ORACLE
PeopleSoft

PeopleTools 8.60: Data Management

July 2024

ORACLE

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://docs.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 https://docs.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit https://docs.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Contents

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.

Hosted PeopleSoft Online Help

You can access the hosted PeopleSoft Online Help on the Oracle Help Center. The hosted PeopleSoft Online Help 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. The hosted PeopleSoft Online Help is available in English only.

To configure the context-sensitive help for your PeopleSoft applications to use the Oracle Help Center, see Configuring Context-Sensitive Help Using the Hosted Online Help Website.

Locally Installed PeopleSoft Online Help

If you're setting up an on-premises PeopleSoft environment, and your organization has firewall restrictions that prevent you from using the hosted PeopleSoft Online Help, you can install the online help locally. Installable PeopleSoft Online Help is made available with selected PeopleSoft Update Images and with PeopleTools releases for on-premises installations, through the Oracle Software Delivery Cloud.

Your installation documentation includes a chapter with instructions for how to install the online help for your business environment, and the documentation zip file may contain a README.txt file with additional installation instructions. See *PeopleSoft 9.2 Application Installation* for your database platform, "Installing PeopleSoft Online Help."

To configure the context-sensitive help for your PeopleSoft applications to use a locally installed online help website, see Configuring Context-Sensitive Help Using a Locally Installed Online Help Website.

Downloadable PeopleBook PDF Files

You can access downloadable PDF versions of the help content in the traditional PeopleBook format on the Oracle Help Center. 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.
& (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.

Typographical Convention	*Description*
⇒	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)

Translations and Embedded Help

PeopleSoft 9.2 software applications include translated embedded help. With the 9.2 release, PeopleSoft aligns with the other Oracle applications by focusing our translation efforts on embedded help. We are not planning to translate our traditional online help and PeopleBooks documentation. Instead we offer very direct translated help at crucial spots within our application through our embedded help widgets. Additionally, we have a one-to-one mapping of application and help translations, meaning that the software and embedded help translation footprint is identical—something we were never able to accomplish in the past.

Using and Managing the PeopleSoft Online Help

Select About This Help in the left navigation panel on any page in the PeopleSoft Online Help to see information on the following topics:

- Using the PeopleSoft Online Help.
- Managing hosted Online Help.
- Managing locally installed PeopleSoft Online Help.

PeopleTools Related Links

PeopleTools 8.60 Home Page

PeopleSoft Search and Insights Home Page

“PeopleTools Product/Feature PeopleBook Index” (Getting Started with PeopleTools)

PeopleSoft Online Help

PeopleSoft Information Portal

PeopleSoft Spotlight Series

PeopleSoft Training and Certification | Oracle University

My Oracle Support

Oracle Help Center

Contact Us

Send your suggestions to psoft-infodev_us@oracle.com.

Please include the applications update image or PeopleTools release that you’re using.

Follow Us

Icon	*Link*
	Watch PeopleSoft on YouTube
	Follow @PeopleSoft_Info on X.
	Read PeopleSoft Blogs
	Connect with PeopleSoft on LinkedIn

Chapter 1

Getting Started with Data Management

Data Management Overview

This section provides a short overview of topics explained further for data management.

PeopleSoft Data Mover

The sections on PeopleSoft Data Mover are shifted to the Application Life Cycle Management book. For related information to the Data Mover see, Lifecycle Management Guide.

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 desupported 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.

- 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 for z/OS.
- Oracle.

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.

   PeopleSoft applications are installed with the PeopleSoft deployment packages and include PeopleTools, database, application server, web server (PIA), and Process Scheduler.

   See the product documentation for *PeopleSoft 9.2 Application 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 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, new feature overviews (previously named release notes), technical briefs, and PeopleSoft product documentation.

Chapter 2

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

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**, and then insert the PSARCHIVE_SBR sub-record.
6. Save the record.
7. Build the table by selecting **Build, Current Object**.

- Select the following build options: **Create Tables and Create Indexes**.
- Select the following build execute options: Execute and Build script.
- Click **Build**.

Understanding Archiving Techniques

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 be accessed through reports only. 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 Pure 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.

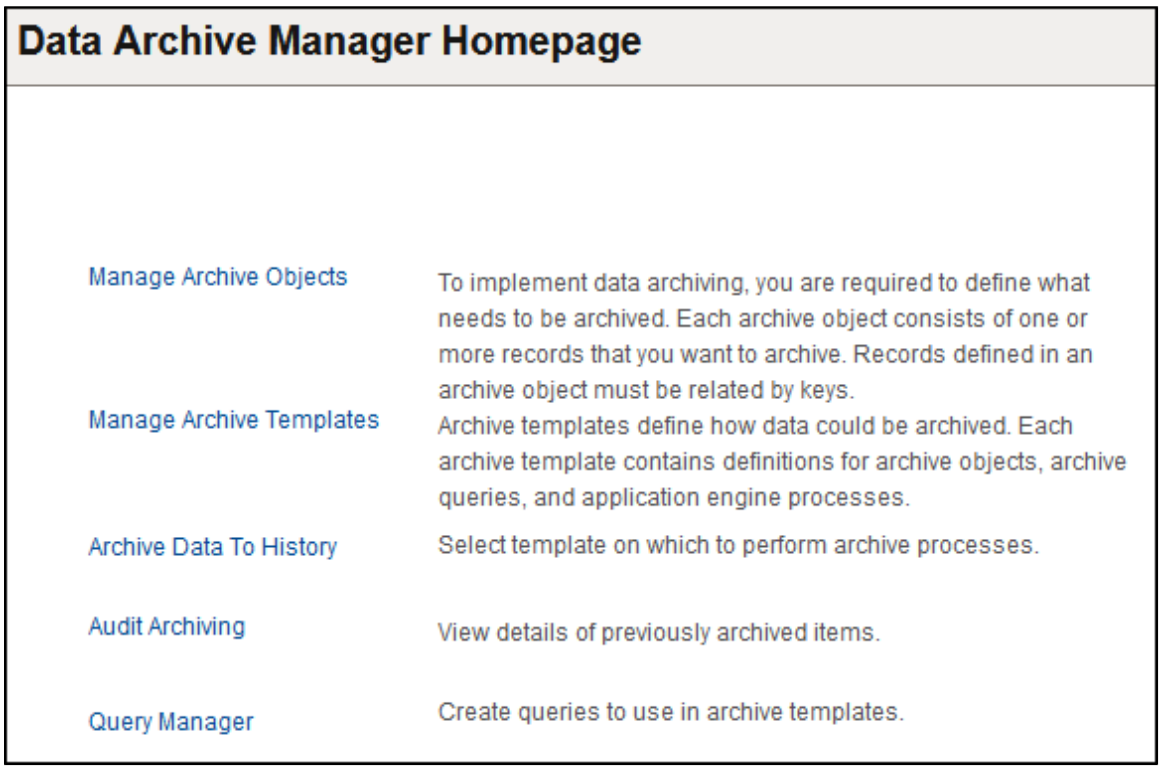
Note: This potential limitation applies *only* to Oracle databases. No other databases are affected.

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** > **Data Archive Manager** to access the PeopleSoft Data Archive Manager Homepage.

This example illustrates the fields and controls on the Data Archive Manager Homepage.



Field or Control	***Description***
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.

Field or Control	*Description*
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**).

Field or Control	*Description*
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.

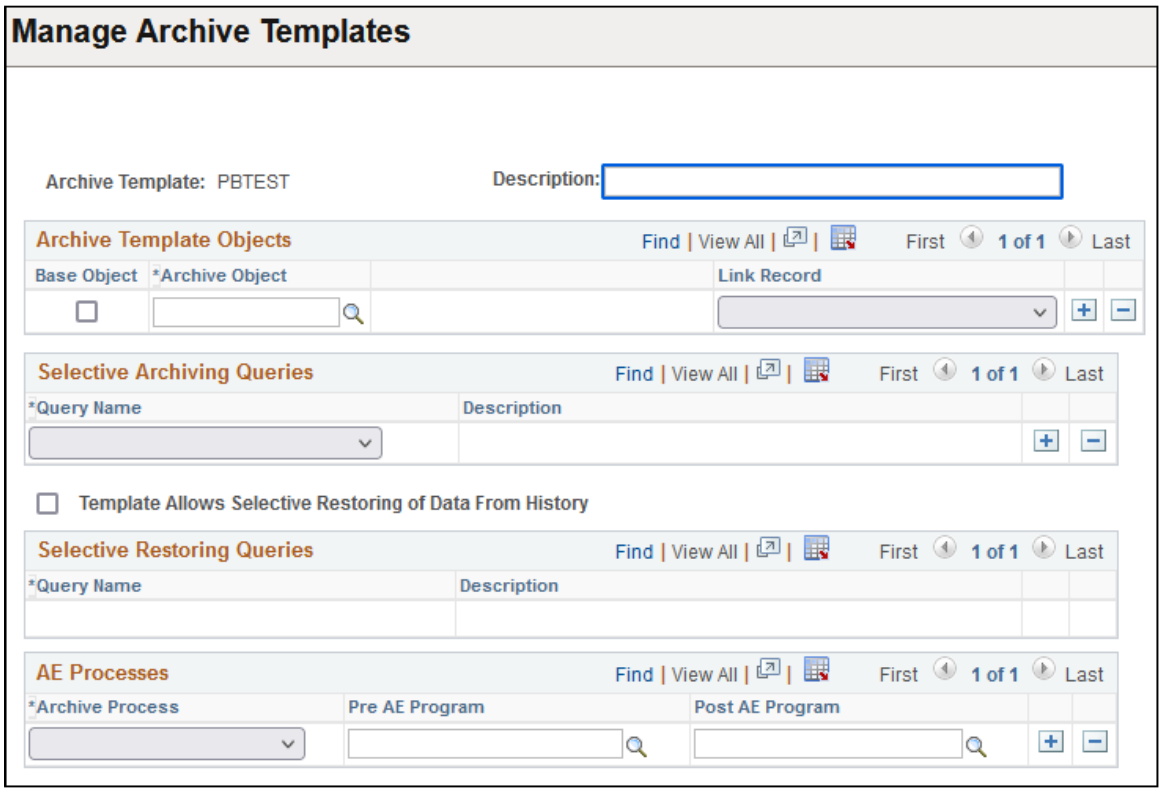
Managing Archive Templates

This section discusses how to manage archive templates.

Managing Archive Templates

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

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



Archive Template Objects

Field or Control	*Description*
Base Object	Select this check box if the archive object that you select is the base object of this archive template. By 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

Field or Control	*Description*
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

Field or Control	*Description*
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

Field or Control	*Description*
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.

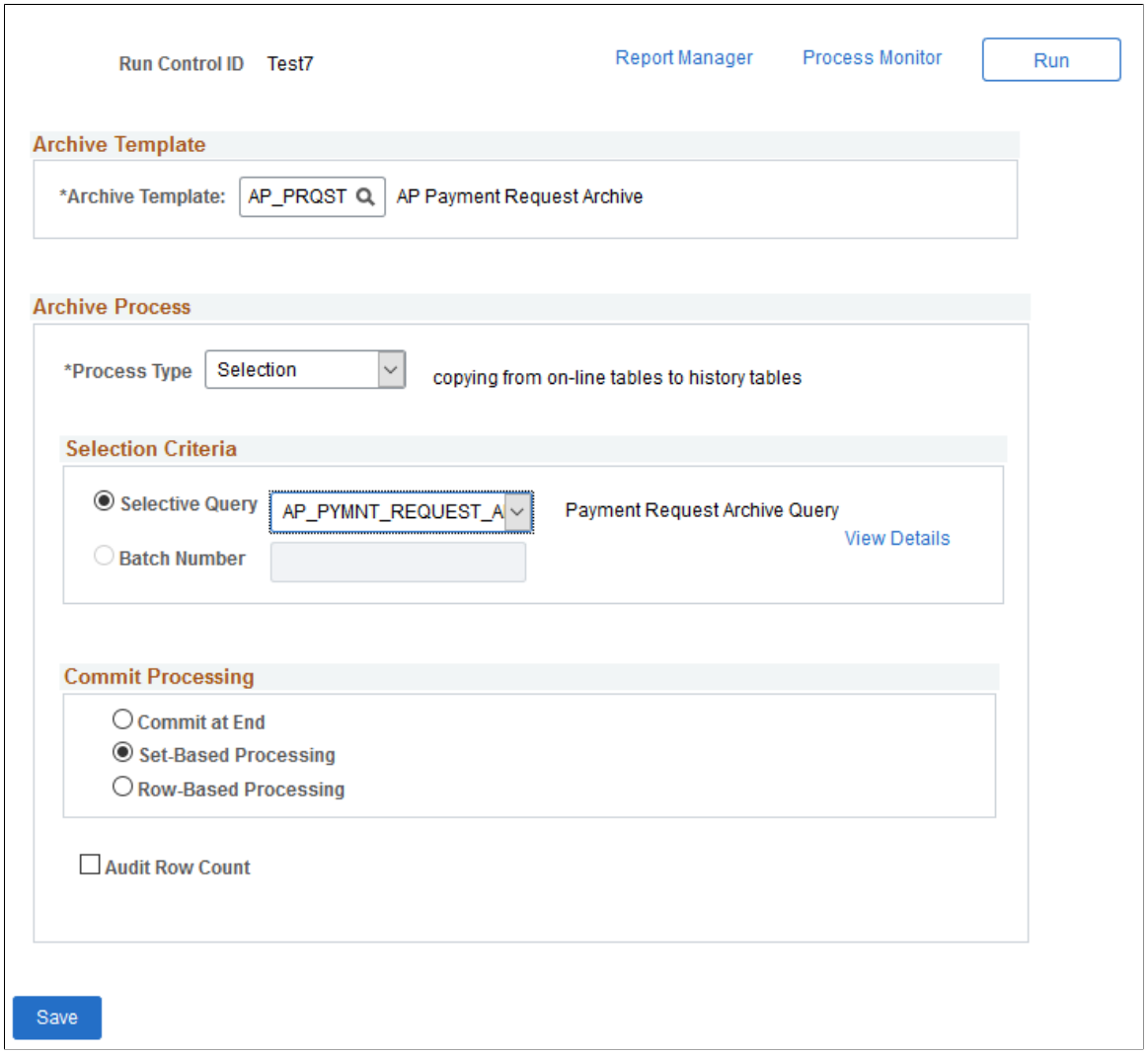
Managing Archive Jobs

This section discusses how to manage the archived jobs.

Defining Archive Jobs

Access the Archive Data To History page (select **PeopleTools** > **Data Archive Manager** > **Archive Data to History**).

This example illustrates the fields and controls on the Archive Data To History page



Field or Control	*Description*
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.

Field or Control	*Description*
Run	Click to run this batch job after defining the archive process and commit processing.
Report Manager	Click this to view reports. For more information about viewing reports, see “Viewing Reports” (Process Scheduler)
Process Monitor	Click this to review the status of scheduled or running processes. For more information about viewing the status of processes, see “Viewing the Status of Processes” (Process Scheduler)

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.

Field or Control	*Description*
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 select the **Define Binds** link.
View Details	Select this link to access the Archive Run Control Details page to view the SQL and row counts of this batch job. You see this link when you pick a relevant Selective Query. **Note:** If you're using bind variables, you must save the run control data before selecting the View Details link. See Viewing Details.
Define Binds	Select this link to access the Define Query Bind Variables page. You see this link when the query contains binds. See Defining Archive Query Binds.

Field or Control	*Description*
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 or search for 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

Field or Control	*Description*
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 run the Pre-AE program, then it will process all of the tables in the Archive Template, then it will run 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.

Field or Control	*Description*
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 (select **PeopleTools** > **Data Archive Manager** > **Archive Data to History** and click View Details).

Field or Control	*Description*

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_BATCH⇒`
	
	`NUM = 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 Archive Query Binds page to define query bind variables. Select **PeopleTools** > **Data Archive Manager** > **Archive Data to History**, and click the Define Binds link.

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 Archive Query Binds 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**).

Field or Control	*Description*
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	Displays the archive IDs returned by the search. If you search by archive ID, this corresponds to that archive ID. If you search by date, there may be several archive IDs.
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.

Field or Control	*Description*
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 3

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 **Build** on 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

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, indexed views, materialized query tables, materialized 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, you can view the .LIS file using any text editor.

The following tables list the names of each query that DDDAUDIT performs on the PeopleSoft system. The tables elaborate on the query type, if any row is returned on running the query then what does it suggest, and how to resolve the inconsistencies.

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

Index Queries

Query	*The Returned Rows Indicate*	*Resolution*
INDEX-1	Indexes are defined in Application Designer and not found in the database.	Use Application Designer to create the index.

Indexed View Queries [Microsoft SQL Server]

Query	*The Returned Rows Indicate*	*Resolution*
IVIEWS-1	Indexed views or summary tables which are defined in Application Designer, but are not found in the database.	Use Application Designer to Build the indexed views or summary tables.
IVIEWS-2	Indexed views or summary tables which are defined in the database, but are not found in the Application Designer.	Drop the indexed view or summary table if it is not valid. Otherwise, define a new indexed view or summary table in the Application Designer.
IVIEWS-3	Indexed views or summary tables which are missing a related language record.	Use the Application Designer to add a related language record for the indexed views or the summary tables.
IVIEWS-4	Indexed views whose keys are not defined.	Use the Application Designer to set the keys for the indexed views.

Materialized Query Table Queries [DB2 ZOS]

Query	*The Returned Rows Indicate*	*Resolution*
MQT-1	Materialized query table that are defined in Application Designer, but are not found in the database.	Use the Application Designer to build the materialized query table.
MQT-2	Materialized query table that are defined in the database, but are not found in Application Designer.	Drop the materialized query table if it is not valid. Otherwise, define a new materialized query table in the Application Designer.
MQT-3	Materialized query table that are missing a related language record.	Use the Application Designer to add a related language record for the materialized query tables.
MQT-4	Materialized query tables whose table spaces are not set.	Use the Application Designer to set table space for the materialized query tables.

Materialized View Queries [Oracle]

Query	*The Returned Rows Indicate*	*Resolution*
MVIEWS-1	Materialized views which are defined in the Application Designer, but are not found in the database.	Use the Application Designer to build the materialized views.
MVIEWS-2	Materialized views which are defined in the database, but are not found in the Application Designer.	Drop the materialized view if it is not valid. Otherwise, define a new materialized view in the Application Designer.
MVIEWS-3	Materialized views which are missing a related language record.	Use the Application Designer to add a related language record for materialized views.

Table Queries

Query	*The Returned Rows Indicate*	*Resolution*
TABLE-1	SQL table names are defined in the Data Designer that are not blank and not the same as the record name.	Use the Application Designer to enter the record name as the **Non-Standard SQL Table Name.**

Query	*The Returned Rows Indicate*	*Resolution*
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.	Drop the table if it is not valid. Otherwise, define a new record in the 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 the Application Designer to assign table spaces to these tables.
TABLE-5	Table contains more than 500 fields.	Use the Application Designer to adjust the number of fields on the table, as needed.

Trigger Queries

Query	*The Returned Rows Indicate*	*Resolution*
TRIGGER-1	Trigger defined in the Application Designer and not found in the database.	Delete the definition if it is not needed. Otherwise, use the Application Designer to create the trigger in the database.

View Queries

Query	*The Returned Rows Indicate*	*Resolution*
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. On Db2 z/OS, if there are dependent view failures after creating materialized views, add those views into a project and recreate views.
VIEWS-2	Views are defined in the database and not found in the Data Designer.	Drop the view if it is not valid. Otherwise, define a new view in the Application Designer.

Running SYSAUDIT

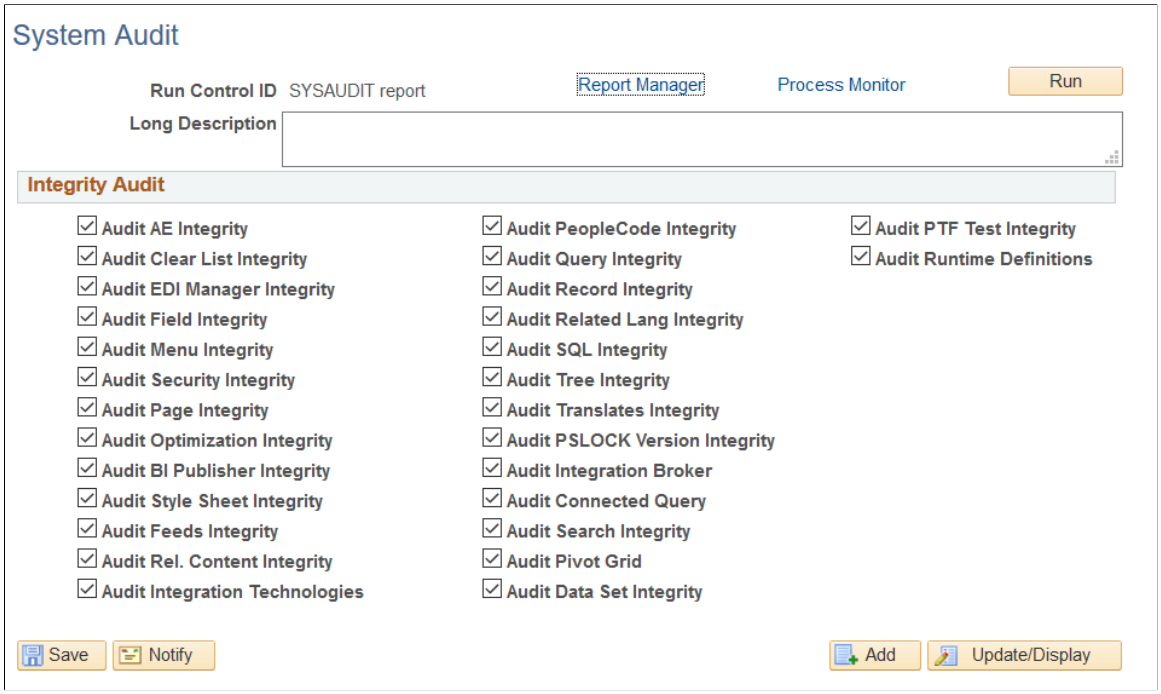
This section provides an overview of how to run SYSAUDIT and discusses the audits in detail.

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.** Add a new Run Control ID or select an existing one to access the System Audit page. Select the appropriate check boxes to run the audits that you want.

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



Term	*Definition*
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.

Term	*Definition*
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.

Term	*Definition*
Audit Data Set Integrity	Audits the definitions for application data sets.
Audit PTF Test Integrity	Audits the PeopleSoft Test Framework tests.
Audit Runtime Definitions	Audits system tables to locate runtime definitions, which should not be combined with design time definitions. See “Understanding the Managed Object Classes” (PeopleCode API Reference)

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” (Process Scheduler)

Application Data Set Integrity

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

Query	*Description*	*Resolution*
ADS-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.
ADS-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.
ADS-03	This audit is run against PSADSDEFN table to check if all the specified extended 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 extended application classes for the data set may become missing.	Copy the associated application package and its PeopleCode together to avoid missing extended application classes.
ADS-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.
ADS-05	This audit is run against PSADSDEFNITEM table to check if a record is included in a data set definition, then its related language record must also be in the same data set definition.	Add the missing related language record to the data set.
ADS-06	This audit is run against PSADSGROUPPROP table to check if all the three Managed Objects (MO) properties — Managed Object Name, Managed Object Delete Table and Managed Object Version are set for a data set definition if one of them is set. PivotGrid is an exception in this case. It only has the Managed Object Name property.	Add missing MO properties to the data set.

Query	*Description*	*Resolution*
ADS-07	This audit is run against PSADSDEFNITEM and PSRECFIELDDB tables to check if the key fields in a parent record in a data set also exist in all of its child records in the same data set, and they are also the key fields in all the child records. This form of inconsistency may occur when the record structure changes after the data set is created.	Redesign the child records so that the flagged fields are key fields.
ADS-08	This audit is run against PSADSDEFN and PSADSGROUPPROP tables to check the owner of the cacheable MO data sets is PeopleTools (PPT).	Set Owner ID to PeopleTools for the flagged data sets.
ADS-09	This audit is run against PSADSGROUPMEMB and PSRECFIELDDB tables to verify if the translatable fields in mergeable groups in base tables are also in the mergeable groups for the corresponding Related Language tables in a data set.	Add related language record field to the mergeable group.
ADS-10	This audit is run against PSADSRELATION table to verify if the To-data sets exist in definition references in an ADS relation. This inconsistency may occur when a data set with defined relations is copied to another database, but its related data sets are missed.	Copy all To-data sets together with the flagged ADS definition.
ADS-11	This audit is run against PSADSRELMAP table to verify the record fields references of From-data sets existing in a relation mapping definition. This inconsistency may occur when the flagged record structure changes.	Redesign the flagged records so that all missing fields are in the record.
ADS-12	This audit is run against PSADSRELMAP table to verify the record fields references of To-data sets in a relation mapping definition exist. This inconsistency may occur when the flagged record structure changes.	Redesign the flagged records so that all missing fields are in the record.

Query	*Description*	*Resolution*
ADS-13	This audit is run against PSADSRELPROP table, to verify the relation properties in an ADS relation mapping are defined in PSADSPROPVALUE table. This inconsistency may occur when updating relation properties in PSADSRELPROP table while missing sync corresponding property definitions in PSADSPROPVALUE table.	In case of creating or upgrading an environment, the corresponding DMS (Data Mover Script) has already been generated for the flagged data, re-run the DMS to sync the properties. In case of updating data set definition from a source environment, export the flagged missing property definitions in PSADSPROPVALUE table from the source database, then import the data to the target database.
ADS-14	This audit is run against PSADSGROUPMEMB table to verify all record fields references in a data set group definition exist. This occurs when the flagged record structure changes.	Redesign the flagged records so that all missing fields are in the record.
ADS-15	This audit is run against PSADSGROUPPROP table to verify the delete table for an MO data set exists. The delete table has the value of Managed Object Delete Table property. This inconsistency may occur when copying a data set for a new MO that do not exist on target.	Copy the missing delete table together with the flagged data set.
ADS-16	This audit is run against PSADSGROUPPROP table to verify existing corresponding row in the tables PSLOCK and PSVERSION for Managed Object Version property value for cacheable MO data sets. The MO_VERSION object property is used to indicate that the object is cached at run time. The property value should be the OBJECTTYPENAME of a row in PSVERSION and PSLOCK tables associated with the managed object. This information is used to synchronize cache for a cacheable managed object when the object is updated. Objects that are not cached should omit this property. All cacheable MOs must have C++ managers.	Remove the MO_VERSION object property if the object is not cached or add the appropriate row to PSVERSION and/or PSLOCK tables.
ADS-17	This audit is run against PSADSDEFNITEM table to check if there are duplicate order keys in data set root record.	Remove the duplicate order key field from the flagged record.

Query	*Description*	*Resolution*
ADS-18	This audit is run against PSPROJADSBIND tables to verify every ADS in a project has the corresponding ADS definition.	Import or create the ADS definition. The database from which the ADS project was exported will have the correct ADS definition.
ADS-19	This audit is run against PSPROJBINDITEM and PSADSDEFNITEM tables to verify the binding fields references in ADS project instances are existing. This occurs when the root record structure for the flagged data set changes.	Redesign the root records so that all flagged missing bind symbols are in the record, and they are also the key fields.
ADS-20	This audit is run against PSADSGROUPMEMB table to verify if any orphaned ADS group members references exist that do not have corresponding data sets. This happens in rare cases, only if the data set definition is corrupt.	Recreate the flagged data sets with the groups and members.
ADS-21	This audit is run against PSADSGROUPPROP table to verify if any orphaned data set properties exist that do not have a corresponding data set. This happens in rare cases, only if the data set definition is corrupt.	Recreate the flagged data sets with the properties.
ADS-22	This audit is run against PSADSRELMAP table to verify if any orphaned relation mappings exist that do not have corresponding From-data set. This can only happen if the data set definition is corrupt.	Recreate the flagged data sets with the relation mappings.
ADS-23	This audit is run against PSADSRELPROP table to verify if any orphaned relation properties exist that do not have corresponding data set definition. This happens in rare cases, only if the data set definition is corrupt.	Recreate the flagged data sets.
ADS-24	This audit is run against PSADSGROUP table to verify if any orphaned data set groups exist that do not have corresponding data set definition. This happens in rare cases, only if the data set definition is corrupt.	Recreate the flagged data sets.

Query	*Description*	*Resolution*
ADS-25	This audit is run against PSADSRELMAP table to verify if any orphaned relation mappings exist that do not have corresponding data set relation definition. This happens in rare cases, only if the data set definition is corrupt.	Recreate the flagged relation with the mappings.
ADS-26	This audit is run against PSADSRELPROP table to verify if any orphaned relation properties exist that do not have corresponding data set relation definition. This happens in rare cases, only if the data set definition is corrupt.	Recreate the flagged relation.
ADS-27	This audit is run against PSADSGROUPMEMB table to verify if any orphaned ADS group members exist that do not have corresponding data set group. This happens in rare cases, only if the data set definition is corrupt.	Recreate the flagged group with the flagged record fields.
ADS-28	This audit is run against PSADSGROUPPROP table to verify if there are any orphaned data set properties that do not have a corresponding data set group. This happens in rare cases, only if the data set definition is corrupt.	Recreate the flagged group with the flagged properties.
ADS-29	This audit is run against PSADSRELMAP table to verify if there are any orphaned relation mappings that do not have corresponding To-data set. This may occur when copying a data set with defined relations, but not related data sets.	Copy all To-data sets together with the flagged ADS definition.

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.
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.

Query	*Description*	*Resolution*
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.
AE-06	This audit lists the AE base section details without 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.
AE-07	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 clean up these orphans.
AE-08	This audit lists the AE Call Section actions referring to nonexistent sections	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 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.

Query	*Description*	*Resolution*
AE-09	This audit lists the AE Log Message 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 actions without an AE step.	If the affected program 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. 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 clean up these orphans.
AE-11	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-12	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.

Query	*Description*	*Resolution*
AE-13	This audit lists the orphaned AE SQL objects.	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-14	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-15	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-16	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.

Query	*Description*	*Resolution*
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 interface.
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.
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.
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.

Query	*Description*	*Resolution*
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.

Query	*Description*	*Resolution*
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.
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.

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

Query	*Description*	*Resolution*

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_IBS⇒`
		
		`O`
		`WHERE PTFP_DATATYPE_ID='<D⇒`
		
		`ATATYPEID>'`

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_ATT⇒`
		
		`R`
		`WHERE PTFP_DATATYPE_ID='<D⇒`
		
		`ATATYPEID>'`

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='<FEEDI⇒`
		
		`D>'`

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='<FEEDI⇒`
		
		`D>'`

Query	*Description*	*Resolution*

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='<FEEDI⇒`
		
		`D>'`

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='<FEEDI⇒`
		
		`D>'`

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='<FEEDI⇒`
		
		`D>'`

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='<FEEDI⇒`
		
		`D>'`

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='<FEEDI⇒`
		
		`D>'`

Query	*Description*	*Resolution*

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='<FEEDI⇒`
		
		`D>'`
		
		Non-Oracle:
		
		`UPDATE PSFP_FEED`
		`SET PTFP_CATEGORY_ID = ''`
		`WHERE PTFP_FEED_ID='<FEEDI⇒`
		
		`D>'`
		
		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.
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.

Query	*Description*	*Resolution*

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='<FEEDI⇒`
		
		`D>' AND PTFT_FIELD_NAME = '<FI⇒`
		
		`ELDNAME>'`

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='<FEEDI⇒`
		
		`D>' AND PTFT_FIELD_NAME = '<QR⇒`
		
		`YFLDNAME>'`

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 Process 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 sysaudit 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” (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 referenced 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 using 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 (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 (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 = <N⇒`
		
		`AME>`
		`AND EFFDT = <EFFDT>`
		
		To delete a duplicate routing, select **PeopleTools** > **Integration Broker** > **Service Utilities** > **Integration Broker Admin**. 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 = <N⇒`
		
		`AME>`
		`AND EFFDT = <EFFDT>`
		
		`DELETE FROM PSIBRTNGSUBDEF⇒`
		
		`N`
		`WHERE ROUTINGDEFNNAME = <N⇒`
		
		`AME>`
		`AND EFFDT = <EFFDT>`
		
		`DELETE FROM PSRTNGDFNPARM`
		`WHERE ROUTINGDEFNNAME = <N⇒`
		
		`AME>`
		`AND EFFDT = <EFFDT>`
		
		`DELETE FROM PSRTNGDFNCONPR⇒`
		
		`P`
		`WHERE ROUTINGDEFNNAME = <N⇒`
		
		`AME>`
		`AND EFFDT = <EFFDT>`

Query	*Description*	*Resolution*
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.
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>`

Query	*Description*	*Resolution*
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.
IBRK-27	Operation with no service name in PSOPERATION table	This audit identifies the service operation entries in PSOPERATION table which does not have the IB_SERVICENAME column set. To resolve this audit issues, run the Application Engine program – UPGPTSERVOPR. This Application Engine program has a step to fix entries in PSOPERATION with right service names.
IBRK-28	Application Service with no service defined.	This audit identifies the Application Service entries that do not have a defined IB Service. To resolve this audit issue, bring up the Application Service in the Application Services Designer and re-save the Application Service. This should re-build the Service and associated metadata.
IBRK-29	Application Service with no service defined.	This audit identifies the Application Service entries that do not have a defined Document. To resolve this audit issue, bring up the Application Service in the Application Services Designer and re-save the Application Service. This should re-build the Document and associated metadata.

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';`
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.

Query	Description	Resolution
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 = 'recordnam⇒ e';`
		`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 definition 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.

Query	*Description*	*Resolution*
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.
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.

Query	*Description*	*Resolution*
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.OBJECTID⇒ 1 = 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 PeopleCode. Select the **Compile all and Save all PeopleCode** option. See “Finding References to Application Packages and Classes” (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) AN⇒`
		
		`D`
		`F.APPCLASSID = OBJECTVA⇒`
		
		`LUE4)`
		
		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 and save all directive PeopleCode in Application Designer when upgrading from PT8.53 or later.

PeopleSoft Test Framework Integrity

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

Query	*Description*	*Resolution*
PTF-01	Lists tests that contain references to tests that do not exist. For example, a test <Test Case> is removed or deleted but is still referenced by a <Test Name>.	Access the PTF client. Find the test by <Test Name> in PTF client, open it in test editor and navigate to problematic steps identified by the <Test Seq> in the SYSAUDIT report and correct the reference or remove the step.
PTF-02	Lists tests that contain references to test cases that do not exist. For example, a <Reference Test Case> is removed or deleted from a <Reference Test> which does exist, and the <Reference Test Case> is referenced by a <Test Name>.	Access the PTF client. Find the test by <Test Name> in PTF client, open it in test editor and navigate to problematic steps identified by the <Test Seq> in the SYSAUDIT report and correct the reference or remove the step.

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.

Query	*Description*	*Resolution*
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, use Pivot Grid Administration to delete the model.
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.
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” (Pivot Grid) for information on deleting Pivot Grid Models.

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 **PeopleTools** > **Process Scheduler** > **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 **PeopleTools** > **Process Scheduler** > **Process Scheduler Processes** > **Destination**. In the Output Destination Options group, set the Type option to *(None)*.
PRCSSCHED-04	Process Definitions, where the API AWARE should be set to true.	For the listed processes, select **PeopleTools** > **Process Scheduler** > **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.

Query	*Description*	*Resolution*
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 **PeopleTools** > **Process Scheduler** > **Process Scheduler Processes** > **Process Definition**. Correct the Process Category.
PRCSSCHED-09	Job Definitions, where the process category is invalid.	For the listed jobs, select **PeopleTools** > **Process Scheduler** > **Process Scheduler Jobs** > **Job Definition**. Correct the Process Category.
PRCSSCHED-10	Process Definitions, where the process category is missing.	For the listed processes, select **PeopleTools** > **Process Scheduler** > **Process Scheduler Processes** > **Process Definition**. Specify a Process Category.
PRCSSCHED-11	Job Definitions, where the process category is missing.	For the listed jobs, select **PeopleTools** > **Process Scheduler** > **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 **PeopleTools** > **Process Scheduler** > **Process Scheduler Servers** > **Server Definition**. Remove the invalid Process Category.

Query	*Description*	*Resolution*
PRCSSCHED-13	Server Categories, where a server is missing a process category definition.	For the listed servers, select **PeopleTools** > **Process Scheduler** > **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 **PeopleTools** > **Process Scheduler** > **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 **PeopleTools** > **Process Scheduler** > **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 **PeopleTools** > **Process Scheduler** > **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” (Application Engine).

Query	*Description*	*Resolution*
PRCSSCHED-19	Process definitions, where a process definition specifies a run time parameter record that does not exist.	For the listed processes, select **PeopleTools** > **Process Scheduler** > **Process Scheduler Processes** > **Runtime Parameters**, and perform one of the following: • select the correct runtime parameter record. • remove the incorrect parameters.
PRCSSCHED-20	Process definitions, where a process definition specifies a run time parameter field that does not exist.	For the listed processes, select **PeopleTools** > **Process Scheduler** > **Process Scheduler Processes** > **Runtime Parameters**, and perform one of the following: • select the correct runtime parameter field. • remove the incorrect parameters.

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.

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-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	*Description*	*Resolution*
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-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	*Description*	*Resolution*

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 non-public 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>)`

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 **PeopleTools** > **Portal** > **Related Content Service** > **Manage Related Content Service** and 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_MNUFLDR⇒`
		
		`S A`
		`WHERE A.PTCS_FOLDERID = A.⇒`
		
		`PTCS_PRNT_FOLDERID`
		`AND NOT EXISTS`
		`(SELECT B.PTCS_FOLDERID  F⇒`
		
		`ROM  PSPTCS_MNULINKS B`
		`WHERE A.PORTAL_NAME = B.PO⇒`
		
		`RTAL_NAME`
		`AND A.PORTAL_OBJNAME = B.P⇒`
		
		`ORTAL_OBJNAME`
		`AND A.PTCS_MENUID = B.PTCS⇒`
		
		`_MENUID`
		`AND A.PTCS_FOLDERID = B.PT⇒`
		
		`CS_FOLDERID)`

RELCONTENT-07	This audit lists all the menus with no folders.	Delete empty menus using this SQL:
		
		`DELETE FROM PSPTCS_MENU_TB⇒`
		
		`L A`
		`WHERE NOT EXISTS`
		`(SELECT 'X'  FROM PSPTCS_M⇒`
		
		`NUFLDRS B`
		`WHERE A.PORTAL_NAME = B.PO⇒`
		
		`RTAL_NAME`
		`AND A.PORTAL_OBJNAME = B.P⇒`
		
		`ORTAL_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.

Query	*Description*	*Resolution*

RELCONTENT-09	This audit lists all the orphaned links.	Delete the orphaned links using this SQL:
		
		`DELETE FROM PSPTCS_MNULINK⇒`
		
		`S A`
		`WHERE A.PTCS_FOLDERID NOT ⇒`
		
		`IN`
		`(SELECT B.PTCS_FOLDERID`
		`FROM PSPTCS_MNUFLDRS B`
		`WHERE A.PORTAL_NAME = B.PO⇒`
		
		`RTAL_NAME`
		`AND A.PORTAL_OBJNAME = B.P⇒`
		
		`ORTAL_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_MNUFLDR⇒`
		
		`S A`
		`WHERE A.PTCS_MENUID NOT IN`
		`(SELECT B.PTCS_MENUID`
		`FROM PSPTCS_MENU_TBL B`
		`WHERE A.PORTAL_NAME = B.PO⇒`
		
		`RTAL_NAME`
		`AND A.PORTAL_OBJNAME = B.P⇒`
		
		`ORTAL_OBJNAME)`

RELCONTENT-11	This audit lists all menu entries pointing to nonexistent service configurations.	Delete menu entries, using this SQL:
		
		`DELETE FROM PSPTCS_MNULINK⇒`
		
		`S A`
		`WHERE NOT EXISTS`
		`(SELECT 'X'`
		`FROM PSPTCSSRVCONF B`
		`WHERE A.PORTAL_NAME = B.PO⇒`
		
		`RTAL_NAME`
		`AND A.PORTAL_OBJNAME = B.P⇒`
		
		`ORTAL_OBJNAME`
		`AND A.PTCS_SERVICEID = B.P⇒`
		
		`TCS_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.

Query	*Description*	*Resolution*

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 = <POR⇒`
		`TAL_NAME> AND`
		`A.PTCS_SERVICEID = <SERVIC⇒`
		`E_DEFN_ID>`
		`DELETE FROM PSPTCS_MNULINK⇒`
		`S A`
		`WHERE A.PORTAL_NAME = <POR⇒`
		`TAL_NAME> AND`
		`A.PTCS_SERVICEID = <SERVI⇒`
		`CE_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_MNULINK⇒`
		`S 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_SRVCF⇒`
		`G AWHERE  A.PORTAL_NAME <>⇒`
		`'_PTCS_PTPG' AND  NOT EXI⇒`
		`STS (SELECT 'X' FROM PSPRS⇒`
		`MDEFNWHEREPORTAL_REFTYPE  ⇒`
		`='C' ANDPORTAL_NAME     = ⇒`
		`A.PORTAL_NAME ANDPORTAL_OB⇒`
		`JNAME = A.PORTAL_OBJNAME)`
		If a new content reference is not copied to the target database then move the content reference using correct mechanism.

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';`

Query	*Description*	*Resolution*
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_R⇒`
		
		`EC`
		`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, assigning multiple LONGs to a single record definition is discouraged. For Db2 z/OS, 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 z/OS and note 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

Query	Description	Resolution
		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)`

Query	*Description*	*Resolution*

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 PSRECDEF⇒`
		
		`N B`
		`WHERE PSRECDEFNLANG.RECNA⇒`
		
		`ME =`
		`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.FIEL⇒`
		
		`DNAME=`
		`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.FI⇒`
		
		`ELDNAME,`
		`A.FIELDVALUE, A.EFFDT`
		`FROM PSXLATITEMLANG A`
		`WHERE NOT EXISTS`
		`  (SELECT 'X'`
		`   FROM PSXLATITEM B`
		`   WHERE A.FIELDNAME = B.F⇒`
		
		`IELDNAME`
		`   AND A.FIELDVALUE = B.FI⇒`
		
		`ELDVALUE`
		`   AND A.EFFDT = B.EFFDT);`
		
		`DELETE FROM PSXLATITEMLANG⇒`
		
		`A`
		`WHERE NOT EXISTS`
		`  (SELECT 'X'`
		`   FROM PSXLATITEM B`
		`   WHERE A.FIELDNAME = B.F⇒`
		
		`IELDNAME`
		`   AND A.FIELDVALUE = B.FI⇒`
		
		`ELDVALUE`
		`   AND A.EFFDT = B.EFFDT);⇒`

Query	*Description*	*Resolution*

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.RELLAN⇒`
		
		`GRECNAME,`
		`A.OBJECTOWNERID`
		`FROM PSRECDEFN A`
		`WHERE A.RELLANGRECNAME <> ⇒`
		
		`' '`
		`AND A.RELLANGRECNAME NOT I⇒`
		
		`N`
		`(SELECT B.RECNAME FROM⇒`
		
		`PSRECDEFN B)`
		`ORDER BY A.RECNAME;`
		
		`UPDATE PSRECDEFN SET RELLA⇒`
		
		`NGRECNAME = ' '`
		`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.RELLAN⇒`
		
		`GRECNAME,`
		`B.RELLANGRECNAME, A.OBJECT⇒`
		
		`OWNERID`
		`FROM PSRECDEFN A, PSRECDEF⇒`
		
		`N B`
		`WHERE A.RELLANGRECNAME <> ⇒`
		
		`' '`
		`AND A.RELLANGRECNAME = B.R⇒`
		
		`ECNAME`
		`AND B.RELLANGRECNAME <>' '`
		`ORDER BY A.RECNAME;`
		
		`UPDATE PSRECDEFN SET RELLA⇒`
		
		`NGRECNAME=' '`
		`WHERE RECNAME IN`
		`(SELECT A.RELLANGRECNAME`
		`FROM PSRECDEFN A, PSRECDEF⇒`
		
		`N B`
		`WHERE A.RELLANGRECNAME <> ⇒`
		
		`' '`
		`AND A.RELLANGRECNAME = B.R⇒`
		
		`ECNAME`
		`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.

Query	*Description*	*Resolution*

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.LANG⇒`
		
		`UAGE_CD`
		`FROM PSDBFLDLABLLANG A`
		`WHERE NOT EXISTS`
		`(SELECT 'X'`
		`FROM PSDBFIELD B`
		`WHERE A.FIELDNAME =`
		`B.FIELDNAME)`
		`ORDER BY 1,2`
		
		`DELETE FROM PSDBFLDLABLLAN⇒`
		
		`G A`
		`WHERE NOT EXISTS`
		`(SELECT 'X'`
		`FROM PSDBFIELD B`
		`WHERE A.FIELDNAME =`
		`B.FIELDNAME)`

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'`

Runtime Definition Integrity

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

Query	*Description*	*Resolution*
SYSRTDEFN-01	System table audit against PSRECFIELD. This audit identifies runtime fields on design time records.	In Application Designer, open the record and remove the reported runtime field.
SYSRTDEFN-02	System table audit against PSRECDEFN. This audit identifies runtime fields that are Set Control Field properties on design time records.	In Application Designer, open the record and remove the reported runtime field.

Query	*Description*	*Resolution*
SYSRTDEFN-03	System table audit against PSRECDEFN. This audit identifies runtime records that are Parent Record properties on design time records.	In Application Designer, open the record and remove the reported runtime record.
SYSRTDEFN-04	System table audit against PSRECDEFN. This audit identifies runtime records that are Related Language Record properties on design time records.	In Application Designer, open the record and remove the reported runtime record.
SYSRTDEFN-05	System table audit against PSRECDEFN. This audit identifies runtime records that are Query Security Record properties on design time records.	In Application Designer, open the record and remove the reported runtime field.
SYSRTDEFN-06	System table audit against PSRECDEFN. This audit identifies runtime records that are Analytic Delete Record properties on design time records.	In Application Designer, open the record and remove the reported runtime record.
SYSRTDEFN-07	System table audit against PSRECDEFN. This audit identifies runtime records that are Audit Record properties on design time records.	In Application Designer, open the record and remove the reported runtime record.
SYSRTDEFN-08	System table audit against PSRECDEFN table. This audit identifies runtime fields that are System ID field properties on design time records.	In Application Designer, open the record and remove the reported runtime field.
SYSRTDEFN-09	System table audit against PSRECDEFN. This audit identifies runtime fields that are Timestamp Field properties on design time records.	In Application Designer, open the record and remove the reported runtime field.
SYSRTDEFN-10	System table audit against PSRECFIELD. This audit identifies runtime records that are Default Value Record properties on design time record fields.	In Application Designer, open the record field and remove the reported runtime record.
SYSRTDEFN-11	System table audit against PSRECFIELD. This audit identifies runtime fields that are Default Value Field properties on design time record fields.	In Application Designer, open the record field and remove the reported runtime field.

Query	*Description*	*Resolution*
SYSRTDEFN-12	System table audit against PSRECFIELD. This audit identifies runtime records that are Prompt Table properties on design time record fields.	In Application Designer, open the record field and remove the reported runtime record.
SYSRTDEFN-13	System table audit against PSRECFIELD. This audit identifies runtime fields that are Set Control Field properties on design time record fields.	In Application Designer, open the record field and remove the reported runtime field.
SYSRTDEFN-14	System table audit against PSRECFIELD. This audit identifies runtime fields that are Currency Control Field properties on design time record fields.	In Application Designer, open the record field and remove the reported runtime field.
SYSRTDEFN-15	System table audit against PSRECFIELD. This audit identifies runtime fields that are Time Zone field properties on design time record fields.	In Application Designer, open the record field and remove the reported runtime field.
SYSRTDEFN-16	System table audit against PSRECFIELD. This audit identifies runtime fields that are Related Time Date field properties on design time record fields.	In Application Designer, open the record field and remove the reported runtime field.
SYSRTDEFN-17	System table audit against PSPNLFIELD. This audit identifies runtime records on pages.	In Application Designer, open the page and remove the reported runtime record.
SYSRTDEFN-18	System table audit against PSPNLTREECTRL. This audit identifies runtime records that are on Tree Control fields of a page.	In Application Designer, open the page and remove the reported runtime record.
SYSRTDEFN-19	System table audit against PSPNLGRPDEFN. This audit identifies runtime records that are Search Record properties on components.	In Application Designer, open the component and remove the reported runtime record.
SYSRTDEFN-20	System table audit against PSPNLGRPDEFN. This audit identifies runtime records that are Add Search Record properties on components.	In Application Designer, open the component and remove the reported runtime record.

Query	*Description*	*Resolution*
SYSRTDEFN-21	System table audit against PSPCMPROG. This audit identifies runtime records with PeopleCode.	To resolve, run this SQL: `DELETE * FROM PSPCMPROG WHERE OBJECTID1 = 1 AND OBJECTVALUE1 = <ru⇒ ntime record name>`
SYSRTDEFN-22	System table audit against PSOPTREC. This audit identifies runtime records on Analytic Types.	In Application Designer, open the Analytic Type and remove the reported runtime record.
SYSRTDEFN-23	System table audit against PS_APPR _RULE_HDR. This audit identifies runtime records as Currency Records on Approval Rules.	In Application Designer, open the object and remove the reported runtime record.
SYSRTDEFN-24	System table audit against PSFLDSEGDEFN. This audit identifies runtime records on File Layouts.	In Application Designer, open the File Layout and remove the reported runtime record.
SYSRTDEFN-25	System table audit against PSOPTSETDIM. This audit identifies runtime records on the Optimization Model Index Set.	In Application Designer, open the Optimization Model and remove the reported runtime record.
SYSRTDEFN-26	System table audit against PSOPTPARARRAY. This audit identifies runtime records on the Optimization Model Parameter Array.	In Application Designer, open the Optimization Model and remove the reported runtime record.
SYSRTDEFN-27	System table audit against PSEVENTDEFN. This audit identifies runtime records on Activity Events.	In Application Designer, open the Activity and remove the reported runtime record.
SYSRTDEFN-28	System table audit against PSEVENTROUTE. This audit identifies runtime records on Activity Event Routes.	In Application Designer, open the Activity and remove the reported runtime record.
SYSRTDEFN-29	System table audit against PSMAPROLEBIND. This audit identifies runtime records on the Activity Map Role Bind table.	In Application Designer, open the Activity and remove the reported runtime record.
SYSRTDEFN-30	System table audit against PSMAPLEVEL. This audit identifies runtime records on Activity Event Level mapping.	In Application Designer, open the Activity and remove the reported runtime record.

Query	*Description*	*Resolution*
SYSRTDEFN-31	System table audit against PSMAPRECFIELD. This audit identifies runtime records on Activity Event Field mapping.	In Application Designer, open the Activity and remove the reported runtime record.
SYSRTDEFN-32	System table audit against PSAEAPPLSTATE. This audit identifies runtime records that are State Record properties on Application Engine programs.	In Application Designer, open the AE program and remove the reported runtime record.
SYSRTDEFN-33	System table audit against PSAEAPPLTEMPTBL. This audit identifies runtime records that are Temp Table properties on Application Engine programs.	In Application Designer, open the AE program and remove the reported runtime record.
SYSRTDEFN-34	System table audit against PSROLEDEFN. This audit identifies runtime records on roles.	Access the Roles Page (select **PeopleTools** > **Security** > **Permission and Roles** > **Roles**). Select the Dynamic Members tab, and remove the runtime record.
SYSRTDEFN-35	System table audit against PSARCHOBJREC. This audit identifies runtime records on Archive Objects.	Access the Manage Archive Object page (select **PeopleTools** > **Data Archive Manager** > **Manage Archive Objects**), and remove the Archive Object.

SYSRTDEFN-36	System table audit against PS_ARCH_TBL table. This audit identifies runtime records on Archive Templates.	Run the following SQL:
		`DELETE FROM PS_ARCH_TBL WH⇒`
		`ERE PSARCH_ID=<Archive ID>⇒`
		`AND RECNAME=<Runtime Reco⇒`
		`rd name>;`

SYSRTDEFN-37	System table audit against PS_ARCH_CTRL. This audit identifies runtime records on the Archive Template Control table.	Run the following SQL:
		`DELETE FROM PS_ARCH_CTRL W⇒`
		`HERE PSARCH_ID=<Archive ID⇒`
		`> AND RECNAME=<Runtime Rec⇒`
		`ord name>`

Query	*Description*	*Resolution*

SYSRTDEFN-38	System table audit against PS_ARCH _OTH_CTRL. This audit identifies runtime records on the Archive Template Other Control table.	Run the following SQL:
		`DELETE FROM PS_ARCH_OTH_CT⇒`
		`RL WHERE PSARCH_ID=<Archiv⇒`
		`e ID> AND RECNAME=<Runtime⇒`
		` Record name>;`

SYSRTDEFN-39	System table audit against PS_ARCH_ SQL_LNG. This audit identifies runtime records on the Archive Template SQL table.	Run the following SQL:
		`DELETE FROM PS_ARCH_SQL_LN⇒`
		`G WHERE PSARCH_ID=<Archive⇒`
		` ID> AND RECNAME=<Runtime ⇒`
		`Record name>;`

SYSRTDEFN-40	System table audit against PSBCITEM. This audit identifies runtime records on Component Interface items.	In Application Designer, open the Component Interface and remove the reported runtime record.
SYSRTDEFN-41	System table audit against PSCONQRSMAP. This audit identifies runtime records on Connected Query Relationships.	Access the Connected Query Manager page (select **Reporting Tools** > **Connected Query** > **Connected Query Manager**). Click Temp Tables, and then click Delete Temp Tables to remove the runtime record.

SYSRTDEFN-42	System table audit against PSPTCONTEXT. This audit identifies runtime records on the Field Context table.	Run the following SQL:
		`DELETE FROM PSPTCONTEXT WH⇒`
		`ERE RECNAME=<Runtime Recor⇒`
		`d name>;`

SYSRTDEFN-43	System table audit against PSOPTVARARRAY. This audit identifies runtime records on the Optimization Model Variable Array.	In Application Designer, open the Optimization Model and remove the reported runtime record.
SYSRTDEFN-44	System table audit against PSMSGREC. This audit identifies runtime records on the Message Definition record.	Access the Message Definition page (select **PeopleTools** > **Integration Broker** > **Integration Setup** > **Message Definitions**), and remove the runtime record.
SYSRTDEFN-45	System table audit against PSOPTREC. This audit identifies runtime records on Optimization Application records.	In Application Designer, open the Analytic Type and remove the reported runtime record.

Query	*Description*	*Resolution*
SYSRTDEFN-46	System table audit against PS_PRCS _RUNPARM. This audit identifies runtime records on the Process Run Control Parameters table.	Access the Process Scheduler Processes page (select **PeopleTools** > **Process Scheduler** > **Process Scheduler Processes**). Select the Runtime Parameters tab and remove the runtime record from the RunTime Parameters grid.
SYSRTDEFN-47	System table audit against PSQRYRECORD. This audit identifies runtime records on the Query Record table.	Access the Query Manager page (select **Reporting Tools** > **Query** > **Query Manager**). Open the Query, select the Query tab, and remove the runtime record.
SYSRTDEFN-48	System table audit against PSQRYFIELD. This audit identifies runtime records on Query fields.	Access the Query Manager page (select **Reporting Tools** > **Query** > **Query Manager**). Open the Query, select the Query tab, and remove the runtime record.
SYSRTDEFN-49	System table audit against PSQRYFIELDDEP. This audit identifies runtime records on Query Fields Dependencies.	Access the Query Manager page (select **Reporting Tools** > **Query** > **Query Manager**). Open the Query, select the Dependency tab, and remove all runtime records from the Field Dependency grid.
SYSRTDEFN-50	System table audit against PSPTCSSRVCONF. This audit identifies runtime records on Related Content Service configurations.	Access the Event Mapping Configuration page (select **PeopleTools** > **Portal** > **Event Mapping** > **Event Mapping Configuration**) and remove the runtime record.
SYSRTDEFN-51	System table audit against PSPTSF_ RECORDS. This audit identifies runtime records on Search Definition Application records.	Access the Define Search Definition page (**PeopleTools** > **Search Framework** > **Designer** > **Define Search Definition**) and remove the runtime record from the Application Records grid.
SYSRTDEFN-52	System table audit against PSPTSF _SRCHATTR. This audit identifies runtime records on Search Definition Search attributes.	Access the Define Search Definition page (**PeopleTools** > **Search Framework** > **Designer** > **Define Search Definition**) and remove the runtime record from the Search Attributes grid.
SYSRTDEFN-53	System table audit against PSPTSF_GN _DCATR. This audit identifies runtime records on the Search Definition child table PSPTSF_GN_DCATR.	Access the Define Search Definition page (**PeopleTools** > **Search Framework** > **Designer** > **Define Search Definition**). Select the Security tab and remove the runtime record from the Field Mapping grid.

Query	*Description*	*Resolution*
SYSRTDEFN-54	System table audit against PSPTTSTCOMMAND. This audit identifies runtime records on the Test Framework Test Commands table.	Open the Test in the PeopleSoft Test Framework (PTF) Client and delete the test step containing the runtime record.
SYSRTDEFN-55	System table audit against PS_APPR _RULE_FIELD. This audit identifies runtime records on the Approval Rule Defn Route Cntl.	In Application Designer, open the Approval Rule and remove the reported runtime record.
SYSRTDEFN-56	System table audit against PS_APPR _RULE_AMT. This audit identifies runtime records on Approval Rule Amounts.	In Application Designer, open the Approval Rule and remove the reported runtime record.
SYSRTDEFN-57	System table audit against PS_APPR _RULE_QTY. This audit identifies runtime records on Approval Rule Quantities.	In Application Designer, open the Approval Rule and remove the reported runtime record.

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 **PeopleTools** > **Search Framework** > **Designer** > **Define Search Definitions.**
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** > **Define 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, 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, 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, verify the related definitions are included in the project.

Query	*Description*	*Resolution*
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, verify the related definitions are included in the project.
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, 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, 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.

Security Integrity

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

Query	*Description*	*Resolution*

SEC-01	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'`

Query	*Description*	*Resolution*

SEC-02	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-03	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);`

SEC-04	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-05	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-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 with 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 = PSROLEUSER.ROLEUSER)`

Query	*Description*	*Resolution*

SEC-19	Role named in a User-Role relationship does not exist in the Role Definition table.
	
	(Verifies the existence of Roles owned by Users.)
		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 = PSROLEUSER.ROLENAME)`

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 = PSROLECLASS.ROLENAME)`

SEC-21	Permission List named in a Role-Permission List relationship does not exist in the Permission List Definition table.
	
	(Verifies existence of Permission Lists assigned to Roles.)
		If this permission list 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 = PSROLECLASS.CLASSID)`

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 with 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 with the following SQL:
		
		`UPDATE PSOPRDEFN SET PRCSPRFLCLS = ' '`
		`WHERE  PRCSPRFLCLS <> ' ' AND NOT EXISTS`
		`( SELECT 'X' FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = PRCSPRFLCLS )`

Query	*Description*	*Resolution*

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 with 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 or Data Mover, or with help from your DBA. Otherwise, manually change the value to a valid Access Profile or blank through PeopleTools 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 with the following SQL:
		
		`DELETE FROM PS_SCRTY_SRCHGRP A`
		`WHERE NOT EXISTS`
		`(SELECT 'X' FROM PSCLASSDEFN B`
		`WHERE B.CLASSID = A.CLASSID)`

SEC-28	Invalid entries in the PSAUTHITEM table.	See the SEC-28 Resolution section following this table.

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.

Query	*Description*	*Resolution*

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 and Roles** > **Delete Roles.**
		
		To activate inactive roles, use the following SQL:
		
		`UPDATE PSROLEDEFN`
		`SET ROLESTATUS = 'A'`
		`WHERE ROLESTATUS <>'A'`

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 with 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)`

Query	*Description*	*Resolution*

SEC-37	Permission List or Permission List Alias in a Permission List-Permission List Alias relationship does not exist in the Permission List Definition table.
	
	(Verifies the existence of Permission List and Permission List Alias defined in the Permission List-Permission List Alias relationship.)
		If the reported Permission List or Permission List Alias should exist, recreate it through PeopleTools Security. Otherwise, delete the Permission List.
		
		Run the following SQL:
		
		`DELETE FROM PS_PTCLASSIDALIAS A`
		`WHERE A.CLASSID NOT IN`
		`(SELECT B.CLASSID FROM PSCLASSDEFN B)`
		`OR A.PTCLASSIDALIAS NOT IN`
		`(SELECT B.CLASSID FROM PSCLASSDEFN B)`

SEC-38	Role or Role Alias in a Role-Role Alias relationship does not exist in the Role Definition table.
	
	(Verifies the existence of Roles and Role Alias defined in the Role-Role Alias relationship.)
		If the orphaned Role or Role Alias should exist, recreate it through PeopleTools Security. Otherwise, delete the Role.
		
		Run the following SQL:
		
		`DELETE FROM PS_PTROLENAMEALIAS A`
		`WHERE A.ROLENAME NOT IN`
		`(SELECT B.ROLENAME FROM PSROLEDEFN B)`
		`OR A.PTROLENAMEALIAS NOT IN`
		`(SELECT B.ROLENAME FROM PSROLEDEFN B)`

SEC-39	Role Workflow Routing Query does not exist.	If the Workflow Routing Query should exist, recreate it through PeopleTools Query Manager. Otherwise, delete the invalid Role Definition.
SEC-40	Role Dynamic Members Query Rule does not exist.	If the Dynamic Members Query Rule should exist, recreate it through PeopleTools Query Manager. Otherwise, delete the invalid Role Definition.
SEC-41	Role Dynamic Members PeopleCode Rule does not exist.	If the Dynamic Members PeopleCode Rule should exist, recreate it through Application Designer. Otherwise, delete the invalid Role Definition.
SEC-42	Role Dynamic Members LDAP Directory Rule does not exist.	If the Dynamic Members LDAP Directory Rule should exist, recreate it through LDAP Role Membership Rules. Otherwise, edit the Role using PeopleTools Security and deselect the check box named Directory Rule Enabled on the Dynamic Members page of the Roles component.
SEC-43	PSROLEGROUP rows exist for role that does not exist.	If the missing Role should exist, recreate it through PeopleTools Security. Otherwise, delete the invalid Role Group entry.
SEC-44	Orphaned Role Grant references	If the missing Role should exist, recreate it through PeopleTools Security. Otherwise, delete the invalid Role Grant entry.

SEC-28 Resolution

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 IN ('APPLICATION_DESIGNER', 'OBJECT_RUNTIME')
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
                              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
                         )
                      )
                  )
         )
      )
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 IN ('APPLICATION_DESIGNER', 'OBJECT_RUNTIME')
      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 = 'OBJECT_RUNTIME'
      AND PSAUTHITEM.BARNAME = ' ')
OR (PSAUTHITEM.MENUNAME = 'REN'
      AND ((PSAUTHITEM.BARNAME <> ' '
            AND PSAUTHITEM.BARNAME NOT IN
              (SELECT OBJNAME
                 FROM PS_APP_DES_OBJECTS
                 WHERE PSAUTHITEM.BARNAME = OBJNAME
              )
           )
            OR PSAUTHITEM.BARITEMNAME <> ' '
            OR PSAUTHITEM.PNLITEMNAME <> ' '
          )
   )
```

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)`

Query	*Description*	*Resolution*

SQL-03	SQL descriptions without a base definition.	Run the following SQL:
		
		`DELETE FROM PSSQLDESCR`
		`WHERE SQLID NOT IN`
		`(SELECT DISTINCT SQLID`
		`FROM PSSQLDEFN)`

SQL-04	SQL descriptions without associated SQL text.	Run the following SQL:
		
		`DELETE FROM PSSQLDESCR`
		`WHERE SQLID NOT IN`
		`(SELECT DISTINCT SQLID`
		`FROM PSSQLTEXTDEFN)`

SQL-05	AE SQL without SQL definitions.	This reveals Application Engine SQL Actions that do not contain any SQL code within them. Open the Application Engine program and complete the entry of the SQL before attempting to run the program. If the empty SQL actions are delivered by PeopleSoft, open an incident with 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 PSCONTDEFNLA⇒ NG WHERE CONTTYPE = 9 AND CONTNAME = %1; DELETE * FROM PSCONTENTLAN⇒ G WHERE CONTTYPE = 9 AND CON⇒ TNAME = %1;

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

Query	*Description*	*Resolution*
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 TREE-09 Resolution section following this table.
TREE-10	Tree Structure table contains Level Menu-Menu Bar combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-11	Tree Structure table contains Node Menu-Menu Bar combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-12	Tree Structure table contains Detail Menu-Menu Bar combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-13	Tree Structure table contains Level Menu-Page combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-14	Tree Structure table contains Node Menu-Page combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.
TREE-15	Tree Structure table contains Detail Menu-Page combination that does not exist.	Use Tree Manager to open the structure and fix the invalid fields.

Query	*Description*	*Resolution*

TREE-16	Tree Definition Level count does not match the record count in Tree Level table.	Set the Count in the Tree Definition table to reflect the actual number of records that are in the PSTREELEVEL table for this tree branch. Note that a problem may occur if some levels are missing and there are still nodes referencing them. In this case, the nodes do not open the tree correctly. The third SELECT checks for the previous situation. If this is the problem, run PSTED, and define the missing levels, save the tree, and then close and reopen it.
		
		`SELECT COUNT(*)`
		`FROM PSTREELEVEL`
		`WHERE TREE_NAME = 'tree_na⇒`
		
		`me'`
		` AND SETID = 'setid'`
		` AND EFFDT = 'effdt';`
		
		`UPDATE PSTREEDEFN`
		`SET LEVEL_COUNT = $count`
		`WHERE TREE_NAME = 'tree_na⇒`
		
		`me'`
		` AND SETID = 'setid'`
		` AND EFFDT = 'effdt';`

TREE-17	Tree Definition Node count does not match the record count in Tree Node table.	See the TREE-17 Resolution section following this table.

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_na⇒`
		
		`me'`
		`AND SETID = 'setid'`
		`AND EFFDT = 'effdt'`
		`AND TREE_BRANCH =`
		` 'tree_branch_name';`
		
		`UPDATE PSTREEDEFN`
		`SET LEAF_COUNT = $count`
		`WHERE TREE_NAME = 'tree_na⇒`
		
		`me'`
		`AND SETID = 'setid'`
		`AND EFFDT = 'effdt'`
		`AND TREE_BRANCH =`
		` 'tree_branch_name';`

Query	*Description*	*Resolution*
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);`

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 PSTREED⇒`
		
		`EFN 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 TREE-22 Resolution section following this table.

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)`

Query	*Description*	*Resolution*

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 PSTREEDE⇒`
		
		`FN 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 PSTREEDE⇒`
		
		`FN 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.

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 PSTREENO⇒`
		
		`DE 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.

Query	*Description*	*Resolution*
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.

TREE-09 Resolution

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.

TREE-17 Resolution

This audit identifies Tree Definition Node counts that do not match the record count in the Tree Node table. Use these methods to correct the issue.

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'
```

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-22 Resolution

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.

Chapter 4

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.

The elements that are involved with database level auditing are:

Term	*Definition*
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 HCM 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. **Note:** If there is a possibility that multiple audit rows would be generated within the same second or millisecond, then AUDIT _STAMP should be made a Duplicate Key.
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.

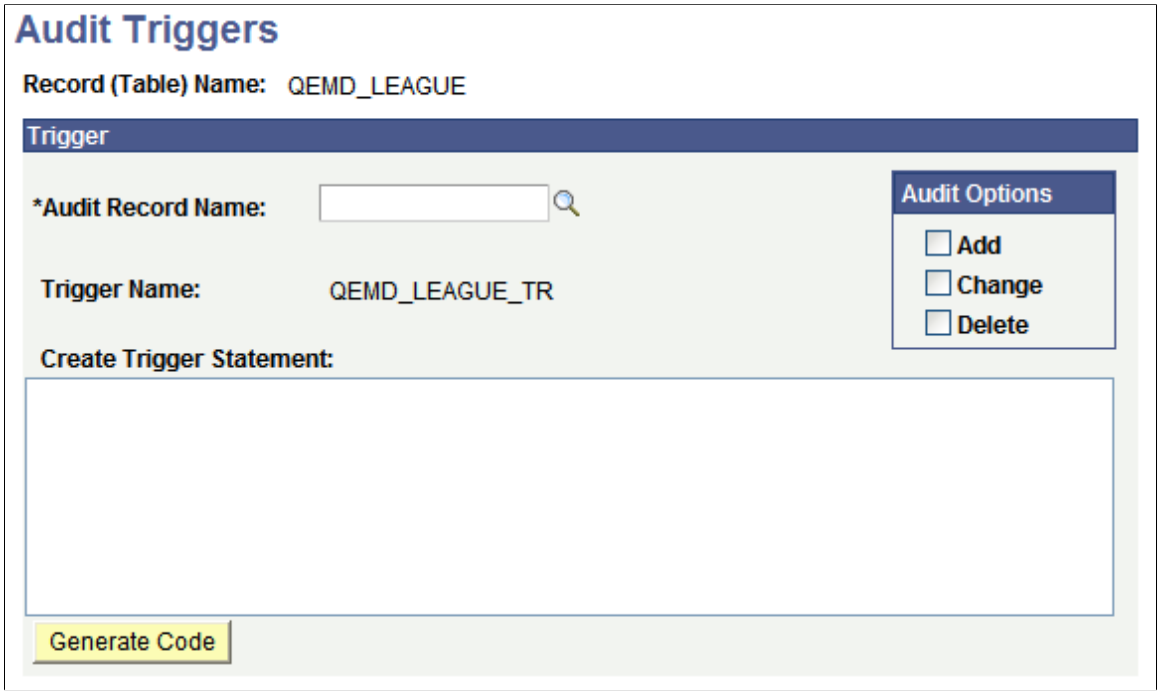
Working With Auditing Triggers

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

Defining Auditing Triggers

Select **PeopleTools** > **Utilities** > **Audit** > **Maintain Audit Triggers**, and search by Record (Table) Name, Audit Record Name, or Trigger Name to access the Audit Triggers page.

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



Field or Control	*Description*
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** > **Maintain Audit Triggers.**
2. Click **Add a New Value.**
3. Enter or browse to find an existing base record.
4. On the Audit Trigger page, choose the record to hold the auditing data, the audit record.

5. Under Audit Options, 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** > **Generate Audit Trigger as SQL**).

Field or Control	*Description*
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** > **Generate Audit Trigger as SQL**.

   Add a Run Control ID or select an existing one to access the Run Audtrgs page.

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** > **Maintain Audit Triggers**.
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.

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.

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

Listing All Audit Records in PS_AUDIT_JOB

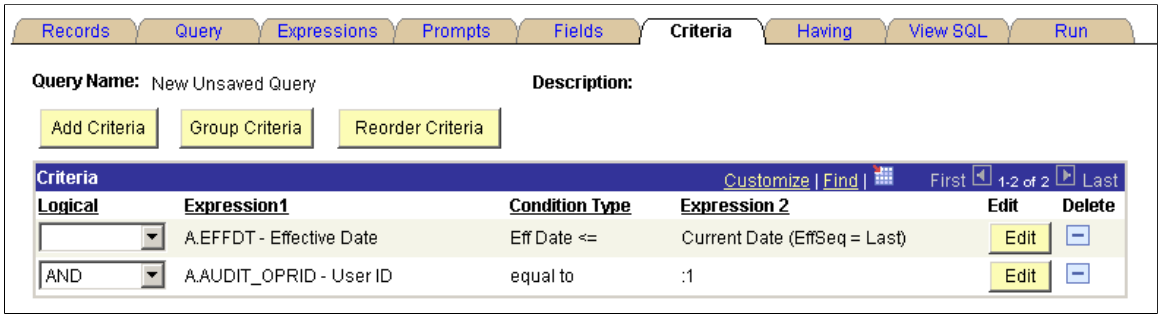
Select all the fields from AUDIT_JOB. There are no extra criteria to add.

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

Listing All Audit Records for a Specified User ID

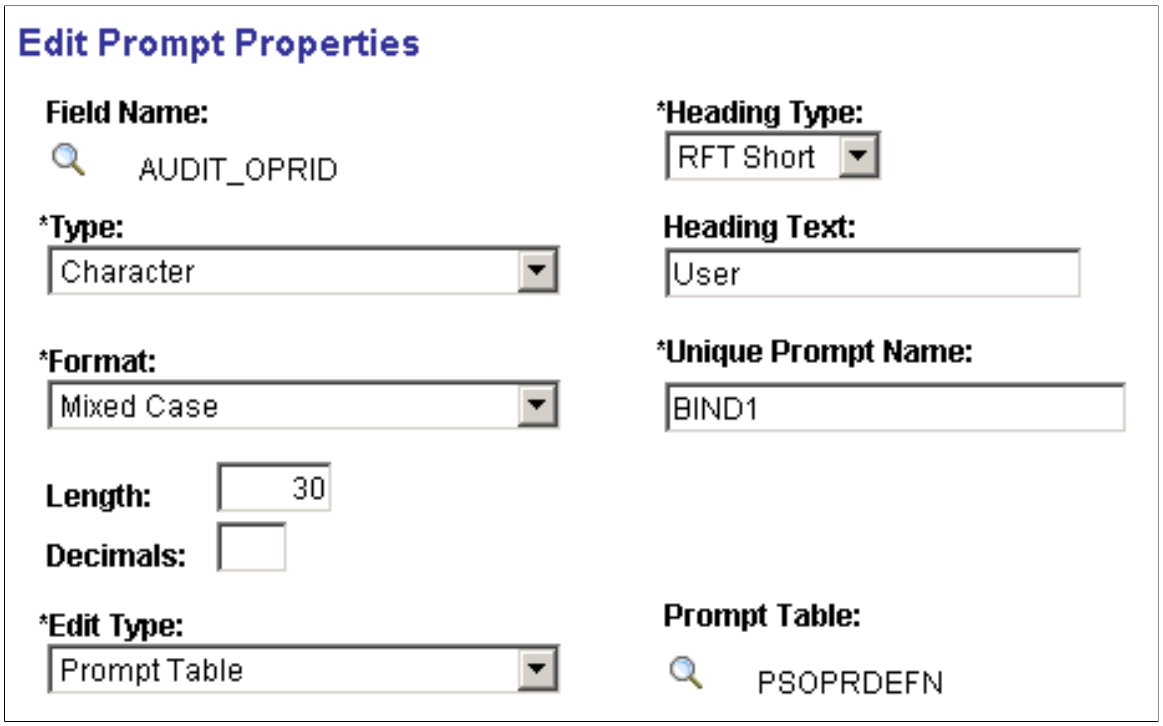
This query is similar to the previous one but with the following criteria added:

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



The example shows the prompt for properties the AUDIT_OPRID field:

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



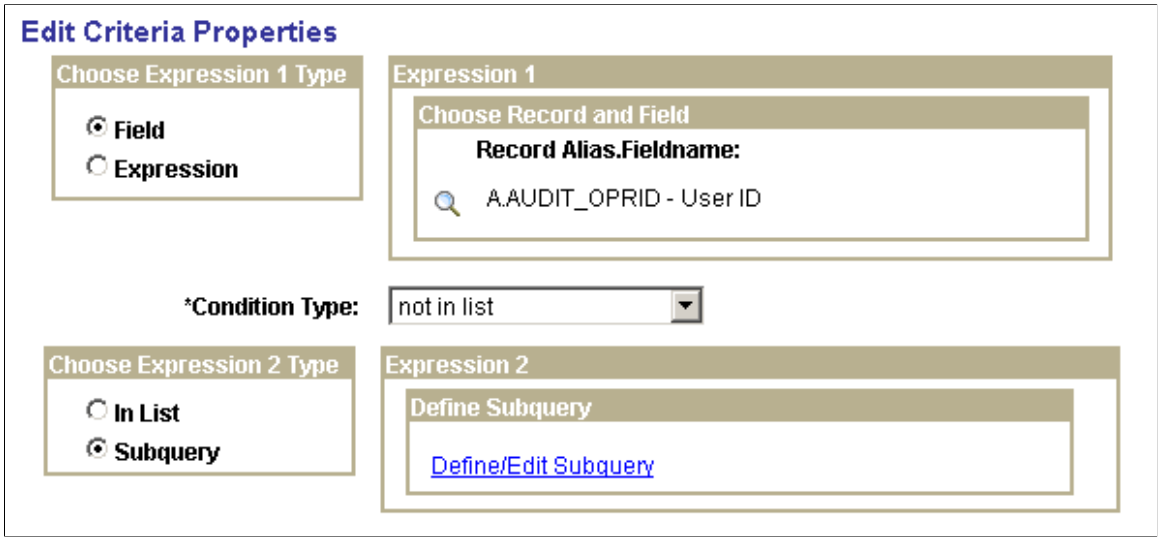
Set up a prompt for User ID against the PSOPRDEFN table. That way, when you run the query, you can specify a particular user ID. In this case, the query focuses on User ID *VP1:*

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

Listing All Audit Records Containing an Invalid OPRID

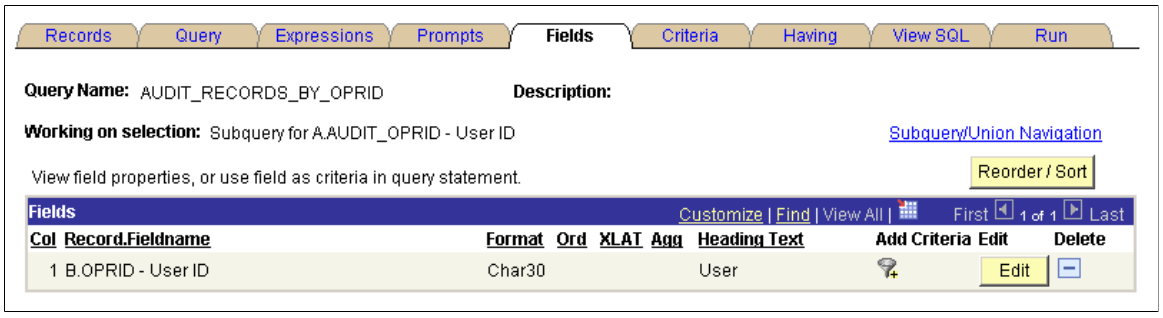
This query is similar to the previous one, but you specify different criteria:

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



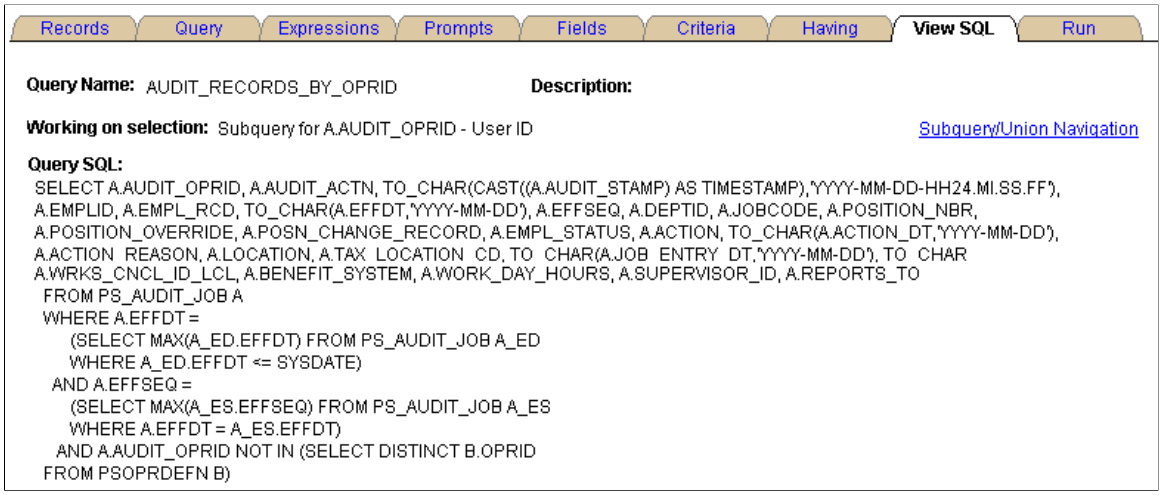
Click the **Define/Edit Subquery** link and select the OPRID field:

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



The subquery selects distinct User ID from PSOPRDEFN.

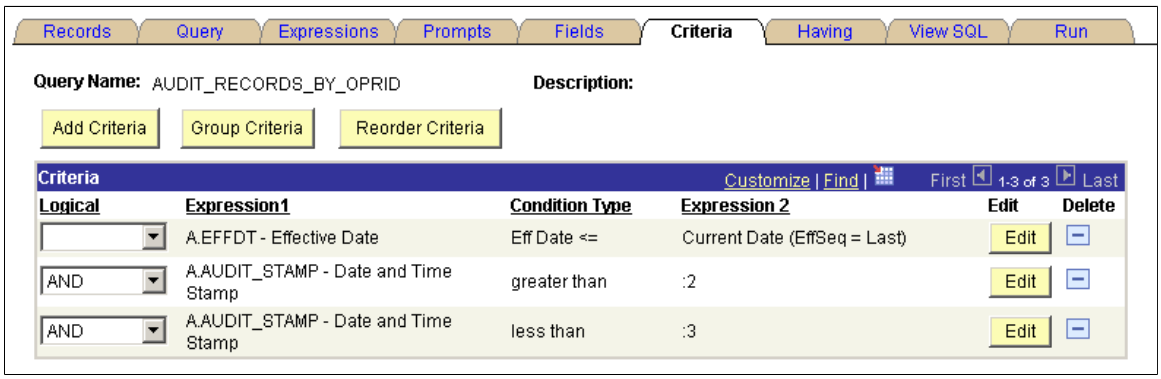
This example shows the SQL for the query.



Listing All Audit Records for a Specified Time Period

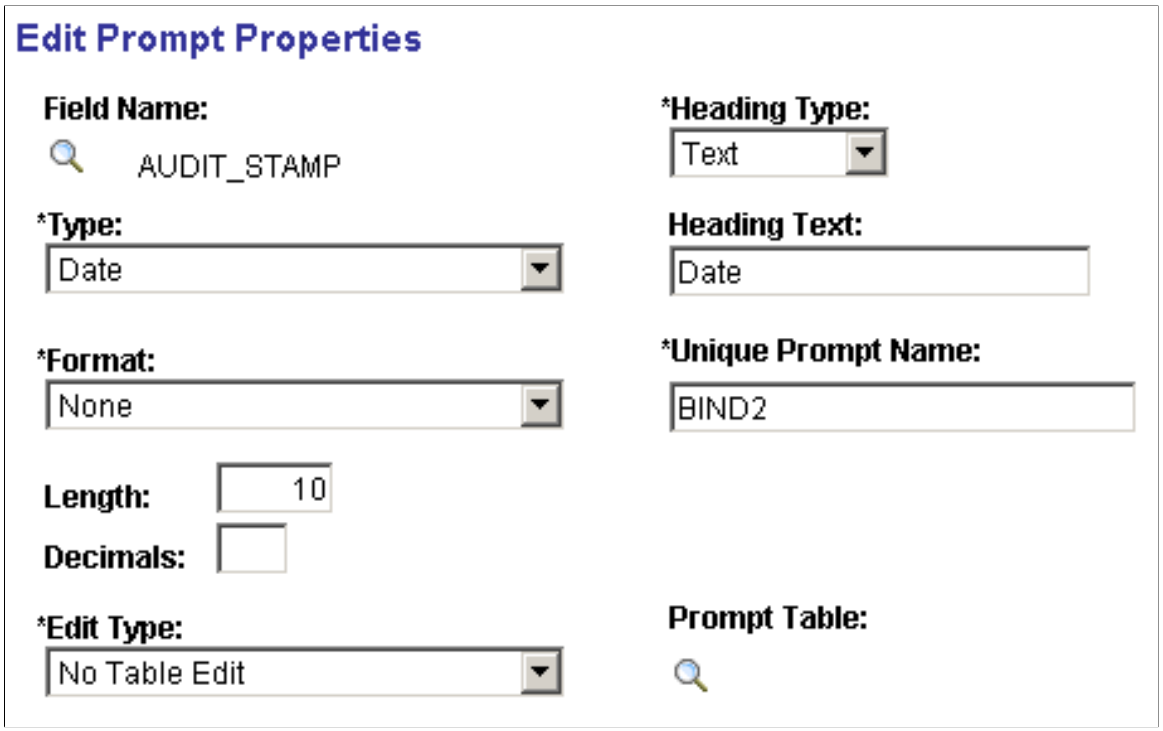
This example shows a query containing the same fields as in the previous queries above, with different criteria.

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



Set the prompt properties to follow this example.

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 usage of Microsoft SQL server trigger information.

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 for z/OS Trigger Information

This section provides an overview of Db2 z/OS trigger information.

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 Oracle Trigger Information

This section discusses using Oracle trigger information.

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.

```
/* 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)                 */
                                      */
drop function GET_PS_OPRID
/
```



```
create function GET_PS_OPRID (v_client_info VARCHAR2 )
        return VARCHAR2 is
        i integer;
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 < 32) loop
                i := i + 1;
        end loop;
        if ( i > 31 ) 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;
```


Chapter 5

Working With The Diagnostics Framework

Understanding Diagnostics Framework

This section provides an overview of the Diagnostics Framework.

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 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.

Diagnostic Plug-In Type	*Description*
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” (PeopleCode API Reference)
Understanding Diagnostic Plug-In Development

Setting Up Security for Diagnostics Framework

This section provides an overview of security for Diagnostics Framework.

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** > **Permissions and Roles** > **Permission Lists**.
2. Select or define a permission list to which you want to add Diagnostics Framework permissions.

   See “Understanding Permission Lists” (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.

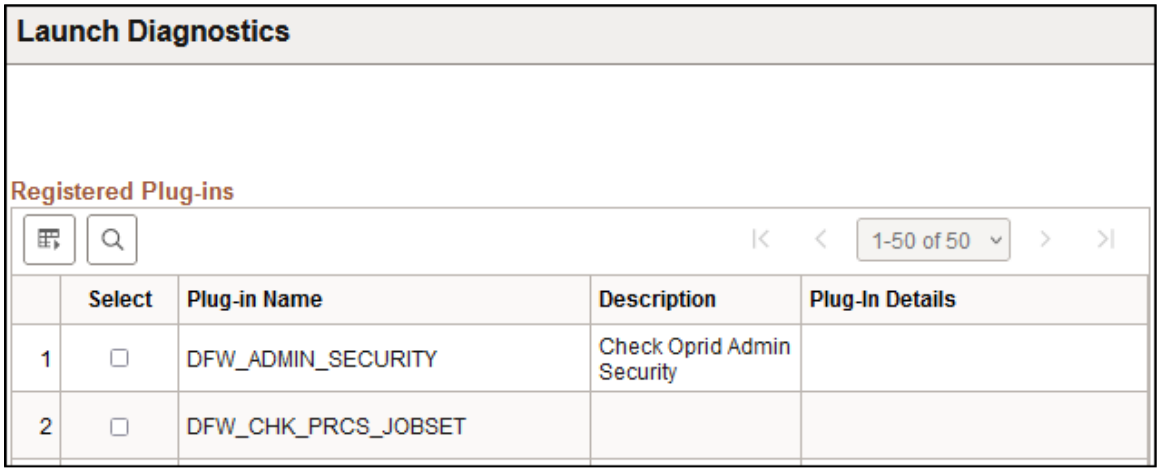
Running Diagnostics

This section discusses how to run diagnostics.

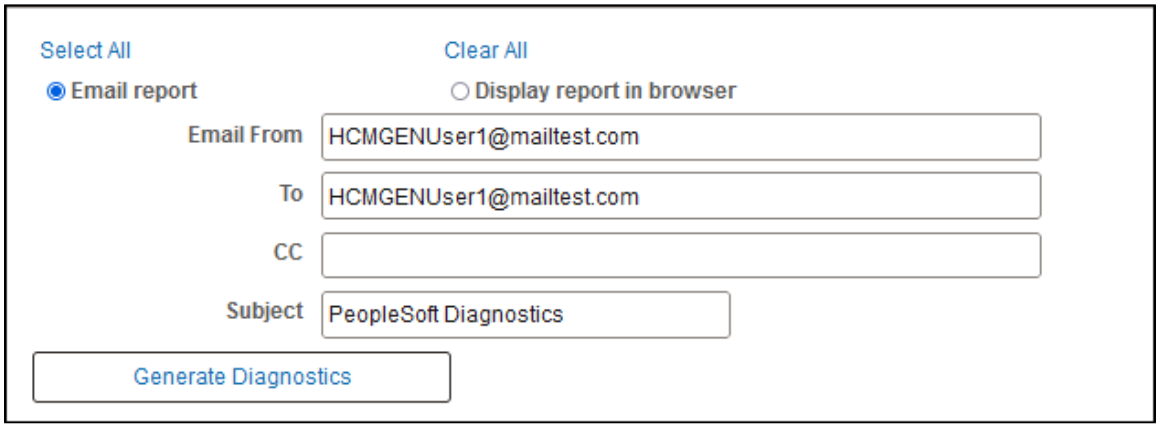
Launching Diagnostic Plug-Ins

Access the Launch Diagnostics page (**Application Diagnostics** > **Launch Diagnostics**).

This example illustrates the fields and controls on the Launch Diagnostics page. This is the top portion of the page.



This example illustrates the fields and controls on the Launch Diagnostics page. This is the bottom portion of the page with the Email report option selected.



This page displays a list of available diagnostic plug-ins. Only registered plug-ins appear.

Field or Control	*Description*
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” (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.

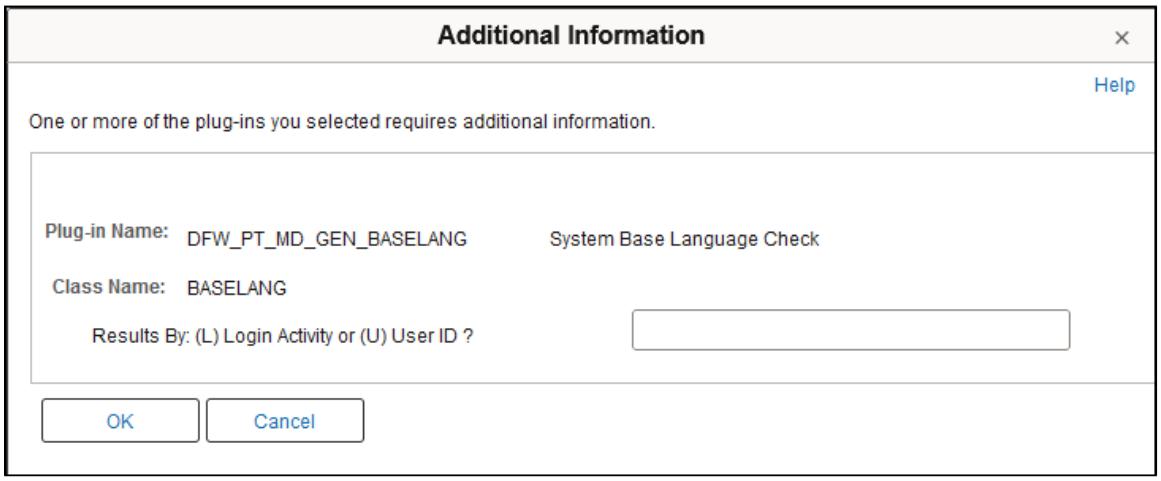
Note: You cannot select both **Display report in browser** and **Email report** simultaneously.

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.

This example illustrates the fields and controls on the Additional Information page for the System Base Language Check plug-in.



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.

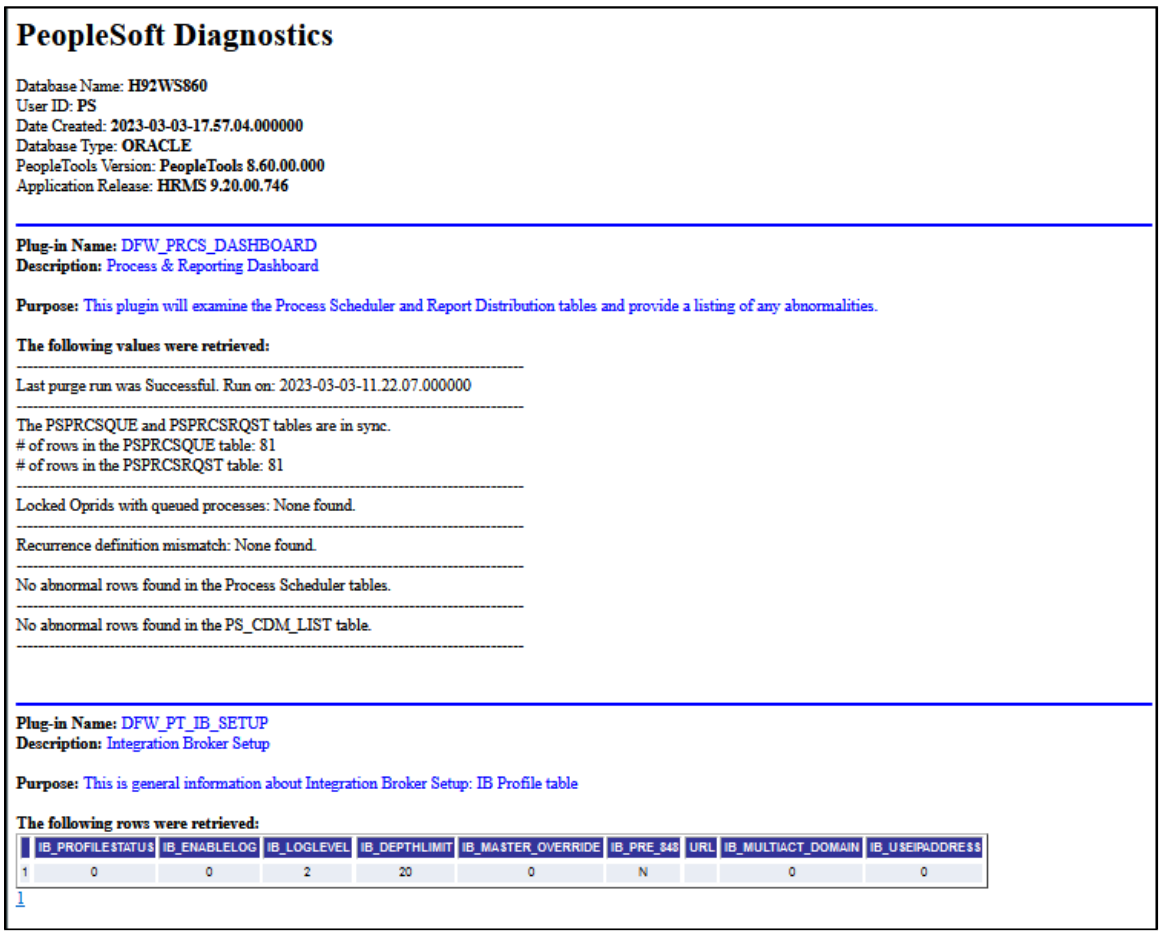
Obtaining Diagnostic Results

When all of the diagnostic results have been gathered, they're disseminated based on the option you selected on the Launch Diagnostics page. Select the option to display the generated HTML report on a browser window or select the option to generate an email containing the HTML and XML copies of the generated diagnostic report. The following sections describe the HTML and Email diagnostic reports.

Understanding the Diagnostic Report Format on the Browser

If you selected **Display report in browser,** the resulting PeopleSoft Diagnostics page appears in HTML format in a new browser window.

This example illustrates the top portion of the PeopleSoft Diagnostics page in HTML format.



PeopleSoft Diagnostics

Database Name: **H92WS860**
User ID: **PS**
Date Created: **2023-03-03-17.57.04.000000**
Database Type: **ORACLE**
PeopleTools Version: **PeopleTools 8.60.00.000**
Application Release: **HRMS 9.20.00.746**

Plug-in Name: DFW_PRCS_DASHBOARD
Description: Process & Reporting Dashboard

Purpose: This plugin will examine the Process Scheduler and Report Distribution tables and provide a listing of any abnormalities.

The following values were retrieved:

Last purge run was Successful. Run on: 2023-03-03-11.22.07.000000

The PSPRCSQUE and PSPRCSRQST tables are in sync.
# of rows in the PSPRCSQUE table: 81
# of rows in the PSPRCSRQST table: 81

Locked Oprids with queued processes: None found.

Recurrence definition mismatch: None found.

No abnormal rows found in the Process Scheduler tables.

No abnormal rows found in the PS_CDM_LIST table.

Plug-in Name: DFW_PT_IB_SETUP
Description: Integration Broker Setup

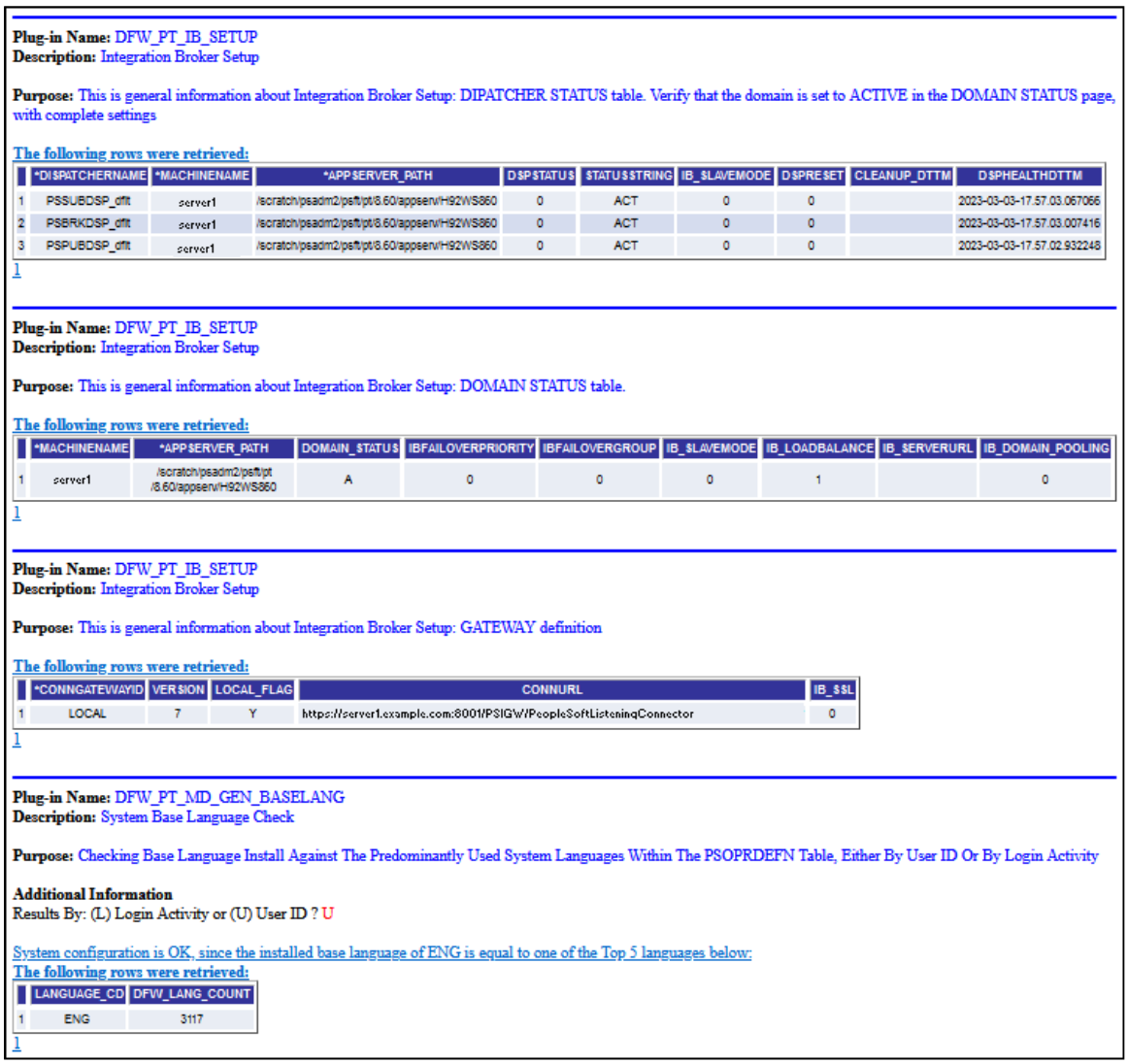
Purpose: This is general information about Integration Broker Setup: IB Profile table

The following rows were retrieved:

	IB_PROFILE$TATU$	IB_ENABLELOG	IB_LOGLEVEL	IB_DEPTHLIMIT	IB_MA$TER_OVERRIDE	IB_PRE_84$	URL	IB_MULTIACT_DOMAIN	IB_U$EIPADDRE$$
1	0	0	2	20	0	N		0	0

1

This example illustrates the bottom portion of the PeopleSoft Diagnostics page in HTML format.



Plug-in Name: DFW_PT_IB_SETUP
Description: Integration Broker Setup

Purpose: This is general information about Integration Broker Setup: DIPATCHER STATUS table. Verify that the domain is set to ACTIVE in the DOMAIN STATUS page, with complete settings

The following rows were retrieved:

	*DISPATCHERNAME	*MACHINENAME	*APPSERVER_PATH	DSPSTATUS	STATUSSTRING	IB_SLAVEMODE	DSPRESET	CLEANUP_DTTM	DSPHEALTHDTTM
1	PSSUBDSP_dflt	server1	/scratch/psadm2/psft/pt/8.60/appserv/H92WS860	0	ACT	0	0		2023-03-03-17.57.03.067066
2	PSBRKDSP_dflt	server1	/scratch/psadm2/psft/pt/8.60/appserv/H92WS860	0	ACT	0	0		2023-03-03-17.57.03.007416
3	PSPUBDSP_dflt	server1	/scratch/psadm2/psft/pt/8.60/appserv/H92WS860	0	ACT	0	0		2023-03-03-17.57.02.932248

1

Plug-in Name: DFW_PT_IB_SETUP
Description: Integration Broker Setup

Purpose: This is general information about Integration Broker Setup: DOMAIN STATUS table.

The following rows were retrieved:

	*MACHINENAME	*APPSERVER_PATH	DOMAIN_STATUS	IBFAILOVERPRIORITY	IBFAILOVERGROUP	IB_SLAVEMODE	IB_LOADBALANCE	IB_SERVERURL	IB_DOMAIN_POOLING
1	server1	/scratch/psadm2/psft/pt/8.60/appserv/H92WS860	A	0	0	0	1		0

1

Plug-in Name: DFW_PT_IB_SETUP
Description: Integration Broker Setup

Purpose: This is general information about Integration Broker Setup: GATEWAY definition

The following rows were retrieved:

	*CONNGATEWAYID	VERSION	LOCAL_FLAG	CONNURL	IB_SSL
1	LOCAL	7	Y	https://server1.example.com:8001/PSIGW/PeopleSoftListeningConnector	0

1

Plug-in Name: DFW_PT_MD_GEN_BASELANG
Description: System Base Language Check

Purpose: Checking Base Language Install Against The Predominantly Used System Languages Within The PSOPRDEFN Table, Either By User ID Or By Login Activity

Additional Information
Results By: (L) Login Activity or (U) User ID ? U

System configuration is OK, since the installed base language of ENG is equal to one of the Top 5 languages below:
The following rows were retrieved:

	LANGUAGE_CD	DFW_LANG_COUNT
1	ENG	3117

1

Displaying rowset and non-rowset information

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 computer.

Displaying retrieved rows and values on the report

The following two subtitles display the retrieved number of rows and the values in the diagnostic report:

- The following rows were retrieved:
- The following values were retrieved:

The titles are hidden by default. If you want to display the retrieved rows or the values, then you need to invoke the following methods in the plug-in to display the titles and the information on the report. The methods are:

- To retrieve rows, add the code:

```
&stat = %Super.SetProperty(%This, "DisplayRows", "Boolean", True);
```

- To retrieve values, add the code:

```
&stat = %Super.SetProperty(%This, "DisplayValues", "Boolean", True);
```

Displaying hyperlinks on the report

If you want to display a hyperlink on the report then add the following code to the plug-in:

```
&status = %Super.InsertData("Link", "For More Information:","<a href='http://www.go⇒
ogle.ca' target='_blank'>go here</a>");
```

In the above code:

- `"For More Information:"` is the static text on the report before the link.
- `<a href='http://www.google.ca' target='_blank'>` is the link that will open in a new window.
- `go here</a>"` is the text which is linked.

Using Font Color to Distinguish Text in Reports

Diagnostic Framework reports in HTML format support four text colors to distinguish between information, errors, and warnings. To support the text colors you will need to add respective methods in the extended plug-in code. The text type, text color and the code are described below:

Information Description	***Text Color***	***Code***
Normal text	Black	`&status = %Super.InsertData("String ", " your subj⇒ect ", " Country Code (3-character ISO format) - o⇒ptional ");`
Information	Green	`&status = %Super.InsertData("info ", " your subjec⇒t ", " Country Code (3-character ISO format) - opt⇒ional ");`
Errors	Red	`&status = %Super.InsertData("Error", "your subject⇒ ", " Country Code (3-character ISO format) - opti⇒onal ");`
Warnings	Yellow	`&status = %Super.InsertData("warning ", " your sub⇒ject ", " Country Code (3-character ISO format) - ⇒optional ");`

Understanding the Diagnostic Report in an Email

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” (Lifecycle Management Guide).

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

Chapter 6

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" (PeopleCode API Reference)

Developing Diagnostic Plug-Ins

This section discusses creation, implementation, and registration of diagnostic plug-ins.

Note: Except for registering the Diagnostic Plug-in, which is performed using a PIA page, you complete all of the 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" (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" (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 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" (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.
sType	Specify as a string the data type of the response required from the user. Select from the following: • String • Number • Date • Boolean

Parameter	*Description*
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.

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;
```


Chapter 7

Administering PeopleSoft Databases on Microsoft SQL Server

Server Options

This section discusses the delivered configuration for the Microsoft SQL server, access IDs, service packs, and quick fix enhancements.

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

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.

Note: If your database has tables containing the MSSCONCATCOL column, you will see SQL to alter the tables and re-create their associated indexes, even though the underlying tables and indexes may not have changed.

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.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 createdb.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 *PeopleSoft 9.2 Application 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.

Working with Indexed Views

This section provides an overview of materialized views in PeopleSoft database on the Microsoft SQL platform. The indexed views and summary tables are materialized views that optimizes performance of a query.

Understanding Indexed Views

Views are virtual tables whose contents are defined by a query. The query may reference tables and views from one or more databases.

Microsoft SQL server provides indexed views to optimize the performance of your database. When the queries need to aggregate lot of data from different tables and involve complex processing, then the performance of the system gets impacted.

If the view is referenced frequently and involves complex queries, create a unique clustered index on a view. When a unique clustered index is created on a view, it is stored in the database just like a table with a clustered index. Queries use the clustered index while retrieving data from a view. The efficiency of retrieving the data is improved.

Clustered indexes sort the data included in the view based on their key value. The key values are the columns included in the index definition. There can only be one clustered index per view, because the data rows themselves can be sorted in only one order.

Another benefit of creating an index on a view is that the optimizer starts using the view index in queries that do not directly name the view in the `FROM` clause. Existing queries can benefit from the improved efficiency of retrieving data from the indexed view without being re-coded.

For more information on indexed views see the Microsoft Developer Network website.

Related Links

Working with Materialized Query Tables
Using Materialized Views

Understanding Summary Tables

Summary tables are physical staging tables. The result of the SQL View is populated into the summary table. Accessing a summary table will return a table thereby improving the performance.

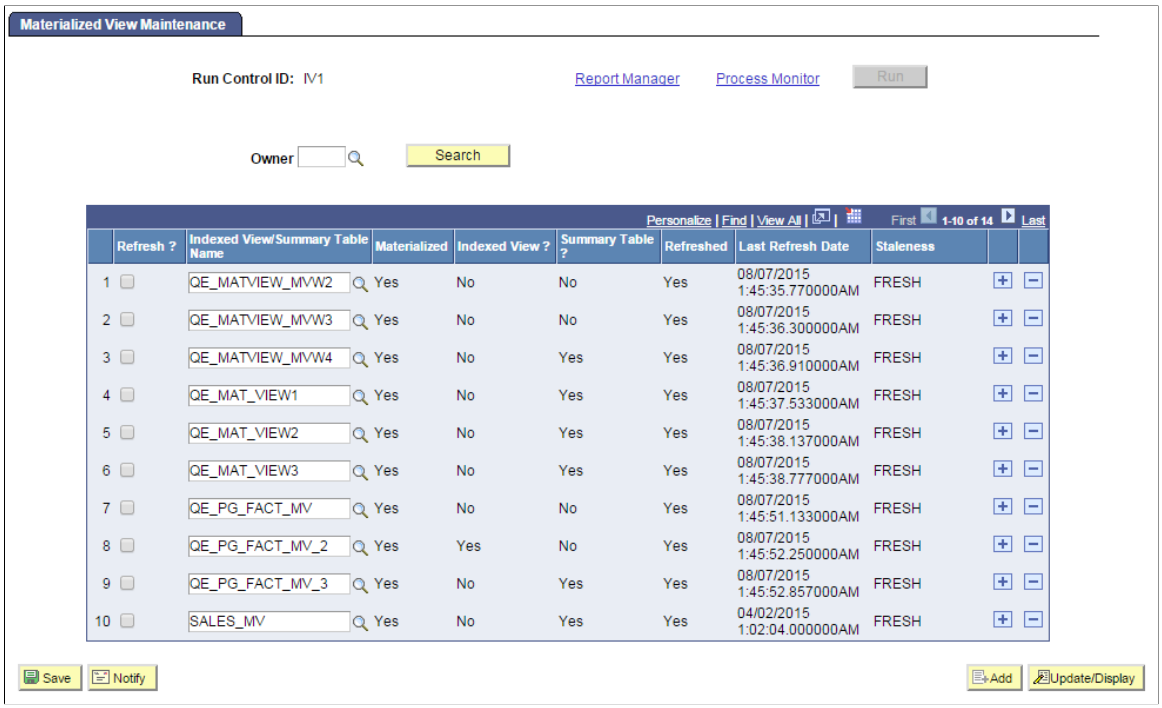
Maintaining Indexed Views/Summary Tables

You can access the Materialized View Maintenance page if you are granted the Materialized View Administrator role. On this page you can select the indexed views and the summary tables for a refresh. A refresh happens when the data in tables are synchronized with the indexed views or summary tables. You can also get information on the staleness and last refresh date of the views.

Access the Materialized View Maintenance run control page (**PeopleTools** > **Utilities** > **Administration** > **Materialized Views** > **Maintain Materialized Views**).

The following example illustrates the Materialized View Maintenance page. The description of the fields and controls follows that:

The page displays information on summary tables and indexed views. You can select a view to refresh its index.



This page contains a grid that is populated with a list of the records in the materialized views table, PSPTMATVWDEFN.

Field or Control	***Description***
Owner	Search and select a specific owner to limit the grid to records that belong to that owner.
Refresh?	Click to schedule a refresh for the record.
Indexed View/Summary Table Name	Enter the name of the indexed view or summary table. You can use the prompt for selecting the name.
Materialized	Displays if the indexed view or summary table is materialized in the database which makes them readily available in the database and the query does not have to be run again.
Indexed View?	Displays Yes, if the materialized view is an Indexed View.
Summary Table?	Displays Yes, if the materialized view is a Summary Table.
Refreshed	Indicates if the record is refreshed (Yes) or not (No).
Last Refresh Date	List the date and time the record was last refreshed.

Field or Control	*Description*
Staleness	Indicates if the record is stale or fresh based on the Microsoft SQL System Catalog.
Run	Click to execute the **PTMATREFVW** Application Engine program, to refresh the selected records.

On the page, select the indexed views/summary tables to refresh. Click the **Save** button. On saving the page, the **Run** button gets enabled.

Click the **Run** button. The Process Scheduler Request page with **PTMATREFVW** process is displayed. Click the **OK** button to initiate the process. The Application Engine program **PTMATREFVW** executes and recreates the indexes for each of the selected indexed views/summary tables.

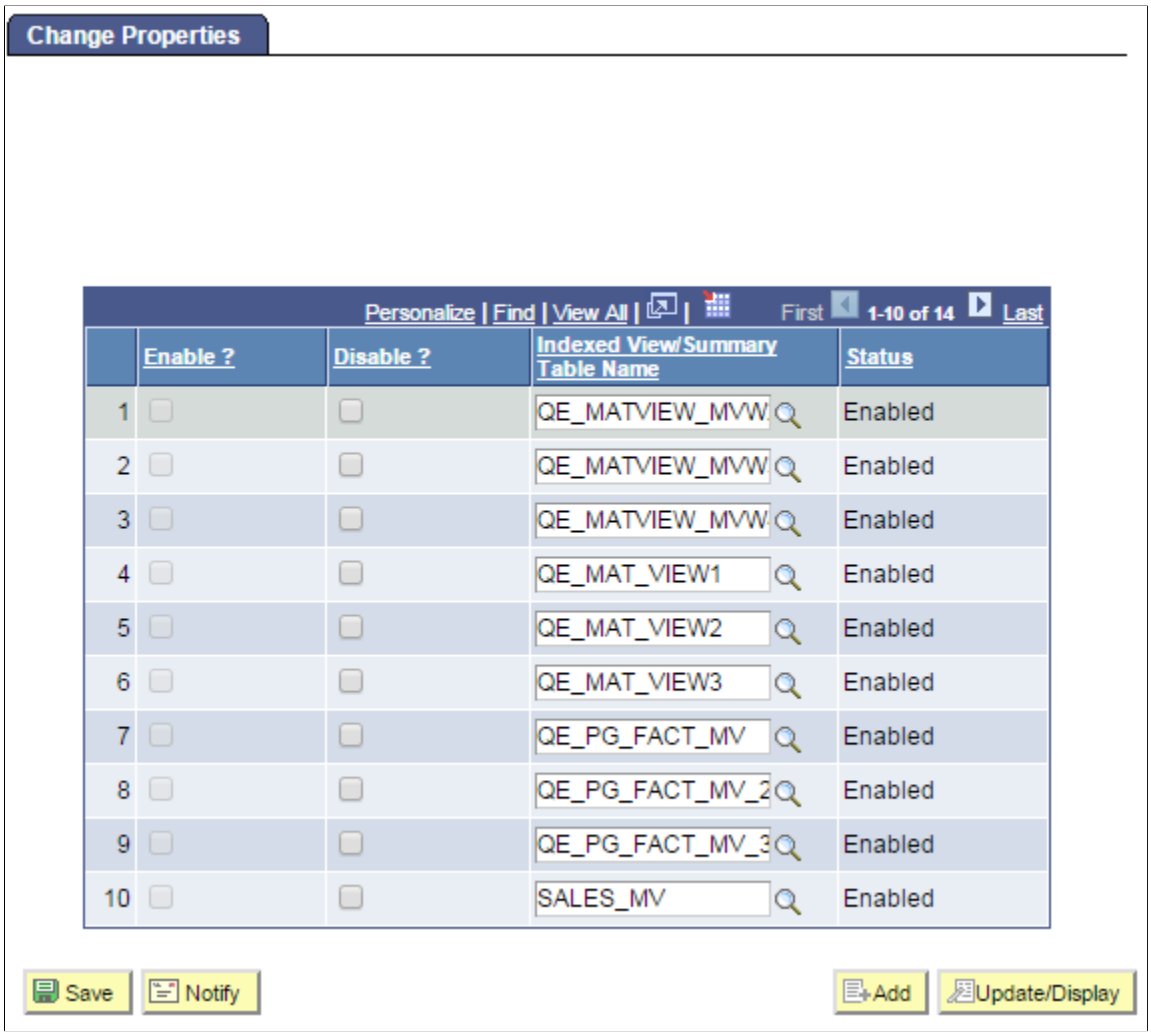
For more information on Process Scheduler Request page see, Maintaining Materialized Views documented under the Oracle platform section.

Enabling Indexed Views/Summary Tables

Access the Change Properties page (**PeopleTools** > **Utilities** > **Administration** > **Materialized Views** > **Enable Materialized Views**).

You can see on the page, a list of the views delivered from applications. You can select specific views to enable or disable indexed views/summary tables feature. This example illustrates the Change Properties page. The columns on the page are explained below:

The page provides option to enable or disable indexed views and summary tables from the PeopleSoft Internet Architecture.



Field or Control	*Description*
Enable?	Select to convert the view as an indexed view or summary table.
Disable?	Select to convert the indexed view or summary table to a normal SQL view.
Indexed View/Summary Table Name	Enter the name of the indexed view or summary table. You can use the prompt for selecting the name.
Status	Displays **Enabled** or **Disabled** depending on the current status of the view.

Save the page after you select the views to enable or disable indexed views/summary tables. Open the ENABLEMV project in the Application Designer and build the project to render the selected views as

indexed views/summary tables. Similarly, open the DISABLEMV project in the Application Designer and build it to convert the indexed views/summary tables to normal SQL views.

Audits for Indexed Views

You can run the following DDDAudit queries for the indexed views to resolve any inconsistency in the database.

- IVIEWS-1
- IVIEWS-2
- IVIEWS-3
- IVIEWS-4

See the Indexed View Queries [Microsoft SQL Server] section for detailed audit queries.

Microsoft SQL Server Feature Considerations

This section discusses features in Microsoft SQL Server (MSS).

Recovery Model

PeopleSoft recommends using the Full recovery model on SQL Server databases. All production databases should use this model for better reliability. The PeopleSoft applications do not require any particular recovery model but using the Full recovery model is considered the best practice.

See the product documentation for Microsoft SQL Server.

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.

See the product documentation for Microsoft SQL Server.

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.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
```

See the product documentation for Microsoft SQL Server.

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" (System and Server Administration)

"Understanding PeopleSoft Configuration Manager" (System and Server Administration)

"Using the PSADMIN Utility to Configure Process Scheduler Tuxedo Servers" (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>
```


Chapter 8

Administering PeopleSoft Databases on DB2 for z/OS

Understanding Db2 for z/OS Administration

This section discusses these topics on concurrency:

- 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 z/OS either through PTPSQLRT (for Cobol), 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 z/OS and potentially reduce deadlock situations. It also provides two extra benefits:

- Block fetch is enabled for ambiguous cursors.
- Db2 z/OS considers parallelism for ambiguous cursors.

RELCURHL

A Db2 z/OS subsystem parameter RELCURHL lets you indicate that you want Db2 z/OS 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 z/OS 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.

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.
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 file.

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
```

Enabling the PTPSQLRT Statistics Report on the Db2 for z/OS Server

For details on creating your own JCL to run COBOL and SQR, see the appendix "Reviewing JCL Samples for Optional Manual Batch Environments" in *PeopleSoft 9.2 Application Installation for Db2 for z/OS* (for the current PeopleTools release).

Enable the mainframe statistics report through your JCL as follows:

Set PARM 3 to *Y* in the PARMFILE instream data set as shown in this section.

```
//* PARMFILE - PARM 1 IS OPRID - LEAVE AS SYMBOLIC
//*            PARM 2 IS A 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 - ALWAYS SET TO  "BATCH" IN THE JCL
//*
//*
//PARMFILE DD  *
PS
DB2SERV
Y

N
N
N
N
BATCH
/*
//*
//
```

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 z/OS during program execution; “cursor” refers not only to open cursors, but deletes, inserts and updates as well.

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.

Note: For details on creating your own JCL to run COBOL and SQR, see the appendix "Reviewing JCL Samples for Optional Manual Batch Environments" in *PeopleSoft 9.2 Application Installation for Db2 for z/OS* (for the current PeopleTools release).

To enable Dynamic Explains in your JCL, set PARM 3 and PARM 5 to *Y* in the PARMFILE instream data set as shown below.

```
//* PARMFILE - PARM 1 IS OPRID - LEAVE AS SYMBOLIC
//*            PARM 2 IS A 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 - ALWAYS SET TO  "BATCH" IN THE JCL
//*
//*
//PARMFILE DD  *
PS
DB2SERV
Y

Y
N
N
N
BATCH
/*
//*
//
```

Enabling Parallelism

The PTPSQLRT API program provides a feature that allows you to enable Db2 parallelism. If enabled, PTPSQLRT issues the following command to Db2 z/OS:

```
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 z/OS parallelism, this does not necessarily mean that the statements in the application programs uses parallelism. What it means is that Db2 z/OS optimizer considers parallelism as a possible option.

To enable parallelism in the JCL, set PARM 6 to *Y* in the PARMFILE instream data set as shown below.

Note: For details on creating your own JCL to run COBOL and SQR, see the appendix "Reviewing JCL Samples for Optional Manual Batch Environments" in *PeopleSoft 9.2 Application Installation for Db2 for z/OS* (for the current PeopleTools release).

```
//* PARMFILE - PARM 1 IS OPRID - LEAVE AS SYMBOLIC
//*            PARM 2 IS A 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 - ALWAYS SET TO  "BATCH" IN THE JCL
//*
//*
//PARMFILE DD  *
PS
DB2SERV
N

N
Y
N
N
BATCH
/*
//*
//
```


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.

Note: For details on creating your own JCL to run COBOL and SQR, see the appendix "Reviewing JCL Samples for Optional Manual Batch Environments" in *PeopleSoft 9.2 Application Installation for Db2 for z/OS* (for the current PeopleTools release).

To enable the PeopleSoft SQL trace in your JCL, set PARM 7 to *Y* in the PARMFILE instream data set as shown below.

```
//* PARMFILE - PARM 1 IS OPRID - LEAVE AS SYMBOLIC
//*            PARM 2 IS A 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 - ALWAYS SET TO  "BATCH" IN THE JCL
//*
//*
//PARMFILE DD  *
PS
DB2SERV
N

N
N
Y
N
BATCH
/*
//*
//
```

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 run a dynamic SQL statement.

In the past, the timings of these dynamically generated statements have been recorded to the DYSQLLOG. In the current PeopleSoft releases 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 for z/OS statements by running the SQR with the -S option.

Note: For details on creating your own JCL to run COBOL and SQR, see the appendix "Reviewing JCL Samples for Optional Manual Batch Environments" in *PeopleSoft 9.2 Application Installation for Db2 for z/OS* (for the current PeopleTools release).

Adding SQR FLAG to SQRSAMP

Add the -S SQR flag to the PARMLIB(SQRPARMS) as shown:

```
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
```

Output from the -S flag is directed to SYSOUT.

Here is 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 z/OS 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 *PeopleSoft 9.2 Application 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.

CLI Connection Attribute	*Description*
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=192.0.2.10.50770.11071904055
  V445-GACC38D8.C652.C816EA9282DD=14772 ACCESSING DATA FOR
    ::192.0.2.10
```

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=192.0.2.10.50588.11071903542
  V445-GACC38D8.C59C.C816E7F87993=14756 ACCESSING DATA FOR
    ::192.0.2.10
```

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=192.0.2.10.50588.11071903542
V445-GACC38D8.C59C.C816E7F87993=14756 ACCESSING DATA FOR   ::192.0.2.10
```

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=192.0.2.10.52097.11071906585
  V445-GACC38D8.CB81.C817113C2D33=14835 ACCESSING DATA FOR
   ::192.0.2.10
```

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.

Understanding COBOL API and Meta SQL

PTPSQLRT is the COBOL API program called by application COBOL programs to prepare and run 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 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 syntax. Stored statements are delivered in directory \SRC\CBL\BASE of the installation file server.

Running COBOL Outside of Process Scheduler

To run COBOL, set PARM 4 to zero (0) or a space.

Note: For details on creating your own JCL to run COBOL and SQR, see the appendix "Reviewing JCL Samples for Optional Manual Batch Environments" in *PeopleSoft 9.2 Application Installation for Db2 for z/OS* (for the current PeopleTools release).

```
//* PARMFILE - PARM 1 IS OPRID - LEAVE AS SYMBOLIC
//*            PARM 2 IS A RUN CONTROL NAME OR PROCESS SCHEDULER
//*                      SERVER 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 - ALWAYS SET TO  "BATCH" IN THE JCL
//*
//*
```



```
//PARMFILE DD  *
PS
DB2SERV
Y

Y
Y
N
N
BATCH
/*
//*
//
```

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. This maintains cursor position after a commit, so that repositioning (reopening and re-fetching) does not need to occur.

The DB2 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 and specify the input and the output files.

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. All SQL is dynamic, therefore no precompiling is necessary for running SQRs.

You can run SQRs on z/OS by submitting them manually using your own JCL.

SQR for z/OS is delivered with the PeopleTools installation. The PeopleSoft 9.2 Application Installation guide for Db2 for z/OS 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.

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)

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 FSCM: 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 run on any operating system by modifying your JCL.

- Add a DSN style file name to the SQR.

  This method alleviates the need to modify JCL, but creates a z/OS operating system specific SQR, because file names specified in this manner are hard-coded directly into the SQR source code.

Use the first method if your SQR should run on any operating system. Use the second method if your SQR runs only on the z/OS operating system and you do not anticipate the need to change file names. Also, always use the DD statement method when indicated in PeopleSoft application-specific documentation.

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 file names 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 JCL must contain a DD statement with this name. Each file used for input or output requires a separate DD statement in the JCL.

In the following example, SQR for z/OS uses VIEWTBL as the DD name in JCL:

```
let $outputfile = '{FILEPREFIX}VIEWTBL{FILESUFFIX}'
open $outputfile as 1 for-writing record=132
```

The DD statement in the execution JCL requires the same DD name:

```
//VIEWTBL DD  DSN=&PSHLQ..OUTFILES(VIEWTBL),DISP=SHR
```

The data set name specified in the JCL may be either sequential or partitioned dataset. If the SQR requires multiple input or output files, you must add a separate DD statement to the JCL for each file.

Note: DD names must reference either sequential datasets or separate partitioned datasets due to the z/OS restriction on writing to more than one PDS member simultaneously.

Adding a DSN Style File Name to the SQR

File names 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)'
```


DD statement modifications are not required to your JCL when using the hard-coded DSN file name.

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 factors related to updating statistics.

Understanding %UpdateStats

The meta-SQL %UpdateStats was introduced for PeopleSoft environments to allow an Application Engine and COBOL programs to update statistics in the Db2 catalog. Updating the statistics in the Db2 catalog 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 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 in your environment, you need to be aware of the following key items:

- Setting up the Db2 z/OS stored procedure: DSNUTILS.
- 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.

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 on non-z/OS platforms for both Application Engine and COBOL. When running COBOL directly on z/OS with your own JCL, enable %UpdateStats by setting PARM 6 to *Y* in the PARMFILE instream data set. 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 Microsoft Windows Process Scheduler servers.

See *PeopleSoft 9.2 Application Installation for DB2 for z/OS*.

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 catalog.

Running the following SQRs ensures that the PeopleTools tables are in sync with the Db2 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 catalog. You should, however, still run SETSPACE and SETTMPIN to keep the PeopleTools metadata synchronized with the Db2 Catalog.

See *IBM's Installation Guide for Db2 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 Microsoft Windows Process Scheduler servers.

Enabling or Disabling %UpdateStats for Batch Processing

To fully enable %UpdateStats for COBOL on z/OS:

1. Set PARM 6 to *Y* in the PARMFILE instream data set of your JCL.
2. Modify the program PSPTSQLRT.

Note: 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.
--
```

3. Uncomment (remove the asterisk at the beginning) the lines noted above so they are activated in the program code.

4. Compile the program PTPSQLRT by submitting the two JCL members found in $PSHLQ.JCLLIB:
   - PSCOBDA
   - PSCOBDE
5. Determine whether you want to bind or rebind two Db2 plans for PeopleSoft.

   Add the following line to the Bind Parameter list in the appropriate JCL members noted below, before submitting them:

   ```
   PKLIST (DSNUTILS.*)
   ```
6. For first time Binding modify the following two members in $PSHLQ.JCLLIB:
   - BINDAADD
   - BINDEADD
7. 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 *PeopleSoft 9.2 Application 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 z/OS (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 working with Alters on Db2 z/OS.

Understanding Alters on Db2 z/OS

PeopleTools Data Administration tools support native Db2 z/OS 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:

- 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:

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 z/OS 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 z/OS 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 z/OS 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 z/OS 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 z/OS 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 z/OS Reorg Tablespace and Modify Recovery utilities, refer to your IBM Db2 documentation.

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

Working with Materialized Query Tables

A Materialized Query Table (MQT) is a table derived from a query result set. MQT optimizes SQL performance against data warehouses.

Related Links

Using Materialized Views
Working with Indexed Views

Understanding Query Optimization

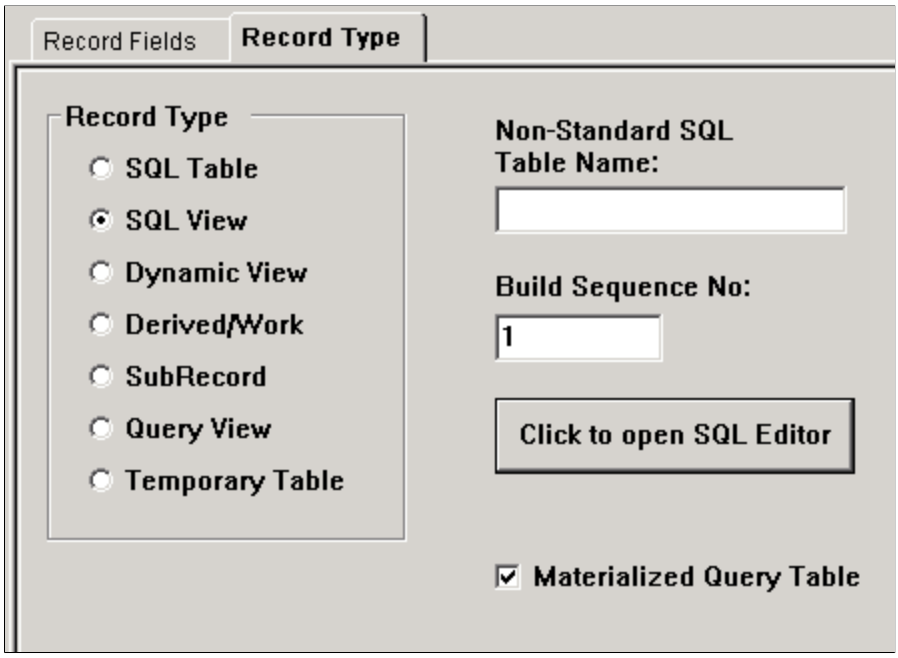
If you enable Query Optimization for a MQT then the SQL is dynamically modified in the database. This happens if another query result set content is completely or partially similar to the MQT. The database compares the SQL with the `FULLSELECT` statement that was used to define the MQT. If it determines that there is either a partial or full overlap between the submitted SQL and the `FULLSELECT` statement of the MQT, then the database automatically rewrites the submitted query to access the MQT instead of updating the base tables that were originally specified in the query. For non-overlapping parts of the query, the database accesses the base tables specified in the query.

For more information on Query Optimization, refer to your IBM Db2 documentation.

Configuring Materialized Query Tables in Application Designer

Materialized Query Tables are enabled from the view definition in the Application Designer.

The following example illustrates a view definition Record Type tab where the Materialized Query Table check box exists:



You can enable or disable the Materialized Query Table for any record type of SQL View or Query View.

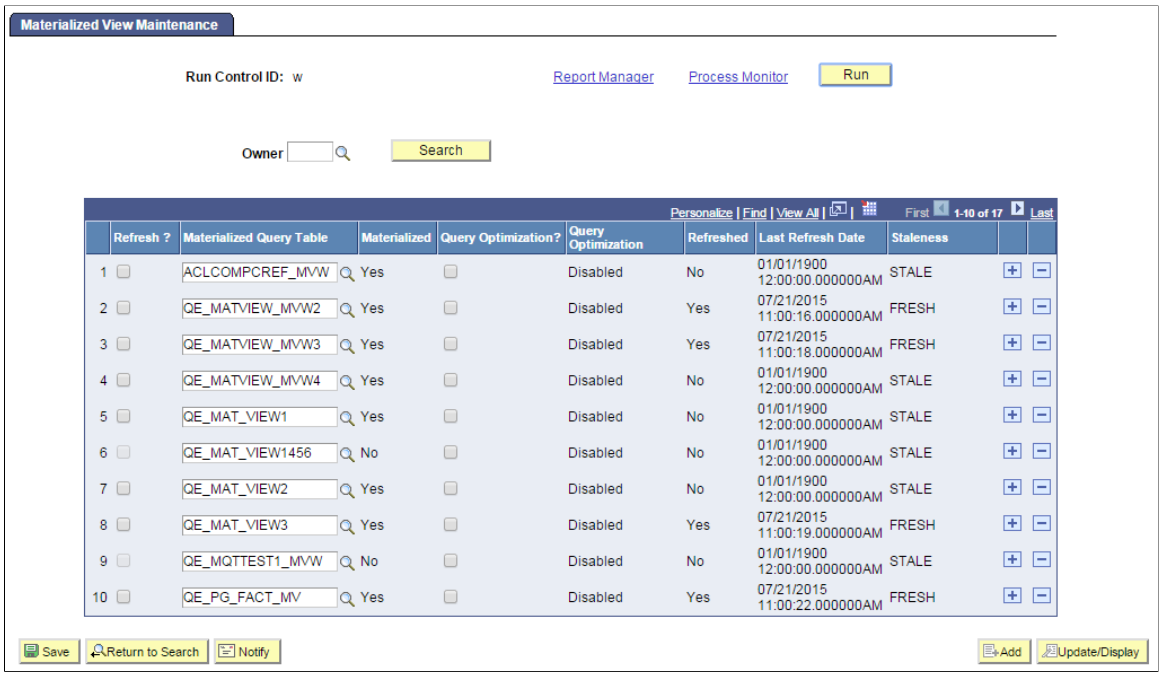
Maintaining Materialized Query Tables

You can access the Materialized View Maintenance page if you are granted the Materialized View Administrator role. Use a run control ID to populate the MQTs on the page. As an administrator, you can refresh the MQTs and enable Query Optimization on them.

Access the Materialized View Maintenance page (**PeopleTools** > **Utilities** > **Administration** > **Materialized Views** > **Maintain Materialized Views**).

Note: Nested MQTs are not supported for Db2 z/OS.

The following example illustrates the Materialized View Maintenance page. The description of the fields and controls follows that:



Field or Control	*Description*
Owner	Search and select an owner of the tables to limit the search result of MQTs.
Refresh?	Select to schedule a refresh on the record.
Materialized Query Table	Displays the name of the record.
Materialized	Displays if the MQT is materialized in the database which makes them readily available in the database and the query does not have to be run again.
Query Optimization?	Select to allow the database to rewrite the SQL if another query result set has similar content in a MQT.
Query Optimization	Displays Enabled, if Query Optimization is enabled for the view. It is a display-only field.
Refreshed	Indicates if the record is refreshed (Yes) or not (No).
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 database catalog.

On the page, select the MQTs to refresh. Click the **Save** button. On saving the page, the **Run** button gets enabled.

Click the **Run** button. The Process Scheduler Request page with **PTMATREFVW** process is displayed. Click the **OK** button to initiate the process. The Application Engine program **PTMATREFVW** executes and refreshes the MQTs.

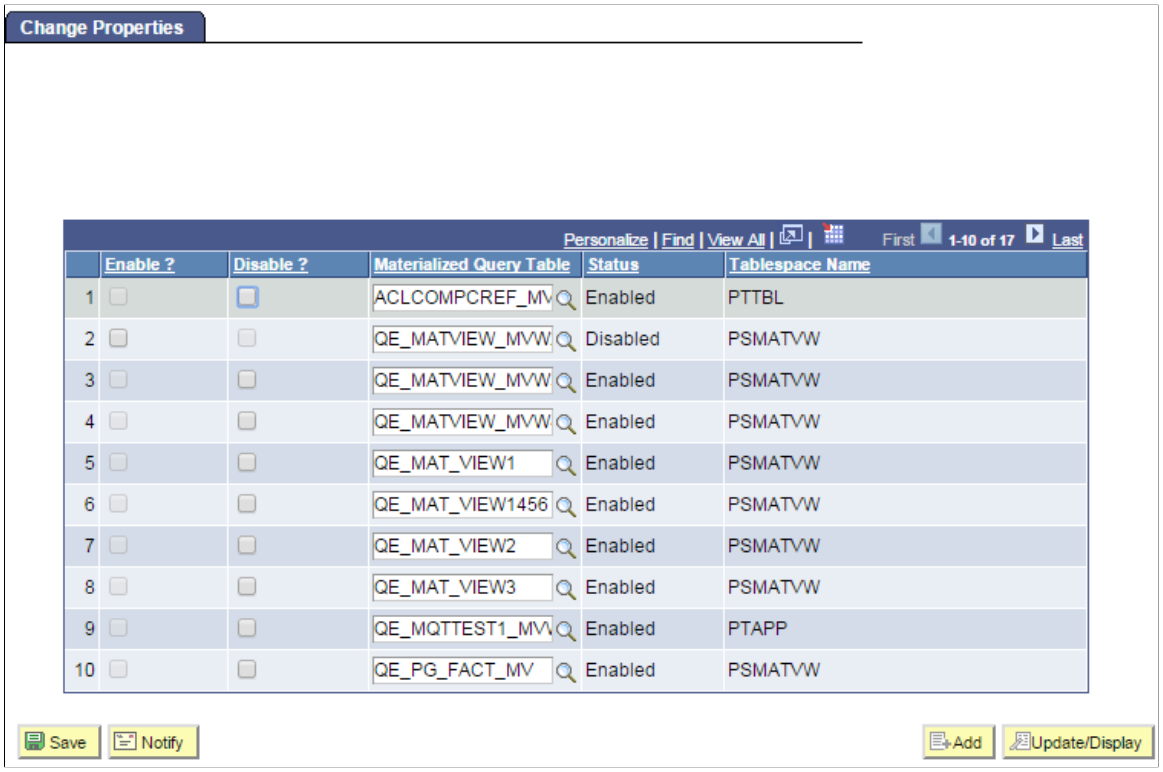
For more information on Process Scheduler Request page see, Maintaining Materialized Views documented under the Oracle platform section.

Enabling Materialized Query Tables

Access the Change Properties page (**PeopleTools** > **Utilities** > **Administration** > **Materialized Views** > **Enable Materialized Views**).

The page displays a list of the views delivered from the applications. You can select specific views to enable or disable materialized view feature.

This example illustrates the Change Properties page. The columns on the page are described below:



This example illustrates the Enable Materialized Query Tables page. The columns on the page are described below:

Field or Control	***Description***
Owner	Search and select an owner of the tables to limit the search result.

Field or Control	*Description*
Enable?	Select the record to render it as a materialized view.
Disable?	Select to convert the MQT to a normal SQL view.
Materialized Query Table	Enter the name of the MQT. You can use the prompt for selecting the name.
Status	Displays **Enabled** or **Disabled** depending on the current status of the view.
Tablespace Name	Displays the tablespace name to which the record is assigned when it is materialized.

Save the page after you select the views to enable or disable MQTs. Open the ENABLEMV project in the Application Designer and build the project to render the selected views as materialized query tables. Similarly, open the DISABLEMV project in the Application Designer and build it to convert the materialized query tables to normal SQL views.

Audits for Materialized Query tables

You can run the following DDDAudit queries for materialized query tables to resolve any inconsistency in the database.

- MQT–1
- MQT–2
- MQT–3
- MQT–4

See the Materialized Query Table Queries [DB2 ZOS] section for detailed audit queries.

Chapter 9

Administering PeopleSoft Databases on Oracle

Working With Oracle Connectivity

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 BEQUEATH_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. **Note:** A LOCAL connection (or BEQ) is not possible with Oracle 12c or higher if a multi-tenant DB configuration is used. If a customer is still using a Single instance DB (non-multi-tenant) then a local connection is still supported, by enabling this and the following parameter Oracle_SID
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.

Working With Oracle Database In-Memory

Oracle Database 12c offers Oracle Database In-Memory, which is also known as the In-Memory Column Store (IMCS). With the IMCS, a single database can deliver high-speed performance for transactions and simultaneously support real-time analytics and reporting. You can enable the IMCS for tables, partitions, tablespaces, and materialized views. This section describes the Dual Memory Format used in IMCS. It also covers benefits, restrictions, and configurations of IMCS.

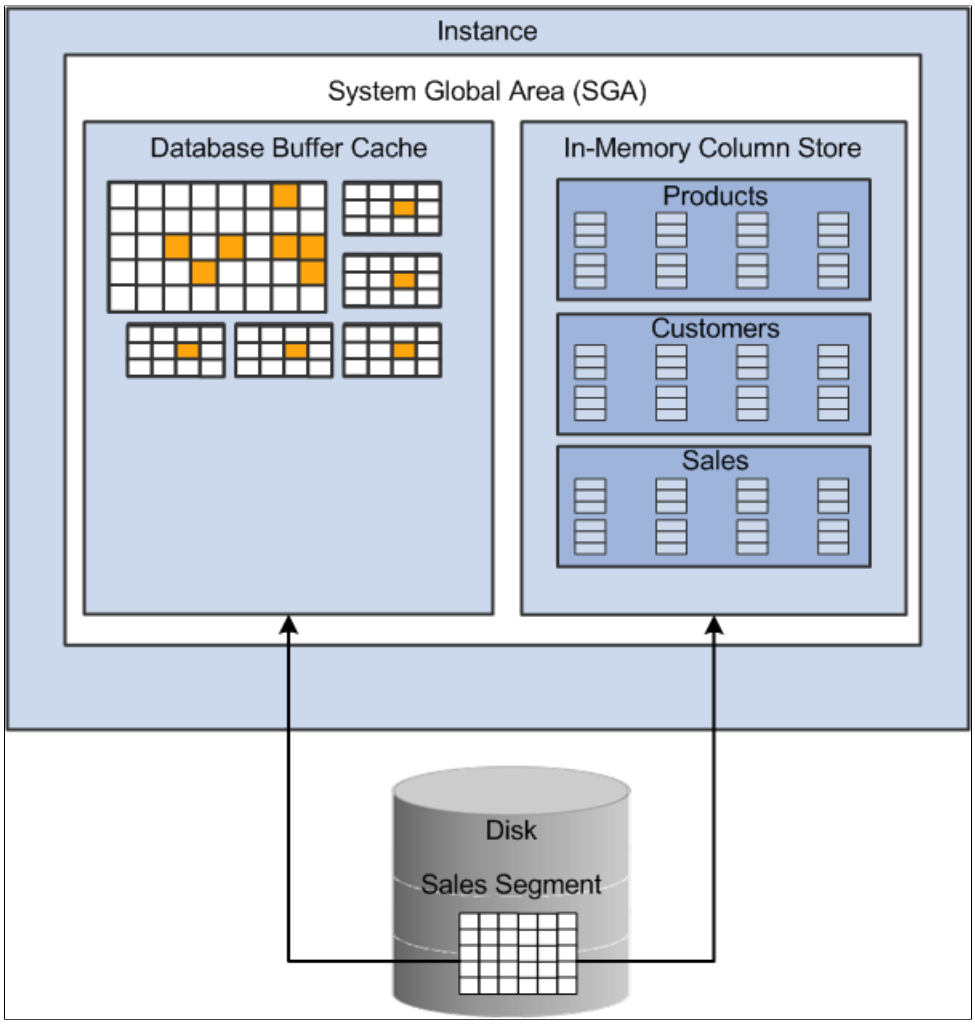
Understanding Oracle Database In-Memory

The IMCS is a static pool in the Oracle System Global Area (SGA) associated with the Oracle Database. It stores copies of objects in the memory in a columnar format. The IMCS does not replace the buffer cache but supplements it so that both the memory areas can store data in different formats.

The Oracle Database reads data from IMCS or the database buffer cache or both within the same query. The Online Transaction Processing (OLTP) queries fetch data from the buffer cache, where the data is formatted in rows. Analytical or reporting queries fetch data from the IMCS. This dual format:

- Improves the performance of the database.
- Allows more analytical queries to run with real-time transaction data without affecting the existing workload.
- Makes the database active and simultaneously consistent in all transactions.
- Structures the data optimized for rapid scans.

This example shows data in Oracle SGA stored in the IMCS in a column format and in the database buffer cache in a row format.



You can specify the amount of memory allocated for IMCS using the *INMEMORY_SIZE* parameter. Only objects specified as `INMEMORY` using DDL statements are populated into the IMCS.

Benefits of Oracle In-Memory Database

Using the IMCS enhances Oracle Database performance. It can perform scans, joins, and aggregates at a rapid speed. Business applications, ad-hoc analytical queries, and data warehouse workloads benefit most; databases that only deal with online transaction processes and run short transactions using index lookups benefit less. The benefits of IMCS are:

- Scans many rows and applies filters that use operators such as =, <, >, and **IN**.
- Queries a subset of columns in a table, for example, selecting 5 of 100 columns.
- Accelerates joins by converting predicates on small dimension tables into filters on a large fact table.
- Supports all existing database features, including High Availability features.
- Requires no application changes.

- Requires fewer indexes.
- Reduces the storage space in memory with fewer pre-built objects that significantly lower processing overhead.

Restrictions Using In-Memory Column Store in PeopleSoft Applications

Note: PeopleSoft does not support Materialized Views in In-Memory database at present.

Configuration Restrictions

- IMCS is not supported in the Active Data Guard configuration. For information on the Active Data Guard, see Implementing Oracle Active Data Guard.
- In-memory is not supported in system and system auxiliary tablespaces.
- The IMCS does not support LONG or LONG RAW columns, out-of-line column (LOB, varray, nested table column), or extended data type columns. If you enable a table for the IMCS and it contains any of these types of columns, then the columns will not be populated in the IMCS.
- You cannot specify this inmemory_column_clause for a LONG or LONG RAW column, an out-of-line column (LOB, varray, nested table column), or an extended data type.

Specific Object Type Restrictions

Global temporary tables do not support IMCS because they cannot be altered in memory.

For information on Global Temporary Tables, see “Understanding Global Temporary Tables” (Application Engine).

PeopleSoft Object Type Restriction

PeopleSoft temporary tables are not supported in IMCS because of the following overheads:

- The in-memory has to be updated after every transaction to keep it consistent with the row-oriented format.
- After initial population of IMCS objects, the temporary table must be scanned so that the data can be loaded into memory. After every subsequent truncation of the temporary table, this has to be repeated.
- After the rows are loaded into the memory, activities on the data such as deletion and insertion will change the columnar structures for the in-memory temporary object.
- For tables with a high rate of data manipulation activity, the MEMCOMPRESS FOR DML compression option (little to no compression) is recommended, which means that PeopleTools would have to explicitly support this compression option rather than just the default of MEMCOMPRESS FOR QUERY.

Configuring the Oracle Database In-Memory

To configure the In-Memory Database (IMDB):

1. Set the *INMEMORY_SIZE* initialization parameter. This parameter specifies the amount of memory reserved for use by the IMCS. For example:

   ```
   INMEMORY_SIZE=60GB
   ```

2. Use DDL statements to specify the tablespaces, tables, partitions, or columns to be read into the IMCS. For example:

   ```
   ALTER TABLE ps_ledger INMEMORY MEMCOMPRESS FOR QUERY DUPLICATE;
   ```

3. Enter SELECT commands to populate the IMCS to load the IMCS faster. For example, the following is an SQL script for **ALTERING** a table (for example,. PS_LEDGER) in memory and parallelizing a **SELECT** command to load the IMCS representation of the TABLE.

   ```
   set timing onset echo onspool load_gl_bal_tables_inmem.logalter system set "_m⇒
   ax_spacebg_slaves"=32 scope=both;alter table ps_ledger inmemory memcompress fo⇒
   r query duplicate;select /*+ full(ps_ledger) parallel(32) */ count(*) fromps_l⇒
   edger;spool off
   ```

Any subsequent SQL operation against this table (PS_LEDGER) that the optimizer determines could benefit from using the IMCS will be directed to use the IMCS.

Configuring the Oracle Database In-Memory for PeopleSoft Applications

In PeopleSoft, you require to designate tables or columns in specific tables to use the In-Memory feature of the database. Then apply the `ALTER IN-MEMORY` DDL command to load them in IMCS. This section describes the following requirements:

- Design time requirements.
- Implementation time requirements.

Design Time Requirements

- You will be able to set TABLES or COLUMNs on TABLES as IN-MEMORY objects in PeopleTools RECORD and FIELD definitions using the Application Designer.
- You will be able to run queries on compressed columns in the Oracle In-Memory compression format using the keyword **MEMCOMPRESS**, a subclause of the INMEMORY attribute. Data populated in IMCS is compressed using a set of compression algorithms that not only save space but allow queries to be run directly against the compressed column for faster performance.

You should keep in mind the restrictions related to IMCS that are described in Restrictions Using In-Memory Column Store in PeopleSoft Applications.

Implementation Time Requirements

During installation or upgrade of PeopleSoft applications, certain `ALTER_INMEMORY` DDL statements are necessary to make use of the IMCS feature.

For information on Install/Upgrade requirements for IMCS, see PeopleSoft application specific Installation or Upgrade documentation.

For information on installing the Oracle Database, see the product documentation for *PeopleSoft 9.2 Application Installation for Oracle*.

Monitoring PeopleSoft Database Connections

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

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*
ABSMITH (uppercase)	NETWORK login ID for Windows workstation.
ABSMITH123199	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 ABSMITH.

```
SQL> /
SPID     OSUSER  USERNAME PROGRAM                        CLIENTINFO

-------- ------- -------- ---------------------------- ------------------
15276    ABSMITH 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    ABSMITH 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    ABSMITH 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    ABSMITH 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    ABSMITH 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    ABSMITH 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 ABSMITH123199, 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    ABSMITH PT844P01 SQLPLUSW.EXE                                             ⇒


15449    ABSMITH 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 ABSMITH123199, submitted from PeopleSoft user ID PS.

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


-------- ------- -------- ------------------------------ --------------------------⇒

----------
15387    TMJONES PT844P01 pside.exe                      PS,TMJONES,TMJONES110299,,p⇒

side.exe,
15276    ABSMITH PT844P01 SQLPLUSW.EXE                                             ⇒


15449    ABSMITH PT844P01 sqrw.exe                       PS,329                    ⇒


15451    ABSMITH PT844P01 PTPTEDIT.exe                   PS,ABSMITH,ABSMITH123199,,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    ABSMITH PT844P01 SQLPLUSW.EXE                                             ⇒


15449    ABSMITH PT844P01 sqrw.exe                       PS,329                    ⇒


15451    ABSMITH PT844P01 PTPTEDIT.exe                   PS,ABSMITH,ABSMITH123199,,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    ABSMITH PT844P01 SQLPLUSW.EXE                                             ⇒


15449    ABSMITH PT844P01 sqrw.exe                       PS,329                    ⇒


15451    ABSMITH PT844P01 PTPTEDIT.exe                   PS,ABSMITH,ABSMITH123199,,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    ABSMITH PT844P01 SQLPLUSW.EXE                                             ⇒


15449    ABSMITH PT844P01 sqrw.exe                       PS,329                    ⇒


15451    ABSMITH PT844P01 PTPTEDIT.exe                   PS,ABSMITH,ABSMITH123199,,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    ABSMITH PT844P01 SQLPLUSW.EXE                                             ⇒


15451    ABSMITH PT844P01 PTPTEDIT.exe                   PS,ABSMITH,ABSMITH123199,,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    ABSMITH 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 prognam 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@myserver.example.com (TNS V1-V3)  QEDMO                       QEDMO,,192.0⇒

.2.10,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,,myserver.example.com,,PSAESRV⇒
```


,

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 maintain partition definitions.

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.

Related Links

Oracle Partitioning Overview

Partitioning Terminology

The following table defines partitioning-related terms, and describes the types of partitioning that are supported.

Term	*Definition*
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.

Term	*Definition*
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.

Term	*Definition*
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 was introduced in Oracle 8i • Range-list partitioning was introduced in Oracle 9i • Range-range partitioning was 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.

Term	*Definition*
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.

Term	*Definition*
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 composite/sub-partitioning types are not supported:

- List-Hash
- Range-Hash
- Interval-Range

- Interval-List
- Interval-Interval

Establishing Partitioning Definitions

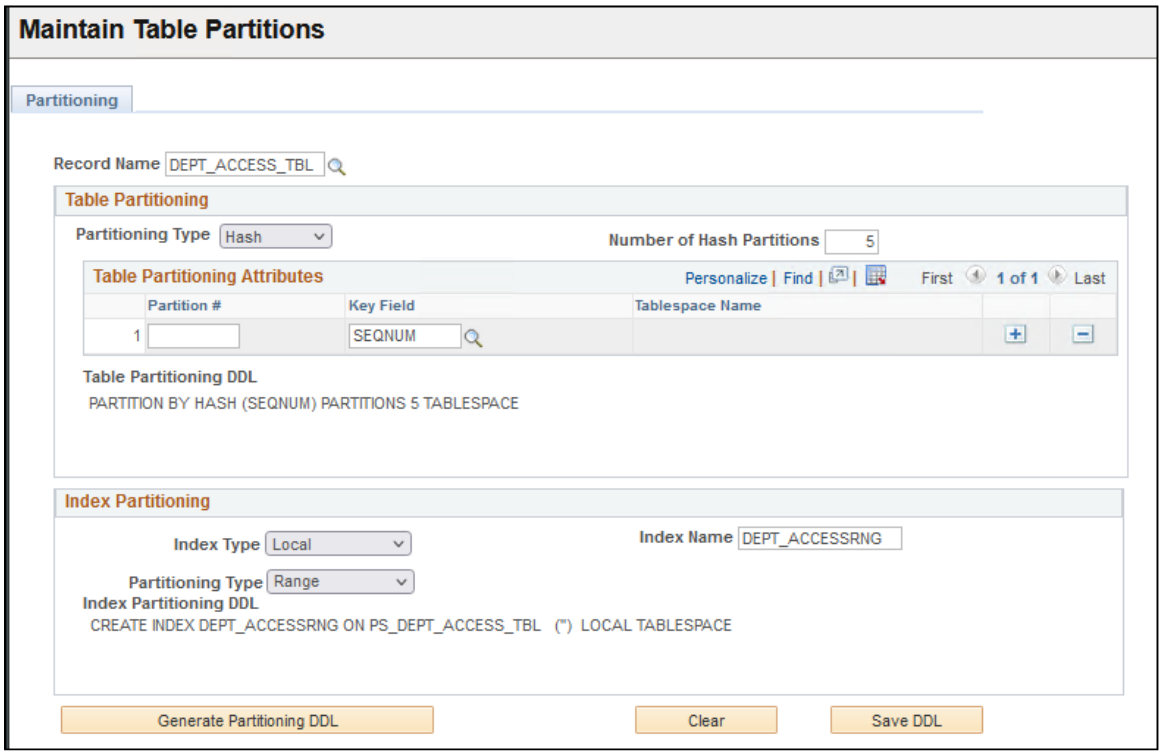
To establish partitioning definitions, use the Partitioning page (PPMU_GEN_DDL_PG).

This page enables you to define partitioning parameters and generates the DDL scripts to create partitioned tables and indexes.

Navigation:

PeopleTools > **Utilities** > **Maintain Table Partitions**

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.

Field or Control	*Description*
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.

Field or Control	***Description***
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*.
Number 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 #** (partition number), **Key Field**, **Tablespace Name**. For list partitions: **Partition #**, **Key Field**, **Key Value**, **Partition Name**, **Tablespace Name**. For range: **Partition #**, **Key Field**, **Key Value**, **Partition Name**, **Tablespace Name**

Field or Control	*Description*
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 Required check box is selected, use the fields within this grid to specify sub-partitioning parameters. For each sub-partition, complete the following fields: **Partition #**, **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.

Field or Control	*Description*
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 #** (partition number), **Key Field**, **Tablespace Name** For list partitions: **Partition #**, **Key Field**, **Key Value**, **Partition Name**, **Tablespace Name** For range: **Partition #**, **Key Field**, **Key Value**, **Partition Name**, **Tablespace Name**
Number 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.

Field or Control	*Description*
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

Field or Control	*Description*
Generate Partitioning DDL	Click to populate the Table Partitioning DDL and Index Partitioning DDL fields.
Clear	Click to clear all values in the page.

Field or Control	*Description*
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.

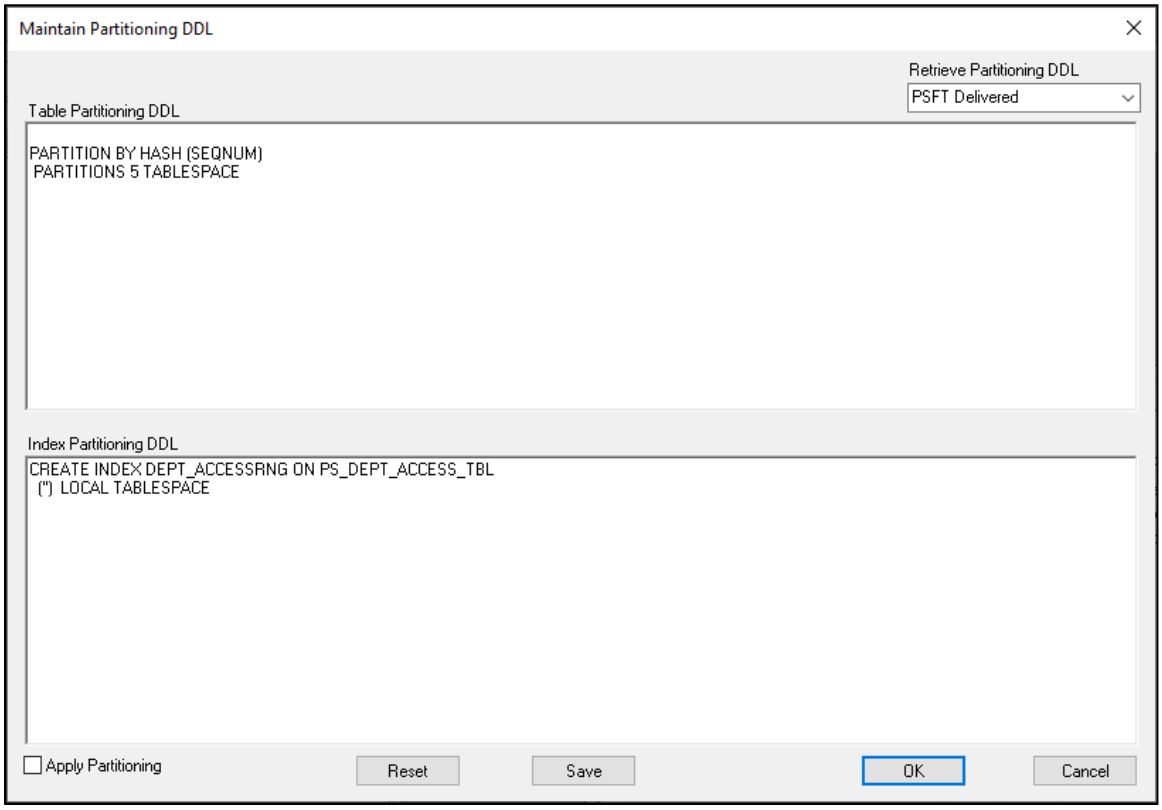
Applying and Maintaining Partitioning DDL

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**

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.

Field or Control	***Description***
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 Partitioning page, in 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.

Field or Control	*Description*
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 ADS Project” (Lifecycle Management Guide).
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 ADS Project File Locations” (Lifecycle Management Guide).

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” (Lifecycle Management Guide).

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, see your Oracle Database documentation.

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.

Multitenant Architecture Overview

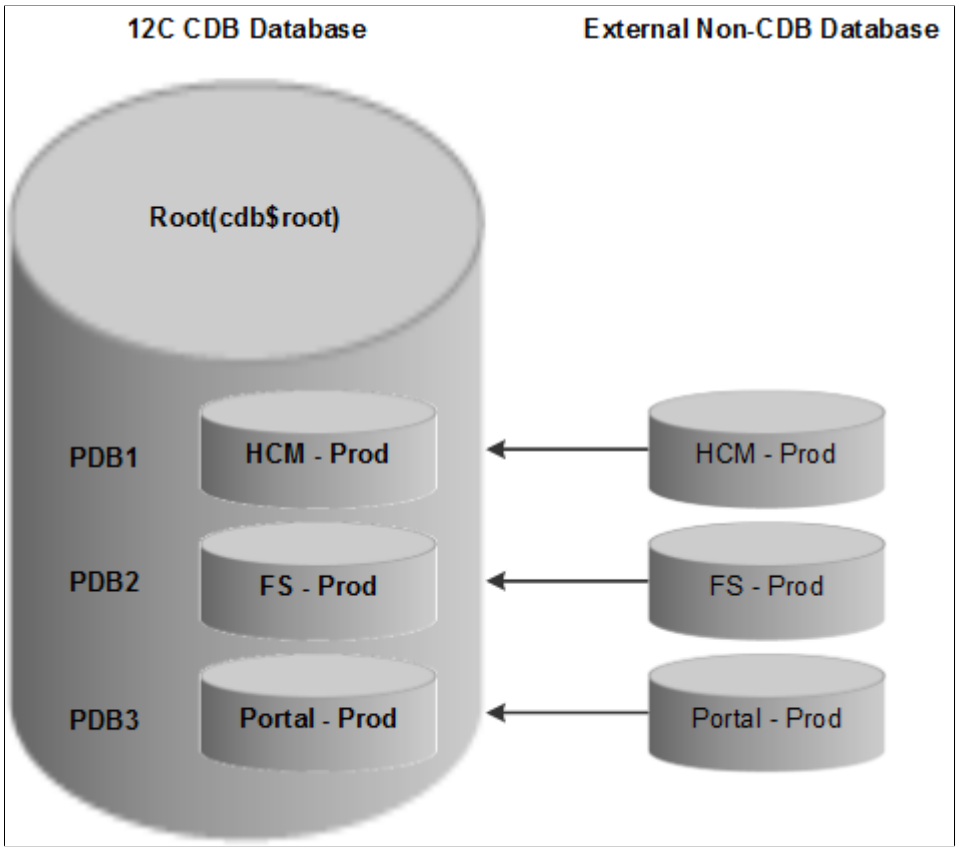
A container is a collection of schemas, objects, and related structures in a (CDB) that appears logically to an application as a separate database. Within a CDB, each container has a unique ID and name.

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.

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).

See *PeopleSoft 9.2 Application Installation for Oracle*.

Related Documentation

For detailed information about pluggable databases and the Oracle Multitenant Architecture see:

- Oracle Multitenant at https://www.oracle.com/database/multitenant/.
- *Oracle Database 19c Administrator's Guide*, "Introduction to the Multitenant Architecture".
- *Oracle Database 19c Administrator's Guide*, "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.

Conceptually similar indexed views and summary tables are present in Microsoft SQL platform and materialized query tables are available in Db2 z/OS platform.

For more information on materialized views, see the latest Oracle Database documentation.

Related Links

Working with Materialized Query Tables
Working with Indexed Views

Defining Materialized Views

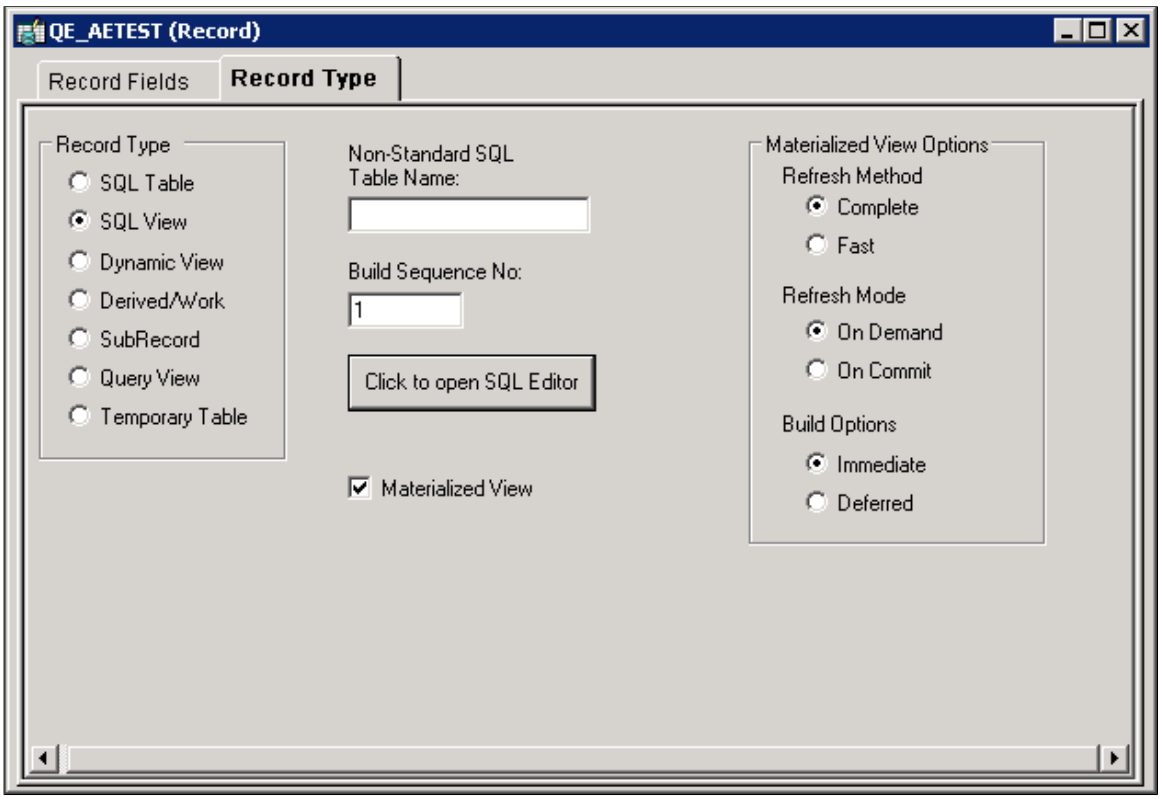
To define a materialized view:

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.

This example shows the Record Type tab when creating a materialized view on the Oracle platform



6. Specify the **Refresh Method**. Options are:

Field or Control	***Description***
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:

Field or Control	***Description***
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.

Field or Control	*Description*
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:

Field or Control	*Description*
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” (Application Designer Developer’s Guide)
- “Running the Build Process” (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 .

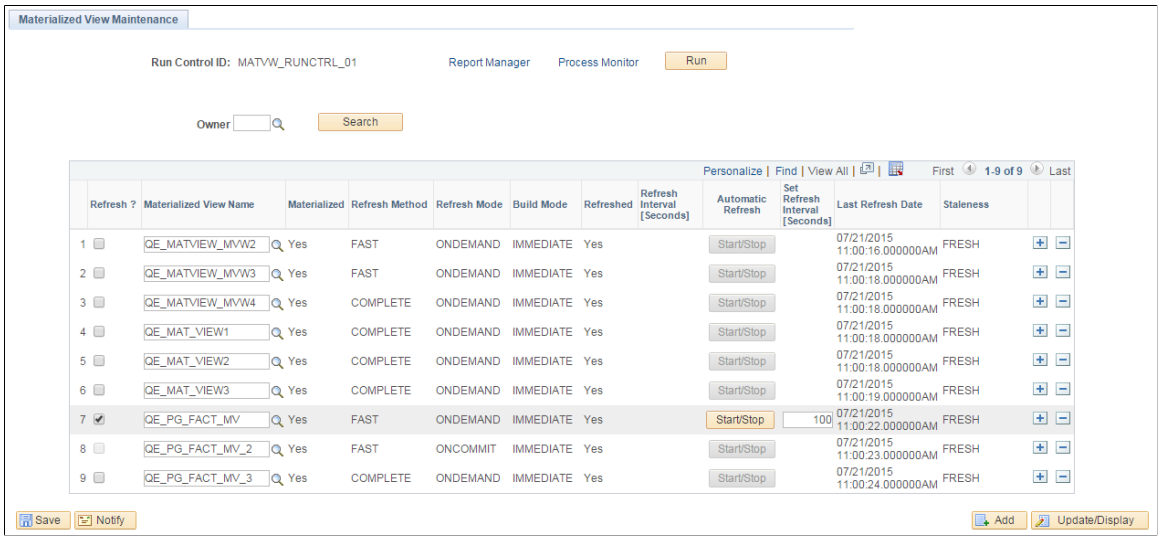
5. Save and build the record.

Maintaining Materialized Views

For ongoing maintenance of materialized views, administrators can use the Materialized View Maintenance page (PTMAT_MAINT). Administrators must have the Materialized View Administrator role to have permission to access this page.

Access the Materialized View Maintenance run control page (**PeopleTools** > **Utilities** > **Administration** > **Materialized Views** > **Maintain Materialized Views**).

This image illustrates the fields and controls on the Materialized View Maintenance page.



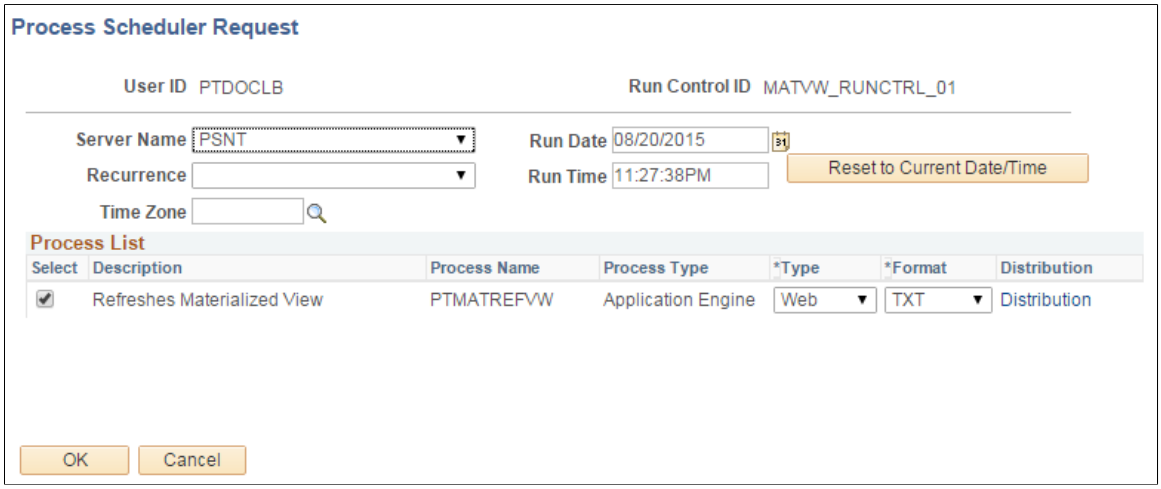
This page contains a grid that is populated with a list of the records in the materialized views table, PSPTMATVWDEFN.

Field or Control	***Description***
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.

Field or Control	*Description*
Refreshed	Indicates if the record is refreshed (Yes) or not (No).
Refresh Interval [Seconds]	Displays the time interval in seconds for the refresh procedure to run.
Automatic Refresh	Toggle the button to start a refresh schedule every time interval in seconds.
Set Refresh Interval [Seconds]	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.

When you click the **Run** button, the PTMATREFVW Application Engine program executes. It brings up the **Process Scheduler Request** page. Enter the appropriate server on the page and click the **OK** button to initiate the process.

The following illustrates the PTMATREFVW Application Engine process that requires to be initiated for refreshing the materialized views.

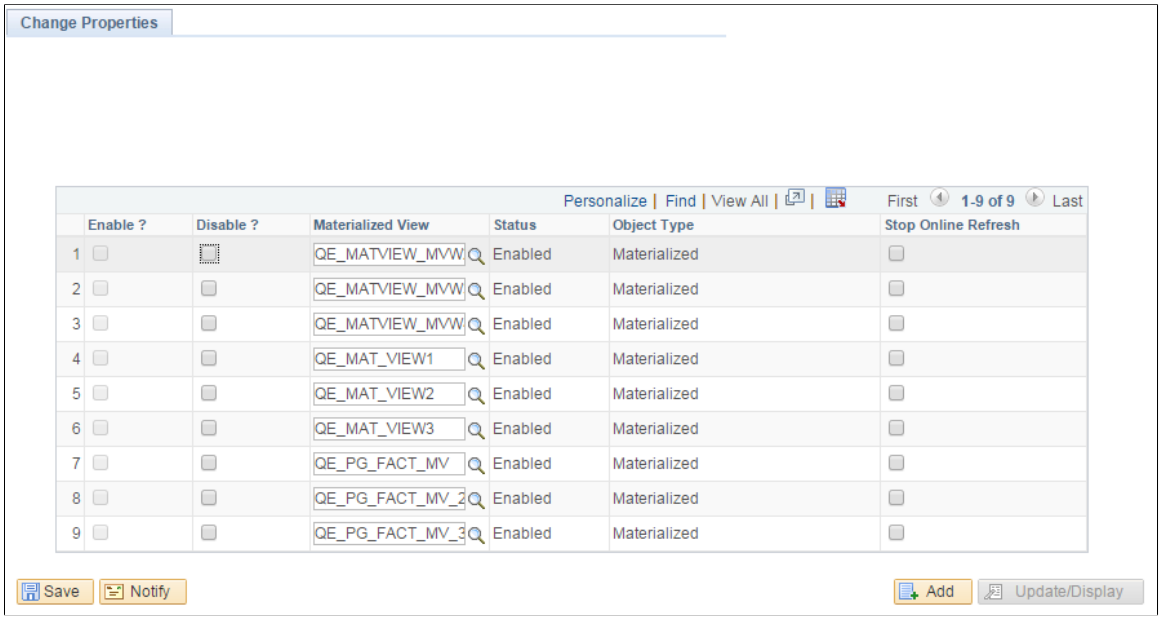


Enabling Materialized Views

Access the Change Properties page (**PeopleTools** > **Utilities** > **Administration** > **Materialized Views** > **Enable Materialized Views**).

You can enable or disable the materialized view feature on an application delivered view.

Change Properties page allows you to select views to enable or disable materialized views on them.



Field or Control	*Description*
Enable?	Select to convert the view to a materialized view.
Disable?	Select to convert a materialized view to a normal SQL view.
Materialized View	Search and select the view. You can use the prompt to select the name.
Status	Displays enabled if the views are already selected to enable the materialized view feature. Displays disabled if the materialized views are already selected to disable the materialized view feature.
Object Type	Displays the current status of the view as materialized or normal SQL view.
Stop Online Refresh	Select to restrict the view from getting refreshed automatically. **Note:** Select to stop Pivot Grid initiated materialized views for automatic refresh.

Note: Once you enable materialized views from PIA, go to the PeopleSoft Application Designer and build record to create the materialized view. Similarly, once you disabled the materialized views from PIA, go to the PeopleSoft Application Designer and build record to create the normal view.

Audits for Materialized Views

You can run DDDAUDIT queries to check for materialized views to resolve any inconsistency in the database:

- MVIEWS-1
- MVIEWS-2
- MVIEWS-3

See the section Materialized View Queries [Oracle] for detailed audit queries.

Understanding Query Rewrite

Note: PeopleSoft enables Query Rewrite only for those materialized views whose refresh method is FAST.

The materialized views are created from complex queries and aggregate data from various tables. Query Rewrite is a process that answers the queries using materialized views. It transforms a SQL statement that references tables and views into a statement accessing one or more materialized views that are defined on the detail tables.

For more information on Query Rewrite, see the latest Oracle Database documentation.

Refresh Mode/Method Combinations

The following table lists the conditions for the various refresh mode/method combinations.

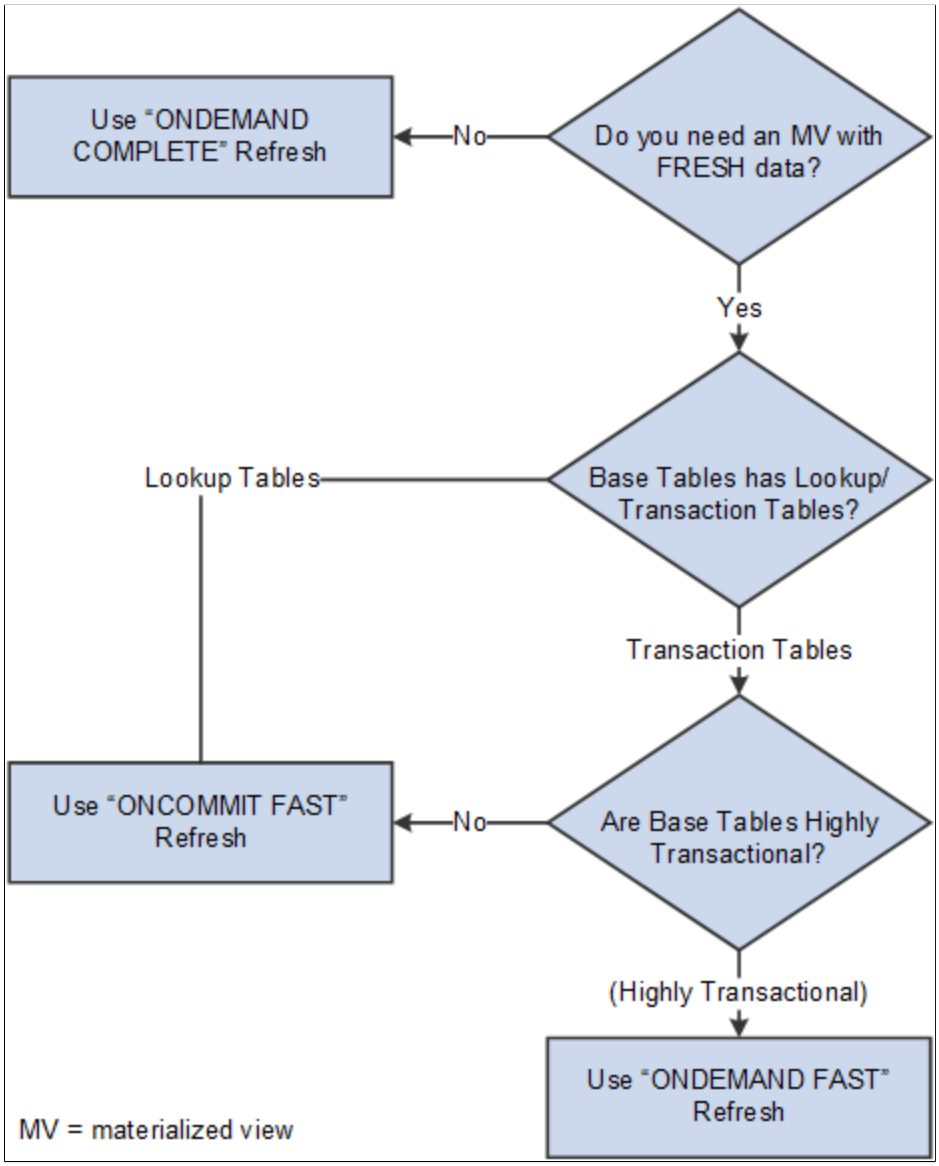
Description	*ON COMMIT/FAST REFRESH*	*ON DEMAND/FAST REFRESH*	*ON DEMAND/COMPLETE REFRESH*
Use Case	Pivot Grid based on lookup and transaction tables	Pivot Grid based on lookup and transaction tables	Staging tables
Data Status (stale/fresh)	Always fresh	Relatively fresh	Stale data
Refresh Cost	Refresh 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.

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 ONDEMAND/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” (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, such as PeopleSoft Configuration Manager. 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

PeopleSoft Resource Name	*Description*	*Mapped Executable(s)*
		PSQED 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

PeopleSoft Resource Name	*Description*	*Mapped Executable(s)*
QUERY SERVER	Executables required to process PeopleSoft Query requests.	PSQRYSRV
QUICK SERVER	Executables required for running SQR for PeopleSoft requests.	PSQCKSRV
SQR	Executables required for running SQR for PeopleSoft requests.	PSSQR

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.

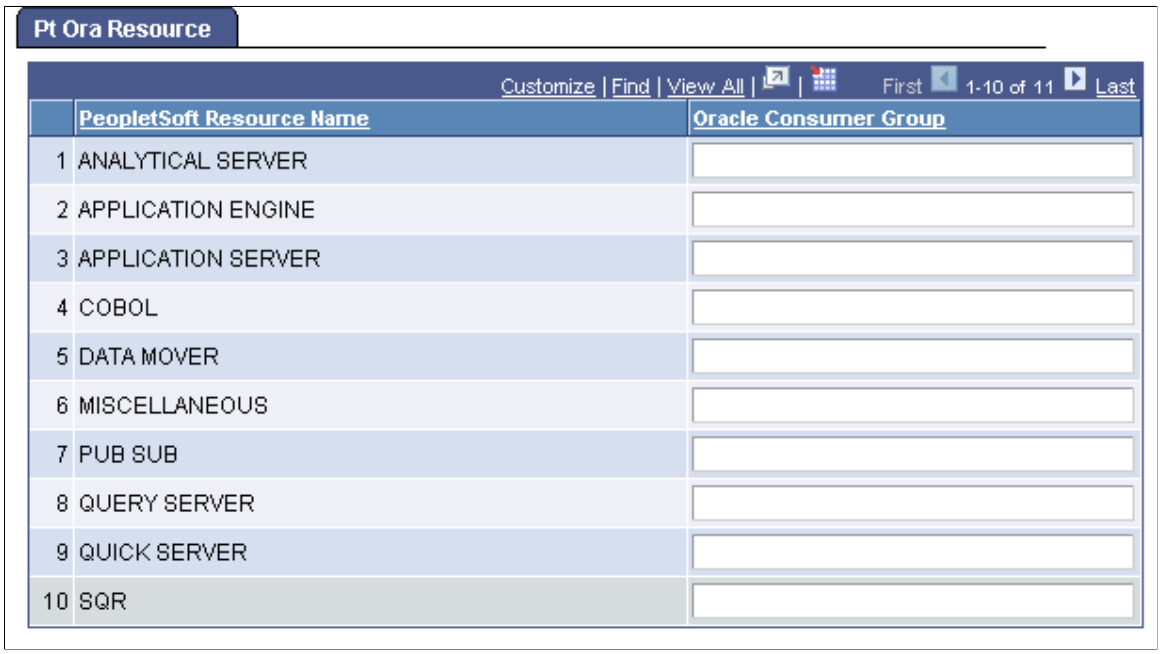
```
BEGIN

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** > **Manage Oracle Resource Groups** to access the Pt Ora Resource page.

This example illustrates the fields and controls on the Pt Ora Resource page.



Field or Control	*Description*
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 set Oracle Transparent Data Encryption.

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.

Oracle Wallet can also be configured using OpenSSL and Orapki tool.

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;
```

Related Links

“Setting Up Oracle Wallet Using OpenSSL” (System and Server Administration)
“Setting Up Oracle Wallet Using ORAPKI” (System and Server Administration)

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 protecting and managing PeopleSoft applications with Database Vault.

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 in the Oracle Database Vault documentation.

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:

- Access ID (default is SYSADM)
- Connect ID (default is people)
- PS (owns the PSDBOWNER table)

The access ID is the schema owner for all database objects related to a specific PeopleSoft application installation. The access ID and access ID password are stored and encrypted in the PeopleSoft security table PSACCESSPROFILE.

```
/*DESCR PSACCESSPROFILE*/
Name                 Null       Type
-------------------  --------   -------------
SYMBOLICID           NOT NULL   VARCHAR2(8)
STM_ACCESS_ID        NOT NULL   VARCHAR2(254)
STM_ACCESS_PSWD      NOT NULL   VARCHAR2(254)
STM_ACCESS_PART1     NOT NULL   VARCHAR2(128)
STM_ACCESS_PART2     NOT NULL   VARCHAR2(128)
ENCRYPTED            NOT NULL   NUMBER(38)
STM_ENCRYPTION_VER   NOT NULL   NUMBER(38)
VERSION              NOT NULL   NUMBER(38)

/*SELECT  * from PSACCESSPROFILE*/
SYMBOLIC ID          STM_ACCESS_ID
<SYMBOLIC_ID_NAME>   mvsrhK/De0GTCI/sNISVxKkeilFx23SYSsZ2Mlxj

STM_ACCESS_PSWD                              STM_ACCESS_PART1
mvsrhK/De0GTCI/sNISVxKkeilFx23SYSsZ2Mlxj    7ZP4zFqIGDc=

STM_ACCESS_PART2      ENCRYPTED        STM_ENCRYPTED_VER    VERSION
O2/Cz9HFQZ4=          1                1                    1040
```

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 access ID 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 access ID 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 access ID 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 access ID 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 access ID 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 access ID 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 access ID 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 access ID 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 access ID or connect ID, the PeopleSoft system will still have the old password stored in the PeopleSoft security meta data tables and configuration files. These changed passwords will have to be reflected in the PeopleSoft security meta data 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/password@QE855C42

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  'password'>
Retype new password: <changed to  'password'>
Password changed

SQL> exit
```

Or,

```
ALTER USER QE855C42 IDENTIFIED BY CHANGEPW ACCOUNT UNLOCK;

User altered.

ALTER USER people IDENTIFIED BY password 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 access ID password) to run the necessary commands to change the access ID passwords on the appropriate PeopleSoft meta data tables. For example,

```
SET LOG c:\temp\changeaccessidpswd.out;
UPDATE PSSTATUS SET OWNERID = <OWNER_ID_NAME>;
UPDATE PSOPRDEFN SET OPERPSWD = <OPRID_PSWD>, ACCTLOCK=0,
   ENCRYPTED = 0;
UPDATE PSACCESSPROFILE SET STM_ACCESS_ID = <ACCESS_ID_NAME>,
   STM_ACCESS_PSWD = <NEW_ACCESS_ID_PSWD>,
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=<DATABASE_NAME>
DBType=<DATABASE_TYPE>
UserId=<USER_ID>
UserPswd=<USER_ID_PSWD>
ConnectId=<CONNECT_ID>
ConnectPswd=<CONNECT_ID_PSWD>
ServerName=<SERVER_NAME>
```

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.
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 submitted just before or during primary instance failure	Oracle reconnects and reconstructs the session on a surviving node but SQR 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 SQR job was not in an open-transaction nor executing DMLs and was performing only SELECT statements, it will fail over and complete successfully.
SQR jobs submitted just after primary instance failure.	Oracle reconnects and reconstructs the session on a surviving node, the SQR job is then submitted on the new primary database and completes successfully.
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 steps to configure the Oracle Active Data Guard.

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.

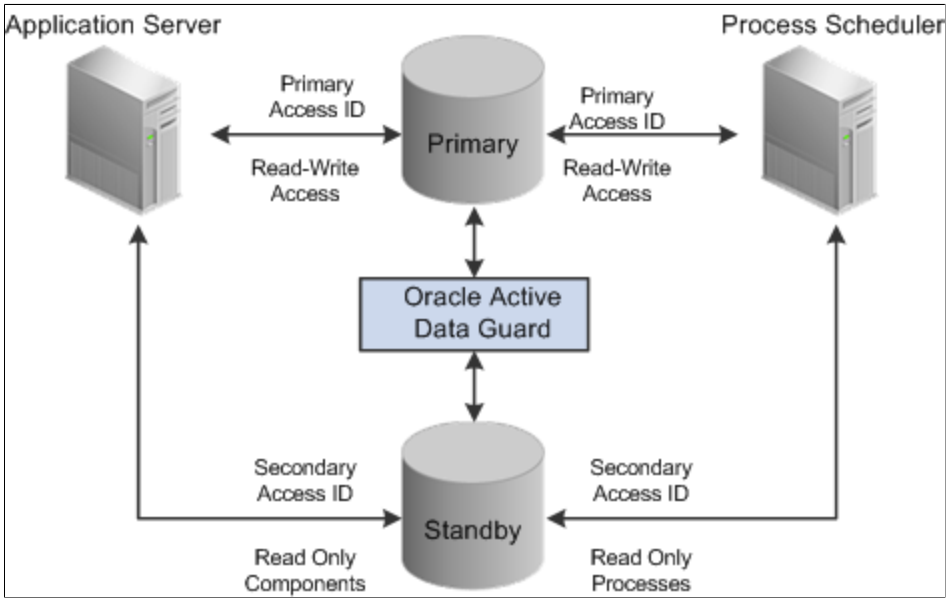
The intent of this feature integration is that *most* SQLs in a mostly-read-only (MRO) marked unit of work are redirected to the standby database in an ADG configuration. The ability to redirect a unit of work (UOW) to a standby database is limited to PeopleSoft components and processes. Within any MRO 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.

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.
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.

Element	*Description*
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: Application Engine 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.

Element	*Description*
	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. **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. 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;
   ```
3. 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** drop-down list on the **General** tab for the user ID you just created, select the symbolic ID associated with your secondary access ID.

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 against the primary database while connected as 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.

Script	*Description*	*Usage*
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 while 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=password
    ConnectId=people
    ConnectPswd=password
    ServerName=
    StandbyDBName=STANDBY
    StandbyDBType=ORACLE
    StandbyUserId=PTSTNDBY
    StandbyUserPswd=password
```

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** > **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
;============================================================================
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 steps to configure Oracle Golden Gate for PeopleSoft Off-Load reporting.

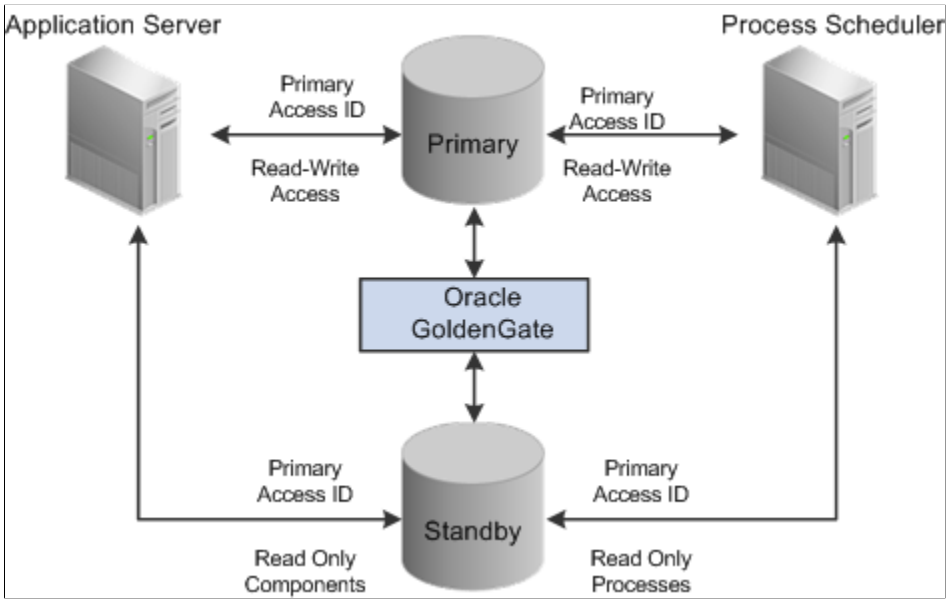
Note: The scripts given below are for Oracle 12c container database.

Note: For non container database replace C##OGGUSER with OGGUSER.

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.

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 EMDBO as your access ID on the primary database, you must also use EMDBO as the access ID on the standby database, and you must keep the passwords for EMDBO 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.

Element	Description
	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. **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.

Related Links

Using Pluggable Databases
Implementing Oracle Golden Gate on Oracle 11g Database

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.

Subdirectory	*Description*
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).

```
<@hostname:>$ cd \data1\ogg\primary
<@hostname:>$ ./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 (<@hostname>) 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:

```
<@hostname:>$ cd \data1\ogg\primary
<@hostname:>$ ./ggsci

GGSCI (<@hostname>) 2> info all

Program     Status      Group       Lag           Time Since Chkpt
```



```
MANAGER     STOPPED

GGSCI (<@hostname>) 3>
```

Configuring PeopleSoft Databases for Oracle GoldenGate

This section explains the 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.

User Privilege	*Extract*	*Replicat*
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:

```
On Primary DB
SQL> create user C##OGGUSER identified by OGGUSER;

User created.
```



```
-----------------------------------------------------------

SQL> exec dbms_goldengate_auth.grant_admin_privilege('C##OGGUSER',container=>'all')⇒

;

PL/SQL procedure successfully completed.

--####Change the pluggable database#####
SQL> alter session set container=PT855GA;

Session altered.

GRANT CONNECT, RESOURCE to C##OGGUSER;
GRANT CREATE SESSION to C##OGGUSER;
GRANT ALTER SESSION to C##OGGUSER;
GRANT SELECT ANY DICTIONARY to C##OGGUSER;
GRANT FLASHBACK ANY TABLE to C##OGGUSER;
GRANT ALTER ANY TABLE to C##OGGUSER;
GRANT SELECT ANY TABLE to C##OGGUSER;
GRANT INSERT ANY TABLE to C##OGGUSER;
GRANT DELETE ANY TABLE to C##OGGUSER;
GRANT UPDATE ANY TABLE to C##OGGUSER;
GRANT CREATE TABLE to C##OGGUSER;
GRANT UNLIMITED TABLESPACE to C##OGGUSER;
GRANT EXECUTE on DBMS_FLASHBACK to C##OGGUSER;
GRANT SELECT ON dba_clusters to C##OGGUSER

Grant succeeded.

On Standby DB
SQL> create user C##OGGUSER identified by OGGUSER;
User created.

SQL> exec dbms_goldengate_auth.grant_admin_privilege('C##OGGUSER',container=>'all')⇒

;

PL/SQL procedure successfully completed.

SQL> alter session set container=STD855GA;

Session altered.

SQL> GRANT CONNECT, RESOURCE to C##OGGUSER;
GRANT CREATE SESSION to C##OGGUSER;
GRANT ALTER SESSION to C##OGGUSER;
GRANT SELECT ANY DICTIONARY to C##OGGUSER;
GRANT FLASHBACK ANY TABLE to C##OGGUSER;
GRANT ALTER ANY TABLE to C##OGGUSER;
GRANT SELECT ANY TABLE to C##OGGUSER;
GRANT INSERT ANY TABLE to C##OGGUSER;
GRANT DELETE ANY TABLE to C##OGGUSER;
GRANT UPDATE ANY TABLE to C##OGGUSER;
GRANT CREATE TABLE to C##OGGUSER;
GRANT UNLIMITED TABLESPACE to C##OGGUSER;
GRANT EXECUTE on DBMS_FLASHBACK to C##OGGUSER;
GRANT SELECT ON dba_clusters to C##OGGUSER;
```

Note: For non container database replace C##OGGUSER with OGGUSER.

Listing the Privileges Granted to the GoldenGate User

To list the privileges granted to the OGG User, you can run the following script for a container database:

```
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('C##OGGUSER')
 /* THE ROLES TO ROLES RELATIONS */
union
   select
  grantee,
granted_role
 from
dba_role_privs
)
start with grantee is null
connect by grantee = prior granted_role;
```

Append the following script to the script above, if you are using non-container database:

```
  /* THE ROLES TO PRIVILEGE RELATIONS */
  union
    select
      grantee,
      privilege
    from
      dba_sys_privs
```

The above script generates following result:

```
C##OGGUSER
ALTER ANY TABLE
ALTER SESSION
CREATE ANY EDITION
CREATE EVALUATION CONTEXT
CREATE JOB
CREATE RULE
CREATE RULE SET
DEQUEUE ANY QUEUE
DROP ANY EDITION
EXECUTE ANY RULE SET
FLASHBACK ANY TABLE
LOGMINING
SELECT ANY TABLE
SELECT_CATALOG_ROLE
HS_ADMIN_SELECT_ROLE

16 rows selected.
```

Enabling Golden Gate Replication

Check if the parameter `enable_goldengate_replication` is set to `True`.

```
SQL> show parameter enable_goldengate_replication
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
enable_goldengate_replication        boolean     FALSE
```

On both the Primary and Standby Database Login as Sysdba and set GoldenGate replication to True.

```
SQL> alter system set enable_goldengate_replication=TRUE;
System altered.
```


Enabling and Viewing Archive Logging and Supplemental Logging

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:

```
Login to Primary DB
set ORACLE_SID=CDBPSFT1
sqlplus / as sysdba

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup mount pfile='C:\PDBATTACH\admin\CDBPSFT1\initCDBPSFT1.ora'
ORACLE instance started.
Total System Global Area 2147483648 bytes
Fixed Size                  3047720 bytes
Variable Size            1207963352 bytes
Database Buffers          922746880 bytes
Redo Buffers               13725696 bytes
Database mounted.
SQL> alter database archivelog;
Database altered.
SQL> alter database open;
Database altered.
```

The following step is only for container database.

```
--#####Lists the pluggable databases in a container####
SQL> show pdbs
CON_ID CON_NAME              OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
 2 PDB$SEED                  READ ONLY     NO
 3 PT855GA                   READ WRITE    NO
```

Viewing Archive Logging

To view the archive logging status:

```
SQL> ARCHIVE LOG LIST;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            C:\oracle12c\product\12.1.0\dbhome_1\RDBMS
Oldest online log sequence     25
Next log sequence to archive   27
Current log sequence           27
```


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;
Database altered.

ALTER SYSTEM SWITCH LOGFILE;
System altered.

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  EMDBO
```

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.

```
################################################################################⇒

#######
-- 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 EMDBO.* with wildcard to capture all tables⇒
```



```
.
-- 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 EMDBO.* statement
-- ###############################################################################⇒

##########
--
-- Edit and modify 'OGGUSER' to Oracle GoldenGate Admin Userid and PW
--
-- Edit and modify 'EMDBO' to PSACCESSID
--
-- Append the output from the PSGGgeneratetrandatadeletes.txt (eg. just the generat⇒

ed
-- DELETE TRANDATA       -- statements) after the ADD TRANDATA EMDBO .* statement
--
dblogin userid C##OGGUSER@PT855GA password OGGUSER
ADD TRANDATA EMDBO.*
-- ###############################################################################⇒

##########
-- This section lists the generated DELETE TRANDATA  statements
-- ###############################################################################⇒

##########
--<Copy in the generated DELETE TRANDATA statements after the preceding ADD TRANDDA⇒

TA statement.>
DELETE TRANDATA emdbo.PS_MENU_LANG_TMP
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM1
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM10
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM2
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM3
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM4
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM5
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM6
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM7
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM8
DELETE TRANDATA emdbo.PS_PSMSFTMPCOM9
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS1
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS10
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS2
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS3
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS4
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS5
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS6
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS7
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS8
DELETE TRANDATA emdbo.PS_PSMSFTMPIDS9
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL1
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL10
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL2
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL3
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL4
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL5
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL6
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL7
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL8
DELETE TRANDATA emdbo.PS_PSMSFTMPTBL9
DELETE TRANDATA emdbo.PS_PSPMCSOAETEMP
DELETE TRANDATA emdbo.PS_PTPP_CPKP_TMP
```

```
DELETE TRANDATA emdbo.PS_PTPP_CPKP_TMP1

-- ##############################################################################⇒

##########
-- This section lists the required static DELETE TRANDATA statements
-- ##############################################################################⇒

##########

DELETE TRANDATA EMDBO.PSLOCK
DELETE TRANDATA EMDBO.PSRECDEFN
DELETE TRANDATA EMDBO.PS_SERVERMONITOR
DELETE TRANDATA EMDBO.PS_SERVERACTVTY
DELETE TRANDATA EMDBO.PS_PRCSSEQUENCE
DELETE TRANDATA EMDBO.PS_MESSAGE_LOGPARM
DELETE TRANDATA EMDBO.PS_MESSAGE_LOG
DELETE TRANDATA EMDBO.PS_AETEMPTBLMGR
DELETE TRANDATA EMDBO.PS_AERUNCONTROLPC
DELETE TRANDATA EMDBO.PS_AERUNCONTROL
DELETE TRANDATA EMDBO.PS_AELOCKMGR
DELETE TRANDATA EMDBO.PSWEBPROFHIST
DELETE TRANDATA EMDBO.PSSERVERSTAT
DELETE TRANDATA EMDBO.PSQRYTRANS
DELETE TRANDATA EMDBO.PSPRCSJOBSTATUS
DELETE TRANDATA EMDBO.PSOPRDEFN
DELETE TRANDATA EMDBO.PSIBSUBSLAVE
DELETE TRANDATA EMDBO.PSIBPUBSLAVE
DELETE TRANDATA EMDBO.PSIBFOLOCK
DELETE TRANDATA EMDBO.PSIBFAILOVER
DELETE TRANDATA EMDBO.PSIBBRKSLAVE
DELETE TRANDATA EMDBO.PSACCESSLOG
DELETE TRANDATA EMDBO.PS_PTFP_ACCESS_LOG
DELETE TRANDATA EMDBO.PS_PTFP_OPTIONS
DELETE TRANDATA EMDBO.PSIBPROFILESYNC
DELETE TRANDATA EMDBO.PSIBLOGHDR
DELETE TRANDATA EMDBO.PSIBLOGERR
DELETE TRANDATA EMDBO.PSIBLOGERRP
DELETE TRANDATA EMDBO.PSIBLOGDATA
DELETE TRANDATA EMDBO.PSIBLOGIBINFO
DELETE TRANDATA EMDBO.PSQASRUN
DELETE TRANDATA EMDBO.PSPRCSRQST
DELETE TRANDATA EMDBO.PSPRCSQUE
DELETE TRANDATA EMDBO.PSPRCSRQSTFILE
DELETE TRANDATA EMDBO.PSPRCSPARMS
DELETE TRANDATA EMDBO.PSPRCSRQSTTEXT
DELETE TRANDATA EMDBO.PSPRCSRQSTTEXT2
DELETE TRANDATA EMDBO.PS_CDM_LIST
DELETE TRANDATA EMDBO.PS_CDM_TRANSFER
DELETE TRANDATA EMDBO.PS_CDM_AUTH
DELETE TRANDATA EMDBO.PS_BAT_TIMINGS_LOG
DELETE TRANDATA EMDBO.PS_BAT_TIMINGS_DTL
DELETE TRANDATA EMDBO.PS_AE_TIMINGS_LG
DELETE TRANDATA EMDBO.PS_AE_TIMINGS_DT
DELETE TRANDATA EMDBO.PS_BAT_TIMINGS_FN
DELETE TRANDATA EMDBO.PSQRYFAVORITES
DELETE TRANDATA EMDBO.PSQRYSTATS
DELETE TRANDATA EMDBO.PSFILE_ATTDET
DELETE TRANDATA EMDBO.PSPTFILE_REF
DELETE TRANDATA EMDBO.PSPTFILE_WART
DELETE TRANDATA EMDBO.PS_PTSF_SCHED_STAT
DELETE TRANDATA EMDBO.PSPGVIEWOPT
DELETE TRANDATA EMDBO.PSPGCHARTOPT
DELETE TRANDATA EMDBO.PSPGCHRTFLRSOPT
DELETE TRANDATA EMDBO.PSPGCHTFLRSLANG
DELETE TRANDATA EMDBO.PSPGDISPOPT
DELETE TRANDATA EMDBO.PSPGGRIDOPT
DELETE TRANDATA EMDBO.PSPGQRYPROMPT
DELETE TRANDATA EMDBO.PSPGQRYPROMPLNG
DELETE TRANDATA EMDBO.PSPGCHARTOPTLNG
DELETE TRANDATA EMDBO.PSPGVIEWOPTLANG
```



```
DELETE TRANDATA EMDBO.PSPGVIEWOPTPERS
DELETE TRANDATA EMDBO.PSPGAXISPERS
DELETE TRANDATA EMDBO.PSPGAXISPERSLNG
DELETE TRANDATA EMDBO.PSPGCHARTOPTPER
DELETE TRANDATA EMDBO.PSPGQRYPRMPTPER
DELETE TRANDATA EMDBO.PSPGGRIDOPTPERS
DELETE TRANDATA EMDBO.PSPGCHTOPTPERLN
DELETE TRANDATA EMDBO.PSPGQRYPRMPTPLN
DELETE TRANDATA EMDBO.PSPGVWOPTPERLN
DELETE TRANDATA EMDBO.PSPGCORE
DELETE TRANDATA EMDBO.PSPGCORELANG
DELETE TRANDATA EMDBO.PSPGMODEL
DELETE TRANDATA EMDBO.PSPGMODELLANG
DELETE TRANDATA EMDBO.PSPGSETTINGS
DELETE TRANDATA EMDBO.PSPGAXIS
DELETE TRANDATA EMDBO.PSPGTHRESHOLDLN
DELETE TRANDATA EMDBO.PSPGNUIOPT
DELETE TRANDATA EMDBO.PSPGNUIDIMOPT
DELETE TRANDATA EMDBO.PSPGSAVEHIST
DELETE TRANDATA EMDBO.PSPTFILE_PRCS
DELETE TRANDATA EMDBO.PSUSEROBJTYPE
DELETE TRANDATA EMDBO.PSUSERSRCHDEFN
DELETE TRANDATA EMDBO.PSUSERPRSNLOPTN
DELETE TRANDATA EMDBO.PSVERSION
DELETE TRANDATA EMDBO.PSUSRTAPAGECUST
DELETE TRANDATA EMDBO.PSBATCHAUTH
DELETE TRANDATA EMDBO.PSBATCHAUTHLONG
DELETE TRANDATA EMDBO.PSPRCSCHLDINFO
DELETE TRANDATA EMDBO.PS_PTNVSLYTQRY
DELETE TRANDATA EMDBO.PSTREESELNUM
DELETE TRANDATA EMDBO.PSTREESELCTL
DELETE TRANDATA EMDBO.PSNVSBATCHRSTRT
DELETE TRANDATA EMDBO.PSNVSDRILLQRY
DELETE TRANDATA EMDBO.PSNVSDRLPROMPTS
DELETE TRANDATA EMDBO.PS_CDM_TEXT
DELETE TRANDATA EMDBO.PS_PRCSRQSTDIST
DELETE TRANDATA EMDBO.PSTREESELECT01
DELETE TRANDATA EMDBO.PSTREESELECT02
DELETE TRANDATA EMDBO.PSTREESELECT03
DELETE TRANDATA EMDBO.PSTREESELECT04
DELETE TRANDATA EMDBO.PSTREESELECT05
DELETE TRANDATA EMDBO.PSTREESELECT06
DELETE TRANDATA EMDBO.PSTREESELECT07
DELETE TRANDATA EMDBO.PSTREESELECT08
DELETE TRANDATA EMDBO.PSTREESELECT09
DELETE TRANDATA EMDBO.PSTREESELECT10
DELETE TRANDATA EMDBO.PSTREESELECT11
DELETE TRANDATA EMDBO.PSTREESELECT12
DELETE TRANDATA EMDBO.PSTREESELECT13
DELETE TRANDATA EMDBO.PSTREESELECT14
DELETE TRANDATA EMDBO.PSTREESELECT15
DELETE TRANDATA EMDBO.PSTREESELECT16
DELETE TRANDATA EMDBO.PSTREESELECT17
DELETE TRANDATA EMDBO.PSTREESELECT18
DELETE TRANDATA EMDBO.PSTREESELECT19
DELETE TRANDATA EMDBO.PSTREESELECT20
DELETE TRANDATA EMDBO.PSTREESELECT21
DELETE TRANDATA EMDBO.PSTREESELECT22
DELETE TRANDATA EMDBO.PSTREESELECT23
DELETE TRANDATA EMDBO.PSTREESELECT24
DELETE TRANDATA EMDBO.PSTREESELECT25
DELETE TRANDATA EMDBO.PSTREESELECT26
DELETE TRANDATA EMDBO.PSTREESELECT27
DELETE TRANDATA EMDBO.PSTREESELECT28
DELETE TRANDATA EMDBO.PSTREESELECT29
DELETE TRANDATA EMDBO.PSTREESELECT30
DELETE TRANDATA EMDBO.PSFP_FEED
DELETE TRANDATA EMDBO.PSFP_FEED_LANG
DELETE TRANDATA EMDBO.PSFP_SETTINGS
DELETE TRANDATA EMDBO.PSFP_ADMN_PREF
DELETE TRANDATA EMDBO.PSFP_USER_PREF
```

```
DELETE TRANDATA EMDBO.PSFP_SECURITY
DELETE TRANDATA EMDBO.PSFP_PUB_SITES
DELETE TRANDATA EMDBO.PSFP_PARMS
DELETE TRANDATA EMDBO.PSFP_PARMS_LANG
DELETE TRANDATA EMDBO.PSFP_PVALS
DELETE TRANDATA EMDBO.PSFP_PVALS_LANG
DELETE TRANDATA EMDBO.PSFP_ATTRS
DELETE TRANDATA EMDBO.PSFP_ATTRS_LANG
DELETE TRANDATA EMDBO.PSFP_FEED_DEL
DELETE TRANDATA EMDBO.PSAPMSGPUBHDR
DELETE TRANDATA EMDBO.PSAPMSGPUBDATA
DELETE TRANDATA EMDBO.PSAPMSGIBATTR
DELETE TRANDATA EMDBO.PSAPMSGSEGATTR
DELETE TRANDATA EMDBO.PSAPMSGPUBSYNC
```

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 C##OGGUSER password OGGUSER
REGISTER EXTRACT primecap DATABASE  CONTAINER (PT855GA)
ADD EXTRACT primecap INTEGRATED TRANLOG, BEGIN NOW
add exttrail ./dirdat/pt, extract primecap
ADD EXTRACT primepmp EXTTRAILSOURCE ./dirdat/pt BEGIN NOW
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 'EMDBO' 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 E⇒

MDBO.*
-- statement
--
-- ###############################################################################⇒

##########
EXTRACT primecap

SETENV (NLS_LANG = "AMERICAN_AMERICA.AL32UTF8")
SETENV (ORACLE_HOME = "C:\oracle12c\product\12.1.0\dbhome_1")
SETENV (ORACLE_SID = CDBPSFT1)

USERID C##OGGUSER PASSWORD OGGUSER

DISCARDFILE ./dirrpt/primecap.dsc, purge
```



```
EXTTRAIL ./dirdat/pt
LOGALLSUPCOLS
UPDATERECORDFORMAT COMPACT
SOURCECATALOG PT855GA
WILDCARDRESOLVE DYNAMIC
TRANLOGOPTIONS USE_ROOT_CONTAINER_TIMEZONE
-- ##############################################################################⇒

##########
-- This section lists the generated TABLEEXCLUDE statements
-- ##############################################################################⇒

##########

-- <Copy in the generated TABLEEXCLUDE statements after the preceding WILDCARDRESOL⇒

VE DYNAMIC
-- statement.>

TABLEEXCLUDE emdbo.PS_MENU_LANG_TMP;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM1;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM10;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM2;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM3;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM4;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM5;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM6;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM7;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM8;
TABLEEXCLUDE emdbo.PS_PSMSFTMPCOM9;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS1;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS10;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS2;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS3;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS4;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS5;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS6;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS7;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS8;
TABLEEXCLUDE emdbo.PS_PSMSFTMPIDS9;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL1;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL10;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL2;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL3;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL4;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL5;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL6;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL7;
ABLEEXCLUDE emdbo.PS_PSMSFTMPTBL8;
TABLEEXCLUDE emdbo.PS_PSMSFTMPTBL9;
TABLEEXCLUDE emdbo.PS_PSPMCSOAETEMP;
TABLEEXCLUDE emdbo.PS_PTPP_CPKP_TMP;
TABLEEXCLUDE emdbo.PS_PTPP_CPKP_TMP1;

-- ##############################################################################⇒

##########
-- This section lists the required static TABLEEXCLUDE statements
-- ##############################################################################⇒

##########


TABLEEXCLUDE EMDBO.PSLOCK;
TABLEEXCLUDE EMDBO.PSRECDEFN;
TABLEEXCLUDE EMDBO.PS_SERVERMONITOR;
TABLEEXCLUDE EMDBO.PS_SERVERACTVTY;
TABLEEXCLUDE EMDBO.PS_PRCSSEQUENCE;
```



```
TABLEEXCLUDE EMDBO.PS_MESSAGE_LOGPARM;
TABLEEXCLUDE EMDBO.PS_MESSAGE_LOG;
TABLEEXCLUDE EMDBO.PS_AETEMPTBLMGR;
TABLEEXCLUDE EMDBO.PS_AERUNCONTROLPC;
TABLEEXCLUDE EMDBO.PS_AERUNCONTROL;
TABLEEXCLUDE EMDBO.PS_AELOCKMGR;
TABLEEXCLUDE EMDBO.PSWEBPROFHIST;
TABLEEXCLUDE EMDBO.PSSERVERSTAT;
TABLEEXCLUDE EMDBO.PSQRYTRANS;
TABLEEXCLUDE EMDBO.PSPRCSJOBSTATUS;
TABLEEXCLUDE EMDBO.PSOPRDEFN;
TABLEEXCLUDE EMDBO.PSIBSUBSLAVE;
TABLEEXCLUDE EMDBO.PSIBPUBSLAVE;
TABLEEXCLUDE EMDBO.PSIBFOLOCK;
TABLEEXCLUDE EMDBO.PSIBFAILOVER;
TABLEEXCLUDE EMDBO.PSIBBRKSLAVE;
TABLEEXCLUDE EMDBO.PSACCESSLOG;
TABLEEXCLUDE EMDBO.PS_PTFP_ACCESS_LOG;
TABLEEXCLUDE EMDBO.PS_PTFP_OPTIONS;
TABLEEXCLUDE EMDBO.PSIBPROFILESYNC;
TABLEEXCLUDE EMDBO.PSIBLOGHDR;
TABLEEXCLUDE EMDBO.PSIBLOGERR;
TABLEEXCLUDE EMDBO.PSIBLOGERRP;
TABLEEXCLUDE EMDBO.PSIBLOGDATA;
TABLEEXCLUDE EMDBO.PSIBLOGIBINFO;
TABLEEXCLUDE EMDBO.PSQASRUN;
TABLEEXCLUDE EMDBO.PSPRCSRQST;
TABLEEXCLUDE EMDBO.PSPRCSQUE;
TABLEEXCLUDE EMDBO.PSPRCSRQSTFILE;
TABLEEXCLUDE EMDBO.PSPRCSPARMS;
TABLEEXCLUDE EMDBO.PSPRCSRQSTTEXT;
TABLEEXCLUDE EMDBO.PSPRCSRQSTTEXT2;
TABLEEXCLUDE EMDBO.PS_CDM_LIST;
TABLEEXCLUDE EMDBO.PS_CDM_TRANSFER;
TABLEEXCLUDE EMDBO.PS_CDM_AUTH;
TABLEEXCLUDE EMDBO.PS_BAT_TIMINGS_LOG;
TABLEEXCLUDE EMDBO.PS_BAT_TIMINGS_DTL;
TABLEEXCLUDE EMDBO.PS_AE_TIMINGS_LG;
TABLEEXCLUDE EMDBO.PS_AE_TIMINGS_DT;
TABLEEXCLUDE EMDBO.PS_BAT_TIMINGS_FN;
TABLEEXCLUDE EMDBO.PSQRYFAVORITES;
TABLEEXCLUDE EMDBO.PSQRYSTATS;
TABLEEXCLUDE EMDBO.PSFILE_ATTDET;
TABLEEXCLUDE EMDBO.PSPTFILE_REF;
TABLEEXCLUDE EMDBO.PSPTFILE_WART;
TABLEEXCLUDE EMDBO.PS_PTSF_SCHED_STAT;
TABLEEXCLUDE EMDBO.PSPGVIEWOPT;
TABLEEXCLUDE EMDBO.PSPGCHARTOPT;
TABLEEXCLUDE EMDBO.PSPGCHRTFLRSOPT;
TABLEEXCLUDE EMDBO.PSPGCHTFLRSLANG;
TABLEEXCLUDE EMDBO.PSPGDISPOPT;
TABLEEXCLUDE EMDBO.PSPGGRIDOPT;
TABLEEXCLUDE EMDBO.PSPGQRYPROMPT;
TABLEEXCLUDE EMDBO.PSPGQRYPROMPLNG;
TABLEEXCLUDE EMDBO.PSPGCHARTOPTLNG;
TABLEEXCLUDE EMDBO.PSPGVIEWOPTLANG;
TABLEEXCLUDE EMDBO.PSPGVIEWOPTPERS;
TABLEEXCLUDE EMDBO.PSPGAXISPERS;
TABLEEXCLUDE EMDBO.PSPGAXISPERSLNG;
TABLEEXCLUDE EMDBO.PSPGCHARTOPTPER;
TABLEEXCLUDE EMDBO.PSPGQRYPRMPTPER;
TABLEEXCLUDE EMDBO.PSPGGRIDOPTPERS;
TABLEEXCLUDE EMDBO.PSPGCHTOPTPERLN;
TABLEEXCLUDE EMDBO.PSPGQRYPRMPTPLN;
TABLEEXCLUDE EMDBO.PSPGVWOPTPERLN;
TABLEEXCLUDE EMDBO.PSPGCORE;
TABLEEXCLUDE EMDBO.PSPGCORELANG;
TABLEEXCLUDE EMDBO.PSPGMODEL;
TABLEEXCLUDE EMDBO.PSPGMODELLANG;
TABLEEXCLUDE EMDBO.PSPGSETTINGS;
TABLEEXCLUDE EMDBO.PSPGAXIS;
```



```
TABLEEXCLUDE EMDBO.PSPGTHRESHOLDLN;
TABLEEXCLUDE EMDBO.PSPGNUIOPT;
TABLEEXCLUDE EMDBO.PSPGNUIDIMOPT;
TABLEEXCLUDE EMDBO.PSPGSAVEHIST;
TABLEEXCLUDE EMDBO.PSPTFILE_PRCS;
TABLEEXCLUDE EMDBO.PSUSEROBJTYPE;
TABLEEXCLUDE EMDBO.PSUSERSRCHDEFN;
TABLEEXCLUDE EMDBO.PSUSERPRSNLOPTN;
TABLEEXCLUDE EMDBO.PSVERSION;
TABLEEXCLUDE EMDBO.PSUSRTAPAGECUST;
TABLEEXCLUDE EMDBO.PSBATCHAUTH;
TABLEEXCLUDE EMDBO.PSBATCHAUTHLONG;
TABLEEXCLUDE EMDBO.PSPRCSCHLDINFO;
TABLEEXCLUDE EMDBO.PS_PTNVSLYTQRY;
TABLEEXCLUDE EMDBO.PSTREESELNUM;
TABLEEXCLUDE EMDBO.PSTREESELCTL;
TABLEEXCLUDE EMDBO.PSNVSBATCHRSTRT;
TABLEEXCLUDE EMDBO.PSNVSDRILLQRY;
TABLEEXCLUDE EMDBO.PSNVSDRLPROMPTS;
TABLEEXCLUDE EMDBO.PS_CDM_TEXT;
TABLEEXCLUDE EMDBO.PS_PRCSRQSTDIST;
TABLEEXCLUDE EMDBO.PSTREESELECT01;
TABLEEXCLUDE EMDBO.PSTREESELECT02;
TABLEEXCLUDE EMDBO.PSTREESELECT03;
TABLEEXCLUDE EMDBO.PSTREESELECT04;
TABLEEXCLUDE EMDBO.PSTREESELECT05;
TABLEEXCLUDE EMDBO.PSTREESELECT06;
TABLEEXCLUDE EMDBO.PSTREESELECT07;
TABLEEXCLUDE EMDBO.PSTREESELECT08;
TABLEEXCLUDE EMDBO.PSTREESELECT09;
TABLEEXCLUDE EMDBO.PSTREESELECT10;
TABLEEXCLUDE EMDBO.PSTREESELECT11;
TABLEEXCLUDE EMDBO.PSTREESELECT12;
TABLEEXCLUDE EMDBO.PSTREESELECT13;
TABLEEXCLUDE EMDBO.PSTREESELECT14;
TABLEEXCLUDE EMDBO.PSTREESELECT15;
TABLEEXCLUDE EMDBO.PSTREESELECT16;
TABLEEXCLUDE EMDBO.PSTREESELECT17;
TABLEEXCLUDE EMDBO.PSTREESELECT18;
TABLEEXCLUDE EMDBO.PSTREESELECT19;
TABLEEXCLUDE EMDBO.PSTREESELECT20;
TABLEEXCLUDE EMDBO.PSTREESELECT21;
TABLEEXCLUDE EMDBO.PSTREESELECT22;
TABLEEXCLUDE EMDBO.PSTREESELECT23;
TABLEEXCLUDE EMDBO.PSTREESELECT24;
TABLEEXCLUDE EMDBO.PSTREESELECT25;
TABLEEXCLUDE EMDBO.PSTREESELECT26;
TABLEEXCLUDE EMDBO.PSTREESELECT27;
TABLEEXCLUDE EMDBO.PSTREESELECT28;
TABLEEXCLUDE EMDBO.PSTREESELECT29;
TABLEEXCLUDE EMDBO.PSTREESELECT30;
TABLEEXCLUDE EMDBO.PSFP_FEED;
TABLEEXCLUDE EMDBO.PSFP_FEED_LANG;
TABLEEXCLUDE EMDBO.PSFP_SETTINGS;
TABLEEXCLUDE EMDBO.PSFP_ADMN_PREF;
TABLEEXCLUDE EMDBO.PSFP_USER_PREF;
TABLEEXCLUDE EMDBO.PSFP_SECURITY;
TABLEEXCLUDE EMDBO.PSFP_PUB_SITES;
TABLEEXCLUDE EMDBO.PSFP_PARMS;
TABLEEXCLUDE EMDBO.PSFP_PARMS_LANG;
TABLEEXCLUDE EMDBO.PSFP_PVALS;
TABLEEXCLUDE EMDBO.PSFP_PVALS_LANG;
TABLEEXCLUDE EMDBO.PSFP_ATTRS;
TABLEEXCLUDE EMDBO.PSFP_ATTRS_LANG;
TABLEEXCLUDE EMDBO.PSFP_FEED_DEL;
TABLEEXCLUDE EMDBO.PSAPMSGPUBHDR;
TABLEEXCLUDE EMDBO.PSAPMSGPUBDATA;
TABLEEXCLUDE EMDBO.PSAPMSGIBATTR;
TABLEEXCLUDE EMDBO.PSAPMSGSEGATTR;
TABLEEXCLUDE EMDBO.PSAPMSGPUBSYNC;
```

```
TABLE EMDBO.*;
```

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 'EMDBO' to PSACCESSID
-- ################################################################################⇒

##########
EXTRACT primepmp
SETENV (ORACLE_SID='CDBPSFT1')

PASSTHRU
PASSTHRUMESSAGES
USERID C##OGGUSER PASSWORD OGGUSER
-- Remote Host and Trail Information
RMTHOST <hostname> MGRPORT 7810
RMTTRAIL ./dirdat/pr

-- Table Mapping Parameters
WILDCARDRESOLVE DYNAMIC
SOURCECATALOG PT855GA
TABLE EMDBO.*;
```

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 C##OGGUSER@STD855GA password OGGUSER
ADD REPLICAT trgtrep INTEGRATED EXTTRAIL ./dirdat/pr
```

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 'EMDBO' to PSACCESSID
-- ################################################################################⇒

###########
REPLICAT trgtrep

SETENV (NLS_LANG = "AMERICAN_AMERICA.AL32UTF8")
SETENV (ORACLE_HOME = "C:\oracle12c\product\12.1.0\dbhome_1")
SETENV (ORACLE_SID = "CDBPSFT1")
DBOPTIONS INTEGRATEDPARAMS(parallelism 6)
USERID C##OGGUSER@STD855GA, PASSWORD OGGUSER

ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/trgtrep.dsc, APPEND
DISCARDROLLOVER ON SUNDAY
ALLOWNOOPUPDATES

MAP PT855GA.EMDBO.*, TARGET STD855GA.EMDBO.*;
```

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:

```
EMDBO.PSLOCK
EMDBO.PS_SERVERMONITOR
EMDBO.PS_SERVERACTVTY
```



```
EMDBO.PS_PRCSSEQUENCE
EMDBO.PS_MESSAGE_LOGPARM
EMDBO.PS_MESSAGE_LOG
EMDBO.PS_AETEMPTBLMGR
EMDBO.PS_AERUNCONTROLPC
EMDBO.PS_AERUNCONTROL
EMDBO.PS_AELOCKMGR
EMDBO.PSWEBPROFHIST
EMDBO.PSSERVERSTAT
EMDBO.PSQRYTRANS
EMDBO.PSPRCSJOBSTATUS
EMDBO.PSOPRDEFN
EMDBO.PSIBSUBSLAVE
EMDBO.PSIBPUBSLAVE
EMDBO.PSIBFOLOCK
EMDBO.PSIBFAILOVER
EMDBO.PSIBBRKSLAVE
EMDBO.PSACCESSLOG
EMDBO.PS_PTFP_ACCESS_LOG
EMDBO.PS_PTFP_OPTIONS
EMDBO.PSIBPROFILESYNC
EMDBO.PSIBLOGHDR
EMDBO.PSIBLOGERR
EMDBO.PSIBLOGERRP
EMDBO.PSIBLOGDATA
EMDBO.PSIBLOGIBINFO
EMDBO.PSQASRUN
EMDBO.PSPRCSRQST
EMDBO.PSPRCSQUE
EMDBO.PSPRCSRQSTFILE
EMDBO.PSPRCSPARMS
EMDBO.PSPRCSRQSTTEXT
EMDBO.PSPRCSRQSTTEXT2
EMDBO.PS_CDM_LIST
EMDBO.PS_CDM_TRANSFER
EMDBO.PS_CDM_AUTH
EMDBO.PS_BAT_TIMINGS_LOG
EMDBO.PS_BAT_TIMINGS_DTL
EMDBO.PS_AE_TIMINGS_LG
EMDBO.PS_AE_TIMINGS_DT
EMDBO.PS_BAT_TIMINGS_FN
EMDBO.PSQRYFAVORITES
EMDBO.PSQRYSTATS
EMDBO.PSFILE_ATTDET
EMDBO.PSPTFILE_REF
EMDBO.PSPTFILE_WART
EMDBO.PS_PTSF_SCHED_STAT
EMDBO.PSPGVIEWOPT
EMDBO.PSPGCHARTOPT
EMDBO.PSPGCHRTFLRSOPT
EMDBO.PSPGCHTFLRSLANG
EMDBO.PSPGDISPOPT
EMDBO.PSPGGRIDOPT
EMDBO.PSPGQRYPROMPT
EMDBO.PSPGCHARTOPTLNG
EMDBO.PSPGVIEWOPTLANG
EMDBO.PSPTFILE_PRCS
EMDBO.PSUSEROBJTYPE
EMDBO.PSUSERSRCHDEFN
EMDBO.PSUSERPRSNLOPTN
EMDBO.PSVERSION
EMDBO.PSUSRTAPAGECUST
EMDBO.PSBATCHAUTH
EMDBO.PSBATCHAUTHLONG
EMDBO.PSPRCSCHLDINFO
EMDBO.PS_PTNVSLYTQRY
EMDBO.PSTREESELNUM
EMDBO.PSTREESELCTL
EMDBO.PSNVSBATCHRSTRT
EMDBO.PSNVSDRILLQRY
EMDBO.PSNVSDRLPROMPTS
```



```
EMDBO.PS_CDM_TEXT
EMDBO.PS_PRCSRQSTDIST
EMDBO.PSTREESELECT01
EMDBO.PSTREESELECT02
EMDBO.PSTREESELECT03
EMDBO.PSTREESELECT04
EMDBO.PSTREESELECT05
EMDBO.PSTREESELECT06
EMDBO.PSTREESELECT07
EMDBO.PSTREESELECT08
EMDBO.PSTREESELECT09
EMDBO.PSTREESELECT10
EMDBO.PSTREESELECT11
EMDBO.PSTREESELECT12
EMDBO.PSTREESELECT13
EMDBO.PSTREESELECT14
EMDBO.PSTREESELECT15
EMDBO.PSTREESELECT16
EMDBO.PSTREESELECT17
EMDBO.PSTREESELECT18
EMDBO.PSTREESELECT19
EMDBO.PSTREESELECT20
EMDBO.PSTREESELECT21
EMDBO.PSTREESELECT22
EMDBO.PSTREESELECT23
EMDBO.PSTREESELECT24
EMDBO.PSTREESELECT25
EMDBO.PSTREESELECT26
EMDBO.PSTREESELECT27
EMDBO.PSTREESELECT28
EMDBO.PSTREESELECT29
EMDBO.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.
- <EMDBO> 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.

   Please make sure that the access ID has "create public database link" privileges or else the above create database link will fail with *ORA-01031: insufficient privileges* . Grant the `CREATE PUBLIC DATABASE LINK` privilege to the access id on the standby database:

```
$ sqlplus / as sysdba
SQL> Alter session set container=PT855GA;
Session Altered.
SQL> Grant create public database link to EMDBO;
```

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

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

Implementing Oracle Golden Gate on Oracle 11g Database

The topic lists scripts to implement for Oracle Golden Gate 11g for PeopleSoft. For detailed information on any of the steps and the scripts required for Oracle 12c database see, Implementing Oracle GoldenGate for PeopleSoft Off-Load Reporting.

You can also find more information on installing Oracle GoldenGate at Oracle® GoldenGate Installing and Configuring Oracle GoldenGate for Oracle Database, " Installing Oracle Golden Gate".

Creating and Granting Privileges to the Oracle GoldenGate User for Oracle 11g

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;
```

For more information on creating a Oracle Golden Gate user and script for Oracle database 12c, see Configuring PeopleSoft Databases for Oracle GoldenGate.

Example to List the Privileges Granted to the GoldenGate User in Oracle 11g

To list the privileges granted to the Oracle GoldenGate 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;
```

The above script generates following result:

```
  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.
```

Enabling Archive Logging for Oracle 11g

For the following information, see Enabling and Viewing Archive Logging and Supplemental Logging:

- Example of Archive Logging on Oracle 12c.
- Examples for viewing archive logging status.
- Example to enable supplemental logging.

Example to Enable Archive Logging in 11g

To enable archive logging for Oracle GoldenGate on Oracle 11g, use these commands in SQL Plus:

```
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.
```

Creating Oracle GoldenGate Parameter Files for the Primary Database on Oracle 11g

This section describes the parameter files that you need to create and modify manually for the primary database on Oracle 11g. See Creating Oracle GoldenGate Parameter Files for the Primary Databasefor examples on Oracle 12c.

- 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

Example to create a file named *priaddtrndata.oby* on Oracle 11g:

```
--
--  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

Example to create a file named *configure_primary.oby* on Oracle 11g:

```
-- ############################################################################
--
-- 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

Example to create a file named *primecap.prm* on Oracle 11g:

```
-- ############################################################################⇒

##########
-- 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

Example to create a file named *primepmp.prm* on Oracle 11g:

```
-- ##############################################################################⇒

##########
-- 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 <hostname> MGRPORT 7810
RMTTRAIL ./dirdat/pr

-- Table Mapping Parameters
WILDCARDRESOLVE DYNAMIC

TABLE SYSADM.*;
```

Creating Oracle GoldenGate Parameter Files for the Standby Database on Oracle 11g

This section describes the parameter files that you need to create and modify manually for the standby database on Oracle 11g. See Creating Oracle GoldenGate Parameter Files for the Standby Database for examples on Oracle 12c.

- mgr.prm
- configure_standby.oby
- trgtrep.prm

Creating mgr.prm

See Creating Oracle GoldenGate Parameter Files for the Standby Database, mgr.prm section.

Creating configure_standby.oby

Example to create a file named *configure_standby.oby*, on Oracle 11g:

```
-- ##############################################################################⇒

##########
-- 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

Example to create a file named *trgtrep.prm*, on Oracle 11g:

```
-- ##############################################################################⇒

##########
-- 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.*;
```

Setting Up the PeopleSoft Installation with Oracle RAC

This section contains an overview and discusses the process to set up PeopleSoft installation with Oracle RAC.

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 *PeopleSoft 9.2 Application Installation for Oracle*: *"Creating a Database Manually on UNIX"*.

See *PeopleSoft 9.2 Application 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:
  - CREATEDB18.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 CREATEDB18.SQL Script

You must edit the CREATEDB18.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 CREATEDB18.SQL

The following is an unmodified CREATEDB18.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: CREATEDB18.SQL Modified for RAC

The following is an example of a CREATEDB18.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 = myserver.example.com) (port
= 1521))
      (ADDRESS = (PROTOCOL = TCP) (host = 192.0.2.10) (port = 1521))
   (CONNECT_DATA =
      (service_name = rac)
   )
  )


rac1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (host = myserver.example.com) (port =
1521))
    (ADDRESS = (PROTOCOL = TCP) (host = myserver.example.com) (port = 1521))
    (CONNECT_DATA =
      (service_name = rac)
    )
  )

rac2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (host = 192.0.2.10) (port = 1521))
    (ADDRESS = (PROTOCOL = TCP) (host = myserver.example.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/192.0.2.10-64bit)
      (PROGRAM = extproc)
    )
  )

SID_LIST_LISTENER_PT-SUN29 =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /products/oracle/192.0.2.10-64bit)
      (PROGRAM = extproc)
    )
  )

LISTENER_PT-SUN30 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = 192.0.2.10)(PORT = 1521))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = 192.0.2.10)(PORT = 1521))
      )
    )
  )

LISTENER_PT-SUN29 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = myserver.example.com)(PORT
= 1521))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = 192.0.2.10)(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.

Script	*Description*
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.
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, 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 Oracle Database File 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

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 *PeopleSoft 9.2 Application 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” (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 Autonomous Database

Oracle Autonomous Database on Oracle Cloud Infrastructure delivers automated patching, upgrades, and tuning, without human intervention, and is supported for Oracle 19c.

PeopleSoft PeopleTools supports the use of Oracle Autonomous Database-Dedicated (ADB-D) and Autonomous Database-Serverless (ADB-S).

For reference information and migration instructions see https://docs.oracle.com/cd/E52319_01/infoportal/autonomous_database.html.

With ADB-S you can take advantage of Transparent Application Continuity to hide planned database maintenance, unplanned outages, and local rebalances of the database, ensuring that your applications are continuously available.

See https://docs.oracle.com/en/cloud/paas/autonomous-database/serverless/adbsb/application-continuity-about.html#GUID-85E2A6BC-5E44-4B03-A558-A1E56B581C55.

Chapter 10

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 or Process Scheduler Servers for Remote Data Access for Oracle

This section discusses initial configurations to use Remote Data Access with Oracle.

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, Certifications tab on My Oracle Support, and your *PeopleSoft 9.2 Application 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 Connections**), 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 the Oracle home directory is C:\Apps\DB\Oracle, add the following lines under the ";JavaVM Shared Library=" section:

```
; RDBA Oracle JDBC driver
Add to CLASSPATH=C:\Apps\DB\Oracle\jdbc\lib\ojdbc14.jar
Add to CLASSPATH=C:\Apps\DB\Oracle\jdbc\lib\orai18n.jar
```

Configuring Application Servers or Process Scheduler Servers for Remote Data Access with DB2

This section discusses configuration steps on Db2 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, Certifications tab on My Oracle Support, and the product documentation for *PeopleSoft 9.2 Application Installation*for version and installation information.

Configuring Remote Data Access for Db2 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 home directory, and specify it in the appropriate configuration file. For example, if DB2 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
```


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.

Chapter 11

Mass Change

Understanding Mass Change

Important! Mass Change is a desupported product. 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” (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 on the Mass Change Types page.

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.

   a. To open an existing type, select **PeopleTools** > **Mass Changes** > **Mass Change Types**.

   b. Search by Mass Change Type ID or PeopleSoft owner.

   c. When you select a value, the Description page appears.

      You use the Description page in the Mass Change Types 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* check box is selected, the Mass Change type will be available to other users. Clear the *Public Use* check box to prevent other users from accessing 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 *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.

- 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 Records area 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.

SQL Action	*Definition*	*Use*
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 (plus sign) and specify a Sequence and SQL Action.

7. Add more *SQL Statements,* as desired.

   For each additional SQL Statement, click the Add button. Then, repeat steps 2 through 5.

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 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. Select View All to review all your join fields.

10. Select 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 on the Records and Fields page, 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 *Free Form 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 Templates* 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.

    a. To open an existing template, select **PeopleTools** > **Mass Changes** > **Mass Change Templates**.

    b. Search by Mass Change Type ID, Mass Change Template ID, or PeopleSoft Owner.

    c. When you select a value, the Description page appears.

    Use the Description page in the Mass Change Templates component to assign a Mass Change type and an owner to the template.

2. Fill out the fields on the Description page.

    - For a new template, select a *Mass Change Type ID.*

      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.

    - Select a *PS Owner*.

      The *PS Owner* field identifies the PeopleSoft system from which the template originates. Each template *must* have an owner.

    - Enter a *Description.*.

      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 in the outermost Criteria and Fields scroll area 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. Clicking the lookup button for *Record* brings up a list of valid records for each statement. When a *Record* is selected, clicking the lookup button for *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 **PeopleTools** > **Mass Changes** > **Mass Change Prompt Records**.

2. Select *Add a New Value*, and enter a name, or search for an existing prompt table by Record (Table) Name, Field Name, or Edit 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 Params page.

To specify the datetime formatting for an RDBMS:

1. Select **PeopleTools** > **Mass Changes** > **Mass Change Datetime Parms**.
2. Select a *Database Platform* and click *OK*.

   The Mass Change Datetime Params page appears.
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 **PeopleTools** > **Mass Changes** > **Mass Change Definitions.**

   Search by Mass Change Definition, Mass Change Template ID, Mass Change Type ID, or Primary Permission List. When you select a value the Mass Change Definitions page appears. Alternatively, enter a value for a new definition.
2. For a new Mass Change Definition, select a *Mass Change Template*.

   If you selected an existing Mass Change Definition, this field is not editable.
3. Use the lookup icon to select an *Archive ID*, specify an *Archive Date,* if desired, and enter a *Description.*
4. Select the *Criteria and Defaults* page to specify the values of the criteria and default fields defined by the template.

   The *Criteria* rows consist of a SQL operator (Mass Change Definition drop-down list on the left) and a field 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. 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 *Defaults* box displays fields that will be updated and allows you to enter a Mass Change Field Value for each field. If you have created prompt tables for the fields on this page, you can select from a list of valid values for each field.

5. Select the *Tools Specific Fields* page, and enter any information required for the definitions for your particular application.

6. Select the *Generate SQL* page.

    - Use this page to create the SQL statement(s) based on the information you specified for 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.

- 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, use the Clear Sw button to delete it, and rewrite the Mass Change definition, template, or type, as needed.

- Use the Count button in the lower left corner of the page to display the total number of rows affected by each statement.

- If you select the Execute SQL Upon Saving check box, 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.

7. 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.

8. Select *Execute SQL Upon Saving,* if desired.

9. Click the *Save* button to save the definition.

   If you selected *Execute SQL Upon Saving,* the definition begins executing.

10. Select Execution History to view a list of SQL Statements, Criteria Fields, and Operations.

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 **PeopleTools** > **Mass Changes** > **Mass Change Group.**
2. Search by Mass Change Group ID or PeopleSoft owner for an existing group, or add a new Mass Change Group ID.

   The *Mass Change Groups* page displays.
3. Select a *PS Owner*.
4. In the Mass Change Definition List area, select a *Mass Change Definition* to add to the group.
5. Assign *Execution Sequence* numbers to each definition.

   These numbers determine the order of execution.
6. Click the add button for additional Mass Change Definitions.
7. 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. (Select *OK To Execute MC Online* on the Permission List - Mass Change page.)

See “Setting Mass Change Permissions” (Security Administration)

To execute a Mass Change online:

1. In the *Mass Change Definition, Generate SQL* page, select the *Execute SQL Upon Saving* check box.

   See Building Mass Change Definitions
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 first specify the definition or group you want to execute. You do this using the Run Mass Change page. When you run the definition or group, you see the Process Scheduler Request dialog.

To execute a definition or group in the background:

1. Select **PeopleTools** > **Mass Changes** > **Process Mass Changes.**
2. Select an existing *Run Control ID* or enter a new one.

   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 fields in the *Execution Parameters* area.
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	***Process***
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.

