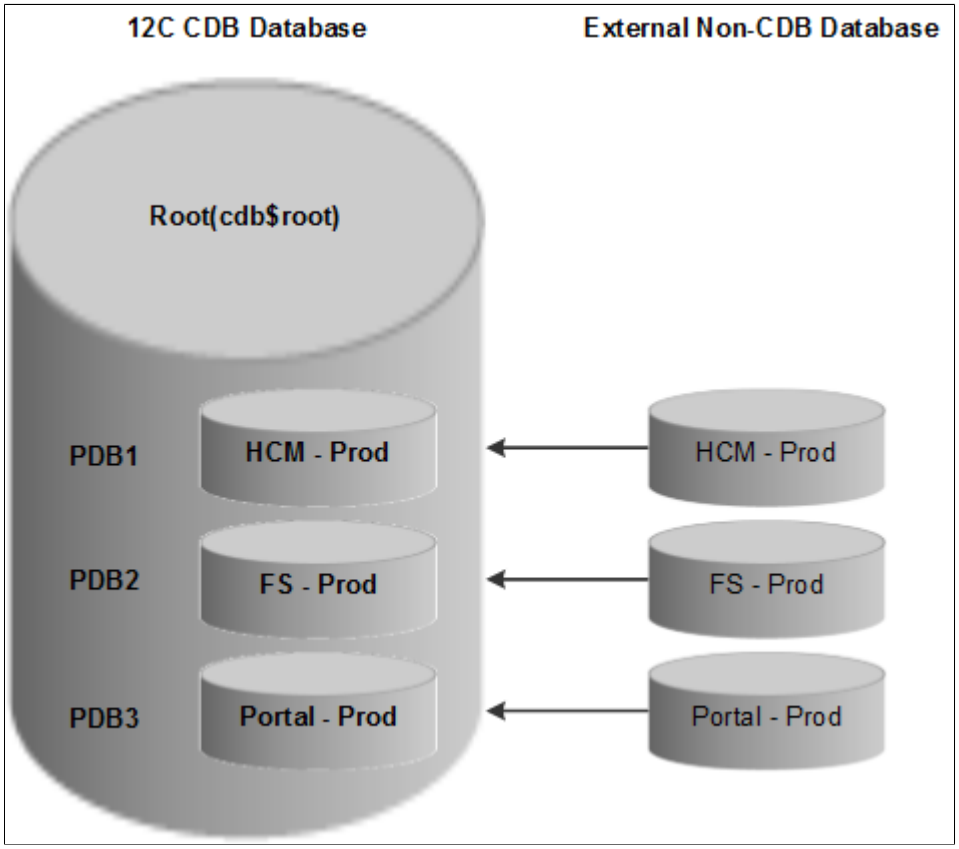
A pluggable database (PDB) is portable collection of schemas, schema objects, and non-schema objects that appears to an Oracle Net client as a non-container database. With PDBs a single database instance can host multiple PeopleSoft Application databases. A PDB is a user-created entity; no PDBs exist at creation of the CDB. You add PDBs based on your business requirements. PDBs isolate data and operations so that, from the perspective of a user or application, each PDB appears as if it were a traditional non-CDB.

The root container, also called the root, is a collection of schemas, schema objects, and nonschema objects to which all PDBs belong. The root, and every PDB are considered to be containers. Every CDB has one and only one root container, which stores the system metadata required to manage PDBs. All PDBs belong to the root.

For example, the following graphic shows how the three production databases: HCM, FS, and Portal, can be managed as 3 PDBs within a single CDB.

Image: Pluggable Database Example

This diagram shows three production databases (HCM, FS, and Portal), managed as pluggable databases.



The owners of PDBs and CDBs and their relationship to PeopleSoft users can be described as follows:

- CDB Administrator (Common User)

  A common user is a database user that has the same identity in the root and in every existing and future PDB. Every common user can connect to and perform operations within the root, and within any PDB in which it has privileges. Every common user is either Oracle-supplied or user-created. Examples of Oracle-supplied common users are SYS and SYSTEM.

  In PeopleSoft PeopleTools, the Oracle database user who belongs to the ORA_DBA group acts as a CDB Administrator.

- PDB Administrator (Local User)

  A local user is a database user that is not common and can operate only within a single PDB. The PeopleSoft PeopleTools Access ID acts as the PDB Administrator.

Advantages of Using Pluggable Databases

Large enterprises may use hundreds or thousands of databases. Often these databases run on different platforms on multiple physical servers. Because of improvements in hardware technology, especially the increase in the number of CPUs, servers are able to handle heavier workloads, and as a result, a single

database may use only a fraction of the server hardware capacity. This approach wastes both hardware and human resources. For example, 100 servers may have one database each, with each database using 10% of hardware resources and 10% of an administrator's time. A team of DBAs must manage the SGA, database files, accounts, security, and so on of each database separately, while system administrators must maintain 100 different computers.

The primary benefit of Pluggable Databases is the ability to consolidate data and code without altering existing schemas or applications. The syntax and semantics of SQL statements executed from a session connected to a non-CDB are identical when executed from a session connected to a PDB. The behavior of an application whose back end is installed in a database released before Oracle Database 12c Release is the same when its back end is installed in a PDB.

Operations that act on an entire non-CDB have the same behavior on an entire CDB; for example, when using Oracle Data Guard, and when preforming administrative tasks such as database backup and recovery. Therefore, users, administrators, and developers of a non-CDB have substantially the same experience after the database has been consolidated.

Implementing Pluggable Databases

To implement pluggable databases with your PeopleSoft system, there are specific steps that you must complete during installation. The PeopleTools installation documentation provides instructions for creating container and pluggable databases, either manually (UNIX or Microsoft Windows), or by using the Database Configuration Wizard (UNIX).

Related Documentation

For detailed information about pluggable databases and the Oracle Multitenant Architecture see:

- The Oracle Database 12C Multitenant Whitepaper at http://www.oracle.com/technetwork/database/multitenant-wp-12c-1949736.pdf
- The *Oracle Database Concepts 12c Release 1 (12.1):* "Introduction to the Multitenant Architecture".
- *Oracle Database Concepts 12c Release 1 (12.1):* "Overview of the Multitenant Architecture".

Using Materialized Views

This section provides an overview of materialized views and describes how to use materialized views with your PeopleSoft database on the Oracle platform.

Understanding Materialized Views

When building SQL views or query views on an Oracle database, you have the option of implementing a materialized view. In contrast to a standard view, which has only a logical existence, a materialized view has a physical existence, and therefore it can be indexed, analyzed, and managed like other database tables. A materialized view takes the results of complex SELECT statements and saves the datasets to disk. The results are then readily available without the need to run the SQL each time.

Using materialized views can provide significant improvements in performance. The SELECT statements that typically define materialized views often contain sizable tables, complex joins, and summary

functions that may take significant time and computing resources to complete. By running the SQL once and saving the results to a table that can be used and reused, a significant savings of CPU and memory consumption can be achieved. Like other PeopleSoft record definitions, materialized views are defined using Application Designer. The data is refreshed on a time period defined in the materialized view record definition. Materialized views are available only for Oracle databases.

Related Links

Materialized Views Concepts and Architecture

Defining Materialized Views

To define a materialized view:

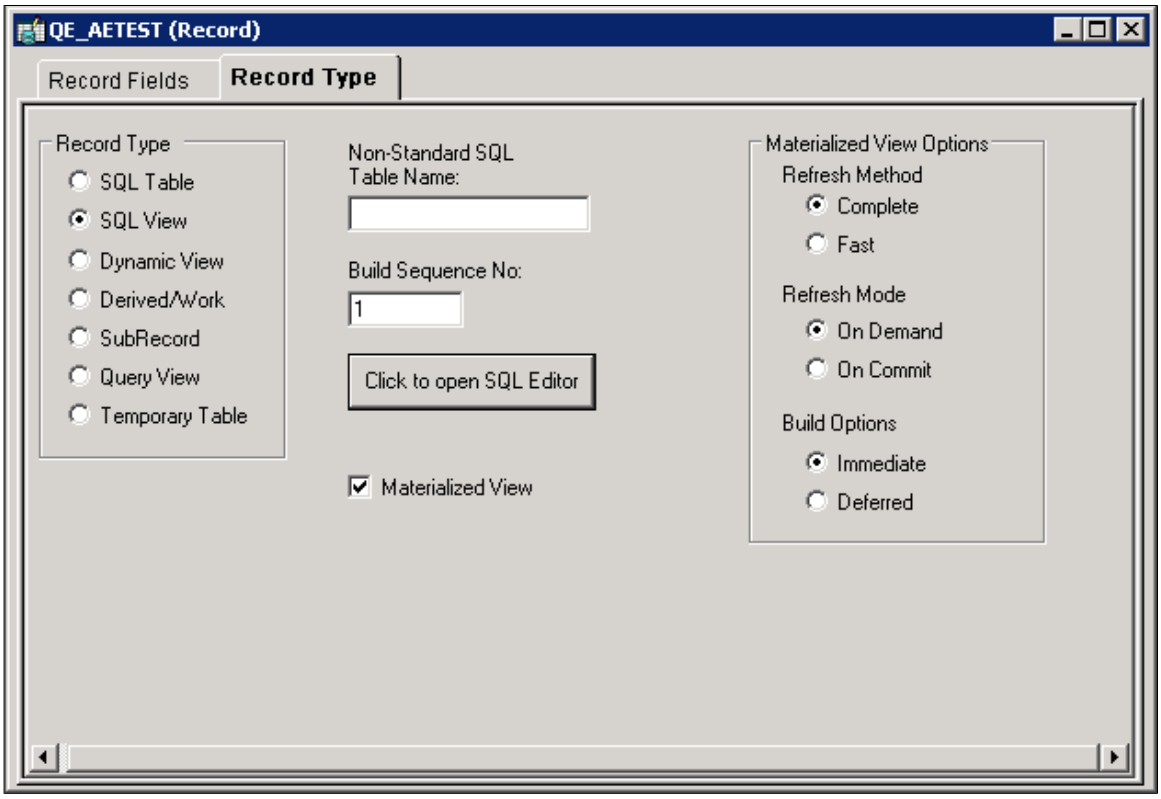
1. In PeopleSoft Application Designer, select File, New, Record to create a new record.
2. Add fields to the record, and insert the SQL query using the Query Editor.
3. Select the Record Type tab.
4. Select SQL View or Query View for the Record Type, whichever is appropriate.
5. Select the Materialized View check box.

   This check box is available only on the Oracle platform.

The Materialized View Options appear.

Image: Record Type tab for Materialized Views

This example shows the Record Type tab when creating a materialized view on the Oracle platform



6. Specify the Refresh Method. Options are:

Complete	The materialized view table will be refreshed completely. Can be done at any time; time consuming. This is the default option.
Fast	Refreshes only modified rows. The materialized view table will be refreshed incrementally when running the Application Designer Build option.

7. Specify the Refresh Mode. Options are:

On Commit	Refreshes occur whenever a commit is performed on one of the view's underlying detail table(s). Available only with single table aggregate or join-based views. This option has a performance impact since commit happens in the base table as well as the materialized view. During a commit, the Oracle system executes triggers and updates the materialized view log tables.

On Demand	Administrators refresh the view through the Materialized View Maintenance page. Can be used with all types of materialized views. This is the default option.

8. Specify the Build Options. Values are:

Immediate	The system creates the view and populates the view from the base tables. This may be time consuming depending upon the complexity of the view and data. This is the default option.
Deferred	The system creates the view but does not populate the view during the build process. An administrator must run a refresh from the Materialized View Maintenance page.

9. Save the record.

   Materialized Views are assigned automatically to the PSMATVW tablespace.

10. Build the record, selecting Create Views in the Build dialog box.

For information about creating and building records, see:

- "Understanding Record Definitions" (PeopleTools 8.54: Application Designer Developer's Guide)
- "Running the Build Process" (PeopleTools 8.54: Application Designer Developer's Guide)

Converting An Existing View to a Materialized View

To convert an existing view to a materialized view:

1. Open the record in Application Designer.
2. Select the Record Type tab.
3. Select the Materialized View check box.
4. Specify the refresh method, refresh mode, and build option.

   For detailed information about these options, see Defining Materialized Views .

5. Save and build the record.

Maintaining Materialized Views

For ongoing maintenance of materialized views, administrators can use the Materialized View Maintenance run control page (PTMAT_MAINT).

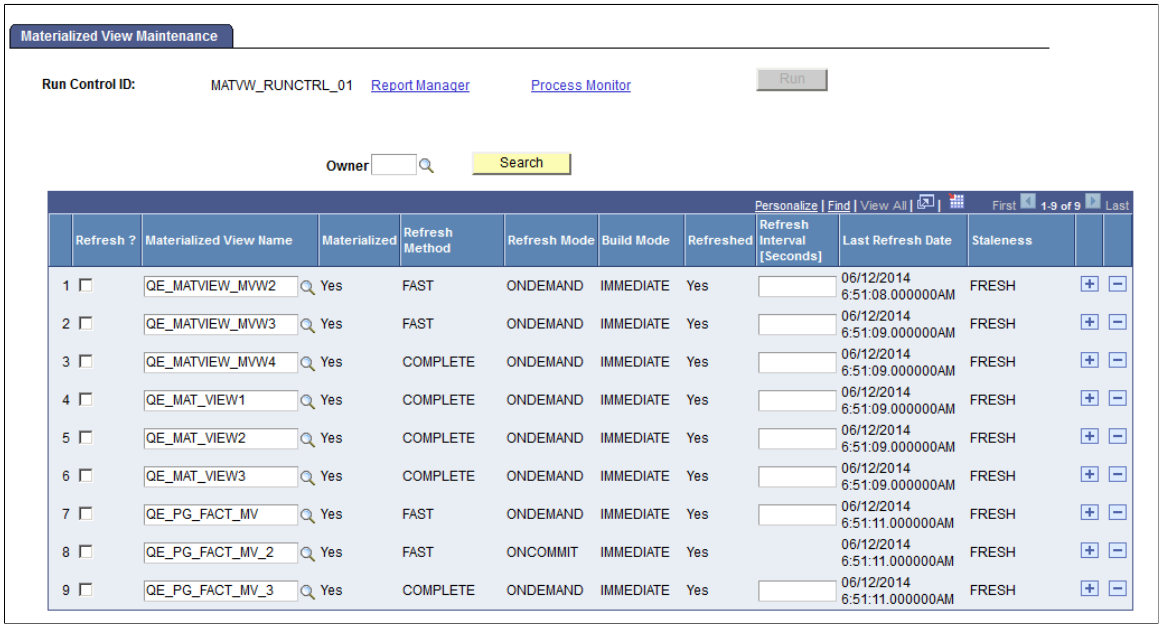
Administrators must have the Materialized View Administrator role to have permission to access this page.

Navigation

PeopleTools, Utilities, Administration, Oracle Materialized Views, Materialized View Maintenance

Image: Materialized View Maintenance Page

This image illustrates the fields and controls on the Materialized View Maintenance page.



This page contains a grid that is populated with a list of the records in the materialized views table, PSPTMATVWDEFN.

Owner To limit the grid to records that belong to a specific owner, enter the owner name and click Search.

Refresh ? Click to schedule a refresh for the record.

Materialized View Name Lists the name of the record. You can change the value in this field.

Materialized A display-only field that indicates if the record is materialized (Yes) or not (No).

Refresh Method Lists the refresh method, either complete or fast.

Refresh Mode Lists the refresh mode, either on demand or on commit.

Build Mode Lists the build mode, either immediate or deferred.

Refreshed Indicates if the record is refreshed (Yes) or not (No).

Refresh Interval The time interval in seconds for the refresh procedure to run. Enter the interval in seconds to refresh the record every *n* seconds.

Last Refresh Date List the date and time the record was last refreshed.

Staleness	Indicates if the record is stale or fresh based on the Oracle System Catalog.
Run	Click to execute the PTMATREFVW Application Engine program, to refresh the selected records.

Audits for Materialized Views

You can run SYSAUDIT to check for materialized views with the following conditions:

- MVIEWS-1

  This audit lists materialized views that are defined in Application Designer, but are not found in the database.

- MVIEWS-2

  This audit lists materialized views that are defined in the database, but are not found in Application Designer.

- MVIEWS-3

  This audit lists materialized views that are missing a related language record.

Guidelines

The following table provides guidelines for the various refresh mode/method combinations.

	ON COMMIT/FAST REFRESH	***ON DEMAND/FAST REFRESH***	***ON DEMAND/COMPLETE REFRESH***
Use Case	Pivot Grid based on lookup and transaction tables	Pivot Grid based on lookup and transaction tables	Staging tables
Data Status (stale/fresh)	Always fresh	Relatively fresh	Stale data
Refresh Cost	Refresh is transparent and automatic. Happens at commit.	Refresh is fast	Refresh is time consuming.
Commit Cost	Commit will be slow depending upon the transaction rate on base tables.	Normal commit.	Normal commit.
DML Cost	Insert to base tables will be slower by a factor of two.	Insert to base tables will be slower by a factor of two.	Normal insertion.

Refresh Mode/Method Recommendations

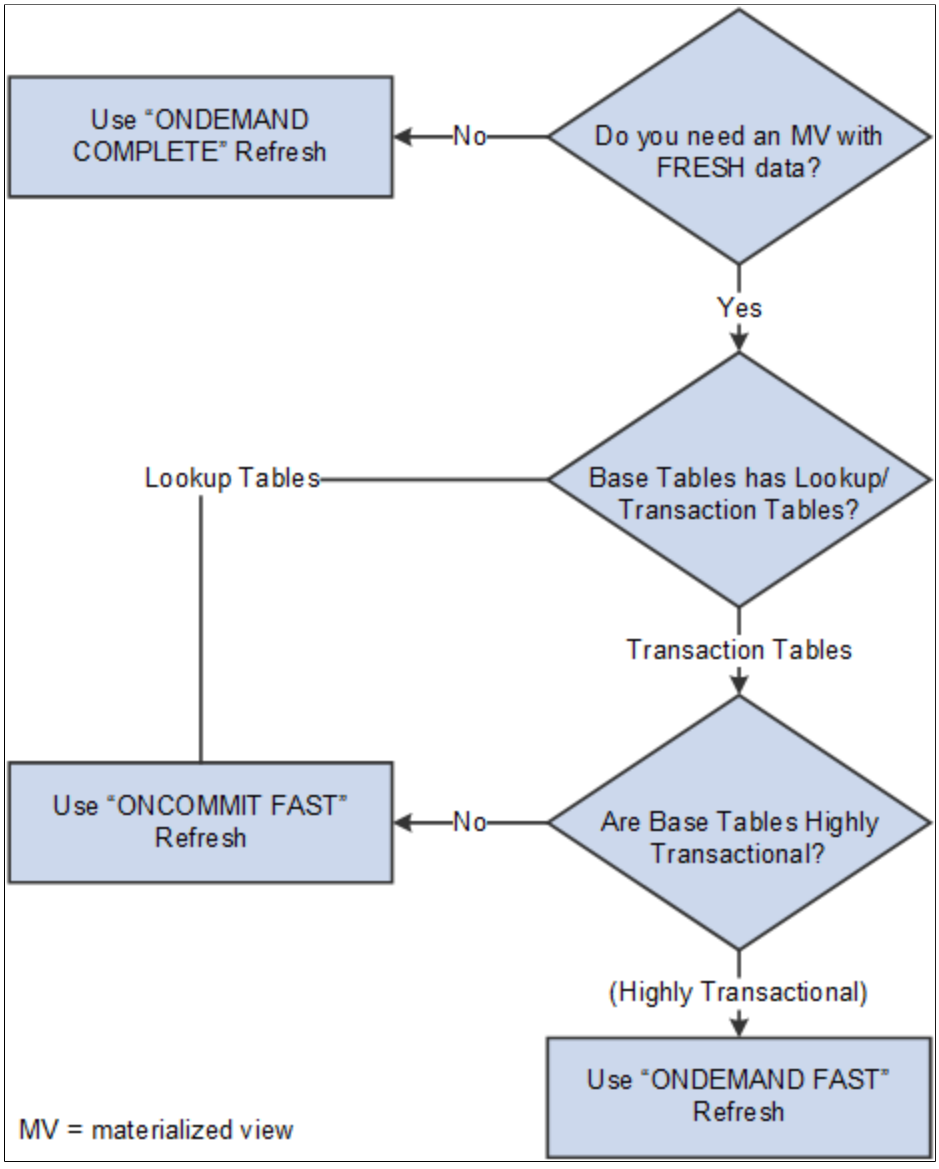
Refer to the following recommendations for determining the appropriate refresh mode/method:

- If requirements are for a materialized view with stale data, use an On Demand/Complete refresh.

- If requirements are for a materialized view with fresh data and the base tables are lookup tables, then use an On Commit/Fast refresh.
- If requirements are for a materialized view with fresh data and the base tables are not highly transactional, then use an On Commit/Fast refresh.
- If requirements are for a materialized view with fresh data and the base tables are highly transactional, then use an On Demand/Fast refresh.

Image: Refresh Mode/Method Flowchart

This graphic provides a flowchart for determining the optimal refresh mode/method.



Using Materialized Views with Oracle Golden Gate or Oracle Active Data Guard

Refresh of materialized views on an Oracle Golden Gate or Oracle Active Data Guard environment is performed in the primary database, not the standby database. The time required to synchronize the

primary database with the secondary database will depend upon the load and the environment. If any PeopleSoft applications are using materialized views on an Oracle Golden Gate or Oracle Active Data Guard environment, the data may be stale, depending upon the last refreshed date. Use the Materialized View Maintenance page to check on the status. Pivot grids are refreshed when they are initialized, as they incorporate materialized views that use On Demand/Fast refreshes, and the data that needs to be refreshed may be small.

Improving Process Performance with Global Temporary Tables

Global Temporary Tables (GTTs) are Oracle database working tables whose data persists for the span of a specific process. Multiple processes are able to use these work tables without any data collision among processes because the data is process-specific and operations on the data in one session will not affect the data in use by any other process. GTTs improve performance by automatically discarding data at the end of a session. GTTs are not logged by design, and they will not incur any logging overhead. You can configure your PeopleSoft database to use Oracle GTTs with application engine processes.

For information about GTTs, see "Understanding Global Temporary Tables" (PeopleTools 8.54: Application Engine)

Working With Oracle Consumer Groups

PeopleTools enables you to map predefined PeopleSoft resource groups to Oracle resource consumer groups that you create, specifically for use with PeopleSoft. Using Oracle Database Resource Manager features, database administrators can monitor and manage the database resource consumption of PeopleTools executables and optimize system performance.

For example, assume that occasionally long running queries run through PeopleSoft Query degrade the performance of the entire system by consuming large amounts of the available database resources. PSQRYSRV.EXE, the application server process dedicated to handling processing requests submitted by PeopleSoft Query, is mapped to the PeopleSoft Resource Group, QUERY SERVER, by default. By creating a ‘PsQuery’ consumer group in your Oracle system, you can limit the percentage of CPU processing available to PSQRYSRV.EXE. If you limit the CPU usage for PSQRYSRV.EXE to 10 percent, for example, other high-priority processing on the system will continue to have system resources available, while PSQRYSRV.EXE is limited only to its allotted 10 percent of CPU processing.

To take advantage of mapping PeopleSoft resource groups to Oracle resource groups, you need to:

- Review the delivered, predefined PeopleSoft resource groups.
- Determine areas of your PeopleSoft system where you’d like to implement control over CPU usage.
- Create the appropriate resource plan and consumer groups in your Oracle database.
- Map PeopleSoft resource groups to the consumer groups you created.

See your *Oracle Database Administrator’s Guide: “Using the Database Resource Manager”* for more information.

Reviewing PeopleSoft Resource Groups

The following table describes the delivered PeopleSoft resource groups and the PeopleTools executables they contain.

Note: How the executables are grouped is not configurable. For example, you can't further subdivide nor combine the delivered PeopleSoft resource groups.

PeopleSoft Resource Name	*Description*	*Mapped Executable(s)*
ANALYTICAL SERVER	Executables required by the PeopleSoft Analytic Calculation Engine.	PSANALYTICSRV
APPLICATION ENGINE	Executables required by the PeopleSoft Application Engine.	PSAESRV PSAE
APPLICATION SERVER	Executables required by core application server processing.	PSAPPSRV PSSAMSRV PSPPMSRV PSPRCSRV
COBOL	Executables required for running COBOL programs.	PSRUN
DATA MOVER	Executables required for running Data Mover.	PSDMTX PSDMT

PeopleSoft Resource Name	*Description*	*Mapped Executable(s)*
MISCELLANEOUS	Executables required for running various PeopleTools executables, from PeopleSoft Configuration Manager to Verity's spider program. These executables are categorized into this category because they are typically used infrequently and/or do not consume enough system resources to warrant their own resource group.	JAVAGEN MKSYD MKVDK PRCSADM PSBITEST PSBOERUN PSCBLUCVRT PSCBLUCVRTZ PSCFG PSCRCONV PSCRRUN PSCVTRPT PSDAEMON P SDOCCGI PSEMAGENTSERVICE PSEMAIL PSIDE PSMAIL PSMBSRV PSMCFLOG PSMONITORSRV PSMSFADMIN PSMSFATTACH PSMSFATTRIBUTES PSNTSRV PSNVS PSOLAP PSOSE PSPALDBG PSPALXML PSPSADM PSQED

PeopleSoft Resource Name	*Description*	*Mapped Executable(s)*
		PSREAPER PSREFRESHENGINE PSRELEASEINFO PSRENSRV PSSRCHSRV PSSVCHARNESS PSTAAT PSTRANS PSUNICONV PSUQSRV PSWATCHSRV PSXFR REAPER REGSVR32 SQLAPI TRC2API UBBGEN VSPIDER
PUB SUB	Executables required for processing and handling the Integration Broker implementation.	PSBRKDSP PSBRKHND PSSUBHND PSSUBDSP PSPUBHND PSPUBDSP PSDBGPRC PSDBGSRV PSDSTSRV PSMSGDSP PSMSGHND PSMSTPRC
QUERY SERVER	Executables required to process PeopleSoft Query requests.	PSQRYSRV

PeopleSoft Resource Name	*Description*	*Mapped Executable(s)*
QUICK SERVER	Executables required for running SQR for PeopleSoft requests.	PSQCKSRV
SQR	Executables required for running SQR for PeopleSoft requests.	PSSQR
VERITY	Executables required for running core Verity search functionality.	PSVERITYEXEC PSVERITYPIPEEXEC

Determining Where to Implement a Consumer Group

While PeopleTools delivers a set of predefined resource groups that you can map to Oracle consumer groups, you only need to create consumer groups for the resource groups where you need to introduce control of system resource usage.

For example, if COBOL and PeopleSoft Analytic Calculation Engine processing are the only areas of your PeopleSoft system that cause unwanted resource usage, then you only need to create Oracle consumer groups to map to ANALYTIC SERVER and COBOL PeopleSoft resource groups.

Creating an Oracle Resource Plan and Consumer Groups

This section describes the process of defining the resource plan, consumer groups, and plan directives to correspond to the PeopleSoft resource groups.

To create the plan, groups, and directives, you can use the tool of your choice, such as SQL Plus, Oracle SQL Developer, or the Resource Manager interface in Oracle Enterprise Manager.

Note: This section covers information specific to PeopleSoft. It does not cover all topics related to Oracle Database Resource Manager. This documentation assumes that you have read and understand the information contained in Oracle Database Administrator's Guide related to Oracle Database Resource Manager.

To create a resource plan, consumer group(s), and directives:

1. Connect to the Oracle SID containing the PeopleSoft schema.
2. Create a pending area.

```
EXECUTE DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
```

3. Create a resource plan, with a name of your choice.

   For example:

```
EXECUTE DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'PeopleSoft_plan',
COMMENT => 'Resource plan/method for PeopleSoft Users Sessions');
```

4. Create the desired consumer groups.

Create the number of consumer groups required to correspond to the PeopleSoft resource groups that need to be controlled. For this example, assume that only the resource usage of executables related to the application server and PeopleSoft Query need to be controlled.

For example:

```
EXECUTE DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP
(CONSUMER_GROUP => 'Application Server',
COMMENT => 'Resource consumer group/method for online users sessions');

EXECUTE DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'PSQuery⇒
',
COMMENT => 'Resource consumer group/method for PSQuery sessions');
```

5. Create the directives for the consumer groups you created.

   For example:

```
EXECUTE DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'PeopleSoft_plan',⇒

GROUP_OR_SUBPLAN => 'Application Server',
COMMENT => 'Applications Server sessions at level 1', CPU_P1 => 50,
CPU_P2=> 0, PARALLEL_DEGREE_LIMIT_P1 => 8);

EXECUTE DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'PeopleSoft_plan',⇒

GROUP_OR_SUBPLAN => 'PSQuery',
COMMENT => 'PSQuery sessions at level 1', CPU_P1 => 10, CPU_P2 => 0,
PARALLEL_DEGREE_LIMIT_P1 => 2);
```

6. Validate the resource plan.

```
EXECUTE DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
```

7. Submit the plan.

```
EXECUTE DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
```

8. Grant the PeopleSoft schema user (PeopleSoft Access ID) these additional Oracle privileges necessary to administer resource plans:

   GRANT_SYSTEM_PRIVILEGE ADMINSTER_RESOURCE MANAGER

```
EXECUTE
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE('<ACCESS_ID>',
'ADMINISTER_RESOURCE_MANAGER',TRUE);
```

   GRANT_SWITCH_CONSUMER_GROUP

```
EXECUTE
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP('<ACCESS_ID>',
'<CONSUMER_GROUP_NAME>', FALSE);
```

9. Enable the Database Resource Manager in your PeopleSoft SID.

   For completing this, you have two options:

   - Set RESOURCE_MANAGER_PLAN='PeopleSoft_plan’ in the init.ora file for PeopleSoft SID.

- Issue the following ALTER statement: `ALTER SYSTEM SET RESOURCE_MANAGER_PLAN='PeopleSoft_plan';`

Example: Creating PeopleSoft Resource Plan and Consumer Groups SQL Script

You can issue each SQL statement separately, or you may elect to create a single script to create the required consumer groups. The following is a sample SQL script for creating a resource plan with consumer groups for your PeopleSoft system.

```
BEGIN

DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();


DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'PeopleSoft_plan',
COMMENT => 'Resource plan/method for PeopleSoft Users Sessions');


DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Application Server',
COMMENT => 'Resource consumer group/method for online users sessions');

DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'PSQuery',
COMMENT => 'Resource consumer group/method for PSQuery sessions');

DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'PubSub',
COMMENT => 'Resource consumer group/method for PUBSUB sessions');

DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Application Engine',
COMMENT => 'Resource consumer group/method for Application Engine');

DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Cobol',
COMMENT => 'Resource consumer group/method for Cobol');


DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'PeopleSoft_plan',
GROUP_OR_SUBPLAN => 'Application Server',
COMMENT => 'Applications Server sessions at level 1', CPU_P1 => 50, CPU_P2=> 0,
 PARALLEL_DEGREE_LIMIT_P1 => 8);

DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'PeopleSoft_plan',
GROUP_OR_SUBPLAN => 'PSQuery',
COMMENT => 'PSQuery sessions at level 1', CPU_P1 => 10, CPU_P2 => 0,
PARALLEL_DEGREE_LIMIT_P1 => 2);

DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'PeopleSoft_plan',
GROUP_OR_SUBPLAN => 'PubSub',
COMMENT => 'PubSub sessions at level 1', CPU_P1 => 10, CPU_P2 => 0,
PARALLEL_DEGREE_LIMIT_P1 => 3);

DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'PeopleSoft_plan',
GROUP_OR_SUBPLAN => 'Application Engine',
COMMENT => 'Application Engine sessions at level 1', CPU_P1 => 10, CPU_P2 => 50,
 CPU_P3 => 50);

DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'PeopleSoft_plan',
GROUP_OR_SUBPLAN => 'Cobol',
COMMENT => 'Cobol sessions at level 1', CPU_P1 => 10, CPU_P2 => 50, CPU_P3 => 50);


DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA()
```

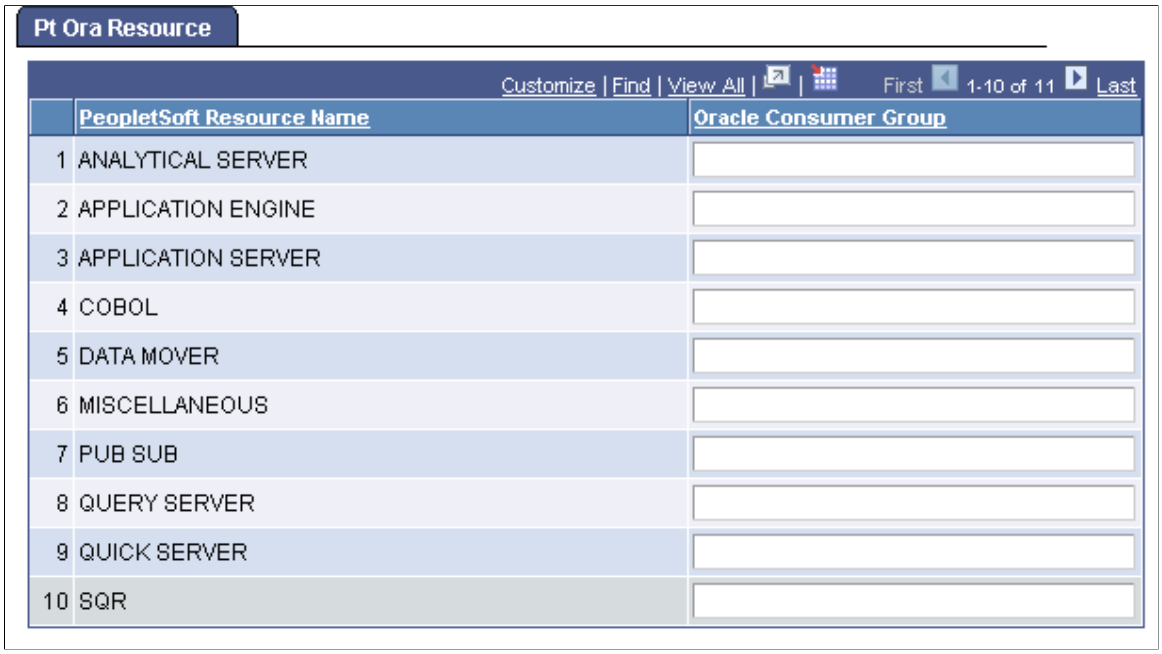
Mapping PeopleSoft Resource Groups to Oracle Consumer Groups

The Pt Ora Resource page enables you to map PeopleSoft resource groups with the Oracle consumer groups you have defined.

Select PeopleTools, Utilities, Administration, Oracle Resource Management.

Image: Pt Ora Resource page

This example illustrates the fields and controls on the Pt Ora Resource page.



PeopleSoft Resource Name

PeopleTools provides predefined PeopleSoft resource groups containing one or more PeopleTools executables. The entire set of delivered PeopleSoft resource groups appears in this list.

Oracle Consumer Group

After you have created the appropriate Oracle resource plan and consumer groups to correspond to the PeopleSoft resource groups, enter the name of the appropriate Oracle resource group in the edit box.

Enter the consumer group name exactly as it appears in the SQL you submitted to create it. For example, if the SQL you used to create the consumer group appeared as:

```
CONSUMER_GROUP => 'PSQuery'
```

Then, in the Oracle Consumer Group edit box, enter *PSQuery*.

Implementing Oracle Transparent Data Encryption

This section contains an overview and discusses how to:

- Determine fields to encrypt.

- Set up the Oracle Wallet.
- Set the encryption algorithm.
- Encrypt fields.
- Manage fields encrypted for TDE.

See also: *Oracle® Database Advanced Security Administrator's Guide.*

Understanding Transparent Data Encryption

PeopleTools enables you to implement Oracle's Transparent data encryption (TDE) feature to encrypt the columns you select, enhancing the security of your PeopleSoft applications.

Transparent data encryption (TDE) enables encryption of sensitive data in database columns as it is stored in the operating system files. It provides for secure storage and management of encryption keys in a security module located outside database, separating ordinary program functions from those that pertain to security, such as encryption.

This separation enables you to divide administration duties between DBAs and security administrators, which is a strategy that enhances security because no administrator is granted comprehensive access to data. For example, one administrator manages only the keys, while another manages only the database.

TDE is a key-based access control system enforcing authorization using these keys:

Key	***Description***
Table	For each database table that contains encrypted columns, there is one encryption key used to encrypt all the columns, regardless of the number of encrypted columns in a given table.
Master	Each table's column encryption key is, in turn, encrypted with the database server's master key. The Master key is stored in an Oracle wallet, which is part of the external security module.

TDE is transparent to the application, and no views or additional tables are required. The application logic associated with SQL and table access will continue to work without modification.

To implement this feature within your PeopleSoft application, you need to:

- Determine the fields that are candidates for TDE.
- Set up the Oracle wallet.
- Set the encryption algorithm.
- Encrypt fields.

Determining Fields to Encrypt

Examples of information that are candidates for TDE include:

- Names.

- Contact information (address, telephone number, email address, and so on).
- Credit card number.
- Passport number.
- Driver's license number.
- Age.
- Salary.
- Academic grades, scores, marks, and so on.

Note: Depending on the type of business and country in which you are running your PeopleSoft applications, there may be specific types of information, PII, that needs to be encrypted to comply with regulatory standards. For more information, see your PeopleSoft application documentation.

Managing the Oracle Wallet

With TDE, each individual table has its own table key, which is used to encrypt the selected columns in that table. Each table key is, in turn, encrypted using the TDE master key. The TDE master key is stored and protected outside the database in an Oracle Wallet, which is a container that stores authentication and signing credentials, including:

- TDE master key.
- PKI private keys.
- Certificates.
- Trusted certificates for SSL.

Encrypted table keys are placed in the data dictionary. When a user enters data into the column defined as encrypted, the Oracle database retrieves the master key from the wallet, decrypts the encryption key for that table from the data dictionary, uses that encryption key on the input value, and stores the encrypted data in the database.

Setting up the Oracle Wallet

Before implementing TDE, creating an Oracle Wallet is required.

Warning! After implementing TDE, the Oracle Wallet must be opened each time a database instance starts (or has been restarted) or else TDE will not work. If the wallet is not open, users will see error messages if they attempt to access any data encrypted using TDE.

To set up an Oracle Wallet for TDE:

1. Specify the wallet location.

   By default, the wallet is created in the directory $ORACLE_BASE/admin/$ORACLE_SID/wallet.

   So, if $ORACLE_BASE is /ds1/product/oracle and $ORACLE_SID is HRDMO, then the wallet will be stored in the directory /ds1/product/oracle/admin/HRDMO/wallet.

You can set a different directory by specifying it in the sqlnet.ora file located in $ORACLE_HOME/network/admin. For instance, if you want the wallet to be in /orawall directory, place the following lines in the sqlnet.ora file:

```
ENCRYPTION_WALLET_LOCATION =
 (SOURCE=
   (METHOD=file)
     (METHOD_DATA=
       (DIRECTORY=/orawall)))
```

Note: Oracle recommends adding this location to regular backup utility.

2. Create the wallet.

   Issue the following SQL as a user with the ALTER SYSTEM privilege, such as SYSTEM, SYS, or SYSDBA. In this example, HRMSTKEY is the password.

   ```
   alter system set encryption key
   authenticated by "HRMSTKEY";
   ```

   The preceding command creates the wallet in the specified location, sets the password of the wallet as HRMSTKEY, and opens the wallet for TDE to store and retrieve the master key.

   Note: The password is case-sensitive and must be enclosed in double quotes. The password doesn't show up in clear text in any dynamic performance views or logs.

Opening and Closing the Wallet

After you create the wallet and set the password, every time you start the database, you'll have to open the wallet explicitly, using SYS, SYSTEM, or SYSDBA accounts.

For example,

```
alter system set encryption wallet open authenticated by "HRMSTKEY";
```

To close the wallet:

```
alter system set encryption wallet close;
```

Setting the Encryption Algorithm

You set the desired encryption algorithm used by TDE on the PeopleTools Options page in the Database Encryption Algorithm edit box.

Access the PeopleTools Options page (PeopleTools, Utilities, Administration, PeopleTools Options).

The algorithms you can enter are:

- Advanced Encryption Standard algorithm with a 128-bit, 192-bit, or 256-bit key.
- Triple DES algorithm with a 168-bit key.

Specify the desired algorithm by entering one of the following values into the Database Encryption Algorithm edit box exactly as it appears below:

- AES128

- AES192
- AES256
- 3DES168

Note: You must specify an encryption algorithm to enable the Encrypt option for a field definition in Application Designer.

Encrypting Fields

You encrypt fields in Application Designer by selecting the Encrypt check box on a field definition, and then creating a table or altering an existing table.

Note: The Encrypt check box is enabled only on Oracle databases that also have an encryption algorithm specified in the Database Encryption Algorithm edit box on the PeopleTools Options page.

These PeopleSoft field types can be encrypted:

- Character
- Long Character (see note below)
- Number
- Signed number
- Date
- DateTime
- Time

Note: Long Character field types may only take advantage of TDE when the following conditions are true: the field length is greater than 0 and less than 1334 *and* the Raw Binary field attribute *is not* set.

These PeopleSoft field types *can not* be encrypted:

- Image
- Image reference
- Attachment

After you define the field to be encrypted, and either create a table or alter an existing table containing that field definition, the Build feature generates DDL SQL containing the ENCRYPT clause in the following syntax:

```
ENCRYPT using 'ALGORITHM'
```

For example,

```
ALTER TABLE  PS_AM_BI_HDR
MODIFY (CR_CARD_NBR ENCRYPT using 'AES256' NO SALT);
```


Note: If you are using Oracle Database version 10.2.0.4 or higher, the syntax includes the NOMAC parameter. For example, `ALTER TABLE PS_AM_BI_HDR MODIFY (CR_CARD_NBR ENCRYPT using 'AES192' 'NOMAC' NO SALT);`
The NOMAC parameter reduces the storage requirements and provides improved performance.

See your Oracle database documentation for more information on NOMAC.

When DDL SQL containing the ENCRYPT clause is run against the database, Oracle:

- creates a cryptographically secure encryption key for the table containing the column.
- encrypts the clear text data in the column, using the specified encryption algorithm.

Managing Fields Encrypted for TDE

This section covers these topics related to the ongoing maintenance of encrypted fields:

- Decrypting fields.
- Regenerating an encryption key.
- Upgrading TDE encrypted fields.

Decrypting Fields

If you decide that you no longer want a field encrypted for TDE, you can issue a SQL ALTER operation using the DECRYPT clause. For example, assume you wanted to decrypt the SSN field on the ACCOUNT table.

```
ALTER TABLE ACCOUNT MODIFY (SSN DECRYPT);
```

Regenerating An Encryption Key

Situations where you might consider regenerating a table encryption key include:

- You suspect a table key has been compromised.
- You want to take advantage of a different encryption algorithm.

You regenerate a table encryption key by issuing a SQL ALTER operation using the REKEY clause. For example, assume you wanted to rekey the PS_AM_BI_HDR table to take advantage of AES256.

```
ALTER TABLE PS_AM_BI_HDR REKEY using 'AES256';
```

This creates a new table key and recreates the encrypted column values using the new table key.

Upgrading TDE Encrypted Fields

All metadata field definitions are delivered with no-encryption attributes enabled. PeopleSoft applications will not deliver any metadata indicating encryption enabled for any field for an initial installation database file, project, or a PeopleTools or PeopleSoft application patch.

If you customize the field by adding TDE encryption, you need to keep track of the fields and associated record definitions and ensure that you maintain the desired encryption status through any upgrades that you perform.

See Your PeopleSoft upgrade documentation

Altering Tables With TDE Encrypted Fields

When altering tables with TDE encrypted fields using the Alter in Place option, Application Designer automatically switches the Index Creation Options selection to Recreate index only if modified even if you specifically select Recreate index if it already exists in the Build Settings dialog box.

Protecting and Managing PeopleSoft Applications with Database Vault

This section provides an overview and discusses:

- Restricting access for the access ID.
- Restricting access for PSFTDBA ID.
- Using multiple, alternate, access IDs.

Understanding Oracle Database Vault

Oracle Database Vault provides an extra layer of security that protects a database against insider security threats. One of Database Vault's key features is that it protects PeopleSoft application data from being accessed by super-privileged users, such as DBA or system administrators, but it still allows them to maintain the Oracle database.

A super-privileged user, such as a DBA, should not have access to PeopleSoft application data. Application data can include salary, identification numbers, credit card numbers, and other personal information. On the other hand, the DBA must still be able to perform database maintenance, such as back up and recovery. Database Vault allows DBAs to do their jobs, but does not allow the DBA to have access to application data.

PeopleTools has validated the use of Oracle Database Value with PeopleSoft applications. From that validation effort we've provided sample PeopleSoft DB Vault security policies. The sample policies are available on Oracle Technology Network (OTN).

These sample policies lock the database to allow all PeopleSoft application processes to access the database, while restricting any super user, like a DBA, from viewing the data using any Oracle delivered query tool. These policies illustrate a minimal usage of Database Vault functionality and may be modified or enhanced based on your specific level of required database security. The following table illustrates how the implementation of the example Database Vault policies affects the PeopleSoft Access ID and end-users, such as VP1 or PS.

User Account	*Database Vault*
SYSADM (Peoplesoft Access ID)	Before Database Vault, the Oracle DBA would use the Access ID for all database maintenance tasks, and they could view all of the data in the database. For example, a DBA might have used the PeopleSoft Access ID during all system testing to query the database when they needed to verify data in the database. Once Database Vault is enabled, the Access ID will no longer be able to access SQL*Plus, for example.
PSFTDBA (Account for DBAs)	With Database Vault enabled, the Oracle DBA responsible for applying PeopleSoft upgrades will no longer use the PeopleSoft Access ID. The DBA will now use the new PSFTDBA account to login to SQL*Plus and perform database maintenance tasks. The PSFTDBA account does not allow the DBA to run SELECT statements on the database tables, but INSERT, UPDATE, and DELETE are allowed.
General PeopleSoft user IDs (VP1, PS, and so on)	The PeopleSoft "end-user" IDs, such as VP1, are *not* affected by Database Vault. Database Vault is transparent to VP1 and other PeopleSoft end-users.

Restricting Access For the Access ID

In the PeopleSoft system, the access ID is the Oracle owner of all schema objects in a PeopleSoft database. With Database Vault you can restrict Oracle users other than the access ID from having 'SELECT' privilege on any access ID objects.

This restrictive usage is supported by using the sample PeopleSoft Database Vault security policies. When the sample PeopleSoft Database Vault security policies are implemented and Database Vault is enabled on a PeopleSoft database running on Oracle, the policies allow the access ID to do everything it currently needs to do on behalf of PeopleSoft components.

By design, all DML including SELECT DML is allowed by the access ID if the DML is issued through a "known" PeopleTools component, as defined in the sample PeopleSoft Database Vault security policies.

SELECT DML access is restricted for the access ID if not executed through a defined PeopleTools component.

SQLPlus and other ad hoc query tools are not explicitly defined in the sample policies and therefore cannot be used to issue SELECT DML against the database.

Restricting Access For PSFTDBA ID

The sample policies and scripts provide for non-access ID access to the database through the Oracle user, PSFTDBA. This user is intended to be used when you need SQLPLUS access to the system.

In order for DBAs to perform system maintenance, upgrade tasks, and so on, the sample policy scripts create the PSFTDBA account. With this account the following actions are allowed on database tables:

- INSERT
- UPDATE

- DELETE

The sample PeopleSoft Database Vault security policies restrict the PSFTDBA ID from performing a SELECT against the access ID's objects. If you use the PSFTDBA account to run a SELECT statement, an error message similar to the following appears:

```
sp-hp15:$ sqlplus PSFTDBA/PSFTDBA@Q8501123

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 9 10:45:36 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - 64bit Production
With the Partitioning, Oracle Label Security, Oracle Database Vault and
Real Application Testing options

SQL> select * from Q8501123.PSSTATUS;
select * from Q8501123.PSSTATUS
                       *
ERROR at line 1:
ORA-01031: insufficient privileges
```

The PSFTDBA ID is designed so that your DBA's use it rather than the access ID to increase security when performing database maintenance. When performing some tasks, keep in mind that PSFTDBA does not have sufficient access to the database to perform all PeopleSoft maintenance tasks, such as all upgrade tasks.

For example, when running SQRs from the workstation, the PSFTDBA user ID cannot run SELECTs on the database to generate reports. This is a defined PeopleSoft Database Vault policy restriction. SQR's should be run as scheduled Process Scheduler jobs on the server. Also, when applying PeopleSoft upgrades involves some steps that require access to the database using the access ID. For example, in some cases you need to run Data Mover in bootstrap mode using the access ID/password. Data Mover scripts cannot be run as PSFTDBA. In these cases, the key limitation to keep in mind is that the PSFTDBA ID cannot run a select against any access ID owned tables, which includes tables required for Data Mover to log in to the system.

In cases, where you need SELECT access for certain features (SQR, Data Mover, and so on) you can configure a set of specific, alternative ID's to be used for PeopleSoft upgrade tasks while still remaining in compliance with the Database Vault policies.

Using Multiple Alternate Access IDs

The sample PeopleSoft Database Vault security policies provide protection of highly sensitive information in the PeopleSoft tables from database "super users." In some cases, you may need a more tailored access, such as in the cases of upgrades, patching, auditing, and the separation of duties for the PeopleSoft Access ID.

You can leverage Database Vault so that PeopleSoft tables, procedures and triggers could be protected can still be protected while allowing special access to complete upgrade and maintenance tasks. The privileges in the Database Vault PeopleSoft template can be given to the multiple, alternate, access IDs. By using multiple, alternate, access IDs to perform PeopleSoft maintenance, you can mitigate the issues involved with distributing the password of the base access ID to multiple users.

The multiple, alternate, access IDs (PSFTDBAnn) technique has been tested with Database Vault in the field on PeopleSoft installations and offers a solution where unique, identifiable accounts can be used to perform PeopleSoft patching and upgrades. These accounts can be limited to the modules and machine

names from which the PSFTDBAnn ID can run. These accounts also can be heavily audited, to make sure that they do not introduce malicious code, which removes the need to implement heavy auditing on the base access ID account.

With multiple, alternate, access IDs you can:

- Use multiple, alternate, access IDs that can be used just for PeopleSoft upgrade/maintenance.
- Create PSFTDBAnn accounts that have many auditing options enabled, not affecting the use of the production access ID (SYSADM).
- Use Oracle's Audit Vault to enhance the separation-of-duties when it comes to centrally managing audit information.
- Configure Database Vault so that the PSFTDBAnn account can be restricted to particular machines and times, and so on.
- Tale advantage of Database Vault's strictly DBA account(s) (PSFTDBA) that can modify the database and system, but not issue selects on tables in the PeopleSoft Realm. The PSFTDBA account can do many DBA activities such as alter tablespaces, examine performance, start and stop the system, but not see sensitive information. The PSFTDBA account can apply Oracle Database Patches, but not apply PeopleSoft type of patches.
- Restrict knowledge of the access ID password, as it is no longer required for PeopleSoft maintenance.
- Address many more of the concerns third party auditors are identifying on systems that contain highly sensitive information in PeopleSoft applications.

In the following examples, the unofficial account "PSFTDBAn" represents multiple access IDs, although it can be almost any name. The PSFTDBAn accounts need to retain the ability to do 'SELECTS' on PeopleSoft objects. This technique leverages a protected Login Trigger that alters the CURRENT_SCHEMA, so that the PSFTDBAn accounts can *act* as the access ID (SYSADM) account, but preserve the user's identity (PSFTDBA1) when running any commands.

To configure multiple, alternate, access IDs:

1. Create one to 'n' multiple, alternate, access IDs (authorized Oracle USERS):

```
create user psftdba1 identified by oracle_1;
create user psftdba2 identified by oracle_1;
create user psftdba3 identified by oracle_1;
```

2. Grant minimal privileges to these alternate authorized USERS:

```
grant connect,resource to psftdba1;
grant connect,resource to psftdba2;
grant connect,resource to psftdba3;
```

3. CREATE an Oracle instance level logon trigger to issue an `ALTER SESSION SET CURRENT_SCHEMA` whenever an alternative authorized user logs into the instance.

```
drop trigger psft_login_trg;
create or replace trigger psft_login_trg
after logon on database

begin
-- * use dvf if in a database vault environemnt.
-- * database vault would also help protect the peoplesoft realm, and
logon trigger, and so on
```



```
-- if dvf.f$session_user in ('PSFTDBA1' , 'PSFTDBA2', 'PSFTDBA3') then
if sys_context('userenv','session_user') in in ('PSFTDBA1' , 'PSFTDBA2',
 'PSFTDBA3') then
execute immediate 'alter session set current_schema=’SYSADM';
end if;
end;
/
```

Every time one of the alternative authorized USERs logs into the instance, an `ALTER SESSION SET CURRENT_SCHEMA=ACCESSID` is issued. From here on in any operation performed that is unqualified would be done in the ACCESSID schema.

For example, if the 'PSFTDBA1' were logged into the database directly using SQLPLUS or indirectly using Data Mover, then any 'VALID' operation performed that is unqualified would be done in the ACCESSID schema. All of ‘PSFTDBA1's actions on the database could be audited if the Oracle Auditing facility (Audit Vault) were used. If you need to verify you have database connectivity, you can use the PSFTDBAn account for your test. Data Mover and SQR testing from the workstation will be able to use the PSFTDBAn account.

Working With Oracle Security Features

Oracle Database 11g introduced security features, which from a database security perspective, increased restrictions for database access. These changes are part of the "Secure By Default" configuration. These changes include setting a defined limit for the PASSWORD_LIFE_TIME and PASSWORD_GRACE_TIME associated with the default profile. This section discusses how PeopleSoft systems are affected and what your options are.

Understanding Default Profiles

All Oracle users created in an instance are assigned a default profile.

Oracle Database Version	***Default Profile Values***
Oracle 11g or higher	PASSWORD_LIFE_TIME: 180 PASSWORD_LOCK_TIME: 1 PASSWORD_GRACE_TIME: 7

For pre-11g Oracle releases, the default profile did not specify a PASSWORD_LIFE_TIME limit. As such, by default, the password for a given Oracle user never expired. PASSWORD_LOCK_TIME and PASSWORD_GRACE_TIME were also unlimited. For 11g or higher, the default profile has a PASSWORD_LIFE_TIME of 180 days. PASSWORD_LOCK_TIME and PASSWORD_GRACE_TIME also have limits.

For a PeopleSoft installation on the Oracle platform, several Oracle user IDs are created during the installation. Those Oracle users are:

- ACCESSID (default is SYSADM)
- CONNECT ID (default is people)

- PS (owns the PSDBOWNER table)

The ACCESSID is the schema owner for all database objects related to a specific PeopleSoft application installation. The ACCESSID and ACCESSID password are stored and encrypted in the PeopleSoft security table PSACCESSPRFL.

```
SQL> descr SYSADM.PSACCESSPRFL
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------

 SYMBOLICID                                NOT NULL VARCHAR2(8 CHAR)
 VERSION                                   NOT NULL NUMBER(38)
 ACCESSID                                  NOT NULL VARCHAR2(16 CHAR)
 ACCESSPSWD                                NOT NULL VARCHAR2(16 CHAR)
 ENCRYPTED                                 NOT NULL NUMBER(38)

SQL> SELECT  * from SYSADM.PSACCESSPRFL;;

SYMBOLIC          VERSION      ACCESSID         ACCESSPSWD        ENCRYPTED
----------------  ------------ ---------------- ----------------- ---------
SYSADM1           7            sBzLcYlPrag=     sBzLcYlPrag=              1
```

The connect ID is a pseudo logon which allows PeopleSoft to associate multiple PeopleSoft user IDs to the same connect ID. The connect ID has the minimum privileges required to connect to the database (only SELECT privileges on specific PeopleTools tables). After a connection has been established using the connect ID, PeopleSoft security uses the PeopleSoft user ID to control access to objects in the database. The PeopleSoft signon process validates the connect ID on the server, rather than the user ID. The connect ID simplifies database security maintenance, as you don't need to maintain access for all PeopleSoft users, just for the connect ID.

The PS ID is used once, during PeopleSoft database creation, to create the PSDBOWNER table. Once this table has been created, read access and write privileges are made public to everyone, then the PS user ID privileges are revoked.

Encountering Issues Related to Oracle Security

When the PASSWORD_LIFE_TIME has been reached, the PeopleSoft Oracle users (in this case the PeopleSoft ACCESSID and CONNECT ID) will be locked out of the database. This means that any PeopleSoft process cannot access the database, such as application server, Process Scheduler, COBOL, Data Mover, and so on.

If this occurs you will see any of the following Oracle database error messages:

```
ORA-28000: the account is locked
Cause: The user has entered wrong password consequently for maximum number of times⇒
 specified by the user's
profile parameter FAILED_LOGIN_ATTEMPTS, or the DBA has locked the account
Action: Wait for PASSWORD_LOCK_TIME or contact DBA

ORA-28001: the password has expired
Cause: The user's account has expired and the password needs to be changed
Action: change the password or contact the DBA

ORA-28002 the password will expire within string days
Cause: The user's account is about to about to expire and the password needs to be
 changed.
Action: Change the password or contact the database administrator.
```

These messages may appear in a SQL trace, an application server log, a Process Scheduler log, or in an error message in the GUI when attempting to access the database (signon to Application Designer or Data Mover). The following are some select examples of what you can expect to see in log and trace files.

The trace will show the login failing as follows:

```
CONNECTID.


2-4     13.06.56    1.581000 Cur#0.6060.QE849C42 RC=28001 Dur=1.581000
Connect=Primary/QE849C42/people/
2-5     13.06.56    0.000000 Cur#0.6060.QE849C42 RC=-1 Dur=0.000000 XER
rtncd=761802124 msg=
2-6     13.06.56    0.000000 Cur#0.6060.QE849C42 RC=0 Dur=0.000000 ERR rtncd=28001
 msg=ORA-28001: the password has expired
```

The following illustrates an application server or Process Scheduler boot with passwords already expired:

```
PeopleTools 8.xx.07 Client Trace - 2008-10-24

PID-Line  Time        Elapsed  Trace Data...
--------  --------  ---------- -------------------->
 1-1      14.25.45             Tuxedo session opened {oprid='QEDMO',
appname='Two Tier', addr='//TwoTier:7000', open at 01C67EC8, pid=4956}
 1-2      14.25.45    0.058000 Cur#0.4956.QE849C41 RC=0 Dur=0.003000 --- router
 PSORA load succeeded
 1-3      14.25.45    0.155000 Cur#0.4956.QE849C41 RC=0 Dur=0.155000 INI
 1-4      14.25.45    0.192000 Cur#0.4956.QE849C41 RC=28002 Dur=0.192000
Connect=Primary/QE849C41/people/
 1-5      14.25.45    0.000000 Cur#0.4956.QE849C41 RC=-1 Dur=0.000000 XER
rtncd=761800508 msg=
 1-6      14.25.45    0.000000 Cur#0.4956.QE849C41 RC=0 Dur=0.000000 ERR
rtncd=28002 msg=ORA-28002:
the password will expire within 7 days
 1-7      14.25.48    2.718000 Cur#0.4956.notSamTran RC=0 Dur=0.000000 DON
 1-8      14.25.51    2.742000 Tuxedo session opened { DisconnectAll at01C67EC8,
 pid=4956}
```

The following illustrates a client trace of a application server or Process Scheduler boot:

```
PeopleTools 8.49.07 Client Trace - 2008-10-24

PID-Line  Time        Elapsed  Trace Data...
--------  --------  ---------- -------------------->
 1-1      14.30.38             Tuxedo session opened {oprid='QEDMO',
appname='Two Tier', addr='//TwoTier:7000', open at 01C67EC8, pid=3328}
 1-2      14.30.38    0.056000 Cur#0.3328.QE849C41 RC=0 Dur=0.004000 --- router
 PSORA load succeeded
 1-3      14.30.38    0.238000 Cur#0.3328.QE849C41 RC=0 Dur=0.238000 INI
 1-4      14.30.38    0.529000 Cur#1.3328.QE849C41 RC=0 Dur=0.529000
Connect=Primary/QE849C41/people/
 1-5      14.30.38    0.036000 Cur#1.3328.QE849C41 RC=0 Dur=0.000000 GET
type=1003 dbtype=4
 1-6      14.30.38    0.000000 Cur#1.3328.QE849C41 RC=0 Dur=0.000000 GET
type=1004 release=11
 1-7      14.30.38    0.076000 Cur#1.3328.QE849C41 RC=0 Dur=0.000000 COM
Stmt=
SELECT OWNERID FROM PS.PSDBOWNER
WHERE DBNAME=:1
.
1-41     14.30.40    0.200000 Cur#1.3328.QE849C41 RC=0 Dur=0.200000 Disconnect
 1-42     14.30.40    0.251000 Cur#0.3328.QE849C41 RC=28002 Dur=0.220000
Connect=Primary/QE849C41/QE849C41/
 1-43     14.30.40    0.000000 Cur#0.3328.QE849C41 RC=-1 Dur=0.000000 XER
rtncd=18874368 msg=
 1-44     14.30.40    0.000000 Cur#0.3328.QE849C41 RC=0 Dur=0.000000 ERR
rtncd=28002 msg=ORA-28002: the password will expire within 7 days
 1-45     14.30.42    2.293000 Cur#0.3328.notSamTran RC=0 Dur=0.000000 DON
 1-46     14.30.43    0.788000 Tuxedo session opened { DisconnectAll
at01C67EC8, pid=3328}
```


The failure and return of the GRACE PERIOD warning message gives you time to react before the password actually expires, enabling you to be proactive and reset or change the ACCESSID and/or the CONNECT ID password(s).

Oracle Security Configuration Options

This section discusses options for dealing with Oracle 11g or higher security, including:

- Setting the PASSWORD_LIFE_TIME to unlimited.
- Creating a PeopleSoft-specific profile.
- Resetting the PeopleSoft installation user IDs.
- Changing the PeopleSoft installation user IDs.

Setting the PASSWORD_LIFE_TIME to Unlimited

You can set the PASSWORD_LIFE_TIME in the default profile to unlimited. If this is done prior to creating the PeopleSoft-specific Oracle user IDs used for the PeopleSoft database installation, then the default behavior will mimic the pre-Oracle 11g behavior.

This can be done by creating the ACCESSID and CONNECT ID using the following command:

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_LIFE_TIME UNLIMITED
;
```

Note: While feasible, this particular solution is counter to the secure by default positioning of Oracle 11g or higher and to regulations requiring periodic changes to important passwords.

Creating a PeopleSoft-Specific Profile

You can create a PeopleSoft-specific profile which sets the PASSWORD_LIFE_TIME to unlimited. Creating the new PeopleSoft profile should be done when you create the database rather than altering PeopleSoft users from the default profiles to the PeopleSoft-specific profiles. Switching the a PeopleSoft-specific profile after you have created the PeopleSoft-specific users expired password limits does not automatically modify the expiry_date column in USER_USERS (done when creating the users with the default profile).

Create the ACCESSID and CONNECT ID user IDs using the delivered scripts, PS_HOME/scripts/PSADMIN.SQL and PS_HOME/scripts/CONNECT.SQL. After doing so, the PeopleSoft Oracle user IDs would have the default profile assigned. Alter the ACCESSID and CONNECT ID user IDs to make use of the alternate profile rather than the default. This can be done using the following commands:

```
CREATE PROFILE PSPROFILE LIMIT  PASSWORD_LIFE_TIME UNLIMITED
;
```

This creates the PSPROFILE profile with password limits values set. All values not explicitly listed are derived from the default profile.

The following statements alter both the default ACCESSID and CONNECT ID to utilize the PSPROFILE profile with the password limit set for PASSWORD_LIFE_TIME to unlimited:

```
ALTER USER SYSADM PROFILE PSPROFILE
;
ALTER USER PEOPLE  PROFILE PSPROFILE
```



```
;
```

Note: While feasible, this solution will allow the profile expiration behavior to mimic the pre-Oracle 11g behavior, but this runs counter to the intent of regulations that require changing critical passwords on a regular basis.

Resetting the PeopleSoft Installation User IDs

You can reset the PeopleSoft installation Oracle user ID passwords (the ACCESSID and CONNECT ID) in all of the places it needs to be reset. After the passwords expire, reset them to the original value. You can reset the password using the PASSWORD command or by ALTER USER command.

Note: If using Database Vault, then only the database vault account manager can reset the account, because the access ID cannot login to SQLPLUS to change the password.

Note: While feasible, this option runs counter to the intent of regulations that require changing critical passwords on a regular basis.

Changing the PeopleSoft Installation User IDs

The recommended option is to change the PeopleSoft installation required Oracle user ID passwords (the ACCESSID and CONNECT ID) after they have expired, and reflect those changes in all required locations. This option enables you to conform to regulations that require changing critical passwords on a regular basis.

If the password expires and an Oracle user ID password is changed within the Oracle database for the ACCESSID or CONNECT ID, the PeopleSoft system will still have the old password stored in the PeopleSoft security metadata tables and configuration files. These changed passwords will have to be reflected in the PeopleSoft security metadata tables and configuration files as well as the database.

At the database level, you can use the PASSWORD and ALTER USER commands to change the ACCESS ID and CONNECT ID passwords. For example:

```
C:\Documents and Settings\>sqlplus people/peop1e@QE849C42

SQL*Plus: Release 10.2.0.3.0 - Production on Tue Oct 21 10:55:57 2008

Copyright (c) 1982, 2006, Oracle.  All Rights Reserved.

ERROR:
ORA-28001: the password has expired

Changing password for people
New password: <changed to  'peop2e'>
Retype new password: <changed to  'peop2e'>
Password changed

SQL> exit
```

Or,

```
ALTER USER QE849C42 IDENTIFIED BY CHANGEPW ACCOUNT UNLOCK;

User altered.

ALTER USER people IDENTIFIED BY peop2e ACCOUNT UNLOCK;

User altered.
```



```
SQL> exit
```

Note: You may also have to include the UNLOCK keyword to unlock the account (if the password retry has been exceeded).

In PeopleTools, open Configuration Manager and change the Connect Password value on the Startup tab.

Then, open Data Mover in bootstrap mode (using the new ACCESSID password) to run the necessary commands to change the ACCESSID passwords on the appropriate PeopleSoft metadata tables. For example,

```
SET LOG c:\temp\changeaccessidpswd.out;
UPDATE PSSTATUS SET OWNERID = 'QE849C42';
UPDATE PSOPRDEFN SET OPERPSWD = OPRID, ACCTLOCK=0, ENCRYPTED = 0;
UPDATE PSACCESSPRFL SET ACCESSID = 'QE849C42', ACCESSPSWD = 'CHANGEPW',
VERSION = 0, ENCRYPTED = 0;
ENCRYPT_PASSWORD *;
```

Note: For Oracle 11g or higher, the password is case sensitive.

Lastly, apply the connect ID changes to the psprcs.cfg and psappsrv.cfg configurations files and rebuild the domains. For example:

```
[Startup]
;============================================================================
; Database Signon settings
;============================================================================
DBName=QEDMO
DBType=ORACLE
UserId=QEDMO
UserPswd==QEDMO
ConnectId=people
ConnectPswd=peop2e
ServerName=
```

Working With Oracle Transparent Application Failover

PeopeTools provides limited support for Oracle Transparent Application Failover (TAF). PeopleTools TAF support includes:

- PeopleSoft servers can be configured to transparently reconnect to a surviving RAC instance in the event of an instance failure with in a RAC cluster.
- PeopleSoft servers can be configured to transparently fail over to an Oracle Database Data Guard standby when the primary database is lost.

Note: In most cases, other than a slight pause in the operation, the failover is transparent to the application end user.

Note: The Oracle TAF feature as implemented in Oracle 11g and earlier versions of Oracle only supports recoverability of in-flight SELECT statements. SELECT statements that are part of an uncommitted transaction block are not supported with TAF. Recoverability of INSERTs, UPDATEs, and DELETEs are not supported with TAF. Given these limitations, PeopleSoft does not support TAF for non-query operations.

PeopleTools is designed to listen for Oracle fast application notification (FAN) events to derive the failover behavior. Upon receipt of a FAN event, PeopleSoft servers break their existing TCP connections and initiate TAF, which references the TNSNAMES.ORA connect alias address list and establishes a connection to the surviving instance.

See Your Oracle RAC and database administration guides for the details of implementing and managing Oracle RAC clusters.

End-User System Behavior With TAF Configured

The following table summarizes PeopleSoft behavior during RAC or Data Guard failover when TAF is configured.

PeopleSoft Client Scenario	***Behavior***
End user is inserting, updating, or deleting data and submits or saves the inserts/updates/deletes during or just after the database failure.	The data manipulation language (DML) will fail. Transactions will not get resubmitted. Oracle reconnects and reconstructs the database session on a surviving node and the end user must resubmit the transaction.
End user is paging through queried data (SELECTs) when the database failure occurs.	Oracle reconnects and reconstructs the database session on a surviving node, re-executes the query, repositions the SQL cursor, and returns the next set of rows.
End user is issuing a new query (SELECTs) or switching screens just after the database failure.	Oracle reconnects and reconstructs the database session on a surviving node.

Batch System Behavior With TAF Configured

The following table summarizes PeopleSoft batch system behavior during RAC or Data Guard failover when TAF is configured.

PeopleSoft Batch System Scenario	***Behavior***
Process Scheduler	Oracle reconnects and reconstructs the session on a surviving node. The Process Scheduler fails over with no administration intervention required.
Application Engine job submitted just *before* primary instance failure	Oracle reconnects and reconstructs the session on a surviving node but Application Engine job may fail and appear in the PeopleSoft Process Monitor with a status of *No Success*. These jobs will need to be resubmitted manually. If the Application Engine job was not in an open-transaction and was performing only SELECT statements, it will fail over and complete successfully.
Application Engine submitted *during* or *just after* primary instance failure	Oracle reconnects and reconstructs the session on a surviving node, the Application Engine job is then submitted on the new primary database and completes successfully.

PeopleSoft Batch System Scenario	*Behavior*
COBOL jobs before, during, or after primary instance failure	The COBOL jobs will not complete successfully on the surviving node. Manual intervention is required to restart the COBOL jobs.
SQR jobs before, during, or after primary instance failure	Manual intervention is required to restart or resubmit the SQR jobs.
Crystal and nVision reports	The behavior is the same as COBOL
PSQUERY, Tree Viewer, BI Publisher Query Report Viewer	Will fail over and complete successfully.

Implementing Oracle Active Data Guard

This section provides an overview and discusses:

- Installing and configuring Oracle Active Data Guard.
- Setting up the secondary access ID.
- Creating the domain boot user.
- Creating synonyms and database links.
- Configuring domains.
- Configuring read-only components.
- Configuring read-only processes.
- Disabling mostly-read-only attributed features.

Understanding Active Data Guard Within PeopleSoft

Oracle Active Data Guard, with Oracle Database Enterprise Edition 11g or higher, enables you to offload resource-intensive activities from a production database to a synchronized standby database. Oracle Active Data Guard (ADG) enables read-only access to a physical standby database for queries, sorting, reporting, web-based access, and so on, while continuously applying changes received from the production database. If you use ADG at your site, PeopleTools provides the infrastructure to use ADG with your PeopleSoft application databases.

The intent of this feature integration is that *most* SQLs in a mostly-read-only (MRO) marked unit of work are redirected to the standby database in an ADG configuration. The ability to redirect a unit of work (UOW) to a standby database is limited to PeopleSoft components and processes. Within any MRO marked component or process, not all SQL is always redirected to a standby database; the percentage of SQL that is redirected depends on the specific component or process and what types of calls are made within that UOW.

The PeopleSoft offload reporting infrastructure using ADG enables the use of a standby database for a subset of the PeopleTools reporting features. For an optimal implementation, you should configure ADG

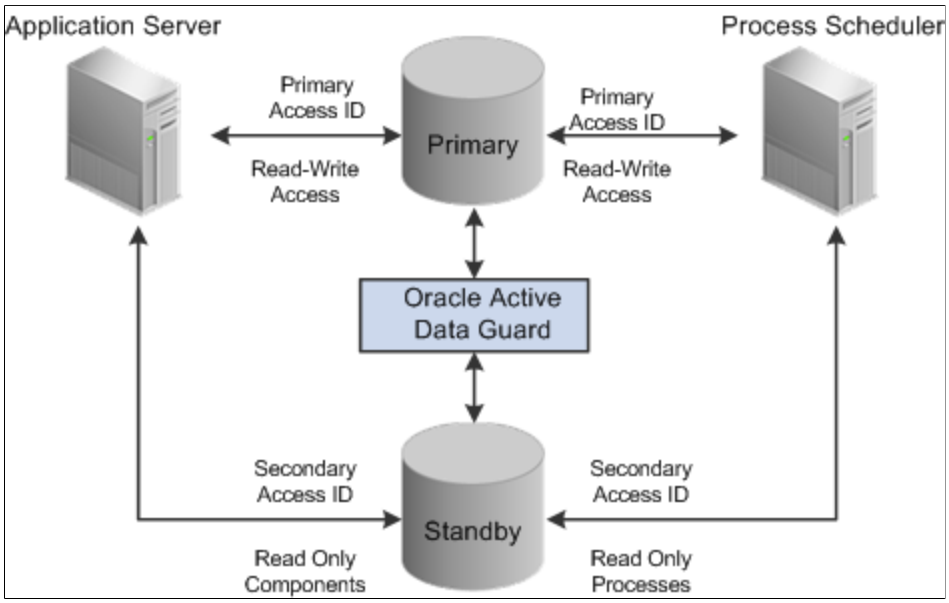
for sub-second replication. Slow performance can be expected in configurations where there is significant network latency between the primary and standby databases. To mitigate potential performance issues over WANs with large network latency, an alternative configuration that can be used with our infrastructure is:

local primary database and *local standby database + remote standby database*

where the local standby database is used for ADG reporting, and the remote standby database is used for high availability purposes.

Image: Oracle Active Data Guard synchronizing the primary and standby database so read-only requests can be routed to the standby database for processing

The following diagram depicts how Oracle Active Data Guard enables you to incorporate the use of a standby database for you to offload designated read-only transactions, freeing up more resources to handle the read-write transactions on your primary (production) database.



Note: This documentation uses the acronyms RO to refer to "read-only" and MRO to refer to "mostly-read-only."

The following table describes the elements within the diagram:

Element	***Description***
Primary Database	Your production database, handling the read-write requests of your transactional system. For example, this database fills orders, updates employee information, adds new product offerings, and so on.
Standby Database	Your clone of the primary database designed to handle read-only, or mostly-read-only (MRO), requests so that those transactions can be offloaded from your primary database, conserving resources on the production system. Examples of MRO requests include, PSQUERY Viewer, Tree Viewer, components that only submit SELECT SQL to display lists of employees, products, and so on.

Element	*Description*
Oracle Active Data Guard	Synchronizes the data stored in the primary and standby databases so that they remain exact duplicates. This is achieved using a combination of Oracle Active Data Guard features and DBLINKS and Remote Synonyms defined by scripts delivered with PeopleTools.
Primary Access ID	The PeopleSoft access ID used for connecting to the primary, production database.
Secondary Access ID	The alternative PeopleSoft database user (access ID), which will be used for connecting to the standby database and subsequently accessing the primary database's tables using LOCAL and REMOTE synonyms.
Read Only components	When Oracle Active Data Guard is enabled and PeopleSoft is configured with a standby database, these components are RO enabled "out of the box:" • Query Manager: PSQUERY Viewer queries will be run on the Standby database. **Note:** The PSQUERY Viewer does not need to be explicitly marked RO. Units of work utilizing ICQueryService are redirected to the Standby DB. Because PSQUERY Viewer uses the ICQueryService, its SQL will be redirected to the standby database. • Tree Manager: Tree Viewer • BI Publisher: Query Report Viewer • Pivot Grid: Pivot Grid Viewer and Pivot Grid viewed as a pagelet • QAS (Query as a Service) • Component Interfaces: Component Interfaces work on any component marked as MRO. Other components can also be made to run against the standby database, by setting the Read Only option in the component properties dialog box in Application Designer. See Configuring Read-Only Components. Limitations: • BI Publisher: If Query Report Scheduler is RO enabled, you cannot schedule a bursted report. • Application Engine: An Application Engine process cannot be marked as RO if it makes any publish call to Integration Broker. There is a limited set of Integration Broker service operations that are enabled, and those operations are specifically related to Feeds and QAS.

Element	*Description*
Read Only processes	When PeopleSoft is configured for Oracle Active Data Guard the following processes are enabled, as delivered, to run against the standby database: PeopleSoft Query: • PSCONQRS: Run Scheduled Connected Queries • PSQUERY: Scheduled Query PS/nVision: • Report • Report drilldown • Report book SQRs: • XRFAPFL • XRFFLPN • XRFFLRC • XRFIELDS • XRFMENU • XRFPANEL • XRFPCFL • XRFPNPC • XRFRCFL • XRFRCPN • XRFWIN • XRFFLPC Audit Utilities: • SWPAUDIT • SYSAUDIT Other processes can also be enabled to run against the standby database by setting the Read Only option on the Process Definition properties page in Process Scheduler. **Note:** Do not run DDDAUDIT report on a standby database. **Note:** For Scheduled Query, if a user attempts to schedule a query to run against the standby database, and selects output type Feeds on the Process Scheduler Request page, that process will be redirected to the primary database. This overrides the RO enabled Run Scheduled Query process.

Element	Description
	Note: The use of Oracle Active Data Guard with PeopleSoft batch processing only applies to the following: Application Engine processes run through the Process Scheduler with PSAESRV configured and SQR processes. **Note:** To enable SQR processes to run against the standby database, refer to Configuring Read-Only Processes. SQR Processes that are generally considered reports are ideal candidates for redirection to the standby database.

Installing and Configuring Oracle Active Data Guard

Oracle Active Data Guard needs to be licensed, installed, and enabled for your server before you can begin setting up your PeopleSoft system to take advantage of this feature. The primary and standby databases need to be set up according to the Oracle Active Data Guard documentation.

Creating the Secondary Access ID

This section discusses how to configure the secondary access ID for use with your primary and standby databases.

To configure the secondary access ID:

1. Run the psadmin.sql script against your primary database.
   - Run the script from PS_HOME\scripts.
   - When prompted, provide the secondary access ID and password.
2. Grant required privileges to the secondary access ID.

   Login as ‘system’ on the primary database and execute the following SQL:

   ```
   GRANT SELECT ANY TABLE TO <secondary access ID>;
   GRANT INSERT ANY TABLE TO <secondary access ID>;
   GRANT UPDATE ANY TABLE TO <secondary access ID>;
   ```

   For example:

   ```
   GRANT SELECT ANY TABLE TO Q855108S;
   GRANT INSERT ANY TABLE TO Q855108S;
   GRANT UPDATE ANY TABLE TO Q855108S;
   ```
3. Insert a row in the PSDBOWNER table for the standby database and the secondary access ID.

   For example:

   ```
   INSERT INTO PS.PSDBOWNER VALUES('DATABASE', '<secondary access ID>’);
   Commit;
   ```
4. Create an access profile to associate the secondary access ID with a new symbolic ID.
   - In Application Designer, select Tools, Misc Def, Access Profile and click New.

- Provide the new symbolic ID, and enter the secondary access ID and password you created using psadmin.sql.

Creating the Domain Boot User

Create or clone a PeopleSoft user ID and associate it with the secondary access ID so that the user profile will be able to start an application server domain and alternate Process Scheduler server connecting to the standby database.

To create the domain boot user:

1. Select PeopleTools, Security, Copy User Profiles, and enter a new user ID and password.
2. From the Symbolic ID dropdown list on the General tab for the user ID you just created, select the symbolic ID associated with your secondary access ID.
3. Open Data Mover and encrypt the new user profile's password by running the following:

```
UPDATE  PSOPRDEFN SET operpswd='<PASSWORD>', encrypted=0
Where oprid='<USER_ID>';
ENCRYPT_PASSWORD *;
```

Creating Synonyms and Database Links

The mechanism that is used to make database objects in the standby database available in the PeopleSoft implementation of Oracle Active Data Guard requires the existence of a secondary access ID. The secondary access ID does not own any objects, so sufficient access to and awareness of objects in the primary database must be made to the secondary access ID.

To associate standby database objects with the equivalents in the primary database, you must create:

- Local synonyms.
- Database links to the primary database.
- Remote synonyms.

PeopleSoft delivers the following scripts (located in PS_HOME\scripts) to create the synonyms and database links:

- createlocalsynonyms.sql
- createdblinktoprimary.sql
- createremotesynonyms.sql

The following table describes each script, and any modifications that you need to make before you run them.

Script	*Description*	*Usage*
createlocalsynonyms.sql	This script generates the following four sql scripts: • createlocaltablesynonyms.sql • createlocalviewsynonyms.sql • grantselectontables.sql • grantselectonviews.sql Execute the generated scripts to create local synonyms for all PeopleSoft tables and views, and to grant select privileges to the secondary access ID for all PeopleSoft tables and views. Run these scripts against the primary database while connected as the secondary access ID: • createlocaltablesynonyms.sql • createlocalviewsynonyms.sql Run these scripts against the primary database while connected as the primary access ID. • grantselectontables.sql • grantselectonviews.sql	Run this script against the primary database while connected as the secondary access ID. Before you run the script, edit it to make the following changes: • Replace <*SYSADM*> with the primary access ID. • Replace <*SYSADMS*> with the secondary access ID.
createdblinktoprimary.sql	This script creates a fixed database link between the secondary access ID and the primary access ID on the primary database.	Run this script connected to the primary database with the secondary access ID Before you run the script, edit it to make the following changes: • Replace <*DBNAME*> with the primary database TNSALIAS. • Replace <*SYSADM*> with the primary access ID. • Replace <*PASSWORD*> with the primary access ID password.
createremotesynonyms.sql	This script creates remote synonyms. Remote synonyms are required for the tables identified as requiring DML access. If you decide to set a component to "read-only" after your analysis, then you need to include the underlying tables that require DML access to this script manually.	Run this script against the primary database connected as the secondary access ID.

Configuring Domains

The domains connecting to your primary database must also be configured to connect to your standby database. This is accomplished by specifying the standby connection credentials in the Startup section of the domain configuration for both the application server and Process Scheduler domains.

For example:

```
Values for config section - Startup
    DBName=PRIMARY
    DBType=ORACLE
    UserId=QEDMO
    UserPswd=QEDMO
    ConnectId=people
    ConnectPswd=peop1e
    ServerName=
    StandbyDBName=STANDBY
    StandbyDBType=ORACLE
    StandbyUserId=PTSTNDBY
    StandbyUserPswd=PTSTNDBY
```

The Startup section enables you to specify signon credentials for both the primary and standby databases. The following table shows which parameters are associated with which database.

Database	***Parameters***
Primary	DBName DBType UserId UserPswd
Standby	StandbyDBName StandbyDBType StandbyUserID StandbyUserPswd

The Standby… parameters are used to maintain the simultaneous connection to the standby database.

Note: Typically, the primary and standby database share a common connect ID.

Note: For Active Data Guard users, the StandbyUserId and StandbyUserPswd values are the User ID and password that you created in Creating the Domain Boot User.
For Golden Gate users, the StandbyUserId and StandbyUserPswd values are the same as your User ID and password. For example, if your User ID is QEDMO, then your StandbyUserId is also QEDMO.

Configuring Read-Only Components

When Oracle Active Data Guard is enabled, a collection of components are supported for running against the standby database. The list of components appears previously in this document.

See Understanding Active Data Guard Within PeopleSoft.

However, you can also set other components to run against the standby database to divert selected requests from your primary database. To set a component to run in "Read-Only" mode (run against the standby database) you need to modify the component properties.

To configure a component for read-only processing:

1. In Application Designer, open the component.
2. Select View, Definition Properties.
3. On the Component Properties dialog box, select the Use tab.
4. Select the Read Only check box.
5. Click OK.

Note: Selecting the Read Only property should only be done after detailed analysis.

Note: It is important to understand the behavior of a mostly read only (MRO) component when it calls other components. If an MRO component is executed, then all components subsequently called by the MRO marked component will inherit the MRO attribute behavior and the SQL calls and the called components will be redirected to the standby database, when possible. Given this behavior, you need to make sure that the requests generated from the component (and called components) perform operations like selecting and displaying lists, rather than inserting, updating, or deleting rows. If there are some DML operations that the component must execute, the affected tables need to be identified, and a remote synonym needs to be created between the standby and primary databases. See the delivered PS_HOME/scripts/CREATEREMOTESYNONYMS.SQL script for an example of how to create a remote synonym.

Configuring Read-Only Processes

If you have Oracle Active Data Guard configured and enabled for your PeopleSoft system, Process Scheduler processes can be set to run against the standby database to divert selected processes from your primary database.

Note: The use of Oracle Active Data Guard with PeopleSoft batch processing only applies to the following: Application Engine processes run through the Process Scheduler with PSAESRV configured and SQR processes.

To configure Process Scheduler processes for read-only processing:

1. Select PeopleTools, Process Scheduler, Processes.
2. Open the process definition.
3. On the Process Definition page, select the Read Only check box.
4. Click OK.

Note: Selecting the Read Only option should only be done after detailed analysis. You need to make sure that the processes perform operations like selecting data or generating reports, rather than inserting, updating, or deleting rows. If there are some DML operations that the process must make, the affected tables need to be identified, and a remote synonym needs to be created between the standby and primary databases.

Note: If an MRO marked SQR program runs and is performing DML on a table not accounted for in the Oracle Active Data Guard configuration, then that SQR process will fail. To fully enable this SQR process, the tables being written to need to have remote synonyms created between the standby and primary databases. See the delivered PS_HOME/scripts/CREATEREMOTESYNONYMS.SQL script for example on how to create a remote synonym.

Disabling Mostly-Read-Only Attributed Features

If you do not want any of the read-only enabled delivered components or processes redirected to the standby database, then you can disable the read-only attribute for the specific component or process by going to the appropriate component definition or process definition and deselecting the read-only check box.

In the following features, the read-only attribute behavior is hard-coded:

- Query as a service (QAS).
- Any component that uses ICQueryService.

Note: The SQL of any component that uses ICQueryService, will be redirected to the standby database. PSQUERY Viewer is an example of one such component; it does not need to be explicitly marked RO, because it uses ICQueryService.

The following sections describe how to disable hard-coded read-only attributes for these features.

Disabling the Read-Only Behavior for QAS

To disable the read-only behavior for QAS Run the PT_SETQASADG Application Engine program on your database.

For example, from the DOS command line the syntax is:

```
<PS_HOME>\bin\client\winx86\psae -CD <dbname> -CT ORACLE -CO <userid> -CP <userpswd⇒
> -R RUN01 -AI PT_SETQASADG -I
```

Use the values for the database name and user ID that you entered on the startup tab of Configuration Manager for <dbname> and <userid> respectively. However, be aware that <userpswd> is not the same as the connect password you entered on the Configuration Manager startup tab. Enter a value for <userpswd> that is the password you want to be associated with the <userid>.

Disabling the Read-Only Behavior for Components that use ICQueryService

To disable routing of components that use ICQueryService to the standby database, modify the following line in the Database Options section of the Application Server configuration file:

```
;Disable ICQueryService Standby Routing=
```

Enable this line by removing the initial comment character (;) and setting the parameter value to 1. For example:

```
 [Database Options]
;============================================================================
; Database-specific configuration options
;============================================================================
SybasePacketSize=
UseLocalOracleDB=0
```



```
;ORACLE_SID=
EnableDBMonitoring=1
PSDB Maximum Cursors=
Disable ICQueryService Standby Routing=1
```

Implementing Oracle GoldenGate for PeopleSoft Off-Load Reporting

This section contains an overview and discusses how to:

- Install Oracle GoldenGate.
- Configure Oracle GoldenGate.
- Configure PeopleSoft databases for use with Oracle GoldenGate.
- Configure PeopleSoft Domains for use with Oracle GoldenGate.
- Configure read-only components.
- Configure read-only process definitions.
- Create synonyms and database links.

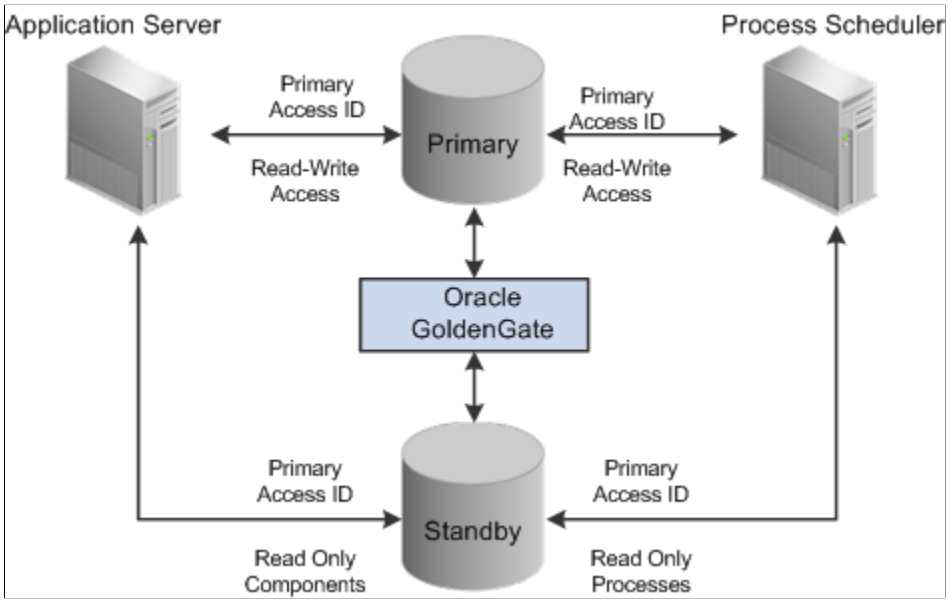
Understanding GoldenGate Within PeopleSoft

Oracle GoldenGate, with Oracle Database 11g or higher, enables you to off-load resource-intensive activities from a production database to a synchronized standby database. Oracle GoldenGate enables access to a physical standby database for queries, sorting, reporting, web-based access, and so on, while continuously applying changes received from the production database. If you use Oracle GoldenGate at

your site, PeopleTools provides the infrastructure to use GoldenGate with your PeopleSoft application databases.

Image: Oracle GoldenGate synchronizing data on the primary and standby databases

The following diagram depicts how Oracle GoldenGate enables you to incorporate the use of a standby database for you to offload designated read-only transactions, freeing up more resources to handle the read-write transactions on your primary (production) database.



The following table describes the elements within the diagram:

Element	***Description***
Primary Database (source)	Your production database, handling the read-write requests of your transactional system. For example, this database fills orders, updates employee information, adds new product offerings, and so on.
Standby Database (target or reporting database)	Your clone of the primary database designed to handle read-only, or mostly-read-only (MRO), requests so that those transactions can be off-loaded from your primary database, conserving resources on the production system. Examples of MRO requests include, PSQUERY Viewer, Tree Viewer, components that only submit SELECT SQL to display lists of employees, products, and so on.
Oracle GoldenGate	Synchronizes the data stored in the primary and standby databases so that they remain exact duplicates. This is achieved using a combination of Oracle GoldenGate features, DBLINKS, and Remote Synonyms defined by scripts delivered with PeopleTools.

Element	*Description*
Primary Access ID	The PeopleSoft access ID used for connecting to the primary production database as well as the standby database. **Note:** Only one access ID is required, unlike Oracle Active Data Guard, where the PeopleSoft implementation requires two access IDs (a primary and a secondary access ID). The access ID and access ID passwords *must* be the same on the primary and standby databases For example, if you use SYSADM as your access ID on the primary database, you must also use SYSADM as the access ID on the standby database, and you must keep the passwords for SYSADM the same on both databases.
Read Only components	When Oracle GoldenGate is enabled and PeopleSoft is configured with a standby database, these components are RO enabled: • Query Manager: PSQUERY Viewer queries will be run on the Standby database. **Note:** The PSQUERY Viewer does not need to be explicitly marked RO. Units of work utilizing ICQueryService are redirected to the Standby DB. Because PSQUERY Viewer uses the ICQueryService, its SQL will be redirected to the standby database. • Tree Manager: Tree Viewer • BI Publisher: Query Report Viewer • Pivot Grid: Pivot Grid Viewer and Pivot Grid viewed as a pagelet • QAS (Query as a Service) • Component Interfaces: Component Interfaces work on any component marked as MRO. Other components can also be made to run against the standby database, by setting the Read Only option in the component properties dialog box in Application Designer. See Configuring Read-Only Components. Limitations: • BI Publisher: If Query Report Scheduler is RO enabled, you cannot Schedule a Bursted report. • Application Engine: An Application Engine process cannot be marked as RO if it makes any publish call to Integration Broker. There is a limited set of Integration Broker service operations that are enabled, and those operations are specifically related to Feeds and QAS.

Element	*Description*
Read Only processes	When PeopleSoft is configured for Oracle GoldenGate the following processes are enabled, as delivered, to run against the standby database: PeopleSoft Query: • PSCONQRS: Run Scheduled Connected Queries • PSQUERY: Scheduled Query PS/nVision: • Report • Report drilldown • Report book SQRs: • XRFAPFL • XRFFLPN • XRFFLRC • XRFIELDS • XRFMENU • XRFPANEL • XRFPCFL • XRFPNPC • XRFRCFL • XRFRCPN • XRFWIN • XRFFLPC Audit Utilities: • SWPAUDIT • SYSAUDIT • DDDAUDIT Other processes can also be enabled to run against the standby database by setting the Read Only option on the Process Definition properties page in Process Scheduler. **Note:** For Scheduled Query, if a user attempts to schedule a query to run against the standby database, and selects output type Feeds on the Process Scheduler Request page, that process will be redirected to the primary database. This overrides the RO enabled Run Scheduled Query process.

Element	Description
	Note: The use of Oracle GoldenGate with PeopleSoft batch processing only applies to the following: Application Engine processes run through the Process Scheduler with PSAESRV configured and SQR processes. **Note:** To enable SQR processes to run against the standby database, refer to Configuring Read-Only Processes. SQR Processes that are generally considered reports are ideal candidates for redirection to the standby database.

Oracle GoldenGate needs to be licensed, installed, and enabled for your server before you can begin setting up your PeopleSoft system to take advantage of this feature. The primary and standby databases need to be set up according to the Oracle GoldenGate documentation.

The basic configuration for PeopleSoft and Oracle GoldenGate (OGG) consists of two databases, your Primary database and the Standby database. Characteristics of the server connections to these databases are:

Server	Connection Characteristics
Application Server	• Primary connection and Standby connection both use the same access ID. • Each server process (PSAPPSRV, PSSAMSRV, and so on) can connect to either database, as needed.
Process Scheduler Server	• Primary connection and Standby connection both use the same access ID. • Each batch server element (PSAESRV, PSAE, COBOL, SQR) can connect to the Primary database as needed. • Only PSAESRV connects to the Standby database as needed.

GoldenGate operates between the two databases, sits between the two databases, and manages a set of components associated with each of the databases. Depending on your implementation, one to many of these components may be in use. GoldenGate transaction replication software consists of several key components, including:

- GoldenGate Manager process: controls background process behavior.
- GoldenGate Extract or Capture component process: extracts data from the online or archive redo log files within an Oracle database.
- GoldenGate trail files: store the extracted data as part of the replication processing.
- GoldenGate Pump process: moves the data in the trail files from the primary database server to the reporting database server.
- GoldenGate Replicate or delivery process: applies the captured data (stored in the trail files) to the target database.

Note: For a GoldenGate environment, the secondary connection to the Standby database utilizes the same login credentials used for the Primary connection.

Note: Whether both databases reside on the same or different servers, GoldenGate binaries need to be installed twice, in two separate directories. One installation is for the Primary database, and the other is for the Standby database.

See your *Oracle Golden Gate* documentation for more information.

Installing Oracle GoldenGate for a PeopleSoft Implementation

The installation described here assumes an Oracle installation where the Primary and Standby database instances are on the same machine. If the Primary and Standby database instances are on separate machines, then many of these steps are duplicated on the database server hosting the standby database.

Before beginning the installation, make sure these items are in place:

- To install on Windows, the user installing GoldenGate must log in as Administrator. Review the Oracle Installation and Setup Guide for Oracle GoldenGate on Windows for any additional requirements.
- To install on UNIX, the user installing GoldenGate must have read and write privileges on the GoldenGate installation directory.
- The GoldenGate processes require an operating system user that has privileges to read, write, and delete files and subdirectories in the GoldenGate directory. In addition, the user for the Manager process requires privileges to control GoldenGate processes.
- The Extract process requires an operating system user that has read access to the log files, both online and archived. On UNIX systems, that user must be a member of the group that owns the Oracle instance.
- It is recommended that these operating system users be dedicated to GoldenGate. Sensitive information might be available to anyone running a GoldenGate process, depending on how database authentication is configured.

To install Oracle GoldenGate for a PeopleSoft Implementation:

1. Download the latest media from E-Delivery: https://edelivery.oracle.com/.
2. FTP or copy the media to database server.
3. Login on the database host as the ‘oracle’ user.
4. On your database server, create a directory that the ‘oracle’ user has read/write/execute access to, such as:

   /data1/ogg
5. Copy the media to the directory you created.
6. Logged in as the ‘oracle’ user, unzip the media file.

For example:

```
unzip V22230-01.zip

rtdc68005spdb:$ unzip V22230-01.zip
Archive:  V22230-01.zip
  inflating: ggs_Linux_x64_ora11g_64bit_v11_1_1_0_0_078.tar
  inflating: OGG_WinUnix_Rel_Notes_11.1.1.0.0_078.pdf
  inflating: README.txt
```

7. Expand the tarball twice: once to a primary directory and once again to a standby directory.

Note: The following examples assume the primary and standby directories already exist.

The primary database will use the primary directory installation and the standby database will use the other installation. For example:

```
tar -xvf  <tar ball> -C primary/

tar -xvf  <tar ball> -C standby/
```

8. View the primary and standby directory to verify the install directory and object creation.

For example:

```
rtdc68005spdb:$ ls
ggs_Linux_x64_ora11g_64bit_v11_1_1_0_0_078.tar  primary
rtdc68005spdb:$

rtdc68005spdb:$ cd primary
rtdc68005spdb:$ ls
UserExitExamples           ddl_remove.sql                  extract
bcpfmt.tpl                 ddl_session.sql                 freeBSD.txt
bcrypt.txt                 ddl_session1.sql                ggMessage.dat
chkpt_ora_create.sql       ddl_setup.sql                   ggsci
cobgen                     ddl_sqlmx.tpl                   help.txt
convchk                    ddl_status.sql                  keygen
db2cntl.tpl                ddl_staymetadata_off.sql        libicudata.so.38
ddl_access.tpl             ddl_staymetadata_on.sql         libicui18n.so.38
ddl_cleartrace.sql         ddl_sybase.tpl                  libicuuc.so.38
ddl_db2.tpl                ddl_tandem.tpl                  libxerces-c.so.28
ddl_db2_os390.tpl          ddl_trace_off.sql               libxml2.txt
ddl_ddl2file.sql           ddl_trace_on.sql                logdump
ddl_disable.sql            ddl_tracelevel.sql              marker_remove.sql
ddl_enable.sql             ddlcob                          marker_setup.sql
ddl_informix.tpl           ddlgen                          marker_status.sql
ddl_mss.tpl                defgen                          mgr
ddl_mysql.tpl              demo_more_ora_create.sql        notices.txt
ddl_nopurgeRecyclebin.sql  demo_more_ora_insert.sql        params.sql
ddl_nssql.tpl              demo_ora_create.sql             replicat
ddl_ora10.sql              demo_ora_insert.sql             reverse
ddl_ora10upCommon.sql      demo_ora_lob_create.sql         role_setup.sql
ddl_ora11.sql              demo_ora_misc.sql               server
ddl_ora9.sql               demo_ora_pk_befores_create.sql  sqlldr.tpl
ddl_oracle.tpl             demo_ora_pk_befores_insert.sql  tcperrs
ddl_pin.sql                demo_ora_pk_befores_updates.sql usrdecs.h
ddl_purgeRecyclebin.sql    emsclnt                         zlib.txt
rtdc68005spdb:$ cd ..
```

Creating Subdirectories for Primary and Standby GoldenGate Installations

In each installation directory (primary and standby) you must create these required subdirectories.

Subdirectory	*Description*
dirchk	GoldenGate checkpoint files.
dirdat	GoldenGate extract and trail files.
dirdef	Source data definitions generated by the DEFGEN utility. (Used to translate heterogeneous data.)
dirpcs	Process status files.
dirout	Directory no longer used.
dirprm	GoldenGate parameter files (run time configuration files).
dirrpt	Process report files.
dirsql	SQL files.
dirtmp	Temporary storage for transactions.
dirver	GoldenGate Veridata directory. (Only used if Veridata is also installed in this GoldenGate instance.)

To install Oracle GoldenGate subdirectories:

1. Change directories to your \primary directory.
2. Launch the GoldenGate command line interface (GGSCI).

```
rtdc68005spdb:$ cd \data1\ogg\primary
rtdc68005spdb:$ ./ggsci

Oracle GoldenGate Command Interpreter for Oracle
Version 11.1.1.0.0 Build 078
Linux, x64, 64bit (optimized), Oracle 11 on Jul 28 2010 13:13:42

Copyright (C) 1995, 2010, Oracle and/or its affiliates. All rights reserved.
```

3. Use the `create subdirs` command to create the required subdirectories.

```
GGSCI (rtdc68005spdb) 1> create subdirs

Creating subdirectories under current directory /data1/ogg/primary

Parameter files                /data1/ogg/primary/dirprm: created
Report files                   /data1/ogg/primary/dirrpt: created
Checkpoint files               /data1/ogg/primary/dirchk: created
Process status files           /data1/ogg/primary/dirpcs: created
SQL script files               /data1/ogg/primary/dirsql: created
Database definitions files     /data1/ogg/primary/dirdef: created
Extract data files             /data1/ogg/primary/dirdat: created
Temporary files                /data1/ogg/primary/dirtmp: created
Veridata files                 /data1/ogg/primary/dirver: created
Veridata Lock files            /data1/ogg/primary/dirver/lock: created
Veridata Out-Of-Sync files     /data1/ogg/primary/dirver/oos: created
Veridata Out-Of-Sync XML files /data1/ogg/primary/dirver/oosxml: created
Veridata Parameter files       /data1/ogg/primary/dirver/params: created
Veridata Report files          /data1/ogg/primary/dirver/report: created
Veridata Status files          /data1/ogg/primary/dirver/status: created
Veridata Trace files           /data1/ogg/primary/dirver/trace: created
Stdout files                   /data1/ogg/primary/dirout: created
```


4. Exit the command line interface.

```
GGSCI (rtdc68005spdb) 2> exit
```

5. View the primary directory to verify the additional subdirectories were created.
6. Repeat these steps for your standby directory.
7. Verify for each installation that the GoldenGate manager is stopped prior to continuing with further configuration instructions.

   For example:

```
rtdc68005spdb:$ cd \data1\ogg\primary
rtdc68005spdb:$ ./ggsci

GGSCI (rtdc68005spdb) 2> info all

Program     Status      Group       Lag           Time Since Chkpt

MANAGER     STOPPED

GGSCI (rtdc68005spdb) 3>
```

Configuring PeopleSoft Databases for Oracle GoldenGate

This section explains these setup steps that need to be performed on the primary and standby databases:

Creating the Oracle GoldenGate User

GoldenGate requires a separate Oracle database user that is dedicated to GoldenGate installation defined in both the Primary and Standby databases. It can be the same user for all of the GoldenGate processes that must connect to a database, such as:

- Extract (source/primary database)
- Replicat (target/standby database)
- Manager (source/primary database, if using DDL support)
- DEFGEN (source or target database)

Note: For the purposes of this document, the same GoldenGate Oracle user is defined on both databases.

Note: To preserve the security of your data, and to monitor GoldenGate processing accurately, do not permit other users, applications, or processes to log on or operate as the GoldenGate database user.

Note: Keep a record of the application database user (PeopleSoft Access ID). It is required in the GoldenGate parameter files, as in, the USERID parameter for the database.

The following table outlines the required database user privileges.

User Privilege	*Extract*	*Replicat*
Create Session, Alter Session	X	X **Note:** If RESOURCE cannot be granted to Replicat, use ALTER USER <user> QUOTA {<size> | UNLIMITED} ON <tablespace>, where <tablespace> represents all tablespaces that contain target objects.
Resource	X	X **Note:** Required only if Replicat owns target objects or any PL/SQL procedures. If CONNECT cannot be granted, grant CREATE <object>for any object Replicat will need to create.
Connect	X	X
Select Any Dictionary	X	X
Flashback Any Table Or Flashback On <owner.table>	X	
Select Any Table Or Select On <owner. table>	X	X
Select on DBA Clusters	X	
Insert, Update, Delete on <target tables>		X
Create Table **Note:** Required if using ADD CHECKPOINTTABLE in GGSCI to use the database checkpoint feature.		X
Execute on DBMS_FLSHBACK package (4) **Note:** GoldenGate must make a call to DBMS_FLASHBACK.GET_SYSTEM_ CHANGE_NUMBER.	X	

Note: Be sure to check the most recent Oracle Installation and Setup Guide for GoldenGate based on the GoldenGate version you are using as permission requirements may change or be appended.

You can create the Oracle GoldenGate user by creating a script similar to the following:

```
set echo on
spool createogguser.log
-- Create the OGG User
GRANT CONNECT, RESOURCE to OGGUSER identified by OGGUSER;
--Grant OGG required privileges
GRANT CREATE SESSION to OGGUSER;
GRANT ALTER SESSION to OGGUSER;
GRANT SELECT ANY DICTIONARY to OGGUSER;
GRANT FLASHBACK ANY TABLE to OGGUSER;
GRANT ALTER ANY TABLE to OGGUSER;
GRANT SELECT ANY TABLE to OGGUSER;
GRANT INSERT ANY TABLE to OGGUSER;
GRANT DELETE ANY TABLE to OGGUSER;
GRANT UPDATE ANY TABLE to OGGUSER;
GRANT CREATE TABLE to OGGUSER;
GRANT UNLIMITED TABLESPACE to OGGUSER;
GRANT EXECUTE on DBMS_FLASHBACK to OGGUSER;
GRANT SELECT ON dba_clusters to OGGUSER;
spool off;
```

To list the privileges granted to the OGG User, you can run the following script:

```
set echo on
set heading off
spool showogguserprivileges.log
-- Show all privileges associated with the OGG User
select
  lpad(' ', 2*level) || granted_role "USER, his roles and privileges"
from
  (
  /* THE USERS */
    select
      null     grantee,
      username granted_role
    from
      dba_users
    where
      username like upper('OGGUSER')
  /* THE ROLES TO ROLES RELATIONS */
  union
    select
      grantee,
      granted_role
    from
      dba_role_privs
  /* THE ROLES TO PRIVILEGE RELATIONS */
  union
    select
      grantee,
      privilege
    from
      dba_sys_privs
  )
start with grantee is null
connect by grantee = prior granted_role;
spool off;
```

When the previous script is run, you should see results similar to:

```
  OGGUSER
    ALTER ANY TABLE
    ALTER SESSION
    CONNECT
      CREATE SESSION
    CREATE TABLE
    DELETE ANY TABLE
    FLASHBACK ANY TABLE
    INSERT ANY TABLE
```

```
    RESOURCE
      CREATE CLUSTER
      CREATE INDEXTYPE
      CREATE OPERATOR
      CREATE PROCEDURE
      CREATE SEQUENCE
      CREATE TABLE
      CREATE TRIGGER
      CREATE TYPE
    SELECT ANY DICTIONARY
    SELECT ANY TABLE
    UNLIMITED TABLESPACE
    UPDATE ANY TABLE

22 rows selected.
```

Enable Archive Logging

Archive logging needs to be enabled in the primary databases.

To enable archive logging for Oracle GoldenGate, use these commands in SQLPlus:

- ALTER DATABASE ARCHIVELOG;
- ALTER DATABASE OPEN;

For example:

```
SQL> startup mount
ORACLE instance started.

Total System Global Area  805933056 bytes
Fixed Size                  2230680 bytes
Variable Size             469763688 bytes
Database Buffers          327155712 bytes
Redo Buffers                6782976 bytes
Database mounted.
SQL> ALTER DATABASE ARCHIVELOG;

Database altered.

SQL> ALTER DATABASE OPEN;

Database altered.
```

To view the archive logging status:

```
SQL>ARCHIVE LOG LIST;

SQL> ARCHIVE LOG LIST;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            /products/oracle/11.2.0.2.0-64bit/dbs/arch
Oldest online log sequence     201
Next log sequence to archive   203
Current log sequence           203
SQL>
```

Enabling Supplemental Logging

Once the Oracle GoldenGate user is created on both databases, you need to enable supplemental logging on the primary database. You can enable supplemental logging using the following SQL on the primary database while logged in as SYSDBA in SQLPlus:

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
ALTER SYSTEM SWITCH LOGFILE;
```



```
SELECT SUPPLEMENTAL_LOG_DATA_MIN FROM V$DATABASE;
Database altered.

SQL>
System altered.

SQL>
SUPPLEME
--------
YES

SQL> exit
```

Verifying the PSDBOWNER Table on the Standby Database

On your standby database, verify the table contents for the PSDBOWNER table. If you've cloned your standby database from the primary database, you may need to update the PSDBOWNER.DBNAME field to reflect the standby database's tnsnames alias. For example, if your newly cloned standby database's name is PSFTSTBY, then the PSDBOWNER table's DBNAME field should be PSFTSTBY:

```
SQL> SELECT * FROM PS.PSDBOWNER;

DBNAME    OWNERID
--------  --------
PSFTSTBY  SYSADM
```

Generating PeopleSoft Parameter File Input

Prior to creating and editing the Oracle GoldenGate configuration files, you need to generate PeopleSoft-specific input parameters for the GoldenGate parameter files.

To generate PeopleSoft-specific GoldenGate parameter files:

1. Open the following SQL script in your SQL editor or text editor: PS_HOME/scripts/psggconfiggenerateparmfilelists.sql.
2. Modify the following variables:
   - Modify all occurrences of <OWNER> to reflect the access ID for the database.
   - Modify all occurrences of <PATH> to reflect the script output directory path (as in, /data1/PT852/scripts/ or c:\temp\).

     Note: The ending slash is mandatory for the path.
3. Save your changes to the SQL file.
4. Log into SQLPlus using your PeopleSoft access ID.
5. Run the psggconfiggenerateparmfilelists.sql script.
6. Verify that these files appear in the output directory:
   - PSGGconfiggenerateparmfilelists.log
   - PSGGgeneratetableexcludes.txt
   - PSGGgeneratetrandatadeletes.txt

Note: You will add the output from the .txt files to the appropriate GoldenGate parameter files manually.

Creating Oracle GoldenGate Parameter Files for the Primary Database

This section describes the parameter files that you need to create and modify manually for the primary database. These files need to be created in the dirprm directory of your primary Oracle GoldenGate installation. For example, C:\OGG\primary\dirprm. The files you need to create for the primary installation are:

- mgr.prm
- priaddtrndata.oby
- configure_primary.oby
- primecap.prm
- primepmp.prm

Creating mgr.prm

Create a file named mgr.prm and add the following:

```
--
--  mgr.prm file
--
PORT 7809
PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPDAYS 3
```

Note: The port number defaults to 7809 or 7810.

Creating priaddtrndata.oby

Create a file named priaddtrndata.oby and add the following.

```
--
--  PRIADDTRNDATA.oby file
--
-- ##############################################################################⇒
##########
-- ADD Trandata Obey file for Primary
-- This file defines the tables which we are interested in having OGG capture chang⇒
es from the Transaction logs.
-- We initially specify an add TRANDATA SYSADM.* with wildcard to capture all table⇒
s.
-- We then direct OGG to ignore specific table trandata.  In our case all of the Pe⇒
opleSoft type '7' temp tables.
-- This is done by appending the output from the PSGGgeneratetrandatadeletes.txt (e⇒
g. just the generated
-- DELETE -- TRANDATA statements) after the ADD TRANDATA SYSADM.* statement
-- ##############################################################################⇒
##########
--
-- Edit and modify 'OGGUSER' to Oracle GoldenGate Admin Userid and PW
--
-- Edit and modify 'SYSADM' to PSACCESSID
--
```



```
-- Append the output from the PSGGgeneratetrandatadeletes.txt (eg. just the generat⇒
ed
-- DELETE TRANDATA       -- statements) after the ADD TRANDATA SYSADM.* statement
--
-- ##############################################################################⇒
##########
DBLOGIN USERID OGGUSER PASSWORD OGGUSER

ADD TRANDATA SYSADM.*

-- ##############################################################################⇒
##########
-- This section lists the generated DELETE TRANDATA  statements
-- ##############################################################################⇒
##########

<Copy in the generated DELETE TRANDATA statements after the preceding ADD TRANDDATA⇒
 statement.>

-- Example:
-- DELETE TRANDATA SYSADM.PS_AR_CMANRT_TAO3
-- DELETE TRANDATA SYSADM.PS_AR_CMANRT_TAO4
-- DELETE TRANDATA SYSADM.PS_AR_CMANRT_TAO5
-- <more>


-- ##############################################################################⇒
##########
-- This section lists the required static DELETE TRANDATA statements
-- ##############################################################################⇒
##########

DELETE TRANDATA SYSADM.PSLOCK
DELETE TRANDATA SYSADM.PS_SERVERMONITOR
DELETE TRANDATA SYSADM.PS_SERVERACTVTY
DELETE TRANDATA SYSADM.PS_PRCSSEQUENCE
DELETE TRANDATA SYSADM.PS_MESSAGE_LOGPARM
DELETE TRANDATA SYSADM.PS_MESSAGE_LOG
DELETE TRANDATA SYSADM.PS_AETEMPTBLMGR
DELETE TRANDATA SYSADM.PS_AERUNCONTROLPC
DELETE TRANDATA SYSADM.PS_AERUNCONTROL
DELETE TRANDATA SYSADM.PS_AELOCKMGR
DELETE TRANDATA SYSADM.PSWEBPROFHIST
DELETE TRANDATA SYSADM.PSSERVERSTAT
DELETE TRANDATA SYSADM.PSQRYTRANS
DELETE TRANDATA SYSADM.PSPRCSJOBSTATUS
DELETE TRANDATA SYSADM.PSOPRDEFN
DELETE TRANDATA SYSADM.PSIBSUBSLAVE
DELETE TRANDATA SYSADM.PSIBPUBSLAVE
DELETE TRANDATA SYSADM.PSIBFOLOCK
DELETE TRANDATA SYSADM.PSIBFAILOVER
DELETE TRANDATA SYSADM.PSIBBRKSLAVE
DELETE TRANDATA SYSADM.PSACCESSLOG
DELETE TRANDATA SYSADM.PS_PTFP_ACCESS_LOG
DELETE TRANDATA SYSADM.PS_PTFP_OPTIONS
DELETE TRANDATA SYSADM.PSIBPROFILESYNC
DELETE TRANDATA SYSADM.PSIBLOGHDR
DELETE TRANDATA SYSADM.PSIBLOGERR
DELETE TRANDATA SYSADM.PSIBLOGERRP
DELETE TRANDATA SYSADM.PSIBLOGDATA
DELETE TRANDATA SYSADM.PSIBLOGIBINFO
DELETE TRANDATA SYSADM.PSQASRUN
DELETE TRANDATA SYSADM.PSPRCSRQST
DELETE TRANDATA SYSADM.PSPRCSQUE
DELETE TRANDATA SYSADM.PSPRCSRQSTFILE
DELETE TRANDATA SYSADM.PSPRCSPARMS
DELETE TRANDATA SYSADM.PSPRCSRQSTTEXT
DELETE TRANDATA SYSADM.PSPRCSRQSTTEXT2
DELETE TRANDATA SYSADM.PS_CDM_LIST
DELETE TRANDATA SYSADM.PS_CDM_TRANSFER
```



```
DELETE TRANDATA SYSADM.PS_CDM_AUTH
DELETE TRANDATA SYSADM.PS_BAT_TIMINGS_LOG
DELETE TRANDATA SYSADM.PS_BAT_TIMINGS_DTL
DELETE TRANDATA SYSADM.PS_AE_TIMINGS_LG
DELETE TRANDATA SYSADM.PS_AE_TIMINGS_DT
DELETE TRANDATA SYSADM.PS_BAT_TIMINGS_FN
DELETE TRANDATA SYSADM.PSQRYFAVORITES
DELETE TRANDATA SYSADM.PSQRYSTATS
DELETE TRANDATA SYSADM.PSFILE_ATTDET
DELETE TRANDATA SYSADM.PSPTFILE_REF
DELETE TRANDATA SYSADM.PSPTFILE_WART
DELETE TRANDATA SYSADM.PS_PTSF_SCHED_STAT
DELETE TRANDATA SYSADM.PSPGVIEWOPT
DELETE TRANDATA SYSADM.PSPGCHARTOPT
DELETE TRANDATA SYSADM.PSPGCHRTFLRSOPT
DELETE TRANDATA SYSADM.PSPGCHTFLRSLANG
DELETE TRANDATA SYSADM.PSPGDISPOPT
DELETE TRANDATA SYSADM.PSPGGRIDOPT
DELETE TRANDATA SYSADM.PSPGQRYPROMPT
DELETE TRANDATA SYSADM.PSPGQRYPROMPLNG
DELETE TRANDATA SYSADM.PSPGCHARTOPTLNG
DELETE TRANDATA SYSADM.PSPGVIEWOPTLANG
DELETE TRANDATA SYSADM.PSPGVIEWOPTPERS
DELETE TRANDATA SYSADM.PSPGAXISPERS
DELETE TRANDATA SYSADM.PSPGAXISPERSLNG
DELETE TRANDATA SYSADM.PSPGCHARTOPTPER
DELETE TRANDATA SYSADM.PSPGQRYPRMPTPER
DELETE TRANDATA SYSADM.PSPGGRIDOPTPERS
DELETE TRANDATA SYSADM.PSPGCHTOPTPERLN
DELETE TRANDATA SYSADM.PSPGQRYPRMPTPLN
DELETE TRANDATA SYSADM.PSPGVWOPTPERLN
DELETE TRANDATA SYSADM.PSPTFILE_PRCS
DELETE TRANDATA SYSADM.PSUSEROBJTYPE
DELETE TRANDATA SYSADM.PSUSERSRCHDEFN
DELETE TRANDATA SYSADM.PSUSERPRSNLOPTN
DELETE TRANDATA SYSADM.PSVERSION
DELETE TRANDATA SYSADM.PSUSRTAPAGECUST
DELETE TRANDATA SYSADM.PSBATCHAUTH
DELETE TRANDATA SYSADM.PSBATCHAUTHLONG
DELETE TRANDATA SYSADM.PSPRCSCHLDINFO
DELETE TRANDATA SYSADM.PS_PTNVSLYTQRY
DELETE TRANDATA SYSADM.PSTREESELNUM
DELETE TRANDATA SYSADM.PSTREESELCTL
DELETE TRANDATA SYSADM.PSNVSBATCHRSTRT
DELETE TRANDATA SYSADM.PSNVSDRILLQRY
DELETE TRANDATA SYSADM.PSNVSDRLPROMPTS
DELETE TRANDATA SYSADM.PS_CDM_TEXT
DELETE TRANDATA SYSADM.PS_PRCSRQSTDIST
DELETE TRANDATA SYSADM.PSTREESELECT01
DELETE TRANDATA SYSADM.PSTREESELECT02
DELETE TRANDATA SYSADM.PSTREESELECT03
DELETE TRANDATA SYSADM.PSTREESELECT04
DELETE TRANDATA SYSADM.PSTREESELECT05
DELETE TRANDATA SYSADM.PSTREESELECT06
DELETE TRANDATA SYSADM.PSTREESELECT07
DELETE TRANDATA SYSADM.PSTREESELECT08
DELETE TRANDATA SYSADM.PSTREESELECT09
DELETE TRANDATA SYSADM.PSTREESELECT10
DELETE TRANDATA SYSADM.PSTREESELECT11
DELETE TRANDATA SYSADM.PSTREESELECT12
DELETE TRANDATA SYSADM.PSTREESELECT13
DELETE TRANDATA SYSADM.PSTREESELECT14
DELETE TRANDATA SYSADM.PSTREESELECT15
DELETE TRANDATA SYSADM.PSTREESELECT16
DELETE TRANDATA SYSADM.PSTREESELECT17
DELETE TRANDATA SYSADM.PSTREESELECT18
DELETE TRANDATA SYSADM.PSTREESELECT19
DELETE TRANDATA SYSADM.PSTREESELECT20
DELETE TRANDATA SYSADM.PSTREESELECT21
DELETE TRANDATA SYSADM.PSTREESELECT22
DELETE TRANDATA SYSADM.PSTREESELECT23
```

```
DELETE TRANDATA SYSADM.PSTREESELECT24
DELETE TRANDATA SYSADM.PSTREESELECT25
DELETE TRANDATA SYSADM.PSTREESELECT26
DELETE TRANDATA SYSADM.PSTREESELECT27
DELETE TRANDATA SYSADM.PSTREESELECT28
DELETE TRANDATA SYSADM.PSTREESELECT29
DELETE TRANDATA SYSADM.PSTREESELECT30
```

Creating configure_primary.oby

Create a file named configure_primary.oby and add the following:

```
-- ############################################################################
--
-- Edit and modify local and remote Trail Directory
--
-- ############################################################################
DBLOGIN USERID OGGUSER PASSWORD OGGUSER
add extract primecap, tranlog, begin now
add exttrail ./dirdat/pt, extract primecap
add extract primepmp, exttrailsource ./dirdat/pt
add rmttrail ./dirdat/pr, extract primepmp
```

Creating primecap.prm

Create a file named primecap.prm and add the following:

```
-- ############################################################################⇒
##########
-- Edit and modify NLS_LANG parameter as required (eg. language.territory.character⇒
)
-- Edit and modify ORACLE_HOME
-- Edit and modify Primary ORACLE_SID
--
-- Edit and modify 'OGGUSER' to Oracle GoldenGate Admin Userid and PW
--
-- Edit and modify 'SYSADM' to PSACCESSID
--
-- Copy the output from the PSGGgeneratetableexcludes.txt (eg. just the generated T⇒
ABLEEXCLUDE
-- statements) between the WILDCARDRESOLVE DYNAMIC statement and before the TABLE S⇒
YSADM.*
-- statement
--
-- ############################################################################⇒
##########
EXTRACT primecap

SETENV (NLS_LANG = "AMERICAN_AMERICA.AL32UTF8")
SETENV (ORACLE_HOME = "/products/oracle/11.2.0.2.0-64bit")
SETENV (ORACLE_SID = pg112064")

USERID OGGUSER PASSWORD OGGUSER

DISCARDFILE ./dirrpt/primecap.dsc, purge

EXTTRAIL ./dirdat/pt

WILDCARDRESOLVE DYNAMIC

-- ############################################################################⇒
##########
-- This section lists the generated TABLEEXCLUDE statements
-- ############################################################################⇒
##########

-- <Copy in the generated TABLEEXCLUDE statements after the preceding WILDCARDRESOL⇒
VE DYNAMIC
```



```
-- statement.>

-- Example:

-- TABLEEXCLUDE SYSADM.PS_AR_CMANRT_TAO3;
-- TABLEEXCLUDE SYSADM.PS_AR_CMANRT_TAO4;
-- TABLEEXCLUDE SYSADM.PS_AR_CMANRT_TAO5;
-- TABLEEXCLUDE SYSADM.PS_AR_CMANRT_TAO6;
-- TABLEEXCLUDE SYSADM.PS_AR_CMANRT_TAO7;
-- TABLEEXCLUDE SYSADM.PS_AR_CMCOLPRC_I;
-- TABLEEXCLUDE SYSADM.PS_AR_CMCOLPRC_I1;


-- ##############################################################################⇒
##########
-- This section lists the required static TABLEEXCLUDE statements
-- ##############################################################################⇒
##########


TABLEEXCLUDE SYSADM.PSLOCK;
TABLEEXCLUDE SYSADM.PS_SERVERMONITOR;
TABLEEXCLUDE SYSADM.PS_SERVERACTVTY;
TABLEEXCLUDE SYSADM.PS_PRCSSEQUENCE;
TABLEEXCLUDE SYSADM.PS_MESSAGE_LOGPARM;
TABLEEXCLUDE SYSADM.PS_MESSAGE_LOG;
TABLEEXCLUDE SYSADM.PS_AETEMPTBLMGR;
TABLEEXCLUDE SYSADM.PS_AERUNCONTROLPC;
TABLEEXCLUDE SYSADM.PS_AERUNCONTROL;
TABLEEXCLUDE SYSADM.PS_AELOCKMGR;
TABLEEXCLUDE SYSADM.PSWEBPROFHIST;
TABLEEXCLUDE SYSADM.PSSERVERSTAT;
TABLEEXCLUDE SYSADM.PSQRYTRANS;
TABLEEXCLUDE SYSADM.PSPRCSJOBSTATUS;
TABLEEXCLUDE SYSADM.PSOPRDEFN;
TABLEEXCLUDE SYSADM.PSIBSUBSLAVE;
TABLEEXCLUDE SYSADM.PSIBPUBSLAVE;
TABLEEXCLUDE SYSADM.PSIBFOLOCK;
TABLEEXCLUDE SYSADM.PSIBFAILOVER;
TABLEEXCLUDE SYSADM.PSIBBRKSLAVE;
TABLEEXCLUDE SYSADM.PSACCESSLOG;
TABLEEXCLUDE SYSADM.PS_PTFP_ACCESS_LOG;
TABLEEXCLUDE SYSADM.PS_PTFP_OPTIONS;
TABLEEXCLUDE SYSADM.PSIBPROFILESYNC;
TABLEEXCLUDE SYSADM.PSIBLOGHDR;
TABLEEXCLUDE SYSADM.PSIBLOGERR;
TABLEEXCLUDE SYSADM.PSIBLOGERRP;
TABLEEXCLUDE SYSADM.PSIBLOGDATA;
TABLEEXCLUDE SYSADM.PSIBLOGIBINFO;
TABLEEXCLUDE SYSADM.PSQASRUN;
TABLEEXCLUDE SYSADM.PSPRCSRQST;
TABLEEXCLUDE SYSADM.PSPRCSQUE;
TABLEEXCLUDE SYSADM.PSPRCSRQSTFILE;
TABLEEXCLUDE SYSADM.PSPRCSPARMS;
TABLEEXCLUDE SYSADM.PSPRCSRQSTTEXT;
TABLEEXCLUDE SYSADM.PSPRCSRQSTTEXT2;
TABLEEXCLUDE SYSADM.PS_CDM_LIST;
TABLEEXCLUDE SYSADM.PS_CDM_TRANSFER;
TABLEEXCLUDE SYSADM.PS_CDM_AUTH;
TABLEEXCLUDE SYSADM.PS_BAT_TIMINGS_LOG;
TABLEEXCLUDE SYSADM.PS_BAT_TIMINGS_DTL;
TABLEEXCLUDE SYSADM.PS_AE_TIMINGS_LG;
TABLEEXCLUDE SYSADM.PS_AE_TIMINGS_DT;
TABLEEXCLUDE SYSADM.PS_BAT_TIMINGS_FN;
TABLEEXCLUDE SYSADM.PSQRYFAVORITES;
TABLEEXCLUDE SYSADM.PSQRYSTATS;
TABLEEXCLUDE SYSADM.PSFILE_ATTDET;
TABLEEXCLUDE SYSADM.PSPTFILE_REF;
TABLEEXCLUDE SYSADM.PSPTFILE_WART;
TABLEEXCLUDE SYSADM.PS_PTSF_SCHED_STAT;
TABLEEXCLUDE SYSADM.PSPGVIEWOPT;
```

```
TABLEEXCLUDE SYSADM.PSPGCHARTOPT;
TABLEEXCLUDE SYSADM.PSPGCHRTFLRSOPT;
TABLEEXCLUDE SYSADM.PSPGCHTFLRSLANG;
TABLEEXCLUDE SYSADM.PSPGDISPOPT;
TABLEEXCLUDE SYSADM.PSPGGRIDOPT;
TABLEEXCLUDE SYSADM.PSPGQRYPROMPT;
TABLEEXCLUDE SYSADM.PSPGQRYPROMPLNG;
TABLEEXCLUDE SYSADM.PSPGCHARTOPTLNG;
TABLEEXCLUDE SYSADM.PSPGVIEWOPTLANG;
TABLEEXCLUDE SYSADM.PSPGVIEWOPTPERS;
TABLEEXCLUDE SYSADM.PSPGAXISPERS;
TABLEEXCLUDE SYSADM.PSPGCHARTOPTPER;
TABLEEXCLUDE SYSADM.PSPGQRYPRMPTPER;
TABLEEXCLUDE SYSADM.PSPGGRIDOPTPERS;
TABLEEXCLUDE SYSADM.PSPGCHTOPTPERLN;
TABLEEXCLUDE SYSADM.PSPGQRYPRMPTPLN;
TABLEEXCLUDE SYSADM.PSPGVWOPTPERLN;
TABLEEXCLUDE SYSADM.PSPGAXISPERSLNG;
TABLEEXCLUDE SYSADM.PSPTFILE_PRCS;
TABLEEXCLUDE SYSADM.PSUSEROBJTYPE;
TABLEEXCLUDE SYSADM.PSUSERSRCHDEFN;
TABLEEXCLUDE SYSADM.PSUSERPRSNLOPTN;
TABLEEXCLUDE SYSADM.PSVERSION;
TABLEEXCLUDE SYSADM.PSUSRTAPAGECUST;
TABLEEXCLUDE SYSADM.PSBATCHAUTH;
TABLEEXCLUDE SYSADM.PSBATCHAUTHLONG;
TABLEEXCLUDE SYSADM.PSPRCSCHLDINFO;
TABLEEXCLUDE SYSADM.PS_PTNVSLYTQRY;
TABLEEXCLUDE SYSADM.PSTREESELNUM;
TABLEEXCLUDE SYSADM.PSTREESELCTL;
TABLEEXCLUDE SYSADM.PSNVSBATCHRSTRT;
TABLEEXCLUDE SYSADM.PSNVSDRILLQRY;
TABLEEXCLUDE SYSADM.PSNVSDRLPROMPTS;
TABLEEXCLUDE SYSADM.PS_CDM_TEXT;
TABLEEXCLUDE SYSADM.PS_PRCSRQSTDIST;
TABLEEXCLUDE SYSADM.PSTREESELECT01;
TABLEEXCLUDE SYSADM.PSTREESELECT02;
TABLEEXCLUDE SYSADM.PSTREESELECT03;
TABLEEXCLUDE SYSADM.PSTREESELECT04;
TABLEEXCLUDE SYSADM.PSTREESELECT05;
TABLEEXCLUDE SYSADM.PSTREESELECT06;
TABLEEXCLUDE SYSADM.PSTREESELECT07;
TABLEEXCLUDE SYSADM.PSTREESELECT08;
TABLEEXCLUDE SYSADM.PSTREESELECT09;
TABLEEXCLUDE SYSADM.PSTREESELECT10;
TABLEEXCLUDE SYSADM.PSTREESELECT11;
TABLEEXCLUDE SYSADM.PSTREESELECT12;
TABLEEXCLUDE SYSADM.PSTREESELECT13;
TABLEEXCLUDE SYSADM.PSTREESELECT14;
TABLEEXCLUDE SYSADM.PSTREESELECT15;
TABLEEXCLUDE SYSADM.PSTREESELECT16;
TABLEEXCLUDE SYSADM.PSTREESELECT17;
TABLEEXCLUDE SYSADM.PSTREESELECT18;
TABLEEXCLUDE SYSADM.PSTREESELECT19;
TABLEEXCLUDE SYSADM.PSTREESELECT20;
TABLEEXCLUDE SYSADM.PSTREESELECT21;
TABLEEXCLUDE SYSADM.PSTREESELECT22;
TABLEEXCLUDE SYSADM.PSTREESELECT23;
TABLEEXCLUDE SYSADM.PSTREESELECT24;
TABLEEXCLUDE SYSADM.PSTREESELECT25;
TABLEEXCLUDE SYSADM.PSTREESELECT26;
TABLEEXCLUDE SYSADM.PSTREESELECT27;
TABLEEXCLUDE SYSADM.PSTREESELECT28;
TABLEEXCLUDE SYSADM.PSTREESELECT29;
TABLEEXCLUDE SYSADM.PSTREESELECT30;


TABLE SYSADM.*;
```

Creating primepmp.prm

Create a file named primepmp.prm and add the following:

```
-- ##############################################################################⇒
##########
-- Pump for Extract primecap
--
-- Edit and modify REMOTE Host, Port and Trail Directory
-- Edit and modify 'SYSADM' to PSACCESSID
-- ##############################################################################⇒
##########
EXTRACT primepmp

PASSTHRU
PASSTHRUMESSAGES

-- Remote Host and Trail Information
RMTHOST rtdc68005spdb.us.oracle.com MGRPORT 7810
RMTTRAIL ./dirdat/pr

-- Table Mapping Parameters
WILDCARDRESOLVE DYNAMIC

TABLE SYSADM.*;
```

Creating Oracle GoldenGate Parameter Files for the Standby Database

This section describes the parameter files that you need to create and modify manually for the standby database. These files need to be created in the dirprm directory of your standby Oracle GoldenGate installation. For example, C:\OGG\standby\dirprm. The files you need to create for the standby installation are:

- mgr.prm
- configure_standby.oby
- trgtrep.prm

Creating mgr.prm

Create a file named mgr.prm, and add the following:

```
--
--  mgr.prm file
--
PORT 7810
PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPDAYS 3
```

Note: The default GoldenGate port is 7809. When installing twice on the same host, 7810 is the other default port.

Creating configure_standby.oby

Create a file named configure_standby.oby, and add the following”

```
-- ##############################################################################⇒
##########
-- Edit and modify 'OGGUSER' to Oracle GoldenGate Admin Userid and PW
-- Edit and modify REMOTE Host and Trail
-- ##############################################################################⇒
##########
```



```
dblogin, userid OGGUSER, password OGGUSER
add checkpointtable OGGUSER.GGSCHKPT
add replicat trgtrep, exttrail ./dirdat/pr, checkpointtable OGGUSER.GGSCHKPT
```

Creating trgtrep.prm

Create a file named trgtrep.prm, and add the following:

```
-- ##############################################################################⇒
##########
-- Edit and modify NLS_LANG parameter as required (eg. language.territory.character⇒
)
-- Edit and modify ORACLE_HOME
-- Edit and modify Primary ORACLE_SID
-- Edit and modify 'OGGUSER' to Oracle GoldenGate Admin Userid and PW
-- Edit and modify 'SYSADM' to PSACCESSID
-- ##############################################################################⇒
##########
REPLICAT trgtrep

SETENV (NLS_LANG = "AMERICAN_AMERICA.AL32UTF8")
SETENV (ORACLE_HOME = "/products/oracle/11.2.0.2.0-64bit")
SETENV (ORACLE_SID = "sg112064")

USERID OGGUSER PASSWORD OGGUSER

ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/trgtrep.dsc, APPEND

DISCARDROLLOVER ON SUNDAY
ALLOWNOOPUPDATES

MAP SYSADM.*, TARGET SYSADM.*;
```

Creating Database Links and Remote Synonyms

This section contains an overview and discusses:

- creating database links
- creating remote synonyms.

Understanding Remote Synonyms with DBLINKS for PeopleSoft and GoldenGate

The default GoldenGate configuration is all changes that occur on the Primary get replicated to the Standby. For the PeopleSoft reporting infrastructure to run correctly we would also need to synchronize the following tables back to the Primary from the Standby should any update be made to the following tables:

```
SYSADM.PSLOCK
SYSADM.PS_SERVERMONITOR
SYSADM.PS_SERVERACTVTY
SYSADM.PS_PRCSSEQUENCE
SYSADM.PS_MESSAGE_LOGPARM
SYSADM.PS_MESSAGE_LOG
SYSADM.PS_AETEMPTBLMGR
SYSADM.PS_AERUNCONTROLPC
SYSADM.PS_AERUNCONTROL
SYSADM.PS_AELOCKMGR
SYSADM.PSWEBPROFHIST
SYSADM.PSSERVERSTAT
SYSADM.PSQRYTRANS
SYSADM.PSPRCSJOBSTATUS
```



```
SYSADM.PSOPRDEFN
SYSADM.PSIBSUBSLAVE
SYSADM.PSIBPUBSLAVE
SYSADM.PSIBFOLOCK
SYSADM.PSIBFAILOVER
SYSADM.PSIBBRKSLAVE
SYSADM.PSACCESSLOG
SYSADM.PS_PTFP_ACCESS_LOG
SYSADM.PS_PTFP_OPTIONS
SYSADM.PSIBPROFILESYNC
SYSADM.PSIBLOGHDR
SYSADM.PSIBLOGERR
SYSADM.PSIBLOGERRP
SYSADM.PSIBLOGDATA
SYSADM.PSIBLOGIBINFO
SYSADM.PSQASRUN
SYSADM.PSPRCSRQST
SYSADM.PSPRCSQUE
SYSADM.PSPRCSRQSTFILE
SYSADM.PSPRCSPARMS
SYSADM.PSPRCSRQSTTEXT
SYSADM.PSPRCSRQSTTEXT2
SYSADM.PS_CDM_LIST
SYSADM.PS_CDM_TRANSFER
SYSADM.PS_CDM_AUTH
SYSADM.PS_BAT_TIMINGS_LOG
SYSADM.PS_BAT_TIMINGS_DTL
SYSADM.PS_AE_TIMINGS_LG
SYSADM.PS_AE_TIMINGS_DT
SYSADM.PS_BAT_TIMINGS_FN
SYSADM.PSQRYFAVORITES
SYSADM.PSQRYSTATS
SYSADM.PSFILE_ATTDET
SYSADM.PSPTFILE_REF
SYSADM.PSPTFILE_WART
SYSADM.PS_PTSF_SCHED_STAT
SYSADM.PSPGVIEWOPT
SYSADM.PSPGCHARTOPT
SYSADM.PSPGCHRTFLRSOPT
SYSADM.PSPGCHTFLRSLANG
SYSADM.PSPGDISPOPT
SYSADM.PSPGGRIDOPT
SYSADM.PSPGQRYPROMPT
SYSADM.PSPGCHARTOPTLNG
SYSADM.PSPGVIEWOPTLANG
SYSADM.PSPTFILE_PRCS
SYSADM.PSUSEROBJTYPE
SYSADM.PSUSERSRCHDEFN
SYSADM.PSUSERPRSNLOPTN
SYSADM.PSVERSION
SYSADM.PSUSRTAPAGECUST
SYSADM.PSBATCHAUTH
SYSADM.PSBATCHAUTHLONG
SYSADM.PSPRCSCHLDINFO
SYSADM.PS_PTNVSLYTQRY
SYSADM.PSTREESELNUM
SYSADM.PSTREESELCTL
SYSADM.PSNVSBATCHRSTRT
SYSADM.PSNVSDRILLQRY
SYSADM.PSNVSDRLPROMPTS
SYSADM.PS_CDM_TEXT
SYSADM.PS_PRCSRQSTDIST
SYSADM.PSTREESELECT01
SYSADM.PSTREESELECT02
SYSADM.PSTREESELECT03
SYSADM.PSTREESELECT04
SYSADM.PSTREESELECT05
SYSADM.PSTREESELECT06
SYSADM.PSTREESELECT07
SYSADM.PSTREESELECT08
SYSADM.PSTREESELECT09
```

```
SYSADM.PSTREESELECT10
SYSADM.PSTREESELECT11
SYSADM.PSTREESELECT12
SYSADM.PSTREESELECT13
SYSADM.PSTREESELECT14
SYSADM.PSTREESELECT15
SYSADM.PSTREESELECT16
SYSADM.PSTREESELECT17
SYSADM.PSTREESELECT18
SYSADM.PSTREESELECT19
SYSADM.PSTREESELECT20
SYSADM.PSTREESELECT21
SYSADM.PSTREESELECT22
SYSADM.PSTREESELECT23
SYSADM.PSTREESELECT24
SYSADM.PSTREESELECT25
SYSADM.PSTREESELECT26
SYSADM.PSTREESELECT27
SYSADM.PSTREESELECT28
SYSADM.PSTREESELECT29
SYSADM.PSTREESELECT30
```

Normally in a GoldenGate configuration you use GoldenGate to replicate changes made on these tables from the standby to the primary database. PeopleSoft cannot take advantage of GoldenGate's bi-directional replication mechanism due to possible issues in the PeopleSoft reporting infrastructure in an environment where both databases are active at all times. The PeopleSoft reporting infrastructure (Process Scheduler, PSAESRV, PSPRCSRV, DISTSRV, and so on) will not accommodate database synchronization lag time with sequence numbers and instance numbers. To address this issue when using GoldenGate, PeopleSoft needs to utilize Remote Synonyms and Database Links to redirect all DML requested on the standby database to the primary database for a subset of the tables that make up the PeopleSoft reporting infrastructure.

Creating Database Links to the Primary Database

To create database links to the primary database:

1. Drop ‘Like’ named redirected tables in the standby database.

   Prior to creating the database links and the remote synonyms, you need to drop ‘like’ named redirected tables on the standby database by running the following script.

   PS_HOME/scripts/dropredirectedtables.sql

   Run this script against the standby database connected as the primary access ID in SQLPlus.

2. Edit the createdblinktoprimary.sql script.

   This script creates a fixed database link between the primary access ID on the standby database to the primary access ID on the primary database. Open the createdblinktoprimary.sql script in PS_HOME \scripts, and modify it as follows:

   - Replace <DBNAME> with the primary database TNSALIAS.
   - <SYSADM> with the primary access ID.
   - <PASSWORD> with the primary access ID password.

3. Run the createdblinktoprimary.sql script.

Run this script against the standby database connected as the primary access ID.

Creating Remote Synonyms

Remote synonyms are required for the tables identified as requiring DML access. If you decide to set a component to "read-only" after your analysis, then you need to include the underlying tables that require DML access to this script manually.

To create remote synonyms, you run the PS_HOME/scripts/createremotesynonyms.sql script. You must run this script against the standby database connected as the primary access ID.

Configuring Oracle GoldenGate for PeopleSoft

This section describes:

- Configuring the primary and standby databases.
- Starting the GoldenGate processes.

Configuring the Primary and Standby Databases

When working with each database, ensure that the ORACLE_HOME and ORACLE_SID environment variables are set before invoking the GoldenGate command line interpreter (GGSCI). Make sure you invoke GGSCI from $OGG_HOME or add $OGG_HOME to the $PATH environment variable.

To configure the primary and standby databases:

1. Add supplemental log groups to the primary database.

```
GGSCI > obey ./priaddtrndata.
```

2. Configure the GoldenGate processes on the primary database.

```
GGSCI > obey ./configure_primary.oby
```

3. Configure the GoldenGate processes on the standby database.

```
GGSCI > obey ./configure_standby.oby
```

Starting the GoldenGate Processes on the Primary and Standby Databases

To start the GoldenGate processes on the primary and standby databases:

1. Issue the following commands on the primary database server to start the GoldenGate processes:

```
GGSCI > start manager
GGSCI> start primecap
GGSCI > info all
```

2. Issue the following commands on the standby database server to start the GoldenGate processes:

```
GGSCI > start manager
GGSCI > info all
```

3. Issue the following command on the primary database server to start the GoldenGate pump processes.

```
GGSCI> start primepmp
GGSCI> info all
```


4. Issue the following command on the standby database server to start the GoldenGate repository processes.

```
GGSCI> start trgtrep
GGSCI> info all
```

Configuring PeopleSoft to Work with GoldenGate

To set up your PeopleSoft system to recognize both the primary and standby databases, as well to recognize the components and processes that will use the GoldenGate implementation, you need to:

- Specify the standby database information in the application server and Process Scheduler configuration files.
- Configure read-only components.
- Configure read-only processes.

The procedures for performing these tasks are identical to those for setting up Oracle ADG.

Note: Its important to understand the inherited behavior for marking a component as RO. If an initial component is marked as RO and this component calls other components, the called components will inherit the RO flag. That is, not only will the initial components SQL be redirected to the STANDBY database, all the SQL from the called components will also be redirected to the STANDBY database. Oracle has accounted for tables that may be involved with DML SQL operations from the supported components and processes listed in Understanding GoldenGate Within PeopleSoft by including those tables in the CREATEREMOTESYNONYM.SQL script. If additional components are marked RO and perform DML SQL operations on any tables not previously accounted for, then REMOTE SYNONYMS must be created for those tables as well.

Related Links

Configuring Domains
Configuring Read-Only Components
Configuring Read-Only Processes

Setting Up the PeopleSoft Installation with Oracle RAC

This section contains an overview and discusses:

- Setting up prerequisites.
- Creating the database.
- Creating Raw Devices.
- Editing the CREATEDB.SQL Script.
- Editing the UTLSPACE.SQL Script.
- Editing the XXDDL Script.
- Configuring Database Security.

- Configuring the Tnsnames and Listener Files.
- Configuring the Application Server.

Understanding the PeopleSoft Installation on Oracle RAC

An Oracle RAC configuration is a multi-Oracle instance environment that uses cluster software to communicate between different Oracle instances and cluster members.

You must use the manual database creation procedure, described in the PeopleTools installation guide if you are installing on an Oracle RAC database. The manual procedure gives you the ability to specify multiple mounting points for raw devices, edit database setup scripts, and edit Data Mover scripts.

See *PeopleTools Installation for Oracle*: *"Creating a Database Manually on UNIX"*.

See *PeopleTools Installation for Oracle*: *"Creating a Database Manually on Windows"*.

Setting Up Prerequisites

To use the Oracle RAC system with PeopleTools, the cluster environment must support the cluster file system. Before beginning the implementation, verify that your system satisfies the following requirements:

- Oracle data files and control files must be raw devices unless a cluster file system is supported by the cluster software.
- Each data file or control file is a single raw device that must be shareable to all cluster members.
- You must have installed Operating System Cluster Software.
- You must have installed the RAC version of the Oracle database.
- You must modify certain operating system parameters.

See *Oracle® Real Application Clusters Installation and Configuration Guide*for your Oracle version.

Creating the Database

Use the following guidelines in setting up the Oracle RAC database:

- Follow the instructions in the PeopleTools installation guide for creating a database manually on either UNIX or Windows.
- Edit the following scripts as per the instructions in the manual database creation instructions:
  - CREATEDB.SQL

    NN represents your Oracle database version.
  - UTLSPACE.SQL
  - *XX*DDL.SQL

    XX is a two-letter code for the PeopleSoft application.

- Create an ORACLE_SID for each Oracle instance.

  Each Oracle instance must have its own ORACLE_SID and its own Oracle initialization file. Within each specific Oracle initialization file, you must define the thread, instance_number, instance_thread, undo tablespaces, the name of a common initialization file, and service name. Within the common initialization file, you must specify the control files and the value cluster_database=true.

Creating Raw Devices

You must create raw volumes for each tablespace and control file which the database uses unless a cluster file system is supported by the cluster software. The following list is an example of raw devices and sizes used in the *XX*DDL.SQL script.

Raw Devices Function	***Example Names***	***Sizes (MB)***
Oracle Instance required:	None	NA
Control file 1	None	NA
Control file 2	None	NA
System	/dev/vg_rac/rlv_system	350
Sysaux	/dev/did/rdsk/rlv_sysaux	500
Log 1	/dev/did/rdsk/rlv_log1	190
Log 2	/dev/did/rdsk/rlv_log2	190
Log 3	/dev/did/rdsk/rlv_log3	190
Log 4	/dev/did/rdsk/rlv_log4	190
Undo 1	/dev/did/rdsk/rlv_psundo1	300
Undo 2	/dev/did/rdsk/rlv_psundo2	300
Temp	/dev/did/rdsk/rlv_pstemp	100
Default	/dev/did/rdsk/rlv_psdefault	250

Editing the CREATEDB.SQL Script

You must edit the CREATEDB.SQL script to so that the database has:

- sufficient maxinstances.
- sufficient maxlogfiles.
- different logfiles for each Oracle instance.
- a different thread for each Oracle instance.

Example: Original CREATDB10.SQL

The following is an unmodified CREATEDB.SQL file.

```
create database   <SID>
    maxdatafiles  1021
    maxinstances  1
    maxlogfiles   8
    maxlogmembers 4
    CHARACTER SET WE8ISO8859P15
    NATIONAL CHARACTER SET AL32UTF8
0240K MAXSIZE UNLIMITED<SID>/system01.dbf' SIZE 2000M REUSE AUTOEXTEND ON NEXT 1EXT⇒
ENT MANAGEMENT LOCAL
SYSAUX DATAFILE '/u01/oradata/<SID>/sysaux01.dbf' SIZE 120M REUSE AUTOEXTEND ON
NEXT  10240K MAXSIZE UNLIMITED
DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE '/u01/oradata/<SID>/temp01.dbf' SIZE
20M REUSE AUTOEXTEND ON NEXT  640K MAXSIZE UNLIMITED
M REUSE AUTOEXTEND ON NEXT  5120K MAXSIZE UNLIMITEDSID>/psundots01.dbf' SIZE 300
LOGFILE GROUP 1 ('/u01/oradata/<SID>/redo01.log') SIZE 100M,
        GROUP 2 ('/u01/oradata/<SID>/redo02.log') SIZE 100M,
        GROUP 3 ('/u01/oradata/<SID>/redo03.log') SIZE 100M;
```

Example: CREATEDB.SQL Modified for RAC

The following is an example of a CREATEDB.SQL used for setting up Oracle RAC.

```
create database   RAC
      maxdatafiles  1021
      maxinstances  2
      maxlogfiles   8
      maxlogmembers 4
      character set WE8ISO8859P15
     datafile
          '/dev/did/rdsk/rlv_system'      size 350M
SYSAUX DATAFILE '/dev/did/rdsk/rlv_sysaux' SIZE 500M
UNDO TABLESPACE PSUNDO1 DATAFILE '/dev/did/rdsk/rlv_psundo1' SIZE 300M
maxinstances  2
     logfile
          '/dev/did/rdsk/rlv_log1'         size 190M,
          '/dev/did/rdsk/rlv_log2'         size 190M;
     alter database add logfile thread 2
          '/dev/did/rdsk/rlv_log3'         size 190M,
          '/dev/did/rdsk/rlv_log4'         size 190M;
     alter database enable thread 2;
```

Editing the UTLSPACE.SQL Script

Edit the UTLSPACE.SQL script to create an additional UNDO tablespace, and to specify the correct raw devices for the tablespaces. Each Oracle instance needs its own UNDO tablespace. The following examples use the names and locations given in the section Creating Raw Devices.

Modify the script to include the following statement for the second UNDO tablespace:

```
CREATE UNDO TABLESPACE PSUNDO2
DATAFILE              '/dev/did/rdsk/rlv_psundo2'
REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED
;
```

Modify the script to specify raw devices. This example is for the PSTEMP tablespace:

```
REM * Create a temporary tablespace for database users.
REM *
CREATE TEMPORARY TABLESPACE PSTEMP
TEMPFILE              '/dev/did/rdsk/rlv_pstemp'           SIZE 300M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K
;
```


Editing the XXDDL Script

Edit the *XX*DDL.SQL script (xx is a two-letter code for your product line) to reflect the correct tablespace naming and tablespace sizing if you are using raw devices. For example, to specify the size of the PSIMAGE tablespace:

```
CREATE TABLESPACE PSIMAGE DATAFILE '/dev/did/rdsk/rlv_psimage' SIZE 100M
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
```

In addition, for Oracle RAC, autoextend of tablespaces is not supported if you are using raw devices. Do not uncomment the autoextend SQL statements in the *xx*ddl.sql script.

Creating Initialization Files

Each Oracle instance must have its own ORACLE_SID and its own Oracle initialization file. For example:

Oracle initialization file for first Oracle Instance where ORACLE_SID=RAC1:

```
InitRAC1.ora:

  instance_name=RAC1
  instance_number=1
  thread=1
  ifile= initRAC.ora
  service_names=RAC
  undo_tablespace=PSUNDO1
```

Oracle initialization file for second Oracle Instance where ORACLE_SID=RAC2:

```
InitRAC2.ora:

  instance_name=RAC2
  instance_number=2
  thread=2
  ifile= initRAC.ora
  service_names=RAC
  undo_tablespace=PSUNDO2
```

The following is the common Oracle initialization file. This file contains the following parameters that must be listed in addition to the regular Oracle initialization file parameters.

```
initRAC.ora

  compatible = 10.1.0
  cluster_database=true
  cluster_database_instances=2
  undo_management=auto
  undo_tablespace=PSUNDO1
  control_files=/dev/did/rdsk/rlv_cnt1
  open_cursors=300
```

Note: Any control_files must be changed to raw devices if there is no cluster file system support.

Configuring Database Security

There must be an entry in ps.psdbowner for each Oracle instance. The Owner ID field must be the same for all entries and the DBNAME must be a name that PeopleSoft software uses in its connection to the Oracle database. There can be multiple entries in ps.psdbowner depending on how tnsnames.ora is set up.

Make sure the ps.psdbowner table contains entries to the multiple DBNAMES and Owner IDs. For example:

```
DBNAMES      Owner ID
--------     ---------
RAC1         RAC1
RAC2         RAC1
```

To add an entry to the ps.psdbowner table, use SQL*Plus; for example:

```
sqlplus>RAC1/RAC1
Insert into ps.psdbowner values ('RAC2','RAC1');
Commit;
```

Configuring the Tnsnames and Listener Files

As a safeguard, the information on database security is defined in two locations. The tnsnames.ora file includes an ADDRESS_LIST containing an IP address and a unique CONNECT_DATA (SERVICE_NAME) for each cluster member. Each corresponding listener.ora file includes this unique SERVICE_NAME and the INSTANCE_NAME that is associated with it.

Example: TNSNAMES.ORA File

The following is a sample tnsnames.ora file:

```
rac =
  (DESCRIPTION =
   (load_balance=on)
      (ADDRESS = (PROTOCOL = TCP) (host = pt-suncluster01.peoplesoft.com) (port
= 1521))
      (ADDRESS = (PROTOCOL = TCP) (host = 192.168.225.80) (port = 1521))
   (CONNECT_DATA =
      (service_name = rac)
   )
  )


rac1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (host = pt-suncluster01.peoplesoft.com) (port =
1521))
    (ADDRESS = (PROTOCOL = TCP) (host = pt-sun29.peoplesoft.com) (port = 1521))
    (CONNECT_DATA =
      (service_name = rac)
    )
  )

rac2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (host = 192.168.225.80) (port = 1521))
    (ADDRESS = (PROTOCOL = TCP) (host = pt-sun30.peoplesoft.com) (port = 1521))
    (CONNECT_DATA =
      (service_name = rac)
    )
)
```

Example: TNSLISTENER.ORA File

The following is a sample listener.ora file.

```
SID_LIST_LISTENER_PT-SUN30 =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
```

```
      (ORACLE_HOME = /products/oracle/10.1.0.3-64bit)
      (PROGRAM = extproc)
    )
  )

SID_LIST_LISTENER_PT-SUN29 =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /products/oracle/10.1.0.3-64bit)
      (PROGRAM = extproc)
    )
  )

LISTENER_PT-SUN30 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.225.80)(PORT = 1521))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.225.78)(PORT = 1521))
      )
    )
  )

LISTENER_PT-SUN29 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = pt-suncluster01.peoplesoft.com)(PORT
= 1521))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.225.77)(PORT = 1521))
      )
    )
  )
```

To start the listener, use the following command, where <*CLUSTER_MEMBER*> is the name of the cluster member in the listener.ora file:

```
lsntrctl start <CLUSTER_MEMBER>
```

For example, to start the listener on the first cluster member in the example above, use the following command:

```
lsntrctl start LISTENER_PT-SUN29
```

Configuring the Server Domains

When configuring application server and Process Scheduler server domains, keep these items in mind:

- Make sure that the PS_MACH field in the application server configuration file corresponds to the proper IP address for the cluster member. It may be using the internal connection IP address for the cluster.
- The DBNAME specified in the domain configuration file must be one of the values in ps.psdbowner.

- Configure additional application servers and Process Scheduler servers to point to the additional RAC instance.

Working with Oracle Fine Grained Auditing

PeopleTools supports the use of Oracle Fine Grained Auditing. Oracle Fine Grained Auditing (FGA) enables you to create policies that define specific conditions that must be met in order for an audit to occur. It provides granular auditing of queries, and INSERT, UPDATE, and DELETE operations.

Using FGA creates more meaningful and focused audit trails. Rather than recording each and every access or update of a table, FGA allows you to set parameters for audits to make them more efficient. For example, you might decide to audit only under these circumstances:

- Tables accessed between 6 p.m. and 6 a.m. or only on Saturday and Sunday.
- An IP address from outside the corporate network is used.
- A specific column has been selected or updated, perhaps with a specific value.

Note: Any policies created using FGA will be preserved after upgrading your PeopleSoft application. During an upgrade, PeopleTools will store your FGA policies and then reapply them to the newly upgraded tables. Use the Oracle FGA documentation to implement FGA on your PeopleSoft implementation.

Important! FGA policies are preserved only during upgrade. If you alter a table using Application Designer, and that table has FGA policies defined for it, you will need to reapply those policies manually.

To enable the preservation of your FGA policies during upgrades, PeopleTools provides the following scripts in PS_HOME\scripts.

Script	***Description***
preupgfgareport.sql	Generates a report showing the current (pre-upgrade) FGA policies.
preupgfgaprocess.sql	Stores the current FGA policy in a temporary table, and creates these scripts: • PSCREATEFGA.SQL: recreates the existing FGA policies. • PSDISABLEFGA.SQL: disables the FGA policies (for improved performance during the upgrade only).
postupgreport.sql	Generates these reports: • Report showing tables untouched during the upgrade with regard to FGA. • Report showing FGA columns dropped during upgrade.

Script	Description
postupgfgavalidation.sql	Generates a report showing the differences between the pre and post-upgrade FGA policies, and drops the temporary table storing the FGA policies.

See also:

- *Oracle® Database Security Guide 11g Release 1 (11.1): "Verifying Security Access with Auditing," Auditing Specific Activities with Fine-Grained Auditing*.
- Your PeopleTools and PeopleSoft application upgrade documentation.

Working with Oracle SecureFiles

PeopleTools supports the use of Oracle SecureFiles. Oracle SecureFiles is a feature introduced with Oracle Database 11g and is designed to deliver high performance storage and retrieval for unstructured data files in your system. Traditionally, relational data is stored in a database while unstructured data is stored as files in the file system. Oracle SecureFiles enables you to store unstructured files in your database while maintaining performance comparable to the performance of traditional file systems, all while retaining the advantages of the Oracle database.

PeopleSoft applications generate and store a variety of unstructured data files, such as Microsoft Word, Microsoft Excel, text files, Crystal reports, SQR reports, and so on. Using Oracle SecureFiles you can apply advanced capabilities, including encryption, compression, and versioning.

To use SecureFiles the following items need to be in place:

- Tablespaces need to be created with ASSM (Automatic Segment Space Management). This is the default for PeopleSoft tablespaces.
- Compatibility level should be set to 11.1 or higher.
- Database Initialization parameter db_securefile must be set in init.ora. For example:

```
ALTER SYSTEM SET db_securefile = 'ALWAYS'
```

Note: To use compression and encryption in LOB storage parameters you must purchase Oracle Advanced Security and Oracle Advanced Compression Packs.

PeopleTools preserves any SecureFiles storage parameters during a PeopleTools upgrade. For example:

- If a table is defined as Secure File the customization would be retained.
- If a table is defined as Basic File, it would be retained as a Basic File during upgrade.

Note: PeopleTools preserves any customization with respect to SecureFiles during an upgrade only. If you have made SecureFiles customizations to a table, and perform a Build for that record in Application Designer you could lose your SecureFiles changes. You can modify the build scripts to make sure underlying LOB objects are stored as secure files, as needed.

See *Oracle® Database SecureFiles and Large Objects Developer's Guide 11g Release:"Using Oracle SecureFiles"* for more information.

Implementing the Oracle Database File System

This section contains an overview and discusses:

- Installing and configuring DBFS.
- Implementing DBFS on your PeopleSoft system.

Understanding Oracle DBFS

The Oracle Database File System (DBFS), included with Oracle Database 11g Enterprise Edition, creates a standard file system interface on top of files and directories that are stored in database tables. DBFS is similar to NFS in that it provides a shared network file system that looks like a local file system. Like NFS, there is a server component and a client component. If you use Oracle DBFS at your site, PeopleTools provides the infrastructure to use DBFS as the Report Repository with your PeopleSoft system.

For detailed information about Oracle DBFS and its architecture, see the Oracle DBFS documentation in Oracle® Database SecureFiles and Large Objects Developer's Guide 11g Release 2 (11.2).

DBFS Prerequisites

The following requirements must be met to implement Oracle DBFS on your PeopleSoft system.

- The dbfs_client host must have the Oracle client libraries installed.
- The dbfs_client can be used as a mount client only on Linux and Linux.X64 platforms.
- The dbfs_client host must have the kernel-devel package installed to configure and build FUSE.
- The dbfs_client host must have the FUSE Linux package installed.
- A group named fuse must be created and the user name that is running the dbfs_client must be a member of the fuse group.

DBFS Limitations

Please note the following DBFS limitations:

- DBFS supports most file system operations with the exception of: ioctl, locking, asynchronous I/O through libaio, O_DIRECT file opens, hard links, pipes, and other special file modes.
- Memory-mapped files are supported except in shared-writable mode.
- For performance reasons, DBFS does not update the file access time every time file data or its attributes are read.
- You cannot run programs from a DBFS-mounted file system if the direct_io option is specified.

DBFS Performance

Like any shared file system, the performance of DBFS for small files lags the performance of a local file system. Each file data or metadata operation in DBFS must go through the FUSE user mode file system, and then be forwarded across the network to the database. Therefore, each operation that is not cached on the client takes a few milliseconds to run in DBFS. For operations that involve an input/output (IO) to disk, the time delay overhead is masked by the wait for the disk IO. Naturally, larger IOs have a lower percentage overhead than smaller IOs. The network overhead is more noticeable for operations that do not issue a disk IO. When you compare the operations on a few small files with a local file system, the overhead is not noticeable, but operations that affect thousands of small files incur a much more noticeable overhead. For example, listing a single directory or looking at a single file produce near instantaneous response, while searching across a directory tree with many thousands of files results in a larger relative overhead.

Installing and Configuring DBFS

Oracle DBFS must be licensed, installed, and enabled before you can set up your PeopleSoft system to take advantage of this feature. For detailed information about installing DBFS, see Installing DBFS in *Oracle® Database SecureFiles and Large Objects Developer's Guide 11g Release 2 (11.2)*

Implementing Oracle DBFS on your PeopleSoft System

The following configuration is required to implement DBFS:

- The DBFS Report Repository must be on a mounted Linux File System.
- The PeopleSoft Webserver must be installed on Linux.
- The PeopleSoft Application Server and the PeopleSoft Database can be installed on any other supported system (Windows/Unix/Linux).

To implement DBFS on your PeopleSoft system, complete the following steps:

1. Using SQLPlus, connect as SYSDBA and grant permission for the creation of secure files.

   For example:

```
SQL> show parameter db_securefile
NAME                          TYPE               VALUE
\-----------------------\  -----------------   \-------------\
db_securefile                string              PERMITTED
```

2. Using SQLPlus, connect as SYSDBA and grant the DBFS role to ACCESSID.

   For example:

```
SQL> grant dbfs_role to SYSADM;
```

3. Using SQLPlus, connect as ACCESSID and create the tablespace for the DBFS Report Repository.

   For example:

```
SQL> connect SYSADM/SYSADM
SQL> create tablespace dbfs_tbs datafile 'd:\orattach\oradata\Pt852GA\dbfs_tbs⇒
1.dbf' size 500M reuse autoextend on next 200M segment space management auto;

Tablespace created.
```


4. *For Oracle 12c only,* using SQLPlus, connect as SYSDBA and grant unlimited quota to the DBFS Report Repository tablespace for ACCESSID.

   For example:

```
SQL> alter user SYSADM quota unlimited on dbfs_tbs;
```

5. Using SQLPlus, connect as ACCESSID and create, register, mount, and set read/write access for the DBFS Report Repository.

   For example:

```
SQL> connect SYSADM/SYSADM

CREATE FILESYSTEM REPORT REPOSITORY:
SQL> @D:\oracle\product\11.2.0\dbhome_1\RDBMS\ADMIN\dbfs_create_filesystem.sql⇒
 dbfs_tbs ReportRepository
No errors.
\-------\
CREATE STORE:
begin dbms_dbfs_sfs.createFilesystem(store_name => 'FS_REPORTREPOSITORY',
tbl_name => 'T_REPORTREPOSITORY', tbl_tbs => 'dbfs_tbs', lob_tbs => 'dbfs_tbs'⇒
,
do_partition => false, partition_key => 1, do_compress => false, compression =⇒
>
'', do_dedup => false, do_encrypt => false); end;
\-------\
REGISTER STORE:
begin dbms_dbfs_content.registerStore(store_name=> 'FS_REPORTREPOSITORY',
provider_name => 'sample1', provider_package => 'dbms_dbfs_sfs'); end;
\-------\
MOUNT STORE:
begin dbms_dbfs_content.mountStore(store_name=>'FS_REPORTREPOSITORY',
store_mount=>'ReportRepository'); end;
\-------\
CHMOD STORE:
declare m integer; begin m := dbms_fuse.fs_chmod('/ReportRepository', 16895);
end;
No errors.
```

6. Mount the DBFS Report Repository using the DBFS Client on Linux.

   For example:

```
[oracle@<hostname>]$export LD_LIBRARY_PATH=/products/oracle/11.2.0/dbhome_1/li⇒
b:/usr/lib
[oracle@<hostname>]$mount

[oracle@<hostname>]$echo "SYSADM" > dbfspassword
[oracle@<hostname>]$nohup dbfs_client SYSADM@PT852GA /mnt/dbfs < dbfspassword ⇒
&

[oracle@<hostname>]$mount

dbfs on /mnt/dbfs type fuse (rw,nosuid,nodev,max_read=1048576,default_permissi⇒
ons,user=oracle)
```

7. Use the mount point /mnt/dbfs/ReportRepository as the Report Repository location while configuring the PeopleSoft Web Server Domain.

   For example:

```
ReportRepository /mnt/dbfs/ReportRepository
```


You can specify the location for the Report Repository Path on the General page of the Web Profile during installation.

See the product documentation for *PeopleTools Installation for Oracle* for information on installing the PeopleSoft Pure Internet Architecture, setting up the Report Repository, and defining ReportRepositoryPath.

See the product documentation for *PeopleTools: Process Scheduler* for more information about "Creating Server Definitions" (PeopleTools 8.54: Process Scheduler).

Using Advanced DBFS Features

Advanced features, including encryption and compression, are available by using Oracle Wallet with Oracle DBFS. To use these advanced features, complete the following steps:

1. Create the Oracle Wallet.

   In $TNS_ADMIN/sqlnet.ora add the following entry:

```
ENCRYPTION_WALLET_LOCATION= (SOURCE=(METHOD=FILE)(METHOD_DATA= (DIRECTORY=$HOM⇒
E/wallet)))
```

2. Open the Wallet; the Wallet needs to be opened to access the Report Repository.

```
SQL> conn / as sysdba
SQL> alter system set encryption wallet open identified by "oracle";
System altered.
```

3. Create the Advanced DBFS Filesystem with Compress Type as "Medium", Encrypted.

```
SQL> conn SYSADM/SYSADM
SQL> @$ORACLE_HOME/rdbms/admin/dbfs_create_filesystem_advanced dbfs_tbs Report⇒
Repository_Encrypt compress-medium nodeduplicate encrypt partition
No errors.
\-------\-
CREATE STORE:
begin dbms_dbfs_sfs.createFilesystem(store_name =>
'FS_REPORTREPOSITORY_ENCRYPT', tbl_name => 'T_REPORTREPOSITORY_ENCRYPT', tbl_t⇒
bs
=> 'dbfs_tbs', lob_tbs => 'dbfs_tbs', do_partition => true, partition_key => 1⇒
,
do_compress => true, compression => 'medium', do_dedup => false, do_encrypt =>
true); end;
\-------\-
REGISTER STORE:
begin dbms_dbfs_content.registerStore(store_name=>
'FS_REPORTREPOSITORY_ENCRYPT', provider_name => 'sample1', provider_package =>
'dbms_dbfs_sfs'); end;
\-------\-
MOUNT STORE:
begin dbms_dbfs_content.mountStore(store_name=>'FS_REPORTREPOSITORY_ENCRYPT',
store_mount=>'ReportRepository_Encrypt'); end;
\-------\-
CHMOD STORE:
declare m integer; begin m := dbms_fuse.fs_chmod('/ReportRepository_Encrypt',
16895); end;
No errors.
```

4. Mount the DBFS Filesystem ReportRepository_Encrypt using dbfs_client. For example:

```
nohup dbfs_client SYSADM@F8538011 /mnt/dbfs < dbfspassword &
```


5. Check if the DBFS Filesystem is mounted. For example:

```
dbfs on /mnt/dbfs type fuse (rw,nosuid,nodev,max_read=1048576,default_permissi⇒
ons,user=oracle)
/mnt/dbfs/ReportRepository_Encrypt
```

6. Use the mount point /mnt/dbfs/ReportRepository_Encrypt as the Report Repository location while configuring the Web Server Domain.

Using Oracle Data Masking

PeopleTools contains pages related to Oracle Data Masking in the following location: PeopleTools, Utilities, Administration, Oracle Data Masking. However, these pages are reserved for Oracle internal use only at this time. Future releases of PeopleTools and PeopleSoft applications may support customer use of the pages.

Appendix F

Administering PeopleSoft Databases on Sybase

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 the *iso_1* character set as the default for your server.

Page Size

Select a 4k or 8k page size during server installation. The 16k page size is not supported.

EBFs

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, including:

- if the statement was recompiled.
- if the statement was using an old query plan.
- the time it took to execute.
- the time between executions.
- if the SQL was parametrized.

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.

Related Links

System and Server Administration

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

Understanding Remote Data Access

Remote Data Access is a feature used in your PeopleSoft environment if you use the Data Transformer feature available with many PeopleSoft applications, as part of Enterprise Components.

Oracle develops, certifies, and tests remote data access using the drivers provided by the supported DBMS vendors. JDBC drivers are either supplied with the database product that you already have, or can be obtained separately from your database vendor. Using these drivers ensures a predictable and well supported environment, with less potential for interoperability issues.

Note: These setup instructions for configuring a JDBC driver on the application and batch servers are needed only if you use remote database sources for the Data Transformer feature of common components. Do not install the driver otherwise.

Configuring Application Servers and Process Scheduler Servers for Remote Data Access for Informix

Before you can use remote data access with Informix on UNIX or Windows, the Informix database connectivity software (the supported version of Informix Client) must be installed on the system where the application or Process Scheduler server is running.

You must also install and configure the supported version of IBM Informix JDBC Driver 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.

See *PeopleTools Installation for Informix*, PeopleSoft Supported Platforms, and *PeopleSoft Hardware and Software Requirements* for version and installation information.

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 edit the application server configuration file, PSAPPSRV.CFG. If you're using the PeopleSoft Process Scheduler (batch server), you must edit the server configuration file, PSPRCS.CFG. These configuration files are located in the *PS_CFG_HOME* directory within the appropriate *domain* or *database name* directory.

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 Application Servers or Process Scheduler Servers for Remote Data Access for Oracle

This section discusses how to:

- Prepare to configure Oracle.
- Configure Oracle connectivity on UNIX.
- Configure Oracle connectivity on Windows.

Preparing to Configure Oracle Remote Data Access

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 Process Scheduler server is running. The supported version of database connectivity software is determined by your database version.

See PeopleSoft Supported Platforms, *PeopleSoft Hardware and Software Requirements*, and your *PeopleTools Installation for Oracle* for version and installation information.

To connect to a remote Oracle database, you must edit the application server configuration file, PSAPPSRV.CFG. If the Process Scheduler server is being used, you must edit the Process Scheduler server configuration file, PSPRCS.CFG, as well. These configuration files can be found in the *PS_CFG_HOME* directory within the appropriate *domain* or *database name* directory.

From the Remote Database Connection page (PeopleTools, Utilities, Administration, Remote Database Connection), you can specify an Oracle data source as "specific" or with TNSNAMES. "Specific" doesn't require a TNSNAMES entry, and will use the Oracle "thin" JDBC driver. However, if TNSNAMES is configured for the remote database, you can use the TNSNAMES style entry.

Configuring Oracle Connectivity for Remote Data Access 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 JDBC driver
Add to CLASSPATH=%ORACLE_HOME%/jdbc/lib/ojdbc14.jar
Add to CLASSPATH=%ORACLE_HOME%/jdbc/lib/orai18n.jar
```

Configuring Oracle Connectivity for Remote Data Access on Windows

Determine the Oracle home directory and specify it in the configuration file. For example, if Oracle the home directory is C:\Apps\DB\Oracle901, add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA Oracle JDBC driver
Add to CLASSPATH=C:\Apps\DB\Oracle901\jdbc\lib\ojdbc14.jar
Add to CLASSPATH=C:\Apps\DB\Oracle901\jdbc\lib\orai18n.jar
```

Configuring Application Servers or Process Scheduler Servers for Remote Data Access with DB2 UDB

This section discusses how to:

- Configure DB2 UDB for Linux, UNIX and Windows.
- Configure DB2 UDB for 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 Process Scheduler server is running. In addition, you must edit the PSAPPSRV.CFG and, the PSPRCS.CFG configuration files.

See PeopleSoft Supported Platforms, *PeopleSoft Hardware and Software Requirements*, and your PeopleTools installation documentation for version and installation information.

Configuring Remote Data Access for 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 edited. If the Process Scheduler server is being used, the Process Scheduler server configuration file, PSPRCS.CFG, must be edited as well. These configuration files reside in the *PS_CFG_HOME* directory within the appropriate *domain* and *database name* directory.

To 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\DB2ODBC8, you should add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA DB2 Add to CLASSPATH=C:\Apps\DB\DB2ODBC8\java1x\db2java.zip
```

Or, if DB2 UDB's home directory is /mnt/db2/v8.1/java12, you should add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA DB2 Add to CLASSPATH= lmnt/db2/v8.1/java1x/db2java.zip
```

Configuring Remote Data Access for DB2 UDB for 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 edited manually. If the Process Scheduler server is being used, the Process Scheduler server configuration file, PSPRCS.CFG, must be edited as well. These configuration files reside in the *PS_CFG_HOME* directory within the appropriate *domain* and *database name* directory..

Determine DB2 UDB's home directory, and specify it in the appropriate configuration file. For example, if DB2 UDB's home directory is C:\Apps\DB\DB2ODBC8, add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA DB2
Add to CLASSPATH=C:\Apps\DB\DB2ODBC8\java1x\db2java.zip
```

Configuring Application or Process Scheduler Servers for Remote Data Access with Sybase

Before you can use remote data access with Sybase on UNIX or Windows, the supported version of Sybase's database connectivity software, Sybase Client, must be installed on the system where the application server or Process Scheduler server is running.

See PeopleSoft Supported Platforms, *PeopleSoft Hardware and Software Requirements*, and your PeopleTools installation documentation for version and installation information.

In addition, if the application server is being used, the application server configuration file, PSAPPSRV.CFG, must be edited. If the batch server is being used, the batch server configuration file, PSPRCS.CFG, must be edited as well. These configuration files are located in the *PS_CFG_HOME* directory within the appropriate *domain* or *database name* directory.

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 by other vendors, such as Oracle. So, you need to 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 Driver for JDBC from Microsoft.

   Note: These drivers are not used by Tuxedo or the web server, so you can ignore any comments on this site about supported versions of various web servers.

2. Save the program to disk in c:\temp or desktop.
3. Double-click the sqljdbc_<version>_enu.exe file.
4. Accept all the defaults.

   The JDBC driver files are installed in C:\program files\microsoft SQL server <ver> JDBC Driver\lib.

5. Copy these files to *PS_HOME*\class:
   - For Microsoft SQL Server 2005 1.0/1.1/1.2 JDBC drivers copy sqljdbc.jar.
   - For Microsoft SQL Server 2008 2.0/3.0 JDBC drivers copy sqljdbc.jar and sqljdbc4.jar.

Appendix H

Archive Data Tool (Deprecated in PeopleTools 8.44)

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).

Note: The Archive Data tool, provided in PeopleTools 8.40-8.43, was deprecated in PeopleTools 8.44. The pages associated with the Archive Data tool are still provided with the current release of PeopleTools for upgrade compatibility, and they appear under select PeopleTools, then select Archive Data. Do not use these pages to create or maintain data archives in the current PeopleTools release.
The deprecated Archive Data tool is superseded by the current Data Archive Manager. Any new archives, should be created and maintained with the Data Archive Manager, which is described within this documentation. Access the Data Archive Manager pages by selecting select PeopleTools, then select Data Archive Manager.

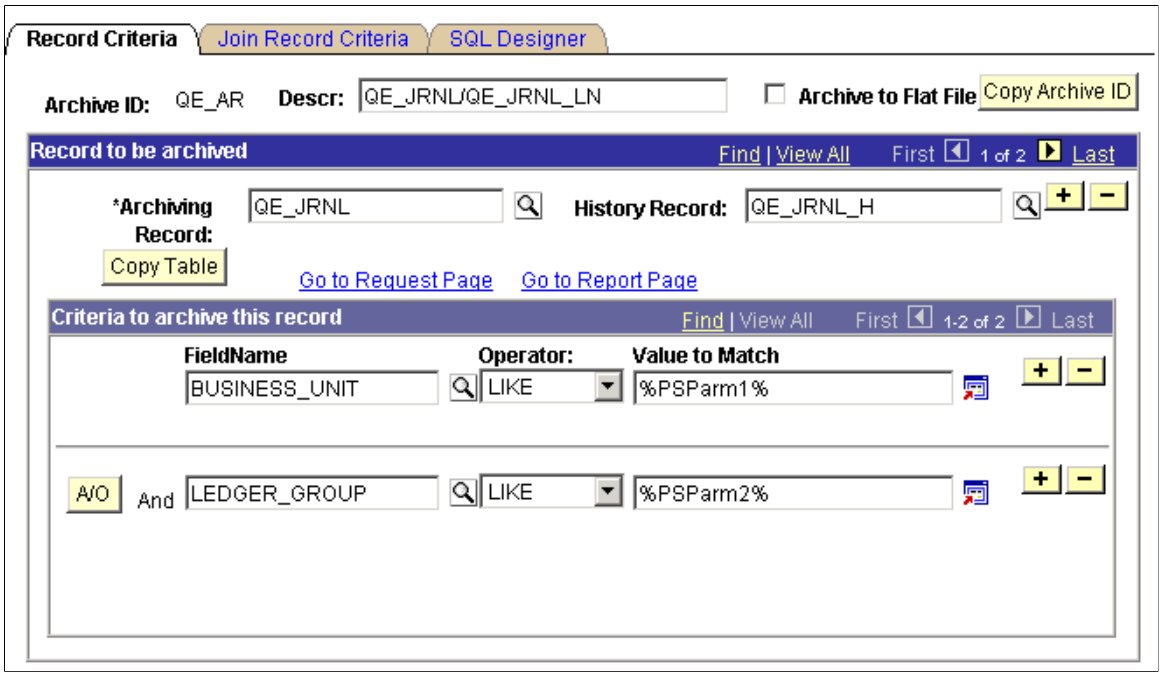
Specifying Fields and Archival Criteria

Access the Record Criteria page (PeopleTools, Archive Data, Archive Designer, Record Criteria).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Record Criteria page

This example illustrates the fields and controls on the 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.

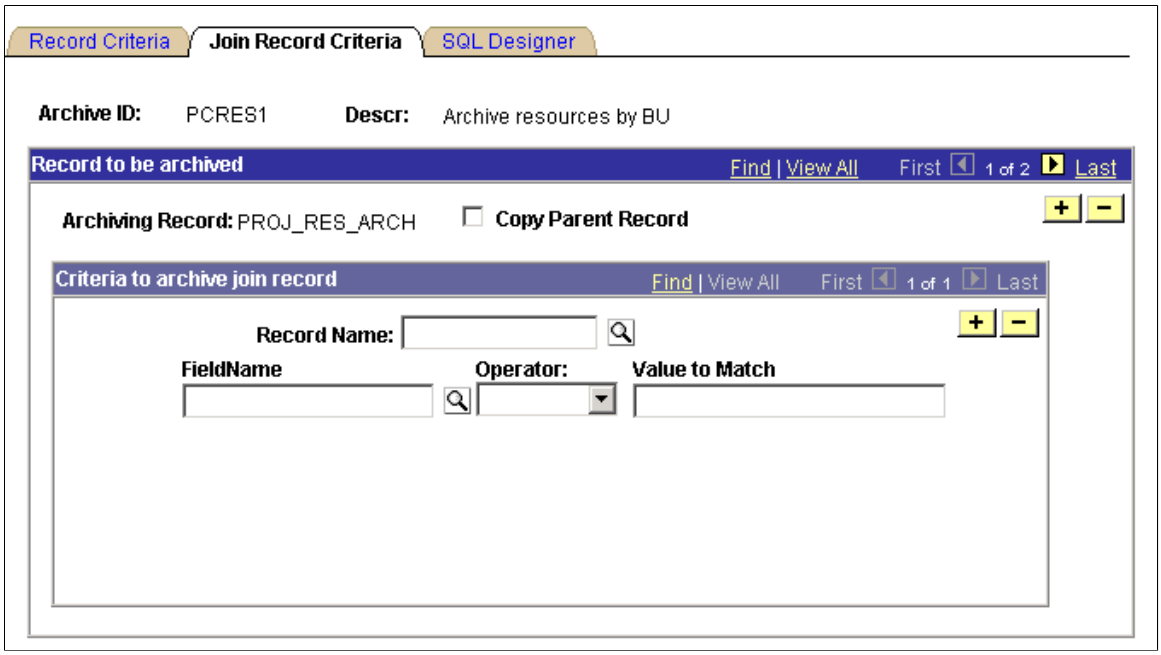
Joining Record Criteria

Access the Join Record Criteria page (PeopleTools, Archive Data, Archive Designer, Join Record Criteria).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Join Record Criteria

This example illustrates the fields and controls on the Join Record Criteria page.



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.

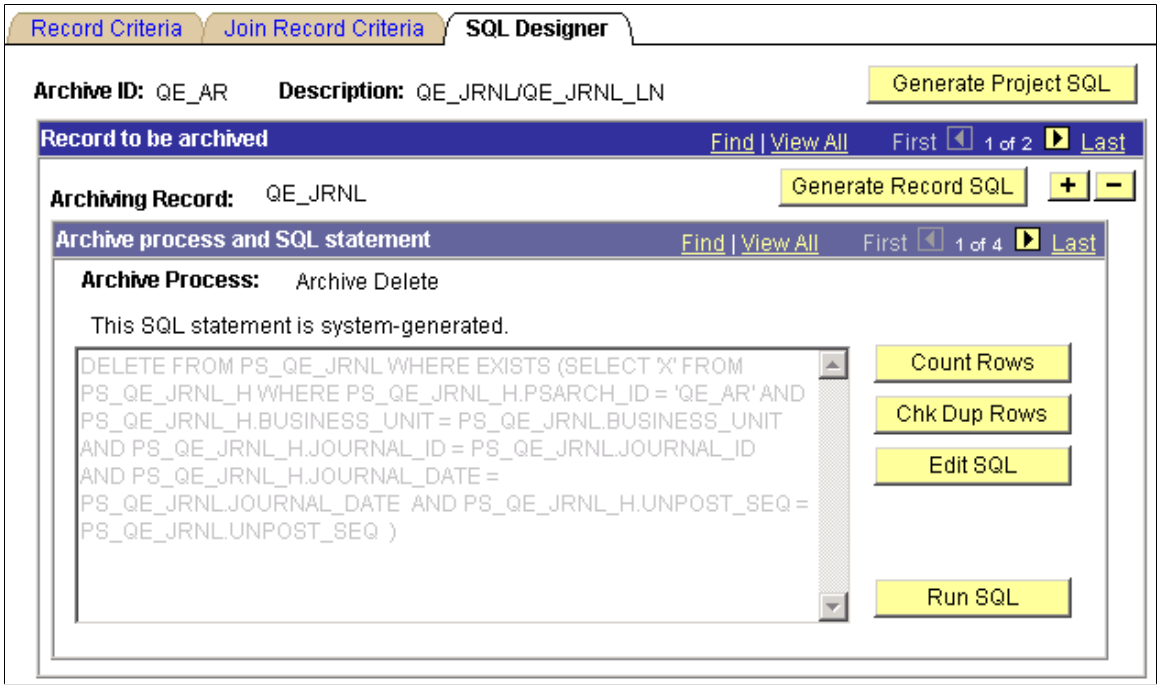
Generating and Editing SQL

Access the SQL Designer page (PeopleTools, Archive Data, Archive Designer, SQL Designer).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: SQL Designer page

This example illustrates the fields and controls on the 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.

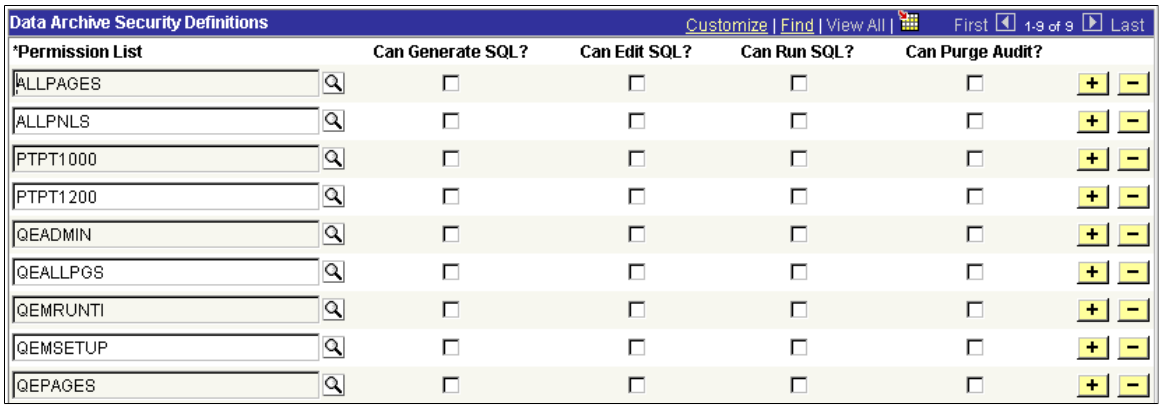
Granting Access Rights

Access the Archive Security page (PeopleTools, Archive Data, Archive Security).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Archive Security page

This example illustrates the fields and controls on the Archive Security page.



This page enables you to grant access rights to the permission lists that use PeopleSoft Archive Data Tool.

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.

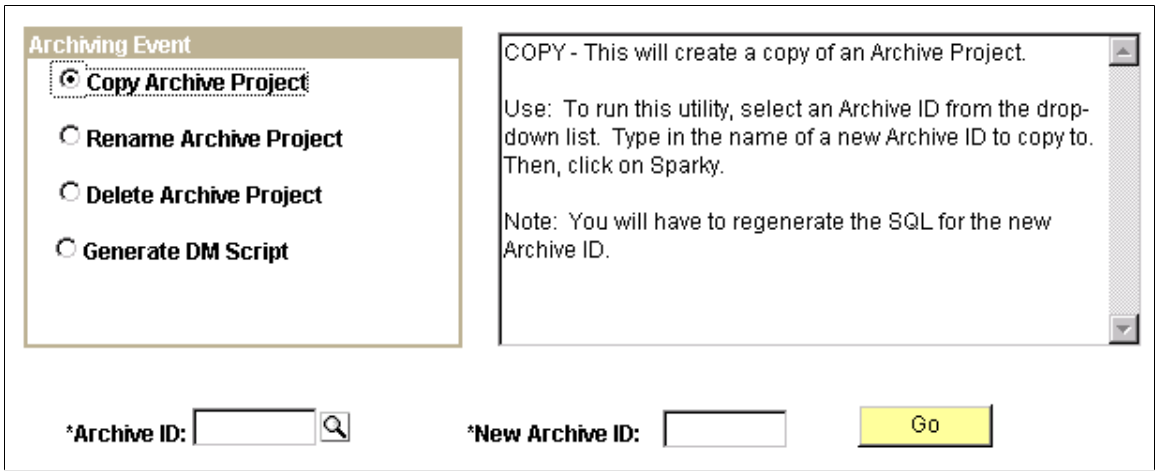
Administering Archive Projects

Access the Archive Utilities page (PeopleTools, Archive Data, Archive Utilities).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Archive Utilities page

This example illustrates the fields and controls on the 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 appear 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.

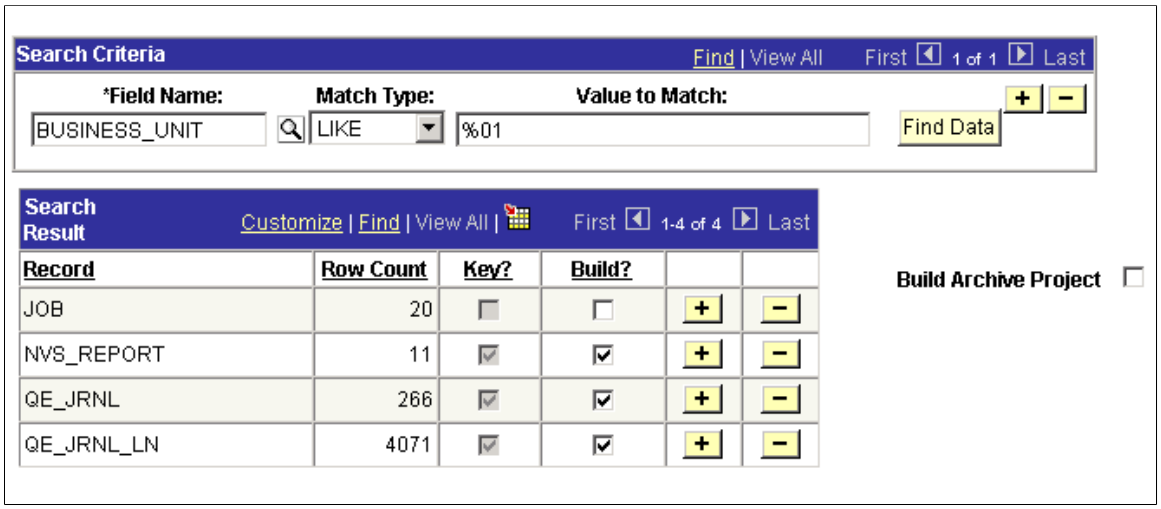
Finding Data That Meets Your Criteria

Access the Find Data page (PeopleTools, Archive Data, Find Data).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Find Data page

This example illustrates the fields and controls on the 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.

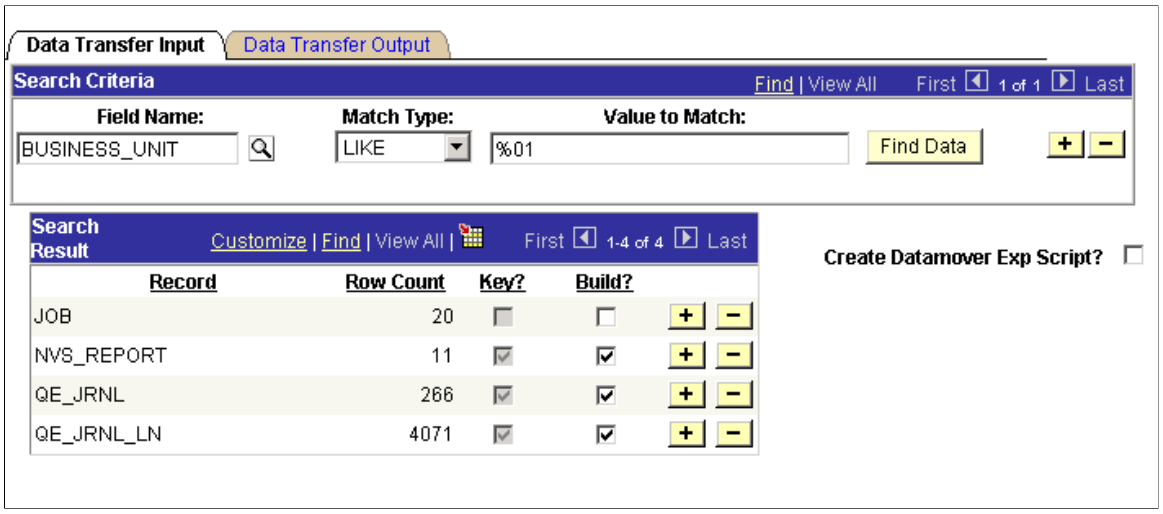
Creating Scripts to Move Data

Access the Data Transfer Input page (PeopleTools, Archive Data, Transfer Data, Data Transfer Input).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Data Transfer Input page

This example illustrates the fields and controls on the 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.

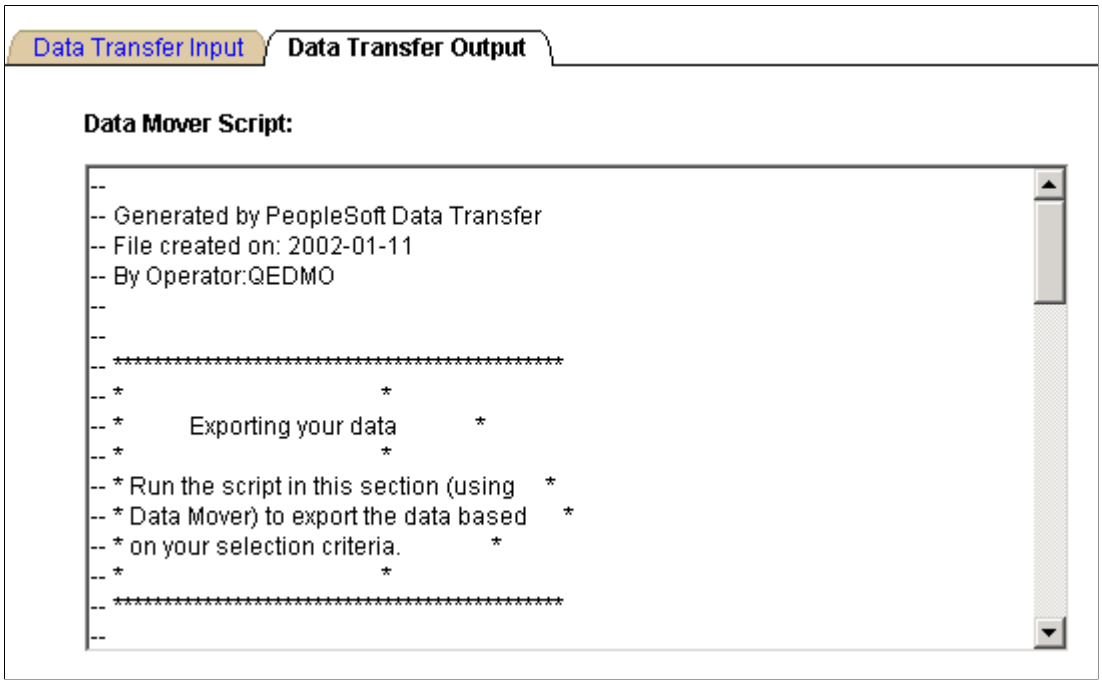
Viewing and Editing Scripts

Access the Data Transfer Output page (PeopleTools, Archive Data, Transfer Data, Data Transfer Output).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Data Transfer Output page

This example illustrates the fields and controls on the 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.

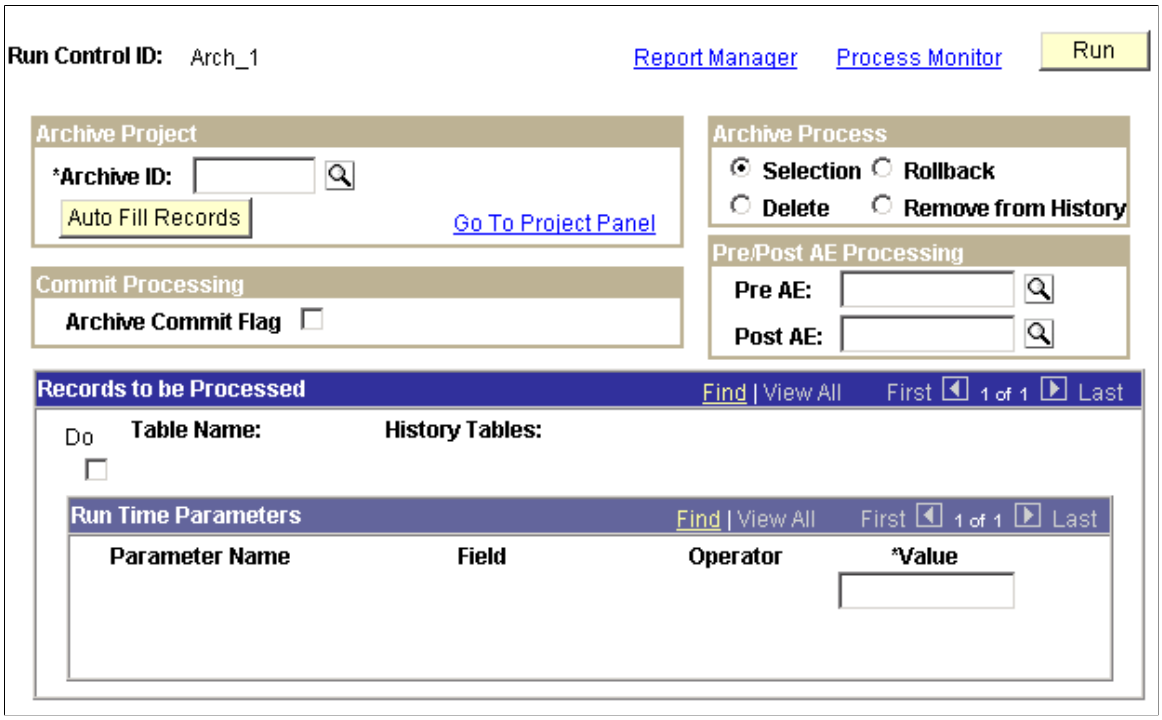
Beginning the Archiving Process

Access the Archive Data page (PeopleTools, Archive Data, Archive Data).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Archive Data page

This example illustrates the fields and controls on the 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.

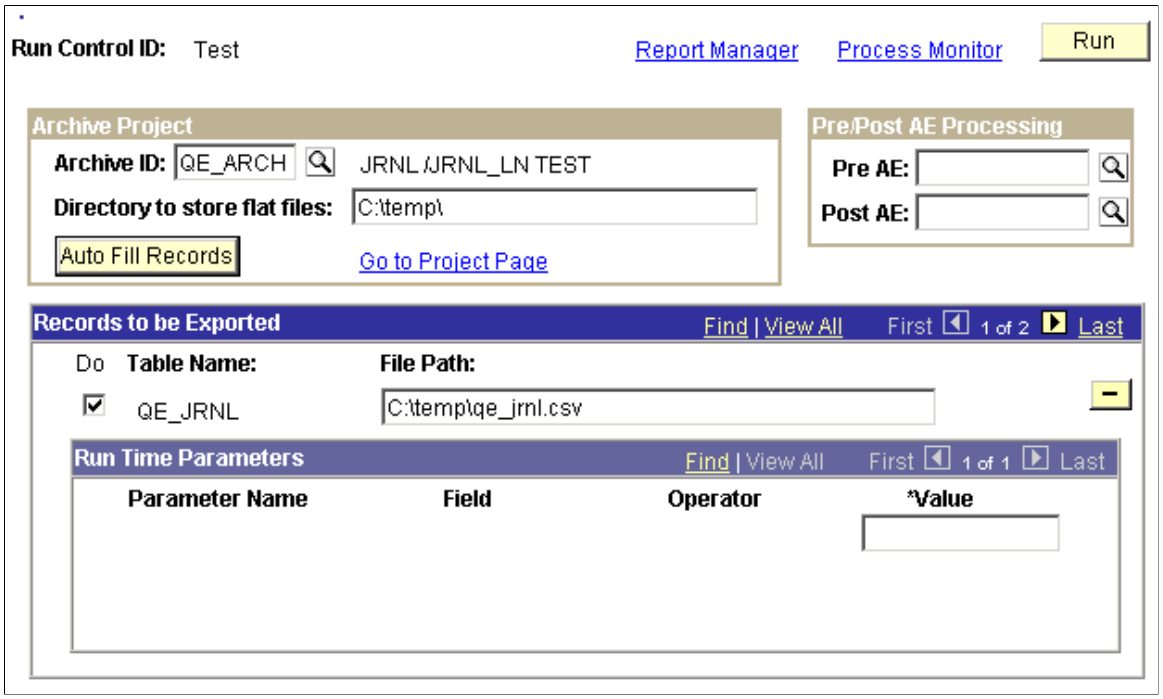
Exporting Data From Online Tables to Flat Files

Access the Archive Online to Flat Files page (PeopleTools, Archive Data, Archive Online to Flat Files).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Archive Online to Flat Files page

This example illustrates the fields and controls on the 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.

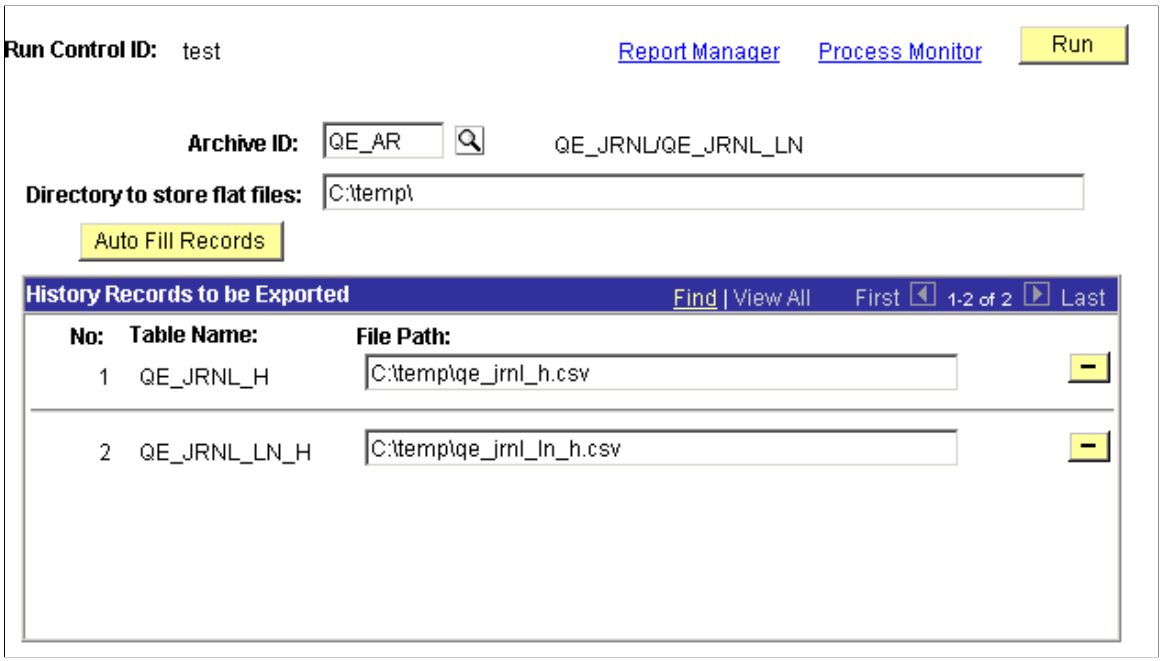
Exporting Data From History Tables to Flat Files

Access the Export History to Flat Files page (PeopleTools, Archive Data, Export History to Flat Files).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Export History to Flat Files page

This example illustrates the fields and controls on the 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.

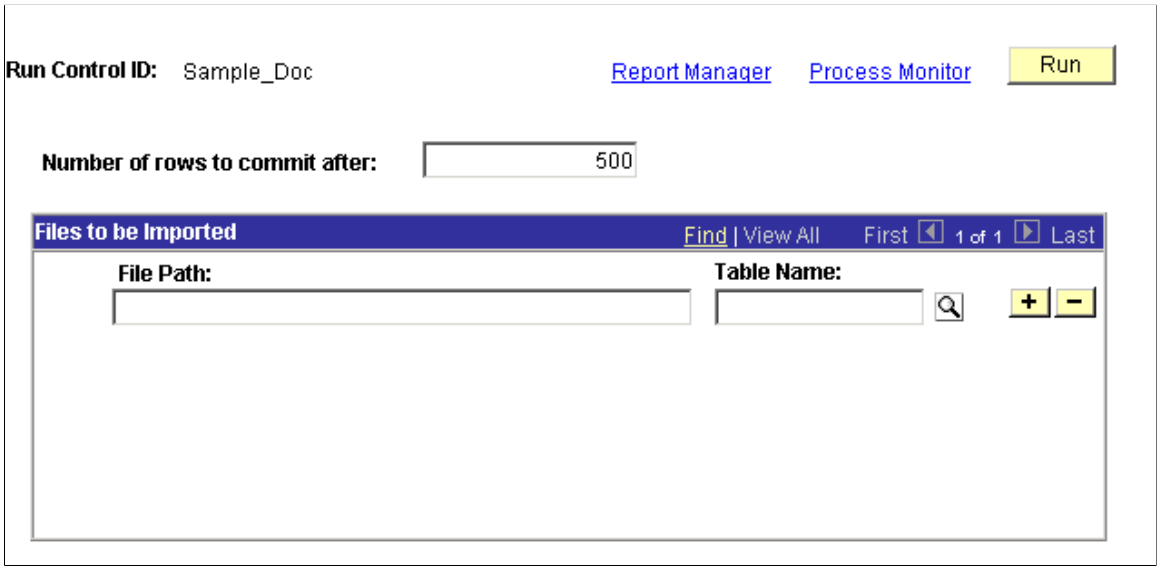
Restoring Archived Data Using Staging Tables

Access the Import From Flat Files page (PeopleTools, Archive Data, Import from Flat Files).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Import From Flat Files page

This example illustrates the fields and controls on the 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.

Running Archive Reports

Access the Archive Report page (PeopleTools, Archive Data, Archive Report).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Archive Report page

This example illustrates the fields and controls on the 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.

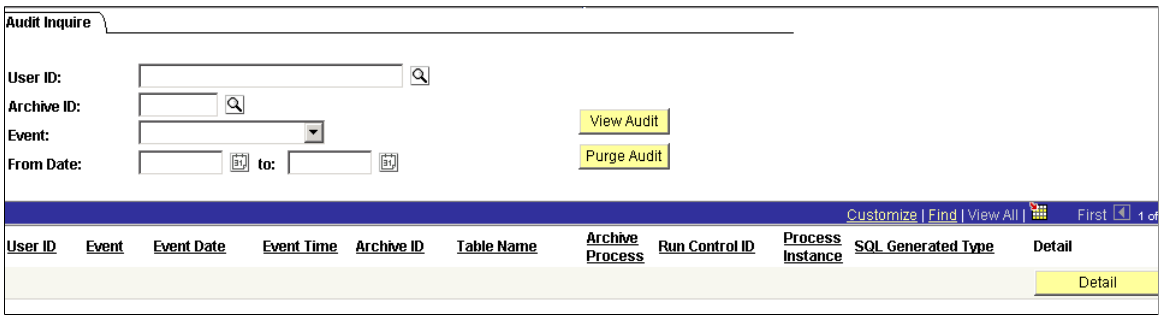
Creating an Audit Inquiry

Access the Audit Inquire page (PeopleTools, Archive Data, Audit Archiving, Audit Inquire).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Audit Inquire page

This example illustrates the fields and controls on the 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.

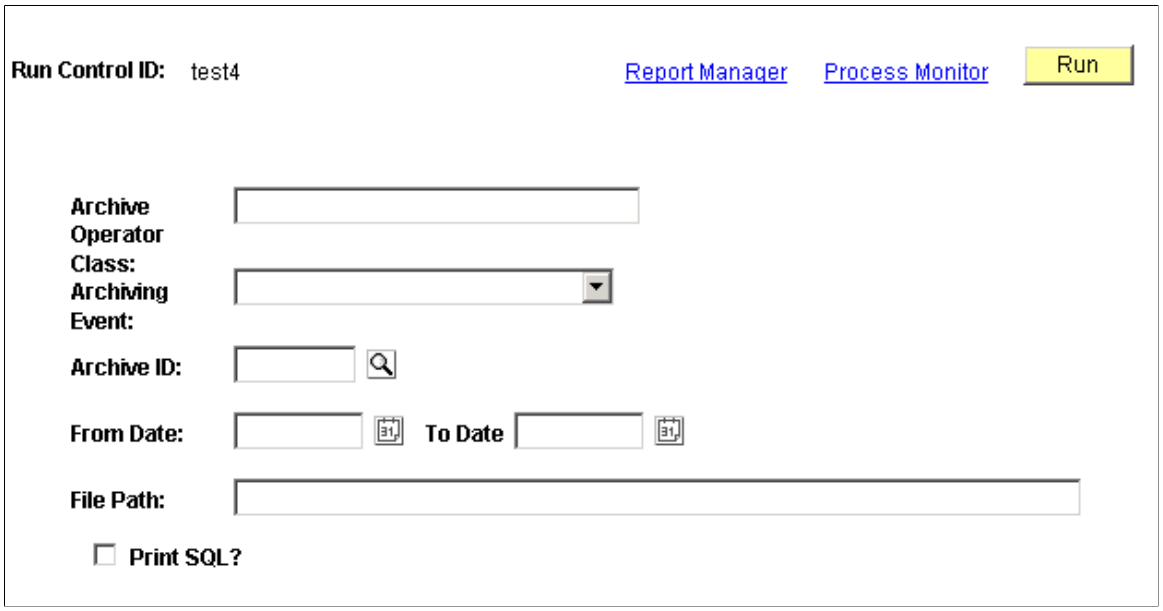
Viewing Audit Results

Access the Audit Report page (PeopleTools, Archive Data, Audit Report).

Warning! This page is associated with the deprecated Archive Data tool and should not be used to create or administer data archives. Use the Data Archive Manager (PeopleTools, Data Archive Manager) to create and administer data archives.

Image: Audit Report page

This example illustrates the fields and controls on the 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

Appendix I

Mass Change

Understanding Mass Change

Important! Mass Change is a deprecated product. Support will be maintained for this product, but no new development will be produced for Mass Change. If you used Mass Change in previous PeopleSoft releases, it is strongly recommended that you use Application Engine instead. For more information on PeopleSoft Application Engine, see "Application Engine Overview" (PeopleTools 8.54: Application Engine).

When end users manipulate the data in a PeopleSoft application, they are essentially executing SQL statements. PeopleSoft provides many of the statements necessary for updating that data, but you might occasionally need to create more.

Mass Change is a SQL generator you can use to develop and perform custom applications. Using Mass Change, a developer can set up a series of INSERT, UPDATE, or DELETE SQL statements that the end user can execute to perform business functions.

The overall structure of Mass Change is similar to that of PeopleSoft Query, except that Query *retrieves* data from the database, while Mass Change actually *updates* the database.

Mass Change is also similar to Application Engine, as far as its end results—updating the database. However, unlike Application Engine, Mass Change generates SQL for you. Also, Mass Change definitions contain no processing logic.

You can use Mass Change to:

- Perform high-volume, set oriented transactions.
- Copy data from table to table.
- Archive table data.
- Perform transactions not normally supported through PeopleSoft pages.

Mass Change design is based on three components:

- *Types* are the lowest level components. A Mass Change type defines the type of SQL statements to be generated, the records involved, and the sequence of execution. Mass Change types are defined by application developers familiar with SQL and the database design.
- *Templates* are built upon Mass Change types. Mass Change templates are used to specify which fields, if any, make up the WHERE clause of the SQL statement and which fields can be hard-coded with a particular value. Templates are typically defined by application developers.

- Mass Change *definitions* are built upon Mass Change templates, and are generally created and executed by end users. Mass Change definitions are used to specify the values and operators for each field in the statement's WHERE clause, and default fields, and to generate the actual SQL statement.

  Anyone who defines Mass Change types or templates should have both a solid understanding of SQL and an extensive knowledge of the PeopleSoft database the SQL will run against. Ideally, your end users should not have to add any Mass Change types or templates. When they create Mass Change definitions, all the necessary information will default from the type and template except for the field and operator values they enter.

Defining Types

Mass Change types determine the basic structure of the SQL statements that a Mass Change definition will generate. They define how many SQL statements will be generated, which records will be operated on, how they will be operated on, and the order in which the operations will take place.

Mass Change types and templates both require a PeopleSoft owner. Assigning an *owner* designates the PeopleSoft system from which the Mass Change type originated. The list of owners to choose from is pre-defined by PeopleSoft.

You define Mass Change types using pages in the Mass Change Type Component.

Note: To create a new Mass Change type, you must create one from scratch. Because there is no *File, Save As* menu option, you cannot clone an existing type.

To define a Mass Change Type:

1. Open or add a type.

   To open an existing type, select *Mass Change, Use, Mass Change Type* you'll be prompted for a *Mass Change Type ID.* Enter letters or numbers in the search field. Click Search to get a list of possible values.

   The *Description* page appears.

   You use the *Description* page in the Mass Change Type Component to enter details about how the type is to be used and who "owns" it.

2. Select a *PeopleSoft Owner*, deselect *Public Use,* if desired, and enter a *Description.*

   The *PeopleSoft Owner* identifies the PeopleSoft system from which the type originates. Each type *must* have an owner. When the *Public Use* checkbox is selected, the Mass Change type will be available to other users. Deselecting *Public Use* prevents other users from being able to access this type. This option is selected by default.

3. Select the Records and Join Fields page.

   This page is where you lay the foundation for the SQL statements that Mass Change generates.

4. Enter the appropriate number in the *Execution Seq* field and define the SQL statement.

   The *Execution Seq* field indicates the order in which the statements will be executed. When you add a SQL Statement, you enter the Execution Seq number manually. The number cannot exist already.

You can insert statements within an existing sequence by renumbering the subsequent steps, starting with the last step. For example, if the type consists of 5 statements and you want to add a new one between statements 3 and 4, you give statement 5 an Execution Seq value of 6, give statement 4 an Execution Seq value of 5, and create the new statement with an Execution Seq of 4.

If you select the *Free Form SQL* option, you bypass Mass Change's automated SQL generation process and will only be able to enter SQL into the last page in this Component. All other fields will be disabled.

If you select *Used for File Download/Upload,* the system will automatically generate a SQL SELECT statement based on the structure of the table you are uploading from or downloading to. This is the only time the system will generate a SQL Select statement on its own initiative.

5. Choose a *Record* and select the *SQL Action* you want to perform against it. Assign it a *Sequence* number.

   The second set of scroll arrows on the page controls which records each SQL statement will operate on, and in what manner. Each record is operated on in order, as indicated by the *Sequence* number. You manipulate record order the in same way as statement order (*Execution Seq*).

   For each *Record* you add to an SQL Statement, you select a *SQL Action* to be performed. The SQL Action you choose tells Mass Change what kind of SQL statement to generate and the Record values specify which records and fields to use in the statement.

   The following table explains the available SQL Actions:

SQL Action	Definition	Use
Delete	Removes specified rows from the record.	Eliminates data.
Insert	Inserts rows of data into the record. Data is supplied by one or more Select-type SQL Actions. If no Select action is added, adds rows with null values and any specified field default values.	Adds data.
Select Distinct	Returns one row for each unique row retrieved from the record.	Adds data without adding duplicate rows.
Select Sum	Returns one row; each field contains the field value total from all rows retrieved from the record.	Adds one row of data reflecting the total of all field values from the specified rows.
Select	When used with Insert, returns specified rows from the record.	Adds data.
Update	Updates specified rows in the record.	Changes data.

6. Add more *Records,* as desired.

   For each additional record, click the add row button. Then, repeat step 4.

7. Add more *SQL Statements,* as desired.

   For each additional SQL Statement, click the add row button. Then, repeat steps 2 through 4.

8. Save your work.

   Click *Save* to save this Mass Change type to the database.

9. Select the *Join Fields* for each *SQL Statement.*

   Once you've selected the records and actions for a Mass Change type and saved your work, you can define the *Join Fields* that Mass Change will use to build a SQL SELECT clauses for the INSERT statement(s). You'll be prompted with a list of all fields common to the records for which you've chosen a Select, Select Distinct, or Select Sum SQL action.

   Only two join fields appear on the page at one time. However, you can add as many additional rows as there are common fields. Use the scroll arrows to view all your join fields.

10. Navigate to the *Fields and Where* page.

    This page is where you enter field defaults for the records you are updating or into which you are inserting information. These are usually system fields that are key to your system processing, but of which the end user is unaware.

11. For each Insert or Update action row, define any system fields to be set.

    Enter a *Field Name* for which you want to set a default value, select the *Field Action* that will be performed on it, and a *Value* appropriate for the Field Action. To add more fields, click the add new row button.

    Field Action specifies how the field value will be updated. The *Value* field is used in a number of ways, depending on your Field Action selection. The following table explains Field Action options and usage:

Field Action	Definition	Use
Append	Adds the text specified in Value to the existing string.	Adds text to an existing text string. CHAR type only.
Count	Count.	CNT (Business_Unit) Don't put quotation marks around values. Mass Change will do this for you on character fields.
Field	Sets the field value equal to that of the field specified in Value.	Copies the field value of one field into another.
Max	Sets the field value equal to the highest value found between the current value and the value of the field specified in Value.	Finds the highest value between two fields.
Sum Field	Sets the field value to equal the current value plus the value of the field specified in Value.	Adds two field values together.

Value	Sets the field value = Value.	Hard-codes a field value.

12. For each Insert, Update, or Delete action row, specify the *Additional Where Clause,* if any.

    You can use the *Fields and Where* page to append an *Additional Where Clause* to each SQL statement. The WHERE clause you specify will be appended to the SQL generated for the record experiencing an update, delete, or insert.

    You can insert substitution parameters into the additional WHERE clause. These parameters are identified by two dollar signs ($$) before and after them. For example:

```
AND COST_BAL_VW.MIN_TRANS_DT <= MC_DEFN_AM_TRANS_DT
AND COST_BAL_VW.PROCESS_INSTANCE = $$PI$$
```

    The system will supply the bind value when the SQL statement is executed. Valid parameters are:

    - $$ARCHIVE_DT$$. Archive date.
    - $$ARCHIVE_ID$$. Archive ID.
    - $$OPRID$$. User ID.
    - $$PI$$. Process Instance.
    - $$RC$$. Return Code.

13. Save your work.

Free Form SQL Page

The last page in the Mass Change Type Component—*Free Form SQL*—enables you to enter customized SQL statements for a Mass Change type.

The only field on this page is *Free Form SQL Statement.* If this field is enabled—meaning you've selected *Freeform SQL* on the *Records and Join Fields* page, you can enter the SQL statement(s) you want the Mass Change type to use.

Note: It is not recommended to use the freeform option, as it can greatly complicate maintenance. You should be able to generate just about any SQL statement you need using the standard Mass Change pages.

Generating SQL

In the sections to come, we explain how to make Mass Change generate SQL statements based on the information you entered in the type, template, and definition pages. However, in order to know what information to provide, you should understand a bit about how Mass Change uses that information.

As we explained earlier, the SQL Action values in the Records and Join Fields page tell Mass Change what *kind* of SQL statement to generate; the Record values specify which records and fields will be used in the statement.

Insert and Select SQL actions work together. If a SQL Statement contains an Insert action, Mass Change creates an INSERT statement and uses all Select-action records for creating the associated SELECT clause. It doesn't matter where in the sequence the Selects and Inserts occur. If more than one Insert action exists in a SQL Statement row, additional INSERT statements will be generated, all using the same SELECT clause. In short, Mass Change can create *multiple* SQL statements using the information from *one* SQL Statement row.

The name SQL Statement on the Records and Join Fields page is a little misleading because Mass Change can actually generate a number of statements from one SQL Statement row, depending on your Record and SQL Action selections.

Typically, we recommend limiting your page selections to create just one SQL statement per SQL Statement row. For example, put an Update-action record in one SQL Statement row, put a Delete-action record in another, and so on. However, in cases where Inserts share a common SELECT clause, it makes sense to put them all in one SQL Statement so you don't have to set up the Select records over and over. If you have multiple Inserts that each require a *different* SELECT clause, then you must put each Insert in its own SQL Statement row.

Note: SQL Statement rows that contain Select actions with no Inserts will not generate any SQL. Select actions must be associated with at least one Insert action.

Mass Change always includes the tables MC_DEFN and MC_DEFN_*owner* in the FROM portion of SELECT clauses. The fields in these tables are used to control execution and to add null values for time, date, and datetime fields, if necessary. Therefore, if you specify an Insert-action record without specifying at least one Select-action record, Mass Change will still generate a SQL statement. However, it will result in added rows that are empty except for any system field defaults or MC_DEFN fields that match up.

In addition, Mass Change always ends each SELECT clause with the following WHERE clause:

```
WHERE MC_DEFN.MC_DEFN_ID = '<definition_ID>'
AND MC_DEFN.MC_DEFN_ID = MC_DEFN_<owner>.MC_DEFN_ID
```

If any other WHERE conditions are supplied—from join fields, criteria fields, or an additional Where clause—Mass Change appends them to this clause.

Defining Templates

Mass change templates take the SQL definition one step further. Templates enable you to control which fields will be available for the user to specify when defining a Mass Change definition, and whether those fields will be used as selection criteria or defaults. Criteria fields are used in the WHERE clause for the statement. Default fields are used in SELECT clauses in INSERT statements and in SET clauses in UPDATE statements.

You define Mass Change templates using the *Mass Change Template* Component.

Note: To create a new Mass Change template, you must create one from scratch. Because there is no File, Save As menu option, you cannot clone an existing template.

To define a Mass Change template:

1. Open or add a template.

To open an existing template, select *Mass Change, Use, Mass Change Template* you'll be prompted for a *Mass Change Template ID.* Enter one and click *OK.* The *Description* page appears.

The *Description* page in the Mass Change Template Component is where you assign a Mass Change type and an owner to the template.

2. Select a *Mass Change Type ID,* and a *PS Owner*. Enter a *Description.*

   The *Mass Change Type ID* specifies the type on which the template will be based. This sets up the default record and field selections in the next page. The *PS Owner* field identifies the PeopleSoft system from which the template originates. Each template *must* have an owner.

   The *Description* should explain how and why you would use a particular template.

3. Navigate to the *Criteria and Fields* page.

   You use the *Criteria and Fields* page to specify which fields will be used as selection criteria, and which will be used as defaults.

4. Enter your *Criteria Fields* and *Default Fields* information.

   Criteria Fields are those fields that the end user will use to retrieve rows from the Select-, Update-, and Delete-action records identified in the associated Mass Change type. In other words, these are the fields to be used in the WHERE clause of the generated SQL statement.

   Default Fields are those to which an end user can assign a default value.

   Use the scroll arrows to view each *SQL Statement.* For each statement, select the *Record* and the *Field Name* for the criteria and default fields for which the end user will enter values.

   The Mass Change type associated with a template limits which *Record* and *Field Name* can be selected for each SQL Statement. Prompting on *Record* brings up a list of valid records for each statement. When a *Record* is selected, prompting on *Field Name* shows the valid fields. For each field selected, enter a descriptive *Field Label* or use the default.

   The *Field Label* text will appear as a display-only label above the corresponding field entry box in the *Mass Change Definition* pages—to guide the end-user.

   To add another field, click the add new row button.

5. Click the *Save* button to save your work.

6. Grant yourself access to the new template.

   Before you can use the template to build a definition, you must update your Mass Change Operator Security profile to include access to the template.

Selecting Prompt Tables

If you want the user to be able to prompt for criteria and default field values when creating a Mass Change definition, you must select a prompt table for each field using the Mass Change Prompt Records page.

To add a prompt record:

1. Select *Mass Change, Use, Mass Change Prompt Records,* and follow the link to *Add a New Value.*
2. Enter the *Record (Table) Name* and *Field Name* for which you want to set up a prompt table.

   The *Prompt Records* page appears.
3. Select the *Prompt Table.*

   Enter the appropriate *Prompt Table* name, or select one from the drop-down list.
4. Click the *Save* button to save your work.

Configuring Date and Datetime Formatting

Each RDBMS handles date and datetime field data in different ways. You can instruct Mass Change to automatically format your date and datetime fields correctly using the Mass Change Datetime Parms page.

To specify the datetime formatting for an RDBMS:

1. Select *Use, Mass Change Datetime Parms.*

   The Mass Change Datetime Parms dialog appears.
2. Select a *Database Platform* and click *OK.*

   The Mass Change Datetime Parms page is displayed.
3. Specify the appropriate prefixes and suffixes.
4. Click the *Save* button to save your work.

Building Mass Change Definitions

Once you've properly set up the types, templates, prompt tables, and security, you can build a Mass Change definition. When you create a definition, all information will default from its Mass Change type and template, except for the criteria and default field values and users

You define a Mass Change definition by using the Mass Change Definition Component. We discuss the first page, Description, in the previous section. The remaining pages are where you identify criteria fields and default fields, and generate the actual SQL statements.

To add a new definition:

1. Select *Mass Change, Use, Mass Change Definition.*

   The Mass Change Definition dialog displays:

   Here is where you identify the Mass Change Template on which the definition will be based.

2. Select a *Mass Change Template,* specify an *Archive ID* and *Archive Date,* if desired, and enter a *Description.*

3. In the *Criteria and Defaults* page, specify the Criteria and Defaults information.

   In the *Criteria and Defaults* page, you specify the values of the criteria and default fields defined by the template.

   The *Criteria* rows consist of a SQL operator (edit box on the left) and a value (on the right). When you specify an operator and a value, you are completing a WHERE clause condition for the field in question. You can only enter one SQL operator per *Criteria Field.* However, you can enter multiple values by adding rows.

   For each criteria field, select an operator and a value. For each default field, enter a value.

   The following table lists the valid operators, how they are used, and an example of each one as used on a set of Business Units that includes: CORP, FRNGN, NEWGN, SUBCO, and WORLD.

Operator	Meaning	Example	Result
<	Less than	< NEWGN	CORP, FRNGN
< =	Less than or equal to	< = NEWGN	CORP, FRNGN, NEWGN
< >	Not equal to	< >NEWGN	CORP, FRNGN, SUBCO, WORLD
=	Equal to	= NEWGN	NEWGN
> =	Greater than or equal to	> = NEWGN	NEWGN, SUBCO, WORLD
>	Greater than	> NEWGN	SUBCO, WORLD
BTW	Between value A and value B	BTW CORP SUBCO	CORP, FRNGN, NEWGN, SUBCO
NBT	Not between value A and value B	NBT CORP SUBCO	WORLD
IN	In a subset of	IN CORP SUBCO	CORP, SUBCO
NIN	Not in a subset of (complement)	NIN CORP SUBCO	FRNGN, NEWGN, WORLD
LIK	Like (used with a % wildcard)	LIK NEW%	NEWGN
NLK	Not like (used with a % wildcard)	NLK %GN	CORP, SUBCO, WORLD

   The *Default Fields* box displays fields that will be updated and allows you to enter a *Value* for each field. If you've created prompt tables for the fields on this page, you can select from a list of valid values for each field.

4. In the *Application Specific Fields* page, enter any information required for the definitions for your particular application.

5. Select the *Generate SQL* page.

   You use this page to create the SQL statement(s) based on the information you've contained in the type, template, and definition. Mass Change gives you the opportunity to check the SQL text generated by a Mass Change definition before actually executing it.

   You view the SQL statement(s) by clicking the Generate SQL button. The SQL text is created and displayed in the large, display-only, edit box.

   If you're unhappy with the SQL, you can delete it, using the Clear SQL button, and rewrite the Mass Change definition, template, or type, as needed.

   Using the Count button in the lower left corner of the page displays the total number of rows affected by each statement.

   In this page, you can also opt to Execute SQL Upon Saving. If you select this checkbox, the SQL will be executed when you save the page if your Mass Change Operator Security profile authorizes you to execute definitions online. If you do not enable this option, you can save the Mass Change definition, then execute it in the background, using a run control.

6. Review the statement(s).

   Check the statement text. Be sure any FROM or WHERE clauses reference the proper tables, fields, and values. As a further test, press *Count*. This displays the total rows affected by each statement.

   Are the totals what you expected? If everything checks out, continue to the next step. If not, redefine the definition, template, or type, as necessary.

7. Select *Execute SQL Upon Saving,* if desired.

8. Click the *Save* button to save the definition.

   If you selected *Execute SQL Upon Saving,* the definition begins executing.

Creating Groups

Quite often, you may need to execute a group of Mass Change definitions in series. Mass Change functionality makes it possible to define groups of definitions and execute them all using one run control ID.

You define groups using the *Mass Change Group* page.

Like types and templates, each group must have a single *PS Owner.* This designates from which PeopleSoft system the mass change group originates.

The *Mass Change Definition* field identities the definitions in the group. You can add as many as you like. Each one should be assigned an *Execution Sequence* number, which determines the execution order.

To create a group:

1. Select *Use, Mass Change Group, Add.*

The *Mass Change Group* page displays.

2. Assign a *PS Owner,* and add *Mass Change Definitions* to the group.

   To add a definition, place your cursor in a definition field and click the add new row button.

3. Assign *Execution Sequence* numbers to each definition.

   These numbers determine the order of execution.

4. Click the *Save* button to save the group.

Executing Mass Change Definitions

You can execute Mass Change definitions either online or in the background.

Executing Online

To execute a Mass Change online, you must have permission granted in PeopleTools security (*OK To Execute MC Online?* selected).

To execute a Mass Change online:

1. In the *Mass Change Definition, Generate SQL* page, select the *Execute SQL Upon Saving* checkbox.
2. Click the *Save* button to save the page.

Executing in the Background

To execute one or more Mass Change definition(s) in the background, you'll first need to specify the definition or group you'd like to execute. You do this using the Run Mass Change page. When you run the definition or group, you'll see the Process Scheduler Request dialog.

To execute a definition or group in the background:

1. Select *Process, Mass Change, Add.*

   The *Add -- Mass Change* dialog appears.

2. Enter a *Run Control ID* and click *OK.*

   The Process -- Mass Change page opens.

   The *Mass Change Run Type* options specify the kind of execution that will occur. In this section we discuss only *Execute Single Mass Change* and *Execute Mass Change Group.*

3. Select either *Execute Single Mass Change* or *Execute Mass Change Group.*
4. Select the desired definition or group from the *Execution Parameters* boxes.
5. Click *Run.*

   The Process Scheduler Request dialog appears.

In this dialog, you specify how, when, and where to execute the mass change.

6. Enter the desired settings and click *OK*.

Performing Mass Changes in PeopleSoft Asset Management

Because of the extreme power Mass Change has to change large amounts of data in the database, we strongly recommend that you do not use Mass Change to write directly to the database. Rather, we suggest you write to intermediary tables, so that you can review the changes and approve, delete, or correct them. Then use another SQR to load the data into the database.

In the Asset Management implementation of Mass Change, we write to three load tables, INTFC_FIN, INTFC_PHY_A, and INTFC_PHY_B. INTFC_FIN holds financial information; INTFC_PHY_A and _B hold non-financial information.

Our AM Utilities window contains pages for reviewing, editing, and approving data in the load tables. This same window also displays pages for running the AM Transaction Loader SQR, which loads the data in the load tables into the Asset Management database.

This setup affords the user some protection against making massive erroneous changes to the database. As an example, we'll look at how to perform mass changes in PeopleSoft Asset Management.

Processing mass changes in PeopleSoft Asset Management consists of a definition phase and a processing phase. First, you define the selection criteria and changes to be made to the selected data, then you run the SQRs that carry out the changes you defined.

In general, you'll complete the following steps.

To run Mass Change in Asset Management:

1. Choose a mass change template and use it to create a mass change definition. Outline the criteria for selecting rows, and identify the columns and values to be changed.
2. Run the mass change SQR to select, change, and transfer the data to the load tables.
3. Preview the data for suitability (optional).
4. Run the Transaction Loader SQR to load the data from the load tables into your PeopleSoft Asset Management tables.

Downloading and Uploading Data with Mass Change

Using Mass Change, you can download data from a single table to a sequential file. Conversely, you can upload that data back to the table from the sequential file. The Download/Upload procedure consists of two phases:

- Preparing the file or table structure.
- Generating the file/populate the table.

Preparing the File or Table Structure

Process	Description
Downloads	The purpose of this phase is to rewrite the MASSLAYO.SQC to reflect the table layout of the table you're downloading. This rewrites part of the Mass Change SQR so that you can do the next step. Do this on a local version of the Mass Change SQCs, not your network copy. The *MASSLAYO.SQC To Be Updated* field is where you specify the actual copy of MASSLAYO.SQC to be rewritten. You may need to modify your SQR environment variables to ensure that you are executing the local copy of Mass Change.
Uploads	This step is similar to the first phase in downloading, explained in the previous section. The Mass Change SQR needs to rewrite itself, so that it can upload the data. Specify the file to be uploaded in the *Download/Upload Data File* field. Remember that this file contains a copy of the MASSLAYO.SQC, which was used to download the table as well as the actual table data.

Generating the File or Populating the Table

Process	Description
Downloads	After completing the first step, you're ready to download the data to a sequential file. This file contains a copy of MASSLAYO.SQC that you generated in the previous step, as well as the actual table data. Specify the filename that will store this data in the *Download/Upload Data File* field. Keep track of this file, in case you need to upload the data later on.
Uploads	After completing the first step, you're ready to upload the file data to the table it was downloaded from. Specify the file to be uploaded in the *Download/Upload Data File* field.

Note: Your SQR environment variables must be set to execute the local copy of Mass Change. A common mistake is to update a local copy of MASSLAYO.SQC in the file preparation step, but then use the network copy of MASSLAYO.SQC when executing Mass Change, due to improper setting of your SQR environment variables.