BEA WebLogic Portal®

Database Administration Guide

Version 9.2
Revised: April 2007

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

Scripts and Properties Files

Sample WLST Scripts

CHAPTER 1

Introduction

This guide describes how to configure and manage databases in BEA WebLogic Portal®. The guide also contains a data dictionary and information about Data Definition Language (DDL) modules, which are the SQL scripts that create WebLogic Portal database objects.

For detailed information about 9.2 database upgrade tasks, see the *Upgrade Guide*.

This chapter includes the following sections:

- Introduction to Using Databases with WebLogic Portal
- How This Guide Fits into the Portal Life Cycle

Introduction to Using Databases with WebLogic Portal

The WebLogic family of products uses Java Database Connectivity (JDBC) to interact with the databases that store program and application data. JDBC is a standard Java API that consists of classes and interfaces written in the Java programming language. Application, tool, and database developers use JDBC to write database applications and execute SQL statements.

WebLogic JDBC enables programmers to interact with different database management systems (DBMSs) including Oracle, Microsoft SQL Server, Sybase, and IBM DB2. WebLogic Portal executes DDL commands for each DBMS it interacts with, so you must ensure that your configuration is supported by reviewing the following database support information:

- *Supported Configurations*
- *BEA Products Installation and Configuration Release Notes*

You can perform basic JDBC setup using the WebLogic Configuration Wizard. The Configuration Wizard allows you to set up JDBC connection pools and data sources for WebLogic Portal. The wizard is described in Creating WebLogic Domains Using the Configuration Wizard.

You can use the WebLogic Server Administration Console to enable, configure, and monitor JDBC connection pools, data sources, and MultiPools. For detailed information on these tasks, see *Configuring JDBC Data Sources* in the WebLogic Server documentation.

The remaining chapters of this book describe database management tasks and reference information that is specific to WebLogic Portal.

How This Guide Fits into the Portal Life Cycle

In general, the tasks that you perform to plan your WebLogic Portal database structure occur as part of the architecture phase of the portal life cycle; these tasks are described in the *WebLogic Portal Overview*. The tasks that you perform to manage your WebLogic Portal database structure can occur during any other phase of the portal life cycle; for this reason, this book is not organized according to the portal life cycle structure.

CHAPTER 2

Database Configuration Overview

This chapter describes how to set up and start using a database management system (DBMS) with WebLogic Portal.

The sample data provided with WebLogic Portal uses the PointBase DBMS. PointBase is a demonstration database ships with WebLogic Server to allow you to use sample user accounts and data. For more information about PointBase, see "Using PointBase" on page 4-1.

Note: PointBase is supported only for the design, development, and verification of applications; it is not supported for enterprise-quality deployment. The evaluation license of PointBase has a database size limit of 30 MB.

To maximize the performance of your database configuration, see "Performance Considerations" on page 3-5.

This chapter contains the following sections:

Overview of Enterprise-Quality Database Configuration for WebLogic Portal

There are several steps to configuring a database for use with WebLogic Portal. You can perform these tasks manually using provided sample scripts, or you can use the WebLogic Configuration Wizard for certain steps.

To configure databases for use with WebLogic Portal, follow these steps:

1. Review related documentation to ensure that your configuration is supported:
   - *Supported Configurations*
   - *BEA Products Installation and Configuration Release Notes*
2. Create your database or database instance. If you want to use behavior event tracking in your enterprise-quality environment, consider using a separate database for behavior event tracking. For more information, see the chapter in this guide pertaining to your database vendor.
3. Prepare the database for use with WebLogic Portal. BEA provides sample initialization scripts that you must modify and run on the vendor database before using the database with WebLogic Portal.

   For more information about these initialization scripts and how to use them, see the chapter in this guide that pertains to your database vendor. For information on the sample scripts and what they do, see "Property Files and Database Scripts" on page B-1.
4. Create appropriate database objects and set JDBC driver settings, either using use the Configuration Wizard or by modifying and running the sample scripts.

   To perform this step using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

   To perform this step manually, see "Manually Creating Database Objects and JDBC Settings", next.
5. Optionally, create the GroupSpace repository database objects and set JDBC driver settings, either using use the Configuration Wizard or by modifying and running the sample scripts.

   To perform this step using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

   To perform this step manually, see "Manually Creating Database Objects and JDBC Settings", next.

6. Optionally, manually create any additional content management repositories.

Manually Creating Database Objects and JDBC Settings

In some cases, you might want to manually create database objects and JDBC settings rather than using the WebLogic Configuration Wizard.

The following are reasons you might want to perform database configuration manually:

- To create additional content management repositories.
- To set up your enterprise-quality (non-PointBase) database.
- If your target database was not configured using the WebLogic Configuration Wizard.
- If, after running the Configuration Wizard, you decide that you want your domain to point to a different database.
- To create a subset of WebLogic Portal database objects; for example, you might want to create only behavior tracking database objects for a particular database.
- To refresh your database with the base configuration data that comes with the product.

WARNING: BEA's database creation scripts first drop all database objects and then they recreate them, which means that all data added since your original installation will be lost. After running the database configuration scripts, only the base configuration data that is needed for the product will exist.

For instructions on manually configuring your database, see the chapter in this guide pertaining to your database vendor.

Creating the GroupSpace Database

For every database other than PointBase (where it is created by default), you need to create a GroupSpace database or database user if you want to use GroupSpace. The instructions for doing this are in the chapter in this guide that pertains to your particular database vendor.

Creating Additional Content Management Databases

You need to manually create any additional content repositories. Instructions for doing this are in the Content Management Guide.

CHAPTER 3

Database Setup and Maintenance Overview

This section identifies considerations for setting up and maintaining WebLogic Portal databases.

For a discussion of database environment considerations for team development, see the *WebLogic Portal Production Operations Guide*.

For additional information on vendor-specific considerations and recommendations, see the chapter in this guide pertaining to your database vendor.

This chapter includes the following sections:

Encrypting Passwords

WebLogic Portal's database creation scripts (`create_db.cmd/sh`) use the domain's salt file when they call JDBCDataLoader.

Database passwords in the database properties files, such as `database.properties` and `groupspace_database.properties`, can be either plain text or encrypted. Use `weblogic.security.Encrypt` to obtain the encrypted passwords. The salt file used to encrypt and decrypt passwords is the domain's *`DOMAIN_HOME`*`/security/SerializedSystemIni.dat` salt file.

For more about `weblogic.security.Encrypt`, see *Using the WebLogic Server Java Utilities* in the WebLogic Server documentation.

Choosing Internationalization and Localization Settings

The Localization Industry Standards Association (LISA) defines internationalization and localization as follows:

"Internationalization is the process of generalizing a product so that it can handle multiple languages and cultural conventions without the need for re-design. Internationalization takes place at the level of program design and document development."

"Localization involves taking a product and making it linguistically and culturally appropriate to the target locale (country/region and language) where it will be used and sold."

This section contains information on choosing language settings, character sets, and collation (sort) settings for your database. It includes the sections:

- Changing the Language Settings in SQL Server
- Supporting Unicode Languages in SQL Server
- Choosing Character Sets and Sort Orders

Changing the Language Settings in SQL Server

To use SQL Server with portal applications that are localized, you must modify the language settings in the *`WL_HOME`*`\portal\db\sql_server\admin\create_database.sql` script before running it to create the database.

For example, to change the language setting from US English to Japanese, perform the following edits:

1. Change the line reading:

   ```
   COLLATE SQL_Latin1_General_CP1_CS_AS
   ```

   to:

   ```
   COLLATE Japanese_CS_AS_KS_WS
   ```

2. Change the line reading:

   ```
   exec sp_addlogin 'WEBLOGIC', 'WEBLOGIC', 'WEBLOGIC', 'us_english'
   ```

   to:

   ```
   exec sp_addlogin 'WEBLOGIC', 'WEBLOGIC', 'WEBLOGIC', 'japanese'
   ```

3. Run the script (after making any additional required modifications).

For more information about language support in SQL Server, refer to the vendor documentation.

Supporting Unicode Languages in SQL Server

If you want to localize portal applications to support a unicode language, you can edit and run the *WL_HOME*\portal\db\sql_server\admin\alter_use_unicode_columns.sql script to alter all columns of type TEXT, CHAR, and VARCHAR to use their unicode equivalents (NTEXT, NCHAR, and NVARCHAR, respectively).

WARNING: Changing these column data types has performance implications. Each column that is modified uses twice as much storage space as the original data type.

Back up the database that contains the portal schema before running this script.

After running the script, you can use `DBCC CHECKDB` command to verify physical storage and table constraints.

Choosing Character Sets and Sort Orders

A database's character set determines which languages can be represented in the database. A database's collation (sort order) determines the rules by which characters are sorted and compared.

For a globalized WebLogic Portal application, choose the database's character set and sort order before you set up the database. Changing an existing database's character set and sort order can be a time- and resource-intensive task. Typically a change can be made only if the target character set is a subset of the source.

The following sections describe character set and sort order considerations for WebLogic Portal-supported databases. Refer to your vendor documentation for more information.

Oracle

For Oracle, you define the character set when you create the database. In globalized Oracle 9i and 10g databases, a commonly used character set is `AL32UTF8`.

You can retrieve information about an Oracle database's character set and sort order by querying the SYS.NLS_DATABASE_PARAMETERS table.

SQL Server

For SQL Server, you can define both instance-level and database-level character sets and sort orders.

The default instance-level character set for SQL Server is determined by the locale setting for the Windows operating system. The following sample output from the `sp_helpsort` stored procedure shows a common instance level sort order:

```
Latin1-General, case-insensitive, accent-sensitive, kanatype-insensitive,
width-insensitive for Unicode Data, SQL Server Sort Order 52 on Code Page
1252 for non-Unicode Data.
```

The sample database creation script is *WL_HOME*`\portal\db\sql_server\admin\create_database.sql`. It defines the WebLogic Portal database with the setting `COLLATE SQL_Latin1_General_CP1_CS_AS`.

This specifies a case-sensitive definition. Setting the database to be case-sensitive allows content management queries to function for SQL Server as they do for other databases. If content management search queries are not written for a case-sensitive environment, you may have unexpected results.

You can use the `sp_helpdb` *dbname* stored procedure to display a SQL Server database's character set and sort order.

Sybase

For Sybase, you define the character set and sort order when you create the Sybase instance. A commonly used Sybase character set and sort order definition is:

```
Character Set = 2, cp850 Code Page 850 (Multilingual)
```

```
Sort Order = 50, bin_cp850 Binary ordering, for use with Code Page 850
(cp850).
```

You can use the `sp_helpsort` stored procedure to display information about an instance's character set and sort order.

DB2

For DB2 databases, you can set character encoding for each database individually. The default character encoding is determined by the operating system.

A commonly used DB2 character encoding is `UTF-8`.

You can use the `DB2 CODESET` configuration parameter to determine a database's codeset.

Performance Considerations

This section details factors to consider when optimizing database performance. It includes the sections:

- Collocating the Database and WebLogic Portal
- Using Caching to Reduce Database Access
- Caching Database Tables Using Database-Specific Settings
- Improving Concurrency Using Sybase Locking
- Improving Content Search Response Time for Oracle
- Using Database Statistics for Table and Index Organization
- Optimizing Table Data and Index Placement

Collocating the Database and WebLogic Portal

To prevent network latency issues, you should locate your database and your WebLogic Portal instances in the same data center. This recommendation is especially important when you use the Propagation Utility, to ensure that a large propagation process can complete successfully.

Using Caching to Reduce Database Access

The frequency of database access varies greatly depending on what functionality is used and how it is used. By reducing the frequency of database access, caching plays a large role in configuring the portal for optimum performance.

For the core portal framework, the portal taxonomy that all users share is loaded in memory during initialization, so database access is typically infrequent after initialization. If a user has customized the portal, a database read is necessary the first time they access it, to retrieve their customizations, user preferences, and so on.

Many other features of the product also use the database, including the content management store. If content is frequently accessed, caching is a good idea where possible; otherwise, database access is necessary to retrieve the content.

Caching Database Tables Using Database-Specific Settings

Some databases provide the ability to cache or *pin* database tables into memory. Choose a database caching implementation based on the WebLogic Portal components that are deployed for your application and how you deploy them.

The following summarizes the support provided by each database type for caching or pinning database tables. Depending on your environment, you might focus your caching strategy on small tables and frequently referenced tables.

- Oracle's touch-count algorithm makes it unnecessary to pin specific tables to buffers or database caches. You can adjust Oracle instance parameters to influence buffer behavior.
- SQL Server provides the `DBCC PINTABLE (database_id, table_id)` command to pin tables to the buffer cache.
- Sybase provides the ability to define multiple database caches and to selectively bind tables, indexes, and logs to those caches. Use `sp_objects_stats` to identify objects that may benefit from their own cache. Use a Sybase monitoring tool or `sp_sysmon` to determine if cache hit ratios are acceptable.
- DB2 uses the `BUFFERPOOL` configuration parameter to cache database data; this should be the focus of tuning efforts. DB2 allows you to assign individual tables and indexes to a buffer pool.

Improving Concurrency Using Sybase Locking

A Sybase instance's default locking mechanism is `ALL PAGES`. For concurrency, locking can also be defined for each table in a Sybase database.

The following WebLogic tables are defined with `LOCK DATAROWS` for Sybase. Other WebLogic Portal tables for Sybase are defined with `LOCK DATAPAGES`.

- CATALOG_ENTITY
- ENTITY
- SEQUENCER
- L10N_INTERSECTION
- PF_DESKTOP_INSTANCE
- PF_PLACEMENT

Based on your usage of WebLogic Portal components with a Sybase database, you may decide to modify additional tables to the setting `LOCK DATAROWS`.

Improving Content Search Response Time for Oracle

Content search within WebLogic Portal typically performs best when using the indexes associated with the content repository. In some situations the Oracle optimizer, based on database statistics, chooses to perform tablespace scans instead of using indexes to access the data. Often this results in a much slower response time than if an index was used.

To improve response time, verify that the `optimizer_mode` Oracle initialization parameter is set to `choose`.

In addition, to ensure that Oracle indexes are given greater preference over tablespace scans, you can modify the following two initialization parameters:

```
optimizer_index_cost_adj (range 0-100)

optimizer_index_caching (range 0-100)
```

Refer to your Oracle documentation for the more information about these settings, and the impact modifications may have on your installation.

Using Database Statistics for Table and Index Organization

Each DBMS has its own utility or commands for updating the database statistics used by its query optimizer. The statistics that you collect for your database often provide information on the organization of tables and indexes in your database. A DBA should schedule periodic jobs to maintain database statistics.

Data Manipulation Language (DML) operations (for example, `DELETE`, `INSERT`, `UPDATE`) executed in WebLogic Portal applications can affect table and index organization; this can affect database performance. Refer to your database vendor documentation for details on utilities that are available for table and index reorganization and for information on determining when a re-organization should occur.

Optimizing Table Data and Index Placement

The following sections describe special considerations, recommendations, and requirements for index placement, CLOB/BLOB/TEXT/IMAGE data storage, and behavior tracking:

Choosing Index Placement

For Oracle databases, use the `rebuild_indexes.sql` script to move indexes into their own tablespace, such as the WEBLOGIC_INDEX tablespace. Run this script manually after creating the WebLogic Portal database to move indexes into their own tablespace. For more information, see "Configuring Oracle Databases" on page 6-2.

For SQL Server and Sybase, the Data Definition Language (DDL) for non-clustered indexes is defined with `ON WEBLOGIC_INDEX` to place indexes into their own file group or segment. You must define the WEBLOGIC_INDEX file group or segment before running WebLogic Portal database creation or upgrade scripts. The provided sample `create_database.sql` scripts define WEBLOGIC_INDEX appropriately.

Choosing Data Storage for Tables with CLOB/BLOB/TEXT/IMAGE Data

This section describes recommendations and special considerations for tables that contain CLOB/BLOB (Oracle/DB2) or TEXT/IMAGE (Sybase/SQL Server) data.

As a general rule, for optimal performance with any of the supported WebLogic Portal database platforms, CLOB/BLOB/TEXT/IMAGE table data should be physically stored separately from other types of data. However, to simplify deployment and to allow for flexibility in your configuration, the default WebLogic Portal database schema is not deployed with separated data.

To determine if your specific application is a candidate for separating CLOB/BLOB/TEXT/IMAGE storage, assess the following considerations in your environment:

- **Content management configuration** - If you are using the out-of-the-box content management system heavily and you are storing and retrieving many large documents, you may want to modify your storage allocations for the tables that have the CM and CMV prefixes.

  The relevant table columns are:

  – CMV_VALUE.BLOB_VALUE

  – CM_PROPERTY.BLOB_VALUE

  – CM_PROPERTY_CHOICE.BLOB_VALUE

- **Behavior tracking configuration** - If you have behavior tracking turned on, you may want to modify storage allocations for behavior tracking tables.

  The relevant table column is BT_EVENT.XML_DEFINITION.

- **Disc controller configuration** - If your environment has a separate disk controller and drive that can be dedicated to CLOB/BLOB/TEXT/IMAGE storage, you might see performance improvements using a separate controller rather than sharing a controller.

- **RAID storage configuration** - If you are using RAID, you may see limited improvement by changing storage allocations. Follow your database vendor recommendations based on your specific configuration.

If you want to change your storage allocations, see your vendor documentation for specific details on changing settings for CLOB/BLOB/TEXT/IMAGE data.

The following additional columns contain CLOB/BLOB or TEXT/IMAGE data types; you can experiment with storage changes to determine the effect on performance.

- AD_BUCKET.AD_QUERY
- CATALOG_PROPERTY_VALUE.BLOB_VALUE
- DATA_SYNC_ITEM.XML_DEFINITION
- DISCOUNT.DISCOUNT_RULE
- MAIL_MESSAGE.MESSAGE_TEXT
- P13N_ANONYMOUS_PROPERTY.PROPERTY_VALUE
- P13N_DELEGATED_HIERARCHY.ENTITLEMENT_DOCUMENT

- PF_CONSUMER_REGISTRY.REGISTRATION_STATE
- PF_PROXY_PORTLET_INSTANCE.PORTLET_STATE
- PLACEHOLDER_PREVIEW.XML_DEFINITION
- PROPERTY_VALUE.BLOB_VALUE

Choosing Behavior Tracking Data Placement

Behavior tracking enables you to develop, manage, and measure personalized portal applications. By recording events, you can customize portal applications and track visitor behavior.

Due to the large number of rows that can be written to the BT_EVENT table when behavior tracking is enabled, you might want to use a separate database (or tablespace and schema, depending upon your DBMS) to store behavior tracking data.

For each database other than PointBase, see the chapter in this guide pertaining to your database vendor for information on creating a separate database for behavior tracking events.

Sizing Considerations

This section describes the considerations for database, page, block, and data type sizing.

Database Sizing

Sizing for your WebLogic Portal databases depends on many factors, including the following:

- The components of WebLogic Portal that are deployed for your application and the method you use to deploy them. By default, all WebLogic Portal database objects are created for each domain. If a component of WebLogic Portal is not deployed, its database objects will exist, but will not contain any data rows.
- The number of WebLogic Portal application users in your environment.
- The degree to which customization or personalization is allowed for your end users' WebLogic Portal resources. For example, you might ask these questions about user personalization:
  - Can end users create their own portlets such as My Mail Portlet, My Task List Portlet, and so on?
  - Can end users create their own portal resource views?

Note: WebLogic Portal resource view data is stored in the Portal Framework tables named `PF_<resource_type>_INSTANCE`.

The following sections describe some WebLogic Portal database tables that you should monitor closely for growth.

Monitoring Behavior Tracking Tables

The BT_EVENT table can grow significantly if behavior tracking is enabled.

Monitoring Personalization Tables

The following database tables store personalization data; monitor them for data growth.

- ENTITY – This table grows as you define new users and groups for your WebLogic Portal application.
- PROPERTY_VALUE – This table grows as property values are added to a user profile. This table can become quite large when a large number of users exist with a large number of property values per user. For a discussion of user and group profile values, see the User Management Guide.

Monitoring Portal Framework and WSRP Tables

The following tables involve the Portal Framework and Web Services for Remote Portlets (WSRP) and can potentially be high volume tables; monitor them for data growth.

- PF_BOOK_INSTANCE – This table identifies instances of the `BOOK_DEFINITION`. There is always at least one book instance, namely the primary instance; all other instances represent customization by administrators or end users. This table can grow significantly if extensive user customization occurs.
- PF_BOOK_GROUP – This table represents child pages or book placements on the parent book. A single record in the table represents one placement on a book. This table can grow significantly if extensive user customization occurs.
- PF_PLACEMENT – This table tracks the portlets on a specific portal page for each desktop. Any time a visitor modifies the position of portlets on a page, a row is inserted for each portlet that exists on the portal page for the user's custom desktop. This table can grow significantly because of portal customization; this growth can be expressed using the equation:

  *number of users * number of pages * number of portlets = number of rows*

- PF_PORTLET_PREFERENCE_VALUE – This table identifies preference values for the portlet instance. This table can grow significantly if extensive user customization occurs.
- PF_PORTLET_INSTANCE – A portlet definition has at least one portlet instance, known as the primary instance. Every time a portlet is dragged onto a page, a new instance of the portlet is created and a column is inserted into this table. If, for example, an administrator drags a portlet onto a page and then a user modifies the portlet by setting the default to "minimized," another instance is created and a row inserted. (Subsequent changes to the portlet instance do not create a new instance/row.) This growth can be expressed using the equation:

  ```
  number of portlet definitions * number of instances = number of rows
  ```

- PF_DESKTOP_INSTANCE – This table identifies a customized or localized instance of a desktop. This table can grow significantly; a row is added for each entitled desktop, and possibly a row for each unique visitor who customizes the site. This growth can be expressed using the equation

  ```
  number of portals * number of desktops + number of visitors who customize
  = number of rows
  ```

Monitoring Content Management Database Tables

The following database tables store content management (CM) data; monitor them for data growth.

- CM_PROPERTY – This table stores property values. This table can grow significantly if content management is used to store large quantities of data.
- CM_NODE – This table uniquely identifies a node within a BEA repository. This table can grow significantly if content management is used to store large quantities of data.
- CMV_NODE – This table uniquely identifies a content-managed node from a BEA repository (from the CM_NODE table) that has been versioned and is being edited within the Virtual Content Repository. This table can grow significantly if content management is used to store large quantities of data.
- CMV_NODE_VERSION – This table uniquely identifies all the versions of a mode within the Virtual Content Repository. This table can grow significantly if content management versioning is enabled (the default).
- CMV_NODE_VERSION_PROPERTY – This table uniquely identifies a relationship between a CMV_NODE_VERSION and CMV_PROPERTY. This table will likely have

the largest number of rows within content management if versioning is enabled. It may not be the largest table because it contains cross-references.

- CMV_PROPERTY – This table uniquely identifies a property that can be associated with a node version. For example, some properties of a book might be author, title, and subject. This table can grow significantly if content management versioning is enabled (the default).
- CMV_VALUE – This table uniquely identifies a value for a given property. For example, a property `SUBJECT` for a `BOOK` might have a value of `FINANCE`. This table can grow significantly if content management is used to store large quantities of data. This table is likely be the largest content management table because it stores the metadata associated with each content node.

Monitoring Community Database Tables

The following database tables store Community data; you should monitor them for data growth. If you are using multiple Communities or have a large numbers of users, you can expect these tables to consume more space than the basic Portal Framework customization tables.

- PF_COMMUNITY_DEFINITION, PF_COMMUNITY_PROPERTY and PF_INVITATION scale in proportion to the number of Community instances.
- PF_COMMUNITY_MEMBER scales in proportion to the number of WebLogic Server users involved in Communities.
- PF_COMMUNITY_MEMBERSHIP, PF_MEMBERSHIP_CAPABILITY, PF_INVITEE, PF_INVITEE_PROPERTY and PF_NOTIFICATION scale in proportion to the number of Community instances multiplied by the number of Community members per Community.

Setting Page and Block Sizes

For Oracle and SQL Server databases, an 8K page/block size is the default.

For DB2 databases, an 8K buffer pool is defined for the WebLogic Portal tables and indexes that require this larger pool size (higher than the 4K default).

For Sybase databases, an 8K page size is required for several WebLogic Portal tables and indexes. For Sybase the default page size is 2K, and Sybase does not allow rows to span pages. If your Sybase instance uses 2k or 4k pages, create a new Sybase instance with an 8K page size. Sybase provides a migration utility to migrate data between servers of different page sizes.

Note: See the chapter in this guide pertaining to your database vendor for additional details on database-specific considerations.

Setting BLOB/CLOB Data Type Size Limits

For DB2 and PointBase databases, WebLogic Portal specifies a maximum BLOB/CLOB size of 30 MB when creating database tables. If you require a larger size, you can edit the appropriate database creation script to change this value. Refer to "Property Files and Database Scripts" on page B-1 and the chapter in this guide pertaining to your database vendor for the location and description of the database scripts.

In general, you should be aware of any size restrictions that might apply to your database. Restrictions can vary from one database version to another; refer to your vendor documentation for more information.

Performing Database Backup and Recovery

Use the same procedures for backup and recovery of WebLogic Portal databases as you use for other database data.

The following are general recommendations for backup and recovery:

- Store your data and J2EE resources in source control, and back up the source control database.
- Back up your WebLogic Portal database according to your database vendor's recommendations.

Perform periodic test restores to ensure that your backups are sound.

Propagating Portal Environments

You use the WebLogic Portal Propagation Utility to move portal metadata from one portal environment to another. You should be aware of the usage of this utility, and back up your databases prior to propagation.

By design, when you move a portal environment, not all database data is propagated. For example, user and group data is not propagated.

For more information about the Propagation Utility, and what database data is propagated, see the *WebLogic Portal Production Operations Guide*. This guide also describes database environment considerations for team development.

Commerce JDBC Configuration Settings

You must use a non-XA data source configuration such as `portalDataSourceNeverXA-jdbc.xml` if you are using commerce functionality.

Remove the jndi-name `weblogic.jdbc.jts.commercePool` from the `portalDataSource-jdbc.xml` file and add it to the `portalDataSourceNeverXA-jdbc.xml` file.

Support for Asynchronous Proliferation Requires XA

JMS and JMS database tables are created automatically by WebLogic Server if they do not already exist in the database to store proliferation information. The Portal Resource Proliferation Mode setting determines how updates to portal resources (pages, books, portlets) are cascaded - or proliferated - to desktops and user-customized instances. You can set the mode to Asynchronous, Synchronous, or Off in the WebLogic Portal Administration Console.

When proliferation is performed asynchronously (the default) a JMS queue is used, which requires an XA driver. In this case, the `portalDataSourceAlwaysXA-xml` data source is used. This means that your database environment must be configured to support XA.

CHAPTER 4

Using PointBase

The sample data provided with WebLogic Portal uses the PointBase DBMS. PointBase is a demonstration database ships with WebLogic Server to allow you to use sample user accounts and data. PointBase is the default database when you create a domain with the Configuration Wizard.

Note: PointBase is supported only for the design, development, and verification of applications; it is not supported for enterprise-quality deployment. The evaluation license of PointBase has a database size limit of 30 MB.

The PointBase server is started automatically when you start WebLogic Server; it must be running for your applications to access it.

This chapter contains the following sections:

PointBase Documentation

PointBase documentation for the version of PointBase supported by WebLogic Server is distributed with WebLogic Server in Adobe PDF form in the *WL_HOME*\common\eval\pointbase\docs\embedded\pdf directory. The PointBase documentation consists of the following manuals:

- PointBase Console Guide
- PointBase Developer's Guide
- PointBase System Guide

PointBase JAR Files

Refer to the section titled "PointBase JAR Files" in the "Before You Begin" chapter of the PointBase System Guide for information on the JAR files provided with WebLogic Server in the *WL_HOME*\common\eval\pointbase\lib directory.

PointBase Tools

Scripts for starting the PointBase Server and the PointBase console are distributed with WebLogic Server in the *WL_HOME*\common\eval\pointbase\tools directory. Scripts are called by start scripts in the sample domains and by start scripts contained in any domain created by the Configuration Wizard. These PointBase start scripts simplify starting the PointBase Server and Console within WebLogic domains.

WebLogic Portal PointBase Databases

PointBase stores all data in .dbn files and all log information in .wal files. Database properties are stored in PointBase.ini files. Data files for WebLogic Portal are named weblogic_eval.dbn and log files for WebLogic Portal are named weblogic_eval$1.wal. Pre-built PointBase data, log, and PointBase.ini files for WebLogic Portal samples are included in the following directory:

WL_HOME\samples\domains\portal

By default domains created using the Configuration Wizard with the Basic WebLogic Portal Domain template cause PointBase data and log files to be created in the following directory:

BEA_HOME\user_projects\domains*DomainName*

PointBase Database Size Restriction

The evaluation license of PointBase has a database size limit of 30 MB. You might find that you reach this limit if you use PointBase extensively. If this happens, you can upgrade your PointBase license by contacting `sales@pointbase.com`, or reduce the amount of data by deleting table rows and running the Database Compress Tool to compress the database to a more compact size. You must shut down the server before running this command line tool; no other process can access the database while the tool runs. For more information, see the PointBase Developer's Guide (*`WL_HOME`*`\common\eval\pointbase\docs\embedded\pdf\pbdeveloper.pdf`).

Administering the WebLogic Portal PointBase Database

You can administer PointBase using the PointBase administration console, or any third-party database visualization and management tool that can connect using JDBC.

You can launch the PointBase Console either from the Windows Start Menu or by executing the `startPointBaseConsole.cmd/.sh` script located in the domain directory.

Prior to launching the PointBase Console, ensure that WebLogic Server for the domain is running. This starts the PointBase Server.

This section contains the following sections:

Launching the PointBase Console from the Windows Start Menu

To launch the PointBase Console from the Windows Start menu:

1. Go to **Start > All Programs > BEA Products > Examples > WebLogic Portal > PointBase Console**.

   Or, if you added Start menu options for a domain created by the Configuration Wizard, navigate to that domain's PointBase Console menu option.

2. When the PointBase Console starts, it prompts you to enter connection parameters to properly connect to the database. Enter the following connection information; you also need this information if you use a third-party product to access the PointBase database:

   – **Driver**: `com.pointbase.jdbc.jdbcUniversalDriver`

- **URL for a portal domain you create using the Configuration Wizard**: `jdbc:pointbase:server://localhost:9093/weblogic_eval`
- **URL for sample portal domain**: `jdbc:pointbase:server://localhost:9052/weblogic_eval`
- **User**: `weblogic` (or `weblogic_groupspace`)
- **Password**: `weblogic` (or `weblogic_groupspace`)

Launching the PointBase Console from the startPointBaseConsole Script

To launch the PointBase Console using the script:

1. Change directories to *`WL_HOME`*`\samples\domains\portal\bin`.
2. For a domain created by the Configuration Wizard, navigate to that domain's home directory.
3. Execute the appropriate start script—`startPointBaseConsole.cmd` or `startPointBaseConsole.sh`—to launch the PointBase Console.
4. When the PointBase Console starts, it prompts you to enter connection parameters to properly connect to the database. Enter the following connection information; you also need this information if you use a third-party product to access the PointBase database:
   - **Driver**: `com.pointbase.jdbc.jdbcUniversalDriver`
   - **URL for a portal domain you create using the Configuration Wizard**: `jdbc:pointbase:server://localhost:9093/weblogic_eval`
   - **URL for sample portal domain**: `jdbc:pointbase:server://localhost:9052/weblogic_eval`
   - **User**: `weblogic` (or `weblogic_groupspace`)
   - **Password**: `weblogic` (or `weblogic_groupspace`)

CHAPTER 5

Using Microsoft SQL Server

This chapter describes the steps necessary to use a Microsoft SQL Server database with WebLogic Portal, and includes the following sections:

- Configuring Microsoft SQL Server Databases
- Configuring the Main WebLogic Portal Database
- Configuring the GroupSpace Database
- Manually Creating a Separate Database and Database Objects for Behavior Tracking
- Database Scripts and Usage Notes

Review this entire chapter and any release notes before proceeding. The tasks in this chapter should be performed by a database administrator.

Configuring Microsoft SQL Server Databases

Before proceeding, read "Overview of Enterprise-Quality Database Configuration for WebLogic Portal" on page 2-2.

The database creation scripts install domain-specific tables. It is recommended that you work with a database administrator to modify the sample scripts and create database devices, file groups, databases, and database users for your SQL Server environment.

Multiple databases are required if you have multiple domains, or to run multiple environments using the same SQL Server instance (for example, if you want to run development and system

test from a single SQL Server installation). GroupSpace requires a separate database, as do any additional content management repositories.

Note: The SQL Server JDBC driver requires additional steps to configure stored procedures for JTA and to place a required DLL on the SQL Server database host. For instructions, see the *WebLogic Server Type 4 JDBC Drivers Guide*.

To configure a SQL Server database:

1. Be sure to back up your database before installing any new database objects. See your database vendor documentation for details.
2. Review the provided sample scripts, located in the *WL_HOME*\portal\db\sql_server\admin directory. See Table 5-1, "Database Scripts and Usage Notes," on page 5-7 and the comments in the scripts for additional information.
3. Copy and modify the sample scripts appropriately for your environment to create each of the following databases:
   a. Follow the steps in "Configuring the Main WebLogic Portal Database" on page 5-2 to create the main WebLogic Portal database and database objects.
   b. If you want to use GroupSpace, follow the steps in "Configuring the GroupSpace Database" on page 5-3 to create the GroupSpace database and database objects.
   c. If you want to create a separate behavior tracking database, follow the steps in "Manually Creating a Separate Database and Database Objects for Behavior Tracking" on page 5-5.

Configuring the Main WebLogic Portal Database

To configure the main WebLogic Portal database, follow these steps:

1. Copy the `create_database.sql` script and modify it appropriately for your environment. See Table 5-1, "Database Scripts and Usage Notes," on page 5-7 and the comments in the script for additional information.
2. Run the modified `create_database.sql` script as a user with System Administrator privileges (normally the `sa` user). For example, from `osql`:

   ```
   osql -Usa -SSQLSERVER -e -icreate_database.sql -ocreate_database.log
   ```

   The output from running `create_database.sql` is written to `create_database.log`. Verify that there are no errors in the log file before proceeding.

Follow the remaining steps only if you want to create database objects manually rather than using the Configuration Wizard. To perform the remaining steps using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

3. Open your domain's `database.properties` file for edit.
4. Set `database=sql_server`.
5. Update the following settings (by replacing the @ symbols and the text between the symbols with the correct values) for your main WebLogic Portal database:

```
sql_server.user=@DB_USER@

sql_server.password=@DB_PASSWORD@

sql_server.url=jdbc:bea:sqlserver://@DB_HOST@:@DB_PORT@;DatabaseName=@DB_NAME@
```

6. Create the database objects.
   a. Navigate to the *BEA_HOME*`\user_projects\domains\`*myPortalDomain* directory.
   b. Double-click `create_db.cmd`.

      If any error messages are displayed, check the `create_db.log` file for additional information.
7. Replace the JDBC data sources in your domain (except for `appsGroupSpaceDataSource`), which point to PointBase by default, with data sources that point to SQL Server. You can configure them using the WebLogic Server Administration Console or choose from the samples provided and update them for your database environment. Sample `jdbc.xml` definition files for each database and driver that BEA supports are available in the *WL_HOME*`\portal\db\jdbc\`*database_driver* directory; for example, `sql_server_bea`. Follow the instructions in the *WL_HOME*`\portal\db\jdbc\README.txt` file.

Configuring the GroupSpace Database

To configure the GroupSpace database, follow these steps:

1. Copy the `create_database.sql` script and modify it appropriately for your environment. See Table 5-1, "Database Scripts and Usage Notes," on page 5-7 and the comments in the script for additional information.
2. Run the modified `create_database.sql` script as a user with System Administrator privileges (normally the `sa` user). For example, from `osql`:

```
osql -Usa -SSQLSERVER -e -icreate_database.sql -ocreate_database.log
```

The output from running `create_database.sql` is written to `create_database.log`. Verify that there are no errors in the log file before proceeding.

Follow the remaining steps only if you want to create database objects manually rather than using the Configuration Wizard. To perform the remaining steps using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

3. Open your domain's `groupspace_database.properties` file for edit.
4. Set `database=sql_server`.
5. Update the following settings (by replacing the @ symbols and the text between the symbols with the correct values) for your GroupSpace database:

   ```
   sql_server.user=@DB_USER@

   sql_server.password=@DB_PASSWORD@

   sql_server.url=jdbc:bea:sqlserver://@DB_HOST@:@DB_PORT@;DatabaseName=@DB_NAME@
   ```

6. Create the database objects.
   a. Navigate to the *BEA_HOME*`\user_projects\domains\`*myPortalDomain* directory.
   b. Enter the command:

      ```
      create_db.cmd -database.properties=groupspace_database.properties
      ```

      If any error messages are displayed, check the `create_db_groupspace.log` file for additional information.
7. Replace the `appsGroupSpaceDataSource` JDBC data source (which points to PointBase by default) with a data source that points to SQL Server. Use the WebLogic Server Administration Console or update the sample `jdbc.xml` definition file provided for each database and driver that BEA supports, in the *WL_HOME*`\portal\db\jdbc\`*database_driver* directory; for example `sql_server_bea`. Follow the instructions in the *WL_HOME*`\portal\db\jdbc\README.txt` file.

Manually Creating a Separate Database and Database Objects for Behavior Tracking

For improved performance, you might want to store behavior tracking events in a different location from other WebLogic Portal database objects. For more information about behavior tracking, see *Setting Up Events and Behavior Tracking* in the Interaction Management Guide.

Note: By default, behavior tracking database objects are created in the same database as other WebLogic Portal database objects. You need to perform these steps only if you are configuring a separate database for behavior tracking events.

To create a separate database for behavior tracking:

1. Modify the `bt_create_database.sql` file for your environment, as indicated in the instructions contained in the scripts and in Table 5-1, "Database Scripts and Usage Notes," on page 5-7.

2. Run `bt_create_database.sql` as a user with system administrator privileges. For example, from `osql`:

   ```
   osql -Usa -SSQLSERVER -e -ibt_create_database.sql
   -obt_create_database.log
   ```

   If any error messages are displayed, check the `bt_create_database.log` file for additional information.

3. Navigate to the appropriate database directory based on your environment; for example, *`WL_HOME`*`\common\p13n\db\sql_server`.

4. Connect as the user `WEBLOGIC_EVENT` and run the following scripts:

   - `bt_create_tables.sql`
   - `bt_create_fkeys.sql`
   - `bt_create_indexes.sql`
   - `bt_create_triggers.sql`

5. Run the following scripts from the path *`WL_HOME`*`\portal\db\data\required`:

   - `bt_insert_system_data.sql`
   - `bt9_insert_system_data.sql`

6. Use the WebLogic Server Administration Console to configure a non-XA JDBC data source to access your behavior tracking database. Associate the JNDI name

`p13n.trackingDataSource` with that data source and then remove `p13n.trackingDataSource` from `p13nDataSource`.

Database Scripts and Usage Notes

Table 5-1 describes the scripts that enable you to configure the Oracle database.

Table 5-1 Database Scripts and Usage Notes

Script Name	Description
`create_database.sql`	Creates a database. You must create a main WebLogic Portal database. If you want to use GroupSpace, you must also create that database. You must also create a database for any additional content management repositories. Make a copy of this script and edit it to replace `<<WEBLOGIC>>` with the appropriate database name, database owner user, and password for each database you create. You must also edit the script to reflect valid disk locations for `DATA` devices, `LOG` devices, and the `WEBLOGIC_INDEX` file group; you may also need to modify file sizes. Put `DATA` and `LOG` files on separate physical disks and away from any system database files, unless you are using RAID devices. **Note:** Do not change the name of the `WEBLOGIC_INDEX` file group. Typical names for the main WebLogic Portal database are: • Database name: `WEBLOGIC` • Database owner user: `WEBLOGIC` • Password: `WEBLOGIC` When you run the script with these values, it creates the `WEBLOGIC` database, the `WEBLOGIC_INDEX` file group, and `WEBLOGIC` database owner (dbo) user login. An alias is created to make `WEBLOGIC` the dbo user in the database. It also sets the `WEBLOGIC` database as the default database for the `WEBLOGIC` user. **Note:** To use GroupSpace, you must create an additional database. Typical names for the GroupSpace repository database are: • Database name: `WEBLOGIC_GROUPSPACE` • Database owner user: `WEBLOGIC_GROUPSPACE` • Password: `WEBLOGIC_GROUPSPACE` **Note:** If you decide to create an additional content management repository, you must create a database with different user names for it. For additional information, see the Content Management Guide.

Table 5-1 Database Scripts and Usage Notes (Continued)

Script Name	Description
statistics.sql	Runs sp_updatestats to compute database statistics needed for the database optimizer. Update statistics periodically and whenever any significant changes in database data occur. (This is done by default for SQL Server databases with the AUTO_UPDATE_STATISTICS database option enabled.) When set to ON (the default), existing statistics are automatically updated when the data in the tables has changed. When set to OFF, existing statistics are not automatically updated. You must manually update statistics. The AUTO_UPDATE_STATISTICS option setting is set in the IsAutoUpdateStatistics property of the DATABASEPROPERTYEX function.
install_report.sql	Builds an informational installation report about the database objects created in the WEBLOGIC schema.
bt_create_database.sql	Creates the WEBLOGIC_EVENT database and WEBLOGIC_EVENT database owner user login. An alias is created to make WEBLOGIC_EVENT the database owner (dbo) user in the database. You must edit the script to reflect valid disk locations for the DATA and LOG devices, or to modify file sizes. Put DATA and LOG files on separate physical disks and away from any system database files. The default names are the following: • Database name: WEBLOGIC_EVENT • Database owner user: WEBLOGIC_EVENT • Password: WEBLOGIC_EVENT

CHAPTER 6

Using Oracle

This chapter describes the steps necessary to use an Oracle database with WebLogic Portal, and includes the following sections:

- Configuring Oracle Databases
- Configuring the Main WebLogic Portal Database
- Configuring the GroupSpace Database
- Manually Creating a Separate Database and Database Objects for Behavior Tracking
- Database Scripts and Usage Notes
- Using WebLogic Portal with Oracle RAC

Review this entire chapter and any release notes before proceeding. The tasks described in this chapter should be performed by a database administrator.

Note: A WebLogic Portal Oracle database user that has DBA privileges causes database create errors and upgrade issues. Before you create database objects or perform an upgrade, you should revoke the Oracle user's DBA privileges and replace them with the following privileges: GRANT CREATE TABLE, CREATE VIEW, CREATE TRIGGER, CREATE SEQUENCE, CREATE SESSION, UNLIMITED TABLESPACE TO <<WEBLOGIC>>.

Configuring Oracle Databases

Before proceeding, read "Overview of Enterprise-Quality Database Configuration for WebLogic Portal" on page 2-2.

The database creation scripts install domain-specific tables. It is recommended that you work with a database administrator to modify the sample scripts, and to create the database users and tablespaces needed for your environment.

Multiple database users are required if you have multiple domains, or to run multiple environments using the same Oracle instance (for example, if you want to run development and system test from a single Oracle installation). GroupSpace requires a separate database user, as do any additional content management repositories.

Note: Oracle configuration settings can impact Content Search performance; for more information, see "Improving Content Search Response Time for Oracle" on page 3-7.

To configure an Oracle database:

1. Be sure to back up your database before installing any new database objects. See your database documentation for details.
2. Review the provided sample scripts, located in the *WL_HOME*/portal/db/oracle/admin directory. See Table 6-1, "Database Scripts and Usage Notes," on page 6-7 and the comments in the scripts for additional information.
3. Copy and modify the sample scripts appropriately for your environment to create each of the following database schemas:
   a. Follow the steps in "Configuring the Main WebLogic Portal Database" on page 6-2 to create the main WebLogic Portal user and database objects.
   b. If you want to use GroupSpace, follow the steps in "Configuring the GroupSpace Database" on page 6-4 to create the GroupSpace user and database objects.
   c. If you want to create a separate behavior tracking database, follow the steps in "Manually Creating a Separate Database and Database Objects for Behavior Tracking" on page 6-5.

Configuring the Main WebLogic Portal Database

To configure the main WebLogic Portal database, follow these steps:

1. If you want to create new tablespaces for the main WebLogic Portal schema:

a. Copy the `create_tablespaces.sql` script and modify it appropriately for your environment. See Table 6-1, "Database Scripts and Usage Notes," on page 6-7 and the comments in the script for additional information.

b. Run the modified `create_tablespaces.sql` script. For example, from SQL*Plus:

```
@create_tablespaces.sql
```

2. Create the main WebLogic Portal database user:

a. Copy the `create_users.sql` script and modify it appropriately for your environment. See Table 6-1, "Database Scripts and Usage Notes," on page 6-7 and the comments in the script for additional information.

b. Run the modified `create_users.sql` script. For example, from SQL*Plus:

```
@create_users.sql
```

Follow the remaining steps only if you want to create database objects manually rather than using the Configuration Wizard. To perform the remaining steps using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

3. Open your domain's `database.properties` file for edit.

4. Set `database=oracle`.

5. Update the following settings (by replacing the @ symbols and the text between the symbols with the correct values) for your main WebLogic Portal database:

```
oracle.user=@DB_USER@

oracle.password=@DB_PASSWORD@

oracle.url=jdbc:bea:oracle://@DB_HOST@:@DB_PORT@;SID=@DB_NAME@
```

6. Create the database objects.

a. Navigate to the *BEA_HOME*`\user_projects\domains\`*myPortalDomain* directory.

b. Enter one of the commands:

- On Windows, double-click `create_db.cmd`.
- On UNIX, run `create_db.sh`.

If any error messages are displayed, check the `create_db.log` file for additional information.

7. Replace the JDBC data sources in your domain (except for `appsGroupSpaceDataSource`), which point to PointBase by default, with data sources that point to Oracle. You can configure

them using the WebLogic Server Administration Console or choose from the samples provided and update them for your database environment. Sample `jdbc.xml` definition files for each database and driver that BEA supports are available in the *`WL_HOME`*`\portal\db\jdbc\`*`database_driver`* directory; for example, `oracle_bea` or `oracle_thin`. Follow the instructions in the *`WL_HOME`*`\portal\db\jdbc\README.txt` file.

8. For improved performance, move indexes to the `WEBLOGIC_INDEX` tablespace by executing `rebuild_indexes.sql`.

   Note: Do this while WebLogic Server is stopped.

Configuring the GroupSpace Database

To configure the GroupSpace database, follow these steps:

1. If you want to create new tablespaces for the GroupSpace schema:

   a. Copy the `create_tablespaces.sql` script and modify it appropriately for your environment. See Table 6-1, "Database Scripts and Usage Notes," on page 6-7 and the comments in the script for additional information.

   b. Run the modified `create_tablespaces.sql` script. For example, from SQL*Plus:

      ```
      @create_tablespaces.sql
      ```

2. Create the GroupSpace user:

   a. Copy the `create_users.sql` script and modify it appropriately for your environment and the GroupSpace settings. See Table 6-1, "Database Scripts and Usage Notes," on page 6-7 and the comments in the script for additional information.

   b. Run the modified `create_users.sql` script. For example, from SQL*Plus:

      ```
      @create_users.sql
      ```

Follow the remaining steps only if you want to create database objects manually rather than using the Configuration Wizard. To perform the remaining steps using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

3. Open your domain's `groupspace_database.properties` file for edit.

4. Set `database=oracle`.

5. Update the following settings (by replacing the @ symbols and the text between the symbols with the correct values) for your GroupSpace database:

   ```
   oracle.user=@DB_USER@
   ```

```
oracle.password=@DB_PASSWORD@
oracle.url=jdbc:bea:oracle://@DB_HOST@:@DB_PORT@;SID=@DB_NAME@
```

6. Create the database objects.
   a. Navigate to the *BEA_HOME*\user_projects\domains*myPortalDomain* directory.
   b. Enter one of the commands:
      - On Windows:

      ```
      create_db.cmd -database.properties=groupspace_database.properties
      ```

      - On UNIX:

      ```
      create_db.sh -database.properties=groupspace_database.properties
      ```

      If any error messages are displayed, check the `create_db_groupspace.log` file for additional information.
7. Replace the `appsGroupSpaceDataSource` JDBC data source (which points to PointBase by default) with a data source that points to Oracle. Use the WebLogic Server Administration Console or update the sample `jdbc.xml` definition file provided for each database and driver that BEA supports, in the *WL_HOME*\portal\db\jdbc*database_driver* directory; for example `oracle_bea` or `oracle_thin`. Follow the instructions in the *WL_HOME*\portal\db\jdbc\README.txt file.

Manually Creating a Separate Database and Database Objects for Behavior Tracking

For improved performance, you might want to store behavior tracking events in a different location from other WebLogic Portal database objects. For more information about behavior tracking, see *Setting Up Events and Behavior Tracking* in the Interaction Management Guide.

Note: By default, behavior tracking database objects are created in the same database as other WebLogic Portal database objects. You need to perform these steps only if you are configuring a separate database for behavior tracking events.

To create a separate database for behavior tracking:

1. Modify the `bt_create_tablespaces.sql` file and the `bt_create_users.sql` file for your environment. See Table 6-1, "Database Scripts and Usage Notes," on page 6-7 and the comments in the script for additional information.
2. Run the modified `bt_create_tablespaces.sql` script.

3. Run the modified `bt_create_users.sql` script.
4. Navigate to the appropriate database directory based on your environment: *`WL_HOME`*`\portal\db\oracle`
5. Connect as the user `WEBLOGIC_EVENT` and run the following scripts:
   - `bt_create_tables.sql`
   - `bt_create_fkeys.sql`
   - `bt_create_indexes.sql`
   - `bt_create_triggers.sql`
6. Run the following scripts from the path *`WL_HOME`*`\portal\db\data\required`:
   - `bt_insert_system_data.sql`
   - `bt9_insert_system_data.sql`
7. Use the WebLogic Server Administration Console to configure a non-XA JDBC data source to access your behavior tracking database. Associate the JNDI name `p13n.trackingDataSource` with that data source and then remove `p13n.trackingDataSource` from `p13nDataSource`.

Database Scripts and Usage Notes

Table 6-1 describes the scripts that enable you to configure the Oracle database and create WebLogic Portal objects in that database.

Table 6-1 Database Scripts and Usage Notes

Script Name	Description
`create_tablespaces.sql`	Creates table and index tablespaces. If you want to use existing tablespaces, you do not need to run this script. You can use new or existing tablespaces for the main WebLogic Portal database, the GroupSpace database, and any additional content management repositories. **Note:** Make a copy of this script and edit it to replace `<<WEBLOGIC>>` with the appropriate data and index tablespaces. You must also edit the script to reflect valid disk locations for the `DATA` and `INDEX` datafiles. Typical names for the main WebLogic Portal database are: • Table data: `WEBLOGIC_DATA` • Indexes: `WEBLOGIC_INDEX` **Note:** Indexes are placed in the table tablespace by default. Rebuild indexes by running `rebuild_indexes.sql` to move them to the index tablespace after initial database object creation.

Table 6-1 Database Scripts and Usage Notes (Continued)

Script Name	Description
`create_users.sql`	Creates a user and password, and sets default and temporary tablespaces. Grants privileges to that user. **Note:** Make a copy of this script and edit it to replace `<<WEBLOGIC>>` with the user name, password and tablespace names. Typical names for the main WebLogic Portal database are: • Database schema user = `WEBLOGIC` • Database password = `WEBLOGIC` • Default tablespace = `WEBLOGIC_DATA` • Temporary tablespace = `TEMP` **Note:** To use GroupSpace, you must create an additional user. Typical names for the GroupSpace repository database are: • Database schema user: `WEBLOGIC_GROUPSPACE` • Password: `WEBLOGIC_GROUPSPACE` • Default tablespace = `WEBLOGIC_GROUPSPACE_DATA` • Temporary tablespace = `TEMP` **Note:** If you decide to create an additional content management repository, you must create another schema for it. For additional information, see the Content Management Guide.
`rebuild_indexes.sql`	Rebuilds schema user (`WEBLOGIC`) indexes to move them from the table (`WEBLOGIC_DATA`) tablespace to the (`WEBLOGIC_INDEX`) tablespace.
`statistics.sql`	Runs `dbms_utility.analyze_schema` to compute database statistics needed for the Oracle optimizer. Update statistics periodically and whenever any significant changes in database data occur.
`install_report.sql`	Builds an informational installation report about the database objects created in the schema.
`dbsize.sql`	Builds a report showing free space in database tablespaces.

Table 6-1 Database Scripts and Usage Notes (Continued)

Script Name	Description
`bt_create_tablespaces.sql`	Creates the tablespace for behavior event tracking. **Note:** You must edit this script to modify the pathnames for the `EVT_DATA_PATHNAME` and `DATA_FILENAME` variables to reflect valid disk locations.
`bt_create_users.sql`	Creates a behavior event tracking schema user and password, and sets default and temporary tablespaces. Grants privileges to that user. **Note:** You can edit this script to change the schema user name, password and tablespace names. The following defaults are used: • Database user: `WEBLOGIC_EVENT` • Password: `WEBLOGIC_EVENT` • Default tablespace = `WEBLOGIC_EVENT_DATA` • Temporary tablespace = `TEMP`

Using WebLogic Portal with Oracle RAC

This section describes the steps necessary to use Oracle Real Application Cluster (RAC) 10g with WebLogic Portal.

Oracle Real Application Cluster is Oracle's database cluster technology. RAC offers scalability, high-availability and load-balancing at the database tier of a WebLogic Portal deployment. For more information on Oracle RAC please refer to the section "Using WebLogic Server with Oracle RAC" in the WebLogic Server document *Configuring and Managing WebLogic JDBC*.

The following sections address the various requirements and configuration choices when using Oracle RAC with WebLogic Portal. These include:

- Supported Configuration
- Configuration Considerations for Oracle RAC
- Procedure for Using Oracle RAC with WebLogic Portal
- Sample Configuration Files

Supported Configuration

WebLogic Portal 9.2 supports the following Oracle RAC and JDBC driver combinations:

Database Type	JDBC Driver	Notes
Oracle 10g R2 RAC (for Oracle 10.2.0.1 and later patch sets of 10.2.x)	• Oracle Thin Driver 10g • Oracle Thin/XA Driver 10g	Only WebLogic Portal 9.2 is supported with Oracle RAC 10g R2

For all the database configurations supported by WebLogic Platform 9.2, refer to "WebLogic Platform Support for Databases."

Configuration Considerations for Oracle RAC

For each JDBC data source used by WebLogic Portal, the following configuration choices need to be considered:

- Multi-Data Source vs. Oracle Thin Driver
- Fail-over vs. Load-Balancing
- XA Considerations
- Supported Configurations

Multi-Data Source vs. Oracle Thin Driver

There are two ways to implement JDBC fail-over and load-balancing: through the WebLogic multi-data source, or through the cluster-aware capabilities built into the Oracle Thin Driver. "Using WebLogic Server with Oracle RAC" includes examples of how these are implemented.

As pointed out in "Using WebLogic Server with Oracle RAC," there are some limitations if you are using the Oracle Thin Driver alone to access Oracle RAC. Specifically, you cannot use load-balancing together with XA support.

On the other hand, WebLogic multi-data source fully supports XA with load-balancing. Multi-data source also offers benefits such as faster fail-over and automatic fail-back. These make multi-data source the favorable choice to access RAC. The rest of the document is based on using multi-data source.

Fail-over vs. Load-Balancing

WebLogic Portal data sources support both fail-over and load-balancing with the exception of the `portalDataSourceNeverXA` data source. The `portalDataSourceNeverXA` is used as JDBC Store. As specified in "Using WebLogic Server with Oracle RAC," the data source of a JDBC Store cannot use the load-balancing algorithm.

XA Considerations

WebLogic Portal data sources can use either the XA or the non-XA JDBC driver, except `p13nDataSource` and `portalDataSourceNeverXA`. The `p13nDataSource` is used in local JDBC transactions only. The `portalDataSourceNeverXA` is used as JDBC Store. As specified in "Using WebLogic Server with Oracle RAC," the data source of a JDBC Store cannot use an XA-compliant JDBC driver.

When the multi-data source is used, it is important to make sure that the multi-data source's value of the global-transaction-protocol attribute is identical to those of all its underlying physical data sources. For a multi-data source, the value of global-transaction-protocol can be configured by either manually editing

```
${DOMAIN_HOME}/config/jdbc/${DATASOURCE_NAME}-jdbc.xml
```

or programmatically through WLST scripts. The WebLogic Administration Console does not provide a way to configure this attribute for a multi-data source.

Supported Configurations

In light of the above considerations, the supported RAC configurations are summarized in Table 6-2.

Table 6-2 WebLogic Portal Data Sources When Using Oracle RAC

Data Source Name	Multi-Data Source	MD-Protocol	XA-Driver	XA-Protocol
p13nDataSource	No		False	None
cgDataSource	Yes	Load-Balance	True	TwoPhaseCommit
cgDataSource-NonXA	No		False	None
portalDataSource	Yes	Load-Balance	True	TwoPhaseCommit
portalDataSourceAlwaysXA	Yes	Load-Balance	True	TwoPhaseCommit

Table 6-2 WebLogic Portal Data Sources When Using Oracle RAC

Data Source Name	Multi-Data Source	MD-Protocol	XA-Driver	XA-Protocol
portalDataSourceNeverXA	Yes	Failover	False	OnePhaseCommit
appsGroupSpaceDataSource	Yes	Load-Balance	True	TwoPhaseCommit
sampleDataSource	Yes	Load-Balance	True	TwoPhaseCommit

Note: The values in the above table are consistent with the JDBC data source configuration files packaged with WebLogic Portal, with the addition of the **Multi-Data Source** column.

Procedure for Using Oracle RAC with WebLogic Portal

The configuration steps for Oracle RAC are very similar to those described in the previous sections "Configuring the Main WebLogic Portal Database" on page 6-2, "Configuring the GroupSpace Database" on page 6-4, and "Manually Creating a Separate Database and Database Objects for Behavior Tracking" on page 6-5. Each of these three sections includes seven steps.

1. If you want to create new tablespaces for the main WebLogic Portal schema:

   Note: To run the SQL script in this step, log on to any one node of the RAC. The specific node is not important since all Oracle instances in the RAC share the same disk storage.

   a. Copy the `create_tablespaces.sql` script and modify it appropriately for your environment. See Table 6-1, "Database Scripts and Usage Notes," on page 6-7 and the comments in the script for additional information.

   b. Run the modified `create_tablespaces.sql` script. For example, from SQL*Plus:

   ```
   @create_tablespaces.sql
   ```

2. Create the main WebLogic Portal database user:

   Note: To run the SQL script in this step, log on to any one node of the RAC. The specific node is not important since all Oracle instances in the RAC share the same disk storage.

   a. Copy the `create_users.sql` script and modify it appropriately for your environment. See Table 6-1, "Database Scripts and Usage Notes," on page 6-7 and the comments in the script for additional information.

b. Run the modified `create_users.sql` script. For example, from SQL*Plus:

```
@create_users.sql
```

Follow the remaining steps only if you want to create database objects manually rather than using the Configuration Wizard. To perform the remaining steps using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

3. Open your domain's `database.properties` file for edit.
4. Set `database=oracle`.
5. Update the following settings (by replacing the @ symbols and the text between the symbols with the correct values) for your main WebLogic Portal database:

```
oracle.user=@DB_USER@
oracle.password=@DB_PASSWORD@
oracle.url=jdbc:bea:oracle://@DB_HOST@:@DB_PORT@;SID=@DB_NAME@
```

Note: In the `oracle.url` string, replace `@DB_HOST@` and `@DB_NAME@` with the hostname and SID of any one node in the RAC. Since all Oracle instances in the RAC share the same disk storage, the database objects created in step 6 will be visible from all other nodes in the RAC.

6. Create the database objects.
   a. Navigate to the *`BEA_HOME`*`\user_projects\domains\`*`myPortalDomain`* directory.
   b. Enter one of the commands:
      - On Windows, double-click `create_db.cmd`.
      - On UNIX, run `create_db.sh`.

      If any error messages are displayed, check the `create_db.log` file for additional information.
7. Replace the JDBC data sources in your domain (except `appsGroupSpaceDataSource`) with multi-data sources pointing to your Oracle RAC.
   a. For each JDBC data source, create one physical data source pointing to each node in the RAC. For example, for `p13nDataSource`, create physical data sources `p13nDataSource-1` to `p13nDataSource-N`, each pointing to one RAC node. Use the WebLogic Server Administration Console to create these data sources.

Alternatively, you could create them by copying and updating the sample `jdbc.xml` files in the `WL_HOME/portal/db/jdbc/oracle_thin` directory following the steps below:

1) Copy the `dataSourceName-jdbc.xml` file to `DOMAIN_HOME/config/jdbc` directory, and name it `dataSourceName-n-jdbc.xml`, where *n* is the sequence number of the *n*-th RAC node.

2) Edit the file `dataSourceName-n-jdbc.xml`. Replace `@DB_USER@`, `@DB_USER_PASSWD@`, `@DB_HOST@`, `@DB_PORT@` and `@DB_NAME@`. For `@DB_HOST@` and `@DB_NAME@`, use the hostname and SID of the *n*-th RAC node.

3) Edit `config.xml` to add this newly created JDBC data source as a JDBC system resource.

b. Create a multi-data source for each of the JDBC data sources, which encompasses the physical data sources created above. For example, create a `p13nDataSource` multi-data source which includes `p13nDataSource-1` to `p13nDataSource-N` created above.

If you use the WebLogic Server Administration Console, you would need to first remove the existing data source with the same name, and then create the multi-data source. Also, since the WebLogic Server Administration Console does not provide a way to configure the value of global-transaction-protocol attribute for a multi-data source, you need to manually add the attribute into the corresponding `jdbc.xml` later.

Alternatively, you could manually edit the corresponding `jdbc.xml` following the steps below:

1) Open the corresponding `jdbc.xml` file, delete the `jdbc-driver-params` element and the `jdbc-connection-pool-params` element, including all their children elements:

2) Between the `jndi-name` and the `global-transactions-protocol` elements, add the following two elements. Determine the value of algorithm-type from Table . The value of data-source-list is a comma-separated list of all the underlying physical data sources.

```
<algorithm-type>Load-Balancing</algorithm-type>
```

or

```
<algorithm-type>Failover</algorithm-type>
```

```
<data-source-list>portalDataSourceAlwaysXA-1,
portalDataSourceAlwaysXA-2 </data-source-list>
```

c. An alternative to the manual procedure described above in steps 1 and 2 is WLST scripting. The sample `build.xml`, `oraracconf.py` and `oraracconf.py.properties` provide an example of how to configure an existing WebLogic Portal domain to point to an Oracle RAC database. These files are listed in Appendix B, "Sample WLST Scripts." To use the sample, follow the steps below:

1) Download these files into your `DOMAIN_HOME` directory.

2) Open the file `oraracconf.py.properties`. Edit the values in the "Database usernames and passwords" and the "ORACLE RAC configuration" sections to reflect your Oracle RAC environment.

3) Open a command prompt, change current directory to `DOMAIN_HOME/bin`. Execute the following script:

. `./setDomainEnv.sh` (or `setDomainEnv.cmd`)

4) Execute `ant`.

Sample Configuration Files

Listing 6-1 p13nDataSource Multi-Data Source

```
<?xml version="1.0" encoding="UTF-8"?>
<jdbc-data-source xmlns="http://www.bea.com/ns/weblogic/90">
  <name>p13nDataSource</name>
  <jdbc-data-source-params>
    <jndi-name>p13n.trackingDataSource</jndi-name>
    <jndi-name>p13n.sequencerDataSource</jndi-name>
    <jndi-name>cm.sequencerDataSource</jndi-name>
    <jndi-name>p13n.leasemanager</jndi-name>
    <jndi-name>p13n.dataSyncDataSource</jndi-name>
    <jndi-name>p13n.entitlementsDataSource</jndi-name>
    <jndi-name>p13n.quiescenceDataSource</jndi-name>
    <algorithm-type>Load-Balancing</algorithm-type>
    <data-source-list>p13nDataSource-1,p13nDataSource-2</data-source-list>
    <global-transactions-protocol>None</global-transactions-protocol>
  </jdbc-data-source-params>
</jdbc-data-source>
```

Listing 6-2 p13nDataSource-1 Data Source

```
<?xml version="1.0" encoding="UTF-8"?>
<jdbc-data-source xmlns="http://www.bea.com/ns/weblogic/90">
  <name>p13nDataSource-1</name>
  <jdbc-driver-params>
    <url>jdbc:oracle:thin:@rnhp380-c11-23-vip:1521:DBSRAC101</url>
    <driver-name>oracle.jdbc.OracleDriver</driver-name>
    <properties>
      <property>
        <name>user</name>
        <value>WEBLOGIC_8</value>
      </property>
    </properties>
   <password-encrypted>{3DES}cIUMOgs5Divb+UWlIFgSoA==</password-encrypted>
  </jdbc-driver-params>
  <jdbc-connection-pool-params>
    <initial-capacity>5</initial-capacity>
    <max-capacity>20</max-capacity>
    <capacity-increment>1</capacity-increment>
    <test-connections-on-reserve>true</test-connections-on-reserve>
    <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
  </jdbc-connection-pool-params>
  <jdbc-data-source-params>
    <jndi-name>p13nDataSource-1</jndi-name>
    <global-transactions-protocol>None</global-transactions-protocol>
  </jdbc-data-source-params>
</jdbc-data-source>
```

Listing 6-3 p13nDataSource-2 Data Source

```
<?xml version="1.0" encoding="UTF-8"?>
<jdbc-data-source xmlns="http://www.bea.com/ns/weblogic/90">
  <name>p13nDataSource-2</name>
  <jdbc-driver-params>
    <url>jdbc:oracle:thin:@rnhp380-c11-25-vip:1521:DBSRAC102</url>
```

```
    <driver-name>oracle.jdbc.OracleDriver</driver-name>
    <properties>
      <property>
        <name>user</name>
        <value>WEBLOGIC_8</value>
      </property>
    </properties>
   <password-encrypted>{3DES}cIUMOgs5Divb+UWlIFgSoA==</password-encrypted>
  </jdbc-driver-params>
  <jdbc-connection-pool-params>
    <initial-capacity>5</initial-capacity>
    <max-capacity>20</max-capacity>
    <capacity-increment>1</capacity-increment>
    <test-connections-on-reserve>true</test-connections-on-reserve>
    <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
  </jdbc-connection-pool-params>
  <jdbc-data-source-params>
    <jndi-name>p13nDataSource-2</jndi-name>
    <global-transactions-protocol>None</global-transactions-protocol>
  </jdbc-data-source-params>
</jdbc-data-source>
```

CHAPTER 7

Using Sybase

This chapter describes the steps necessary to use a Sybase database with WebLogic Portal, and includes the following sections:

- Configuring Sybase Databases
- Configuring the Main WebLogic Portal Database
- Configuring the GroupSpace Database
- Manually Creating a Separate Database and Database Objects for Behavior Tracking
- Database Scripts and Usage Notes

Review this entire chapter and any release notes before proceeding. The tasks in this chapter should be performed by a database administrator.

Configuring Sybase Databases

Before proceeding, read "Overview of Enterprise-Quality Database Configuration for WebLogic Portal" on page 2-2.

The database creation scripts install domain-specific tables. It is recommended that you work with your database administrator to modify the sample scripts, and to create the database users and devices needed for your Sybase environment.

Multiple databases are required if you have multiple domains, or to run multiple environments using the same Sybase instance (for example, if you want to run your test environment and

enterprise-quality system from a single Sybase installation). GroupSpace requires a separate database, as do any additional content management repositories.

Notes: You must define a page size of at least 8K to support WebLogic Portal's use of wide tables, wide columns, and larger indexes. For Sybase the default page size is 2K, and Sybase does not allow rows to span pages. If your Sybase instance uses 2k or 4k pages, create a new Sybase instance with an 8K page size. Sybase provides a migration utility to migrate data between servers of different page sizes.

If a Sybase instance is defined with a page size smaller than 8K, the WebLogic Portal tables are created but warning messages may be issued at creation time, indicating that the row size could exceed the row size limit. These warnings may result in runtime exceptions, depending on the data being inserted or updated. Indexes will fail to create if they are larger than the maximum page size for the Sybase instance. This could result in data issues as well as performance problems.

To configure a Sybase database:

1. Be sure to back up your database before installing any new database objects. See your database vendor documentation for details.
2. Review the provided sample scripts, located in the *WL_HOME*`\portal\db\sybase\admin` directory. See Table 7-1, "Database Scripts and Usage Notes," on page 7-7 and the comments in the scripts for additional information.
3. Copy and modify the sample scripts appropriately for your environment to create each of the following databases:
   a. Follow the steps in "Configuring the Main WebLogic Portal Database" on page 7-2 to create the main WebLogic Portal database and database objects.
   b. If you want to use GroupSpace, follow the steps in "Configuring the GroupSpace Database" on page 7-4 to create the GroupSpace database and database objects.
   c. If you want to create a separate behavior tracking database, follow the steps in "Manually Creating a Separate Database and Database Objects for Behavior Tracking" on page 7-5.

Configuring the Main WebLogic Portal Database

To configure the main WebLogic Portal database, follow these steps:

1. Copy the `create_devices.sql` script and modify it appropriately for your environment. See Table 7-1, "Database Scripts and Usage Notes," on page 7-7 and the comments in the script for additional information.

2. Run the modified `create_devices.sql` as a user with system administrator privileges (normally the `sa` user). For example, using `isql`:

   ```
   isql -Usa -SMYSYBASE -e -icreate_devices.sql -ocreate_devices.log
   ```

3. Copy the `create_database.sql` script and modify it appropriately for your environment. See Table 7-1, "Database Scripts and Usage Notes," on page 7-7 and the comments in the script for additional information.

4. Run the modified `create_database.sql` as a user with system administrator privileges (normally the `sa` user). For example, using `isql`:

   ```
   isql -Usa -SMYSYBASE -e -icreate_database.sql -ocreate_database.log
   ```

   The output from running `create_database.sql` is written to `create_database.log`. Verify that there are no errors in the log file before proceeding.

Follow the remaining steps only if you want to create database objects manually rather than using the Configuration Wizard. To perform the remaining steps using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

5. Open your domain's `database.properties` file for edit.

6. Set `database=sybase`.

7. Update the following settings (by replacing the @ symbols and the text between the symbols with the correct values) for your main WebLogic Portal database:

   ```
   sybase.user=@DB_USER@

   sybase.password=@DB_PASSWORD@

   sybase.url=jdbc:bea:sybase://@DB_HOST@:@DB_PORT@;DatabaseName=@DB_NAME@
   ```

8. Create the database objects. You can use the Configuration Wizard or follow these steps:

   a. Navigate to the *BEA_HOME*`\user_projects\domains\`*myPortalDomain* directory.

   b. Enter one of the commands:

      - On Windows, double-click `create_db.cmd`.
      - On UNIX, run `create_db.sh`.

      If any error messages are displayed, check the `create_db.log` file for additional information.

9. Replace the JDBC data sources in your domain (except for `appsGroupSpaceDataSource`), which point to PointBase by default, with data sources that point to SQL Server. You can

configure them using the WebLogic Server Administration Console or choose from the samples provided and update them for your database environment. Sample `jdbc.xml` definition files for each database and driver that BEA supports are available in the *`WL_HOME`*`\portal\db\jdbc\`*`database_driver`* directory; for example, `sybase_bea`. Follow the instructions in the *`WL_HOME`*`\portal\db\jdbc\README.txt` file.

Configuring the GroupSpace Database

To configure the GroupSpace database, follow these steps:

1. Copy the `create_devices.sql` script and modify it appropriately for your environment. See Table 7-1, "Database Scripts and Usage Notes," on page 7-7 and the comments in the script for additional information.
2. Run the modified `create_devices.sql` as a user with system administrator privileges (normally the `sa` user). For example, using `isql`:

   ```
   isql -Usa -SMYSYBASE -e -icreate_devices.sql -ocreate_devices.log
   ```

3. Copy the `create_database.sql` script and modify it appropriately for your environment. See Table 7-1, "Database Scripts and Usage Notes," on page 7-7 and the comments in the script for additional information.
4. Run the modified `create_database.sql` as a user with system administrator privileges (normally the `sa` user). For example, using `isql`:

   ```
   isql -Usa -SMYSYBASE -e -icreate_database.sql -ocreate_database.log
   ```

   The output from running `create_database.sql` is written to `create_database.log`. Verify that there are no errors in the log file before proceeding.

Follow the remaining steps only if you want to create database objects manually rather than using the Configuration Wizard. To perform the remaining steps using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

5. Open your domain's `groupspace_database.properties` file for edit.
6. Set `database=sybase`.
7. Update the following settings (by replacing the @ symbols and the text between the symbols with the correct values) for your GroupSpace database:

   ```
   sybase.user=@DB_USER@
   sybase.password=@DB_PASSWORD@
   sybase.url=jdbc:bea:sybase://@DB_HOST@:@DB_PORT@;DatabaseName=@DB_NAME@
   ```

8. Create the database objects.

   a. Navigate to the *BEA_HOME*\user_projects\domains*myPortalDomain* directory.

   b. Enter one of the commands:

   – On Windows:

   ```
   create_db.cmd -database.properties=groupspace_database.properties
   ```

   – On UNIX:

   ```
   create_db.sh -database.properties=groupspace_database.properties
   ```

   If any error messages are displayed, check the `create_db_groupspace.log` file for additional information.

9. Replace the `appsGroupSpaceDataSource` JDBC data source (which points to PointBase by default) with a data source that points to Sybase. Use the WebLogic Server Administration Console or update the sample `jdbc.xml` definition file provided for each database and driver that BEA supports, in the *WL_HOME*\portal\db\jdbc*database_driver* directory; for example `sybase_bea`. Follow the instructions in the *WL_HOME*\portal\db\jdbc\README.txt file.

Manually Creating a Separate Database and Database Objects for Behavior Tracking

For improved performance, you might want to store behavior tracking events in a different location from other WebLogic Portal database objects. For more information about behavior tracking, see *Setting Up Events and Behavior Tracking* in the Interaction Management Guide.

Note: By default, behavior tracking database objects are created in the same database as other WebLogic Portal database objects. You need to perform these steps only if you are configuring a separate database for behavior tracking events.

To create a separate database for behavior tracking:

1. Edit the `bt_create_devices.sql` file and the `bt_create_database.sql` file for your environment, as indicated in the instructions contained in the scripts and in Table 7-1, "Database Scripts and Usage Notes," on page 7-7.

2. Run `bt_create_devices.sql` as a user with system administrator privileges. For example, using `isql`:

   ```
   isql -Usa -SMYSYBASE -e -ibt_create_devices.sql
   -obt_create_devices.log
   ```

3. Run `bt_create_database.sql` as a user with system administrator privileges. For example, using `isql`:

   ```
   isql -Usa -SMYSYBASE -e -ibt_create_database.sql
   -obt_create_database.log
   ```

4. Navigate to the appropriate database directory based on your environment; for example, *`WL_HOME`*`\portal\db\sybase`.

5. Connect as the user `WEBLOGIC_EVENT` and run the following scripts:

   - `bt_create_tables.sql`
   - `bt_create_fkeys.sql`
   - `bt_create_indexes.sql`
   - `bt_create_triggers.sql`

6. Run the following scripts from the path *`WL_HOME`*`\portal\db\data\required`:

   - `bt_insert_system_data.sql`
   - `bt9_insert_system_data.sql`

7. Use the WebLogic Server Administration Console to configure a non-XA JDBC data source to access your behavior tracking database. Associate the JNDI name `p13n.trackingDataSource` with that data source and then remove `p13n.trackingDataSource` from `p13nDataSource`.

Database Scripts and Usage Notes

Table 7-1 describes the scripts that enable you to configure the Sybase database.

Table 7-1 Database Scripts and Usage Notes

Script Name	Description
`create_devices.sql`	Creates database devices. Database devices must be created by a user with system administrator privileges (normally the `sa` user). 1. Make a copy of this script and edit it to replace `<<WEBLOGIC>>` with the appropriate database name and device names for each database you create. 2. Edit the script to reflect valid disk locations for `D:\DATAFILE`, `E:\LOGFILE`, and `F:\INDEXFILE` device locations. 3. Modify file sizes as needed depending on the WebLogic Portal functionality you are using. The following values for a production system are provided as a general guide, but you can adjust these values as needed (for example, an individual developer's database can have smaller files): log segment: 2048mb data segment: 2048mb index segment: 500mb Optimally, locate data, log, and index devices on separate physical disks (with separate controllers) and away from any system database files, unless you are using RAID devices. **Note:** Do not change the name of the `WEBLOGIC_INDEX` index segment. Typical device names for the main WebLogic Portal database are: • Data device: `WEBLOGIC_DATA` • Log device: `WEBLOGIC_LOG` Typical device names for the GroupSpace database are: • Data device: `WEBLOGIC_GROUPSPACE_DATA` • Log device: `WEBLOGIC_GROUPSPACE_LOG`

Table 7-1 Database Scripts and Usage Notes (Continued)

Script Name	Description
`create_database.sql`	Creates the database and login. You must create a main WebLogic Portal database. If you want to use GroupSpace, you must create that database. You must also create a database for each additional content management repository. The devices created by `create_devices.sql` are used and indexes are placed in their own `WEBLOGIC_INDEX` segment. An alias is added to the database owner (dbo) user of the database. Make a copy of this script and edit it to replace `<<WEBLOGIC>>` with the appropriate database name, database user, and password for each database you create. Also adjust database and log sizes as appropriate. If the database you are creating is a development database (and therefore database recovery is not a concern), you can uncomment and set the `truncate log on checkpoint` option to `true`. If your applications will use WebLogic Workshop for create database or RowSet controls, uncomment and set the `DDL in transaction` option to `true`; otherwise `create` commands will not work properly. Typical names for the main WebLogic Portal database are: • Database name: `WEBLOGIC` • Database schema user: `WEBLOGIC` • Password: `WEBLOGIC` **Note:** Do not change the name of the `WEBLOGIC_INDEX` index segment. When you run the script with these values, it creates the `WEBLOGIC` database, the `WEBLOGIC_INDEX` file group, and `WEBLOGIC` database owner (dbo) user login. An alias is created to make `WEBLOGIC` the dbo user in the database. It also sets the `WEBLOGIC` database as the default database for the `WEBLOGIC` user. To use GroupSpace, you must create an additional database. Typical names for the GroupSpace database are: • Database name: `WEBLOGIC` • Database schema user: `WEBLOGIC` • Password: `WEBLOGIC` **Note:** If you decide to create an additional content management repository, you must create a database with different user names for it. See the Content Management Guide.

Table 7-1 Database Scripts and Usage Notes (Continued)

Script Name	Description
statistics_build.sql	Builds statistics.sql to update table and index statistics for the database optimizer. Update statistics periodically and whenever any significant changes in database data occur.
install_report_build.sql install_report_static.sql	Builds an informational installation report about the database objects created by the WEBLOGIC user.
bt_create_devices.sql	Creates behavior tracking database devices. Database devices must be created by a user with system administrator privileges (normally the sa user). You must also edit the script to reflect valid disk locations for D:\DATAFILE, E:\LOGFILE, and F:\INDEXFILE device locations; you may also need to modify file sizes. Optimally, locate data, log, and index devices on separate physical disks (with separate controllers) and away from any system database files, unless you are using RAID devices. The following default names are used: • Data device: WEBLOGIC_EVENT_DATA • Log device: WEBLOGIC_EVENT_LOG
bt_create_database.sql	Create the WEBLOGIC_EVENT database and WEBLOGIC_EVENT database owner user login. An alias is created to make WEBLOGIC_EVENT the database owner (dbo) user in the database. Modify the script to reflect devices names bt_create_devices.sql. Also adjust database and log sizes for your site specific needs. Put DATA and LOG files on separate physical disks and away from any system database files. The following defaults are used: • Database name: WEBLOGIC_EVENT • Database owner user: WEBLOGIC_EVENT • Password: WEBLOGIC_EVENT

CHAPTER 8

Using DB2

This chapter describes the steps necessary to use a DB2 database with WebLogic Portal, and includes the following sections:

- Configuring DB2 Databases
- Configuring the Main WebLogic Portal Database
- Configuring the GroupSpace Database
- Manually Creating a Separate Database and Database Objects for Behavior Tracking
- Database Scripts and Usage Notes

Review this entire chapter and any release notes before proceeding. The tasks in this chapter should be performed by a database administrator.

Configuring DB2 Databases

Before proceeding, read "Overview of Enterprise-Quality Database Configuration for WebLogic Portal" on page 2-2.

The database creation scripts install domain-specific tables. It is recommended that you work with a database administrator to modify the sample scripts, and to create the database users and tablespaces needed for your environment.

Multiple database schemas are required if you have multiple domains, or to run multiple environments using the same DB2 instance (for example, if you want to run development and

system test from a single DB2 installation). GroupSpace requires a separate user, as do any additional content management repositories.

Note: To ensure that portal applications run correctly on DB2, you must set some minimum configuration parameters. If you do not, heavy portal activity might exceed database capacity. Use the following settings as guidelines to configure your DB2 database:

- Dynamic Sections: 20,000
- applheapsz: 24,000
- pckcachesz: 2,500

To configure a DB2 database:

1. Be sure to back up your database before installing any new database objects. See your database documentation for details.
2. Review the provided sample scripts, located in the *WL_HOME*`/portal/db/oracle/admin` directory. See Table 8-1, "Database Scripts and Usage Notes," on page 8-8 and the comments in the scripts for additional information.
3. Copy and modify the sample scripts appropriately for your environment to create each of the following database schemas:
   a. Follow the steps in "Configuring the Main WebLogic Portal Database" on page 8-2 to create the main WebLogic Portal user and database objects.
   b. If you want to use GroupSpace, follow the steps in "Configuring the GroupSpace Database" on page 8-4 to create the GroupSpace user and database objects.
   c. If you want to create a separate behavior tracking database, follow the steps in "Manually Creating a Separate Database and Database Objects for Behavior Tracking" on page 8-6.

Configuring the Main WebLogic Portal Database

To configure the main WebLogic Portal database, follow these steps:

1. If you do not already have an 8K buffer pool to use:
   a. Copy the `create_bufferpool.sql` script and modify it appropriately for your environment. See Table 8-1, "Database Scripts and Usage Notes," on page 8-8 and the comments in the script for additional information.
   b. Connect to the database that you want to work with. For example, from the CLP:

```
Db2 connect to database user username password password
```

c. Run `create_bufferpool.sql`. For example, from the CLP:

```
Db2 -tf create_bufferpool.sql -v
```

d. Restart your database instance.

2. If you do not have an existing an 8K temporary tablespace, or you want to a create new 8K temporary tablespace:

   a. Copy the `create_temp_tablespace.sql` script and modify it appropriately for your environment. See Table 8-1, "Database Scripts and Usage Notes," on page 8-8 and the comments in the script for additional information.

   b. Run `create_temp_tablespaces.sql`. For example, from the CLP:

```
Db2 -tf create_temp_tablespaces.sql -v
```

3. If you do not have existing 4K and 8K regular tablespaces, or you want to a create new regular tablespaces:

   a. Copy the `create_tablespace.sql` script and modify it appropriately for your environment. See Table 8-1, "Database Scripts and Usage Notes," on page 8-8 and the comments in the script for additional information.

   b. Run `create_tablespaces.sql`. For example, from the CLP:

```
Db2 -tf create_tablespaces.sql -v
```

4. Create the main WebLogic Portal database user:

   a. Copy the `create_users.sql` script and modify it appropriately for your environment. See Table 8-1, "Database Scripts and Usage Notes," on page 8-8 and the comments in the script for additional information.

   b. Run `create_user.sql`. For example, from the CLP:

```
Db2 -tf create_user.sql -v
```

Follow the remaining steps only if you want to create database objects manually rather than using the Configuration Wizard. To perform the remaining steps using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

5. Open your domain's `database.properties` file for edit.

6. Set `database=db2`.

7. Update the following settings (by replacing the @ symbols and the text between the symbols with the correct values) for your main WebLogic Portal database:

   ```
   db2.user=@DB_USER@
   db2.password=@DB_PASSWORD@
   db2.url=jdbc:bea:db2://@DB_HOST@:@DB_PORT@;DatabaseName=@DB_NAME@
   ```

8. Create the database objects.

   a. Navigate to the *BEA_HOME*\user_projects\domains*myPortalDomain* directory.

   b. Enter one of the commands:

      - On Windows, double-click `create_db.cmd`.
      - On UNIX, run `create_db.sh`.

      If any error messages are displayed, check the `create_db.log` file for additional information.

9. Replace the JDBC data sources in your domain (except for `appsGroupSpaceDataSource`), which point to PointBase by default, with data sources that point to DB2. You can configure them using the WebLogic Server Administration Console or choose from the samples provided and update them for your database environment. Sample `jdbc.xml` definition files for each database and driver that BEA supports are available in the *WL_HOME*\portal\db\jdbc*database_driver* directory; for example, `db2_bea`. Follow the instructions in the *WL_HOME*\portal\db\jdbc\README.txt file.

Configuring the GroupSpace Database

To configure the GroupSpace database, follow these steps:

1. If you do not already have an 8K buffer pool to use:

   a. Copy the `create_bufferpool.sql` script and modify it appropriately for your environment. See Table 8-1, "Database Scripts and Usage Notes," on page 8-8 and the comments in the script for additional information.

   b. Connect to the database that you want to work with. For example, from the CLP:

      ```
      Db2 connect to database user username password password
      ```

   c. Run `create_bufferpool.sql`. For example, from the CLP:

      ```
      Db2 -tf create_bufferpool.sql -v
      ```

d. Restart your database instance.

2. If you do not have an existing an 8K temporary tablespace, or you want to a create new 8K temporary tablespace:

   a. Copy the `create_temp_tablespace.sql` script and modify it appropriately for your environment. See Table 8-1, "Database Scripts and Usage Notes," on page 8-8 and the comments in the script for additional information.

   b. Run `create_temp_tablespaces.sql`. For example, from the CLP:

      ```
      Db2 -tf create_temp_tablespaces.sql -v
      ```

3. If you do not have existing 4K and 8K regular tablespaces, or you want to a create new regular tablespaces:

   a. Copy the `create_tablespace.sql` script and modify it appropriately for your environment. See Table 8-1, "Database Scripts and Usage Notes," on page 8-8 and the comments in the script for additional information.

   b. Run `create_tablespaces.sql`. For example, from the CLP:

      ```
      Db2 -tf create_tablespaces.sql -v
      ```

4. Create the GroupSpace user:

   a. Copy the `create_users.sql` script and modify it appropriately for your environment and the GroupSpace settings. See Table 8-1, "Database Scripts and Usage Notes," on page 8-8 and the comments in the script for additional information.

   b. Run `create_user.sql`. For example, from the CLP:

      ```
      Db2 -tf create_user.sql -v
      ```

Follow the remaining steps only if you want to create database objects manually rather than using the Configuration Wizard. To perform the remaining steps using the Configuration Wizard, see *Creating WebLogic Configurations Using the Configuration Wizard*.

5. Open your domain's `groupspace_database.properties` file for edit.

6. Set `database=db2`.

7. Update the following settings (by replacing the @ symbols and the text between the symbols with the correct values) for your GroupSpace database:

   ```
   db2.user=@DB_USER@
   ```

   ```
   db2.password=@DB_PASSWORD@
   ```

```
db2.url=jdbc:bea:db2://@DB_HOST@:@DB_PORT@;DatabaseName=@DB_NAME@
```

8. Create the database objects.

   a. Navigate to the *BEA_HOME*\user_projects\domains*myPortalDomain* directory.

   b. Enter one of the commands:

      - On Windows:

      ```
      create_db.cmd -database.properties=groupspace_database.properties
      ```

      - On UNIX:

      ```
      create_db.sh -database.properties=groupspace_database.properties
      ```

      If any error messages are displayed, check the `create_db_groupspace.log` file for additional information.

9. Replace the `appsGroupSpaceDataSource` JDBC data source (which points to PointBase by default) with a data source that points to DB2. Use the WebLogic Server Administration Console or update the sample `jdbc.xml` definition file provided for each database and driver that BEA supports, in the *WL_HOME*\portal\db\jdbc*database_driver* directory; for example `db2_bea`. Follow the instructions in the *WL_HOME*\portal\db\jdbc\README.txt file.

Manually Creating a Separate Database and Database Objects for Behavior Tracking

For improved performance, you might want to store behavior tracking events in a different location from other WebLogic Portal database objects. For more information about behavior tracking, see *Setting Up Events and Behavior Tracking* in the Interaction Management Guide.

Note: By default, behavior tracking database objects are created in the same database as other WebLogic Portal database objects. You need to perform these steps only if you are configuring a separate database for behavior tracking events.

To create a separate database for behavior tracking:

1. Edit the `bt_create_tablespaces.sql` file and the `bt_create_users.sql` file for your environment, as indicated in the instructions contained in the scripts and in Table 8-1, "Database Scripts and Usage Notes," on page 8-8.

2. From the CLP, run the `bt_create_tablespaces.sql` script. For example:

   ```
   Db2 -tf bt_create_tablespaces.sql -v
   ```

3. From the CLP, run the `bt_create_users.sql` script. For example:

   ```
   Db2 -tf bt_create_users.sql -v
   ```

4. Navigate to the appropriate database directory based on your environment: *`WL_HOME`*`\portal\db\db2`

5. Connect as the user `WEBLOGIC_EVENT` and run the following scripts:

   - `bt_create_tables.sql`
   - `bt_create_fkeys.sql`
   - `bt_create_indexes.sql`
   - `bt_create_triggers.sql`

6. Run the following scripts from the path *`WL_HOME`*`\portal\db\data\required`:

   - `bt_insert_system_data.sql`
   - `bt9_insert_system_data.sql`

7. Use the WebLogic Server Administration Console to configure a non-XA JDBC data source to access your behavior tracking database. Associate the JNDI name `p13n.trackingDataSource` with that data source and then remove `p13n.trackingDataSource` from `p13nDataSource`.

Database Scripts and Usage Notes

Table 8-1 describes the scripts that enable you to configure the Oracle database and create WebLogic Portal objects in that database.

Table 8-1 Database Scripts and Usage Notes

Script Name	Description
`create_user.sql`	Grants DB2 `createtab`, `bindadd`, and `connect` privileges to the schema owner user. Because IBM DB2 databases authenticate users using the operating system, you must create an operating system user that owns database schema objects. **Note:** Make a copy of this script and edit it to replace `<<WEBLOGIC>>` with the schema user name. For the main WebLogic Portal database, the schema user name is typically `WEBLOGIC`. **Note:** To use GroupSpace, you must create an additional user. For the GroupSpace database, the schema user name is typically `WEBLOGIC_GROUPSPACE`. **Note:** If you decide to create an additional content management repository, you must create a schema with different user names for it. For additional information, see the Content Management Guide.
`create_bufferpool.sql`	Creates an 8K buffer pool, if one does not already exist. You must stop and restart DB2 to utilize new buffer pools. Edit the script to change the 8K buffer pool name. The default buffer pool name is `BP8K`.

Table 8-1 Database Scripts and Usage Notes (Continued)

Script Name	Description
`create_tablespaces.sql`	Creates 4K and 8K regular tablespaces. If you want to use existing regular tablespaces of the correct sizes, you do not need to run this script. You can use new or existing tablespaces for the main WebLogic Portal database, the GroupSpace database, and any additional content management repositories. **Note:** Make a copy of this script and edit it to replace `<<WEBLOGIC>>` with the appropriate tablespace and user names. You must also edit the script to reflect valid disk locations for your environment. Typical names for the main WebLogic Portal database are: • Tablespace for tables with a rowsize smaller than 4K: `WEBLOGIC_DATA_4K` • Tablespace for tables with a rowsize larger than 4K and smaller than 8K: `WEBLOGIC_DATA_8K` • Database user: `WEBLOGIC` Typical names for the main GroupSpace database are: • Tablespace for tables with a rowsize smaller than 4K: `WEBLOGIC_GROUPSPACE_DATA_4K` • Tablespace for tables with a rowsize larger than 4K and smaller than 8K: `WEBLOGIC_GROUPSPACE_DATA_8K` • Database user: `WEBLOGIC_GROUPSPACE`
`create_temp_tablespaces.sql`	Creates an 8K temporary tablespace. If you want to use an existing temporary tablespace of the correct size, you do not need to run this script. You can use new or existing tablespaces for the main WebLogic Portal database, the GroupSpace database, and any additional content management repositories. The default temporary tablespace name is `TEMPSPACE_8K`. **Note:** Edit this script to use the correct buffer pool name. You must also edit the script to reflect valid disk locations for your environment.
`statistics_build.sql`	Builds a file of `runstats` commands for each table that will compute database statistics needed for the database optimizer. Update statistics periodically and whenever any significant changes in database data occur.

Table 8-1 Database Scripts and Usage Notes (Continued)

Script Name	Description
`install_report.sql`	Builds an informational installation report about the database objects created in the `WEBLOGIC` schema.
`bt_create_tablespace.sql`	Creates the `WEBLOGIC_EVENT_DATA` tablespace. **Note:** Edit the script to specify valid physical disk locations for your environment (*`event_container`*), and to use a buffer pool other than the DB2 default buffer pool.
`bt_create_users.sql`	Creates the `WEBLOGIC_EVENT` schema owner user and grants DB2 `createtab`, `bindadd`, and `connect` privileges to the schema owner user. Because IBM DB2 databases authenticate users using the operating system, you must create an operating system user that owns database schema objects. Edit the script to change the schema user name and system administrator password. The default schema owner user name is `WEBLOGIC_EVENT`.

CHAPTER 9

Data Dictionary

This chapter describes the database objects for each component of WebLogic Portal. The information in this section is collectively known as the data dictionary.

For each component of WebLogic Portal, the following information is provided:

- An entity-relationship diagram
- A detailed description of each database table, including:
  - **Table Name** – The predefined name for the Table.
  - **Table Description** – A detailed description of the contents and purpose for the table in WebLogic Portal database schema.
  - **Column Name** – The predefined name for the column.
  - **Data Type** – The predefined characteristics for the column.

    Data types vary slightly by DBMS. For instance, columns defined as BLOB data types in Oracle, DB2, and PointBase would be defined as TEXT columns in Microsoft SQL Server and Sybase.
  - **Null Value** – Indicates whether or not null values can be stored for the column.
  - **Column Description** – A detailed description of the contents and purpose for the column including Primary Key (PK-) and Foreign Key (FK-) designations.

    Note: The term *hint* in the descriptions refers to available capabilities that are not supported in the default skeletons provided with WebLogic Portal.

Change bars in the left column indicate tables or table rows that are new or changed in WebLogic Portal version 9.2.

This chapter contains the following sections:

- Personalization Database Objects
- Portal Services Database Objects
- Portal Framework Database Objects
- Communities Framework Database Objects
- Content Management Database Objects
- Content Management Virtual Database Objects

Note: The appendix "Scripts and Properties Files" on page A-1 identifies the filenames and location of DDL (database definition language) files for each set of WebLogic Portal database objects.

Personalization Database Objects

Personalization enables you to develop, manage, and measure personalized portal applications. These functions help you target content to a desired audience. This includes the tracking of anonymous users and behavior tracking.

Click to view, and use the magnifying tool to inspect, individual tables in the entity-relation diagram for the Personalization database objects.

Base Personalization Database Tables

The following tables support basic personalization functionality:

- USER_PROFILE Database Table
- ENTITY Database Table
- PROPERTY_KEY Database Table
- PROPERTY_VALUE Database Table
- SEQUENCER Database Table
- WEBLOGIC_IS_ALIVE Database Table

USER_PROFILE Database Table

This table associates users with profiles (such as the `WLCS_CUSTOMER` user profile). User profiles use property sets to organize the properties that they contain.

Table 9-1 USER_PROFILE Table Metadata

Column Name	Data Type	Null Value	Description
USER_NAME	VARCHAR(200)	Not Null	PK – The user name.
PROFILE_TYPE	VARCHAR(100)	Not Null	A type of profile associated with the user (such as `WLCS_Customer`).
CREATION_DATE	DATE	Not Null	The date and time this record was created.

ENTITY Database Table

Some objects in WebLogic Portal implement a Java interface called ConfigurableEntity. Any ConfigurableEntity within the system has an entry in this table.

Table 9-2 ENTITY Table Metadata

Column Name	Data Type	Null Value	Description
ENTITY_ID	NUMBER(15)	Not Null	PK – A unique, sequence-generated number used as the record identifier.
ENTITY_NAME	VARCHAR(200)	Not Null	The name of the ConfigurableEntity.
ENTITY_TYPE	VARCHAR(100)	Not Null	Defines the type of ConfigurableEntity.
CREATION_DATE	DATE	Not Null	The date and time this record was created.
MODIFIED_DATE	DATE	Not Null	The date and time this record was last modified.

PROPERTY_KEY Database Table

Any property assigned to a ConfigurableEntity has a unique `PROPERTY_ID`. The identifier and associated information is stored in the following table.

Table 9-3 PROPERTY_KEY Table Metadata

Column Name	Data Type	Null Value	Description and Recommendations
PROPERTY_KEY_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number used as the record identifier.
PROPERTY_NAME	VARCHAR(100)	Not Null	The property name.
CREATION_DATE	DATE	Not Null	The date and time this record was created.
MODIFIED_DATE	DATE	Not Null	The date and time this record was last modified.
PROPERTY_SET_NAME	VARCHAR(100)	Null	The name of the property set.
PROPERTY_SET_TYPE	VARCHAR(100)	Null	The type the property set.

PROPERTY_VALUE Database Table

This table stores property values for boolean, datetime, float, integer, text, and user-defined properties.

Table 9-4 PROPERTY_VALUE Table Metadata

Column Name	Data Type	Null Value	Description
PROPERTY_VALUE_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number used as the record identifier.
PROPERTY_KEY_ID	NUMBER(15)	Not Null	FK to `PROPERTY_KEY.PROPERTY_KEY_ID`
ENTITY_ID	NUMBER(15)	Not Null	FK to `ENTITY.ENTITY_ID`
PROPERTY_TYPE	NUMBER(1)	Not Null	Valid entries are: 0=Boolean, 1=Integer, 2=Float, 3=Text, 4=Date and Time, 5=User-Defined (BLOB).
CREATION_DATE	DATE	Not Null	The date and time this record was created.
MODIFIED_DATE	DATE	Not Null	The date and time this record was last modified.

Table 9-4 PROPERTY_VALUE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
BOOLEAN_VALUE	NUMBER(1)	Null	The value for each boolean property identifier.
DATETIME_VALUE	DATE	Null	The value for each date and time property identifier.
DOUBLE_VALUE	NUMBER	Null	The value associated with each float property identifier.
LONG_VALUE	NUMBER(20)	Null	The value associated with the integer property.
TEXT_VALUE	VARCHAR(254)	Null	The value associated with the text property.
BLOB_VALUE	BLOB	Null	The value associated with the user-defined property.

SEQUENCER Database Table

The `SEQUENCER` table maintains all of the sequence identifiers (for example, property_meta_data_id_sequence, and so on) used in the application.

Table 9-5 SEQUENCER Table Metadata

Column Name	Data Type	Null Value	Description
SEQUENCE_NAME	VARCHAR(50)	Not Null	PK – A unique name used to identify the sequence.
CURRENT_VALUE	NUMBER(15)	Not Null	The current value of the sequence.
IS_LOCKED	NUMBER(1)	Not Null	This flag identifies whether or not the particular SEQUENCE_ID has been locked for update, and has a value of 0 or 1. This column is used as a generic locking mechanism that can be used for multiple database environments.

WEBLOGIC_IS_ALIVE Database Table

This table is used by the JDBC connection pools to ensure that the connection to the database is still alive.

Table 9-6 WEBLOGIC_IS_ALIVE Table Metadata

Column Name	Data Type	Null Value	Description
NAME	VARCHAR(100)	Not Null	Used by the JDBC connection pools to ensure that the connection to the database is still alive.

P13N_QUIESCENT_STATE Database Table

This table was added in WebLogic Portal version 9.2. It allows an Administration Console user or visitor tools user to disable modifications to portal resources, and allows you to send multicast messages to a member of a cluster.

Table 9-7 P13N_QUIESCENT_STATE Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
ENTAPP_NAME	VARCHAR(255)	Not Null	PK – Name of the enterprise application.	9.2
WEBAPP_NAME	VARCHAR(255)	Not Null	PK – Name of the web application.	9.2
RESOURCE_NAME	VARCHAR(255)	Not Null	PK – The name of an entitled or delegated resource; for example, CONTENT, PORTAL.	9.2
CAPABILITY_NAME	VARCHAR(80)	Not Null	PK – The name of allowed operations/capabilities; anything that allows you to the manipulate the system.	9.2

Table 9-7 P13N_QUIESCENT_STATE Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
MODE_NAME	VARCHAR(80)	Not Null	The name of the quiescence mode; for example, BLOCKED.	9.2
MODE_ENABLED_USER_NAME	VARCHAR(200)	Not Null	The name of the person who turned on quiescence for DT, or who set the BLOCKED mode for the resource.	9.2

P13N_LEASE Database Table

This table was added in WebLogic Portal version 9.2. It is used internally to maintain an exclusive lease request across any application connecting to the same database, in order to prevent multiple conflicting processes. The values in the table are transient and rows in the table should exist only as long as the lease is of value. If for some reason the row is not deleted automatically, WebLogic Portal will update any row as needed when another request with the same name occurs.

The following examples show how this lease data is used:

- A WebLogic Portal lease manager locks a resource for startup/update for a specified period of time, and a propagation is running on one node in the cluster. If another node attempts to start a propagation, the lease prevents the second process from beginning at that time.
- A WebLogic Portal lease manager uses this table to synchronize distribution of portal framework data within a cluster.

This table is used for internal processing only and is not accessible using the WebLogic Portal Administration Console.

Table 9-8 P13N_LEASE Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
LEASE_NAME	VARCHAR(80)	Not Null	PK – Unique name of the lease requested.	9.2
CREATION_DATE	TIMESTAMP	Not Null	Date and time the lease was requested. The default value is the current timestamp.	9.2
EXPIRATION_DATE	TIMESTAMP	Not Null	Date and time the lease will expire	9.2
SERVER_NAME	VARCHAR(80)	Not Null	Name of the server requesting the lease.	9.2

Tracked Anonymous User Dictionary Tables

This section documents the database objects for the WebLogic Portal package. For more information about anonymous users, refer to the *User Management Guide*.

The following tables support tracking of anonymous users:

- P13N_ANONYMOUS_PROPERTY Database Table
- P13N_ANONYMOUS_USER Database Table

P13N_ANONYMOUS_PROPERTY Database Table

This table stores the properties associated with the tracked anonymous user.

Table 9-9 P13N_ANONYMOUS_PROPERTY Table Metadata

Column Name	Data Type	Null Value	Description
ANONYMOUS_PROPERTY_ID	VARCHAR(128)	Not Null	PK – A unique, system-generated number to use as the record ID.

Table 9-9 P13N_ANONYMOUS_PROPERTY Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
PROPERTY_SET_NAME	VARCHAR(100)	Not Null	The name of the property set for which the tracked anonymous user data is set.
PROPERTY_NAME	VARCHAR(100)	Not Null	The name of the property being tracked for the anonymous user.
ANONYMOUS_USER_ID	VARCHAR(128)	Not Null	FK - A foreign key to P13N_ANONYMOUS_USER.
PROPERTY_VALUE	BLOB	Not Null	The value of the property that must implement `java.io.Serializable`.

P13N_ANONYMOUS_USER Database Table

This table stores the tracked anonymous user data.

Table 9-10 P13N_ANONYMOUS_USER Table Metadata

Column Name	Data Type	Null Value	Description
ANONYMOUS_USER_ID	VARCHAR(128)	Not Null	FK – maps to the primary key of the anonymous user.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
LAST_VISIT_DATE	DATE	Null	Date when the tracked anonymous user last updated the data.

Behavior Tracking Database Tables

Each DBMS-specific chapter of this guide describes how to set up a separate behavior tracking database. Three tables are provided for behavior tracking data. The BT_EVENT table stores all event data. The BT_EVENT_ACTION table logs actions used by third-party vendors against the recorded event data, and the BT_EVENT_TYPE table references event types and categories in the EVENT table.

For more information about behavior tracking, refer to the *Interaction Management Guide*.

BT_EVENT_TYPE Database Table

This table references event types and categories in the BT_EVENT table. This table is static.

Table 9-11 BT_EVENT_TYPE Database Table

Column Name	Data Type	Null Value	Description
EVENT_TYPE	VARCHAR(30)	Not Null	PK - A unique, system-generated number used as the record ID.
EVENT_GROUP	VARCHAR(10)	Not Null	The event category group associated with the event type.
DESCRIPTION	VARCHAR(50)	Null	A description of the EVENT_TYPE.

To record custom events, you must create an entry in this table. If a custom event does not have a record in this table, you cannot persist it to the BT_EVENT table.

BT_EVENT Database Table

This table stores all behavior tracking event data.

Table 9-12 BT_EVENT Database Table

Column Name	Data Type	Null Value	Description
EVENT_ID	NUMBER	Not Null	PK - A unique, system-generated number used as the record ID.
APPLICATION	VARCHAR (30)	Not Null	The application that created the event.
EVENT_TYPE	VARCHAR(30)	Not Null	FK - Set to BT_EVENT_TYPE. A string identifier showing which event was fired.
EVENT_DATE	DATE	Not Null	The date and time of the event.
WLS_SESSION_ID	VARCHAR(254)	Not Null	A unique, WebLogic Server-generated number assigned to the session.
XML_DEFINITION	CLOB	Null	An XML document that contains the specific event information for each event type. It is stored as a CLOB (Character Large Object). See Table 9-13.
USER_ID	VARCHAR(50)	Null	The user ID associated with the session and event. If the user has not logged in this column is null.
ANONYMOUS_USER_ID	VARCHAR(128)	Null	The user ID of the anonymous user associated with the session and event, if applicable.

As shown in Table 9-12, the BT_EVENT table has eight columns; each column corresponds to a specific event element. Seven of the BT_EVENT table's columns contain data common to every event type. The XML_DEFINITION column contains information about the application, event-date, event-type, session-id, and user-id, plus event data that is unique to each event type. An XML document is created specifically for each event type. The data elements corresponding to each event type are captured in the XML_DEFINITION column of the EVENT table. These elements are listed in Table 9-13.

Table 9-13 XML_DEFINITION Data Elements

Event	Data Element
AddToCartEvent	application event-date event-type session-id user-id sku quantity unit-list-price currency application-name
BuyEvent	application event-date event-type session-id user-id sku quantity unit-price currency application-name order-line-id
CampaignUserActivityEvent	application event-date event-type session-id user-id campaign-id scenario-id

Table 9-13 XML_DEFINITION Data Elements (Continued)

Event	Data Element
ClickCampaignEvent	application event-date event-type session-id user-id document-type document-id campaign-id scenario-id application-name placeholder-id
ClickContentEvent	application event-date event-type session-id user-id document-type document-id
ClickProductEvent	application event-date event-type session-id user-id document-type document-id sku category-id application-name
DisplayCampaignEvent	application event-date event-type session-id user-id document-type document-id campaign-id scenario-id application-name placeholder-id

Table 9-13 XML_DEFINITION Data Elements (Continued)

Event	Data Element
DisplayContentEvent	application event-date event-type session-id user-id document-type document-id
DisplayProductEvent	application event-date event-type session-id user-id document-type document-id sku category-id application-name
PurchaseCartEvent	application event-date event-type session-id user-id total-price order-id currency application-name
RemoveFromCartEvent	application event-date event-type session-id user-id sku quantity unit-price currency application-name

Table 9-13 XML_DEFINITION Data Elements (Continued)

Event	Data Element
RuleEvent	application event-date event-type session-id user-id ruleset-name rule-name
SessionBeginEvent	application event-date event-type session-id user-id
SessionEndEvent	application event-date event-type session-id user-id
SessionLoginEvent	application event-date event-type session-id user-id
UserRegistrationEvent	application event-date event-type session-id user-id

BT_EVENT_ACTION Database Table

This table logs actions used by third-party vendors against the recorded event data.

Table 9-14 BT_EVENT_ACTION Table Metadata

Column Name	Data Type	Null Value	Description
EVENT_ACTION	VARCHAR(30)	Not Null	The event action taken such as `BEGIN EXPORT` or `END EXPORT`. This field is one of the table's primary keys.
ACTION_DATE	DATE	Not Null	The date and time of the event. This field is one of the table's primary keys.
EVENT_ID	NUMBER	Null	The `ID` of the event that corresponds with the event action taken.

Data Synchronization Database Tables

In this section, WebLogic Portal data synchronization objects tables are arranged alphabetically as a data dictionary.

The following tables compose the data synchronization database:

- DATA_SYNC_APPLICATION Database Table
- DATA_SYNC_ITEM Database Table
- DATA_SYNC_SCHEMA_URI Database Table
- DATA_SYNC_VERSION Database Table

DATA_SYNC_APPLICATION Database Table

This table holds the various applications available for the data synchronization process.

Table 9-15 DATA_SYNC_APPLICATION Table Metadata

Column Name	Data Type	Null Value	Description
APPLICATION_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number used as the record identifier.
APPLICATION_NAME	VARCHAR(100)	Not Null	The deployed J2EE application name. This should match the name in the WebLogic Server console.

Table 9-15 DATA_SYNC_APPLICATION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
CREATION_DATE	DATE	Not Null	The date and time this record was created.
MODIFIED_DATE	DATE	Not Null	The date and time this record was last modified.

DATA_SYNC_ITEM Database Table

This table stores all the data items to be synchronized.

Table 9-16 DATA_SYNC_ITEM Table Metadata

Column Name	Data Type	Null Value	Description
DATA_SYNC_ITEM_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number used as the record identifier.
APPLICATION_ID	NUMBER(15)	Not Null	FK to `DATA_SYNC_APPLICATON.APPLICATION_ID`
SCHEMA_URI_ID	NUMBER(15)	Not Null	FK to `DATA_SYNC_SCHEMA_URI.SCHEMA_URI_ID`
VERSION_MAJOR	NUMBER(15)	Not Null	FK to `DATA_SYNC_VERSION.VERSION_MAJOR`
VERSION_MINOR	NUMBER(15)	Not Null	FK to `DATA_SYNC_VERSION.VERSION_MINOR`
ITEM_CHECKSUM	NUMBER(15)	Not Null	A generated number representing the contents of the `XML_DEFINITION` column.
CREATION_DATE	DATE	Not Null	The date and time this record was created.
MODIFIED_DATE	DATE	Not Null	The date and time this record was last modified.
XML_MODIFIED_DATE	DATE	Not Null	The date and time the XML file was last modified.

Table 9-16 DATA_SYNC_ITEM Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
XML_CREATION_DATE	DATE	Not Null	The date and time the XML file was created.
XML_DEFINITION	CLOB	Not Null	The XML representation of the data item to be synchronized.
ITEM_URI	VARCHAR(254)	Not Null	The path on the file system of the data item to be synchronized.
ITEM_AUTHOR	VARCHAR(200)	Null	Metadata information – the OS login.
ITEM_NAME	VARCHAR(100)	Null	Metadata information – the full path to the item.
ITEM_DESCRIPTION	VARCHAR(254)	Null	Metadata information – a general description of the item to be synchronized.

DATA_SYNC_SCHEMA_URI Database Table

This table holds information pertaining to each of the governing schemas used by various documents.

Table 9-17 DATA_SYNC_SCHEMA_URI Table Metadata

Column Name	Data Type	Null Value	Description
SCHEMA_URI_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number used as the record identifier.
SCHEMA_URI	VARCHAR(254)	Not Null	The governing schema of the document.
CREATION_DATE	DATE	Not Null	The date and time this record was created.
MODIFIED_DATE	DATE	Not Null	The date and time this record was last modified.

DATA_SYNC_VERSION Database Table

This table is not being used currently. It is reserved for future use and is expected to accommodate data synchronization versioning. As a result, this table holds only one record.

Table 9-18 DATA_SYNC_VERSION Table Metadata

Column Name	Data Type	Null Value	Description
VERSION_MAJOR	NUMBER(15)	Not Null	PK – The current record has a value of zero.
VERSION_MINOR	NUMBER(15)	Not Null	PK – The current record has a value of zero.
CREATION_DATE	DATE	Not Null	The date and time that the record was created.
MODIFIED_DATE	DATE	Not Null	The date and time that the record was last modified.
BUILD_NUMBER	NUMBER(15)	Null	The build number associated with the version.
VERSION_DESCRIPTION	VARCHAR(30)	Null	A description of the particular sync version.

Portal Services Database Objects

Portal Services database objects are used for placeholders, ads, and ad bucket services.

Click to view, and use the magnifying tool to inspect individual tables in the entity-relation diagram for Portal Services database objects.

The following tables compose the Portal Services database:

- AD_BUCKET Database Table
- AD_COUNT Database Table
- PLACEHOLDER_PREVIEW Database Table
- MAIL_ADDRESS Database Table
- MAIL_BATCH Database Table
- MAIL_BATCH_ENTRY Database Table

- MAIL_HEADER Database Table
- MAIL_MESSAGE Database Table
- SCENARIO_END_STATE Database Table

AD_BUCKET Database Table

This table maintains content queries for ads.

Table 9-19 AD_BUCKET Table Metadata

Column Name	Data Type	Null Value	Description
AD_BUCKET_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number used as the record identifier.
USER_NAME	VARCHAR (200)	Not Null	The user's name associated with the ad.
PLACEHOLDER_XML_REF	VARCHAR(254)	Not Null	The location identifier of the XML-based placeholder definition file.
APPLICATION_NAME	VARCHAR(100)	Not Null	The name of the application for which the ad has been scoped.
CONTEXT_REF	VARCHAR(254)	Null	The scenario-unique identifier.
CONTAINER_REF	VARCHAR(254)	Null	The campaign-unique identifier.
CONTAINER_TYPE	VARCHAR(50)	Null	Identifies the service associated with the `CONTAINER_REF`.
WEIGHT	NUMBER(15)	Null	A weighted scheme used in prioritizing one placeholder over another.
VIEW_COUNT	NUMBER(15)	Null	Disabled. Reserved for future use.
EXPIRATION_DATE	DATE	Null	The date and time the ad expires or becomes invalid.
CREATION_DATE	DATE	Not Null	The date and time this record was created.
MODIFIED_DATE	DATE	Not Null	The date and time this record was last modified.
AD_QUERY	CLOB	Null	The actual content query.

AD_COUNT Database Table

This table tracks the number of times the ads are displayed and clicked though.

Table 9-20 AD_COUNT Table Metadata

Column Name	Data Type	Null Value	Description
AD_ID	VARCHAR(254)	Not Null	PK – A unique, system-generated number used as the record identifier.
CONTAINER_REF	VARCHAR(254)	Not Null	PK – The campaign-unique identifier.
APPLICATION_NAME	VARCHAR(100)	Not Null	PK – The name of the application for which the ad clicks or views were scoped.
DISPLAY_COUNT	NUMBER(15)	Not Null	The number of times the ad has been displayed.
CLICK_THROUGH_COUNT	NUMBER(15)	Not Null	The number of times the ad has been clicked.

PLACEHOLDER_PREVIEW Database Table

This table is used as a mechanism to hold the placeholder for previewing purposes only.

Table 9-21 PLACEHOLDER_PREVIEW Table Metadata

Column Name	Data Type	Null Value	Description
PREVIEW_ID	NUMBER	Not Null	PK – A unique, system-generated number used as the record identifier.
XML_DEFINITION	CLOB	Null	The representation of the expression to be previewed.

MAIL_ADDRESS Database Table

This table stores all of the address information for e-mail purposes.

Table 9-22 MAIL_ADDRESS Table Metadata

Column Name	Data Type	Null Value	Description
MAIL_ADDRESS_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number to use as the record ID.
MESSAGE_ID	NUMBER(15)	Not Null	FK – Foreign key to the `MAIL_MESSAGE` table.
ADDRESS	VARCHAR(254)	Not Null	Stores the various e-mail addresses on the distribution list.
SEND_TYPE	VARCHAR(4)	Not Null	Determines how the `ADDRESS` should be included on the distribution. Possible values are `TO`, `CC`, or `BCC`.

MAIL_BATCH Database Table

This table establishes a batch for each mailing.

Table 9-23 MAIL_BATCH Table Metadata

Column Name	Data Type	Null Value	Description
BATCH_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number to use as the record `ID`.
BATCH_NAME	VARCHAR(254)	Not Null	The name of the mail message batch.

MAIL_BATCH_ENTRY Database Table

This table correlates the mail batch with the specific mail message.

Table 9-24 MAIL_BATCH_ENTRY Table Metadata

Column Name	Data Type	Null Value	Description
BATCH_ID	NUMBER(15)	Not Null	PK and FK – A unique, system-generated number to use as the record ID.
MESSAGE_ID	NUMBER(15)	Not Null	PK and FK – Foreign key to the `MAIL_MESSAGE` table.

MAIL_HEADER Database Table

This table contains all of the header information specific to the e-mail message.

Table 9-25 MAIL_HEADER Table Metadata

Column Name	Data Type	Null Value	Description
HEADER_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number to use as the record ID.
MESSAGE_ID	NUMBER(15)	Not Null	FK – Foreign key to the `MAIL_MESSAGE` table.

Table 9-25 MAIL_HEADER Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
HEADER_NAME	VARCHAR(50)	Null	The name of the mail message header.
HEADER_VALUE	VARCHAR(254)	Null	The value of the mail message header.

MAIL_MESSAGE Database Table

This table contains the specifics of the mail message (for example, the subject line, text, and so on).

Table 9-26 MAIL_MESSAGE Table Metadata

Column Name	Data Type	Null Value	Description
MESSAGE_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number to use as the record ID.
FROM_ADDRESS	VARCHAR(254)	Null	Identifies who is sending the message.
SUBJECT	VARCHAR(128)	Null	Stores the mail message subject.
MESSAGE_TEXT	CLOB	Null	Holds the content of the mail message.

SCENARIO_END_STATE Database Table

This table identifies when a user is no longer eligible to participate in a particular scenario.

Table 9-27 SCENARIO_END_STATE Table Metadata

Column Name	Data Type	Null Value	Description
SCENARIO_XML_REF	VARCHAR(20)	Not Null	PK – The identifier for the XML-based scenario definition file.
USER_NAME	VARCHAR(200)	Not Null	PK—the user `ID`. FK to `WLCS_USER.IDENTIFIER`.
CONTAINER_REF	VARCHAR(254)	Not Null	PK—the campaign unique identifier. FK to `CAMPAIGN.CAMPAIGN_UID`.

Table 9-27 SCENARIO_END_STATE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
CONTAINER_TYPE	VARCHAR(50)	Not Null	PK—At this time this column always holds the string `Campaign`.
APPLICATION_NAME	VARCHAR(100)	Not Null	PK – The deployed J2EE application name. This should match the name in the WebLogic Server console.

Entitlement Reference Dictionary Tables

The following tables are used by the Administration Console to maintain security policy reference data as policies are created, edited, and deleted. These tables allow efficient searching for policies given a role name, resource ID web application name, and so on. The Entitlement Policy system has the following tables:

- P13N_ENTITLEMENT_APPLICATION Database Table
- P13N_ENTITLEMENT_POLICY Database Table
- P13N_ENTITLEMENT_RESOURCE Database Table
- P13N_ENTITLEMENT_ROLE Database Table
- P13N_DELEGATED_HIERARCHY Database Table
- USERS Database Table
- GROUPS Database Table
- GROUPMEMBERS Database Table

P13N_ENTITLEMENT_APPLICATION Database Table

This table uniquely identifies an application for which entitlements can be applied.

Table 9-28 P13N_ENTITLEMENT_APPLICATION Table Metadata

Column Name	Data Type	Null Value	Description
ENTITLEMENT_APP_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number to use as the record ID.

Table 9-28 P13N_ENTITLEMENT_APPLICATION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
ENTAPP_NAME	VARCHAR(255)	Not Null	The name of the enterprise application.
WEBAPP_NAME	VARCHAR(255)	Null	The name of the web application.
ENTITLEMENT_APP_DESCRIPTION	VARCHAR(255)	Null	The description of the enterprise application.
CREATION_DATE	DATE	Not Null	The date and time the record was created; the default is the current time stamp.
MODIFIED_DATE	DATE	Not Null	The date and time the record was last modified; the default is the current time stamp.

P13N_ENTITLEMENT_POLICY Database Table

This table uniquely identifies an entitlement policy. An entitlement policy is created when an entitlement role is associated with an entitlement resource and capability.

Table 9-29 P13N_ENTITLEMENT_POLICY Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
RESOURCE_ENT_APP_ID	NUMBER(15)	Not Null	PK and FK to `P13N_ENTITLEMENT_APPLICATION`.	
RESOURCE_ID	NUMBER(15)	Not Null	PK and FK to `P13N_ENTITLEMENT_RESOURCE`.	
ROLE_ENT_APP_ID	NUMBER(15)	Not Null	PK and FK to `P13N_ENTITLEMENT_APPLICATION`.	
ROLE_ID	NUMBER(15)	Not Null	PK and FK to `P13N_ENTITLEMENT_ROLE`.	

Table 9-29 P13N_ENTITLEMENT_POLICY Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
POLICY_RESOURCE_CAPABILITY	VARCHAR(80)	Not Null	PK – Identifies the unique capability of this policy instance.	
CREATION_DATE	DATE	Not Null	The date and time the record was created; the default is the current time stamp.	
MODIFIED_DATE	DATE	Not Null	The date and time the record was last modified; default value is the current time stamp.	
ROLE_PROVIDER_NAME	VARCHAR(80)	Not Null	The name given the deployed Role Mapping provider holding the role definition.	
ATZ_PROVIDER_NAME	VARCHAR(80)	Not Null	The name of the authorization provider.	9.2

P13N_ENTITLEMENT_RESOURCE Database Table

This table uniquely identifies an application resource that can have an entitlement associated with it.

Table 9-30 P13N_ENTITLEMENT_RESOURCE Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
ENTITLEMENT_APP_ID	NUMBER(15)	Not Null	PK and FK to `P13N_ENTITLEMENT_APPLICATION`.	

Table 9-30 P13N_ENTITLEMENT_RESOURCE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
RESOURCE_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number to use as the record ID.	
RESOURCE_NAME	VARCHAR(882)	Not Null	The name of the resource having a policy applied on it.	9.2
RESOURCE_DESCRIPTION	VARCHAR(255)	Null	Optional description of resource.	
RESOURCE_METADATA	VARCHAR(255)	Null	Optional application-defined metadata.	
CREATION_DATE	DATE	Not Null	The date and time the record was last modified; the default is the current time stamp.	
MODIFIED_DATE	DATE	Not Null	The date and time the record was last modified; the default is the current time stamp.	

P13N_ENTITLEMENT_ROLE Database Table

This table uniquely identifies entitlement and delegated administration roles for a given application.

Table 9-31 P13N_ENTITLEMENT_ROLE Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
ENTITLEMENT_APP_ID	NUMBER(15)	Not Null	PK and FK to `P13N_ENTITLEMENT_APPLICATION`.	

Table 9-31 P13N_ENTITLEMENT_ROLE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
ROLE_ID	NUMBER(15)	Not Null	PK – A unique, system-generated number to use as the record ID.	
ROLE_NAME	VARCHAR(255)	Not Null	The name of the security role.	
ROLE_DESCRIPTION	VARCHAR(255)	Null	Optional description of the role.	
ROLE_SEGMENT	VARCHAR(255)	Null	Optional role expression name.	
CREATION_DATE	DATE	Not Null	The date and time the record was last modified; the default is the current time stamp.	
MODIFIED_DATE	DATE	Not Null	The date and time the record was last modified; the default is the current time stamp.	
ROLE_PROVIDER_NAME	VARCHAR(80)	Not Null	Identifies the policy role mapper provider; default value = DefaultRoleMapper	9.2

P13N_DELEGATED_HIERARCHY Database Table

This table uniquely identifies an entitlement hierarchy. An entitlement hierarchy is associated with P13N_ENTITLEMENT_APPLICATION.

Table 9-32 P13N_DELEGATED_HIERARCHY Table Metadata

Column Name	Data Type	Null Value	Description
ENTITLEMENT_APP_ID	NUMBER(15)	Not Null	PK and FK to `P13N_ENTITLEMENT_APPLICATION`.
ENTITLEMENT_DOCUMENT	CLOB	Not Null	An XML document containing the Delegated Administration role hierarchy.

Table 9-32 P13N_DELEGATED_HIERARCHY Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
CREATION_DATE	DATE	Not Null	The date and time the record was last modified; the default is the current time stamp.
MODIFIED_DATE	DATE	Not Null	The date and time the record was last modified; the default is the current time stamp.

USERS Database Table

This table was added in WebLogic Portal version 9.2. It maintains user, group, and password information for the WebLogic Server SQLAuthenticator.

Table 9-33 USERS Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
U_NAME	VARCHAR(200)	Not Null	PK – Unique name of the user.	9.2
U_PASSWORD	VARCHAR(50)	Not Null	Password of the user.	9.2
U_DESCRIPTION	VARCHAR(1000)	Null	Optional description.	9.2

GROUPS Database Table

This table was added in WebLogic Portal version 9.2. It maintains information on groups and their descriptions.

Table 9-34 GROUPS Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
G_NAME	VARCHAR(200)	Not Null	PK – Unique name of the group.	9.2
G_DESCRIPTION	VARCHAR(1000)	Null	Optional description.	9.2

GROUPMEMBERS Database Table

This table was added in WebLogic Portal version 9.2. It maintains group hierarchy information.

Table 9-35 GROUPMEMBERS Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
G_NAME	VARCHAR(200)	Not Null	PK and FK to the GROUPS table. Unique name of the group.	9.2
G_MEMBER	VARCHAR(200)	Not Null	PK – Member of the group.	9.2

Portal Framework Database Objects

The Portal Framework is the portion of WebLogic Portal that is responsible for the rendering and customization of the portal.

Click to view, and use the magnifying tool to inspect individual tables in the logical entity-relation diagram for the Portal Framework tables. This includes tables that support WSRP and localization.

Portal Framework Database Tables

In this section, tables are arranged alphabetically as a data dictionary. The following tables compose the generic Portal Framework database:

- PF_BOOK_DEFINITION Database Table
- PF_BOOK_GROUP Database Table
- PF_BOOK_INSTANCE Database Table
- PF_DESKTOP_DEFINITION Database Table
- PF_DESKTOP_INSTANCE Database Table
- PF_LAYOUT_DEFINITION Database Table
- PF_LOOK_AND_FEEL_DEFINITION Database Table
- PF_MARKUP_DEFINITION Database Table
- PF_MARKUP_XML Database Table
- PF_PAGE_DEFINITION Database Table
- PF_PAGE_INSTANCE Database Table
- PF_PLACEHOLDER_DEFINITION Database Table
- PF_PLACEMENT Database Table
- PF_PORTAL Database Table
- PF_PORTLET_CATEGORY Database Table
- PF_PORTLET_CATEGORY_DEFINITION Database Table
- PF_PORTLET_DEFINITION Database Table
- PF_PORTLET_INSTANCE Database Table
- PF_PORTLET_PREFERENCE Database Table
- PF_PORTLET_PREFERENCE_VALUE Database Table
- PF_PORTLET_USER_PROPERTIES Database Table
- PF_SHELL_DEFINITION Database Table

- PF_THEME_DEFINITION Database Table
- PF_URL_COMPRESSION Database Table

To view the communities-related portal framework tables, see "Communities Framework Database Objects" on page 9-70. To view the tables related to localization, refer to "Localization Dictionary Tables" on page 9-57.

PF_BOOK_DEFINITION Database Table

This table defines a `BOOK` portal library resource. Books are used to aggregate `PAGES` and other `BOOKS`.

Table 9-36 PF_BOOK_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
BOOK_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
MARKUP_DEFINITION_ID	NUMBER	Not Null	FK to `PF_MARKUP_DEFINITION`.
IS_PUBLIC	NUMBER	Not Null	A boolean flag indicating whether this book definition displays to the public. When end users create books they are not marked as public.
IS_HIDDEN	NUMBER	Not Null	A boolean flag indicating whether this book definition is hidden from the menu. Marking a page or book as hidden does not prevent it from being displayed; this indicator is only a hint to the menu control not to display a tab for the given book or page. The page or book can be activated using a link or a backing file.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to `L10N_INTERSECTION`.

Table 9-36 PF_BOOK_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application (as defined in the `config.xml`) to which the portal resource is scoped.
BOOK_LABEL	VARCHAR(80)	Null	A moniker used to reference this portal resource for development purposes. This is the same as the bookDefinitionLabel in WebLogic Workshop. If a label is not supplied at creation time, the `BOOK_DEFINITION_ID` prefixed with a `B` is used. This label can be supplied to APIs to activate books or pages.

PF_BOOK_GROUP Database Table

This table represents a child page or book placement on the parent book. A single record in the table represents one placement on a book. This table also identifies a customized grouping of books and pages. Customized groupings are represented and aggregated around the `DESKTOP_INSTANCE_ID`.

Table 9-37 PF_BOOK_GROUP Table Metadata

Column Name	Data Type	Null Value	Description
BOOK_GROUP_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
PARENT_BOOK_ID	NUMBER	Not Null	FK to `PF_BOOK_INSTANCE` that identifies the parent `BOOK_INSTANCE_ID`.
ALIGNMENT	NUMBER	Not Null	The alignment is a hint to the menu skeleton JSP to indicate whether the tab should be aligned on the left or right of the tab bar. A skeleton can either implement this feature or ignore it.

Table 9-37 PF_BOOK_GROUP Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
MENU_POSITION	NUMBER	Not Null	The order, in the tab menu, in which this page or book will appear on the parent book.The order does not need to be contiguous.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
IS_DEFAULT	NUMBER	Not Null	A boolean flag indicating that this is the default page or book on the parent book.
CHILD_BOOK_ID	NUMBER	Null	FK to `PF_BOOK_INSTANCE` that identifies the child `BOOK_INSTANCE_ID`.
PAGE_INSTANCE_ID	NUMBER	Null	FK to `PF_BOOK_INSTANCE`.
DESKTOP_INSTANCE_ID	NUMBER	Null	FK to `PF_DESKTOP_INSTANCE`. If this book grouping is an administrator or end user customization, this value is non null and points to the administrator's or user's desktop. If this field is null, it represents the library's view.

PF_BOOK_INSTANCE Database Table

This table identifies an instance of the `BOOK_DEFINITION`. There is always at least one book instance, namely the primary instance. All other instances represent customization by administrators or end users.

Table 9-38 PF_BOOK_INSTANCE Table Metadata

Column Name	Data Type	Null Value	Description
BOOK_INSTANCE_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.

Table 9-38 PF_BOOK_INSTANCE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
MENU_ORIENTATION	NUMBER	Not Null	The orientation is a hint to the book skeleton JSP and the menu skeleton JSP to display the tabs on the top, left, right, or bottom of the main book. The skeletons can choose to ignore this field.
INSTANCE_TYPE	NUMBER	Not Null	The type of book instance: 1=Primary, 3=Admin, 4=User.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INSTANCE_TITLE	VARCHAR(255)	Null	An end-user-customized title for this `BOOK`. This title is not internationalized as it is used only by the end user. If the end user does not customize the book title, then the value is null and the `L10N_RESOURCE` title is used.
BOOK_DEFINITION_ID	NUMBER	Not Null	FK to `PF_BOOK_DEFINITION`.
MENU_DEFINITION_ID	NUMBER	Null	FK to `PF_MENU_DEFINITION`. Can be null as not every book must have a menu.
THEME_DEFINITION_ID	NUMBER	Null	FK to `PF_THEME_DEFINITION`.

PF_DESKTOP_DEFINITION Database Table

This table defines a desktop definition. You can create Desktops from template (`.portal`) files or from existing resources.

Table 9-39 PF_DESKTOP_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
DESKTOP_PATH	VARCHAR(40)	Not Null	Part of the PK – identifies the partial URL path to the desktop.

Table 9-39 PF_DESKTOP_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
PORTAL_PATH	VARCHAR(40)	Not Null	Part of the PK and FK to PF_PORTAL – identifies the partial URL path to this desktop and parent portal.
WEBAPP_NAME	VARCHAR(80)	Not Null	Part of the PK and FK to PF_PORTAL. This is the name of the web application (as defined in the config.xml file) to which this desktop is scoped.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to L10N_INTERSECTION. The BOOK_INSTANCE_ID of the main or default PF_BOOK_INSTANCE for the desktop.
MARKUP_DEFINITION_ID	NUMBER	Not Null	FK to PF_MARKUP_DEFINITION.
IS_TREE_OPTIMIZED	NUMBER	Not Null	Indicates whether tree optimization is active for a desktop. Acceptable values are 0 (off; the default) or 1 (on).
DESKTOP_TYPE	NUMBER	Not Null	Indicates whether the desktop is a community; the default is 0 (not a community).
IS_TEMPLATE	NUMBER	Not Null	Indicates whether the desktop is a template; the default is 0 (not a template).
IS_GLOBAL	NUMBER	Not Null	Indicates whether the desktop is a global desktop; the default is 0 (not global).

PF_DESKTOP_INSTANCE Database Table

This table identifies a customized or localized instance of a desktop.

Table 9-40 PF_DESKTOP_INSTANCE Table Metadata

Column Name	Data Type	Null Value	Description
DESKTOP_INSTANCE_ID	NUMBER	Not Null	PK – identifies the partial URL path to the desktop.
DESKTOP_PATH	VARCHAR(40	Not Null	FK to PF_DESKTOP_DEFINITION.
PORTAL_PATH	VARCHAR(40)	Not Null	FK to PF_DESKTOP_DEFINITION.
WEBAPP_NAME	VARCHAR(80)	Not Null	FK to PF_DESKTOP_DEFINITION.
MAIN_BOOK_ID	NUMBER	Not Null	FK to BOOK_INSTANCE_ID of the main or default PF_BOOK_INSTANCE for the desktop.
USER_NAME	VARCHAR(200)	Null	The name of the user if the user has customized his/her desktop. This value is null if the desktop instance is not for a particular user or administrator.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
LOOK_FEEL_DEFINITION_ID	NUMBER	Null	FK to PF_LOOK_AND_FEEL_DEFINITION.
INSTANCE_TITLE	VARCHAR(20)	Null	An end-user-customized title for this DESKTOP. This title is not internationalized as it is used only by the end user. If the end user does not customize the desktop title, then the value is null and the L10N_RESOURCE title is used.
SHELL_DEFINITION_ID	NUMBER	Not Null	FK to PF_SHELL_DEFINITION.

Table 9-40 PF_DESKTOP_INSTANCE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
IS_COMMUNITY	NUMBER	Not Null	Indicates whether the desktop is a community; possible values are: 0 (the default, not a community), or 1 (community).

PF_LAYOUT_DEFINITION Database Table

This table defines a `LAYOUT` portal library resource that is used as a specification for determining the location of items on a page. For every layout definition there is a corresponding `.layout` file. By updating the `.layout` file, you are updating this record.

Table 9-41 PF_LAYOUT_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
LAYOUT_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
MARKUP_DEFINITION_ID	NUMBER	Not Null	FK to `PF_MARKUP_DEFINITION`.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to `L10N_INTERSECTION`.
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application to which the portal resource is scoped.

Table 9-41 PF_LAYOUT_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
IS_LAYOUT_FILE_ DELETED	NUMBER	Not Null	A boolean indicating that the file associated with this layout was removed from the file system. If the layout is not being used, then the record is deleted outright. This flag is set to true only when the `.layout` file is deleted and the layout is still in use. You can either return the `.layout` file and this flag is automatically reset, or remove the layout with a replacement layout in the admin tools.
LAYOUT_FILE	VARCHAR(255)	Null	The name and location of the file associated with this layout definition.
ICON_URI	VARCHAR(255)	Null	The URI that identifies the `ICON` for this layout definition.
HTML_LAYOUT_URI	VARCHAR(255)	Null	The URI for the HTML for this layout definition. The htlp file is used by the admin and visitor tools to provide a visual display that emulates the real layout.

PF_LOOK_AND_FEEL_DEFINITION Database Table

This table defines a `LOOK and FEEL` portal library resource or template for assignment to `DESKTOPs` that control how a portal renders.

Table 9-42 PF_LOOK_AND_FEEL_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
LOOK_FEEL_ DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
LOOK_FEEL_LABEL	VARCHAR(80)	Not Null	A moniker used to reference this portal resource for development purposes.
CREATION_DATE	DATE	Not Null	The date and time the row was created.

Table 9-42 PF_LOOK_AND_FEEL_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to L10N_INTERSECTION.
MARKUP_DEFINITION_ID	NUMBER	Not Null	FK to PF_MARKUP_DEFINITION.
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application to which the portal resource is scoped.
IS_LOOK_FEEL_FILE_ DELETED	NUMBER	Not Null	A boolean indicating that the file associated with this look and feel was removed from the file system. If the look and feel is not being used, then the record is deleted outright. This flag is set to true only when the `.laf` file is deleted and the look and feel is still in use. You can either return the `.laf` file and this flag is automatically reset, or remove the look and feel with a replacement look and feel in the Administration Console.
LOOK_FEEL_FILE	VARCHAR(255)	Not Null	The fully qualified file path (from the web app) to the location of the `.laf` file associated with this look and feel definition.

PF_MARKUP_DEFINITION Database Table

This table defines the `MARKUP` (blueprint, design, model) for a portal library resource.

Table 9-43 PF_MARKUP_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
MARKUP_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
CREATION_DATE	DATE	Not Null	The date and time the row was created.

Table 9-43 PF_MARKUP_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
MARKUP_NAME	VARCHAR(255)	Not Null	The filename and location that contains the definition of this portal object.
MARKUP_TYPE	VARCHAR(20)	Not Null	The type of portal resource that this markup defines.
BEGIN_XML	VARCHAR(2000)	Not Null	The first 2000 characters of XML definition of this portal object.
END_XML	VARCHAR(2000)	Null	The last 2000 characters of the XML definition of this portal object.
MARKUP_FILE	VARCHAR(255)	Null	Location of the file containing the markup definition.
WEBAPP_NAME	VARCHAR(80)	Null	Name of the J2EE web application to which the portal resource is scoped.

PF_MARKUP_XML Database Table

This table was added in WebLogic Portal version 9.2. It defines additional XML markup information for portal resources.

Table 9-44 PF_MARKUP_XML Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
MARKUP_DEFINITION_ID	INTEGER	Not Null	PK – A unique, system-generated number to use as the record ID.	9.2
XML_TYPE	CHAR(1)	Not Null	Identifies if this block is a begin or end block. Values are "B" for begin or "E" for end.	9.2

Table 9-44 PF_MARKUP_XML Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
BLOCK_NUMBER	INTEGER	Not Null	Determines the order of the markup.	9.2
MARKUP_XML	VARCHAR(2000)	Not Null	The markup value.	9.2

PF_MENU_DEFINITION Database Table

This table defines a `MENU` portal library resource or template that can be assigned to a `BOOK INSTANCE`.

Table 9-45 PF_MENU_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
MENU_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
MARKUP_DEFINITION_ID	NUMBER	Not Null	FK to `PF_MARKUP_DEFINITION`.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to `L10N_INTERSECTION`.
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application to which the portal resource is scoped.

Table 9-45 PF_MENU_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
IS_MENU_FILE_DELETED	NUMBER	Not Null	A boolean indicating that the file associated with this menu was removed from the file system. If the menu is not being used then the record is deleted outright. This flag is set to true only when the `.menu` file is deleted and the menu is still in use. You can either return the `.menu` file and this flag is automatically reset, or remove the menu with a replacement menu in the Administration Console.
MENU_FILE	VARCHAR(255)	Not Null	The fully qualified path (from the web application) to the location of the .menu file associated with this menu definition.

PF_PAGE_DEFINITION Database Table

This table defines a `PAGE` portal library resource or template that can be assigned to a `PAGE INSTANCE`.

Table 9-46 PF_PAGE_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
PAGE_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
MARKUP_DEFINITION_ID	NUMBER	Not Null	FK to `PF_MARKUP_DEFINITION`.
IS_PUBLIC	NUMBER	Not Null	A boolean indicating this page definition is public. Only public page definitions are ever exposed to site visitors.
IS_HIDDEN	NUMBER	Not Null	A boolean indicating this page is hidden. The hidden flag is a hint to the menu not to render a tab for this page. The page can still be displayed by other methods (links, events).

Table 9-46 PF_PAGE_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to `L10N_INTERSECTION`.
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application to which the portal resource is scoped.
PAGE_LABEL	VARCHAR(80)	Null	A moniker used to reference this portal resource for development purposes.

PF_PAGE_INSTANCE Database Table

This table identifies an instance of the page definition; at least one instance per definition always exists.

Table 9-47 PF_PAGE_INSTANCE Table Metadata

Column Name	Data Type	Null Value	Description
PAGE_INSTANCE_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INSTANCE_TYPE	NUMBER	Not Null	The type of page instance: 1=Primary, 3=Admin, 4=User.
LAYOUT_DEFINITION_ID	NUMBER	Not Null	FK to `PF_LAYOUT_DEFINITION`.
PAGE_DEFINITION_ID	NUMBER	Not Null	FK to `PF_PAGE_DEFINITION`.
THEME_DEFINITION_ID	NUMBER	Null	FK to `PF_THEME_DEFINITION`.

Table 9-47 PF_PAGE_INSTANCE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
INSTANCE_TITLE	VARCHAR(255)	Null	A desktop- or user-customized title for this page. This instance title is valid only to end users as it cannot and need not be localized.

PF_PLACEHOLDER_DEFINITION Database Table

This table defines a `PLACEHOLDER` portal library resource or template that has a `LAYOUT` definition and can be assigned to a `PLACEMENT`.

Table 9-48 PF_PLACEHOLDER_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
PLACEHOLDER_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
MARKUP_DEFINITION_ID	NUMBER	Not Null	FK to `PF_MARKUP_DEFINITION`.
LAYOUT_LOCATION	NUMBER	Not Null	The location of this placeholder in the layout. This is used when swapping layouts, as portlets in one layout's location are moved to the other layout's location with the same ID. If the other layout does not have the same number of placeholders, the modulus of the location by number of locations are used.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to `L10N_INTERSECTION`.
LAYOUT_DEFINITION_ID	NUMBER	Not Null	FK to `PF_LAYOUT_DEFINITION`.

PF_PLACEMENT Database Table

Each record in this table represents a single placement of a book or portlet on a page.

Table 9-49 PF_PLACEMENT Table Metadata

Column Name	Data Type	Null Value	Description
PLACEMENT_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
PAGE_INSTANCE_ID	NUMBER	Not Null	FK to `PF_PAGE_INSTANCE`.
POSITION	NUMBER	Not Null	The position within the placeholder where this placement lies. Placeholders can contain more than one placement.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
PLACEHOLDER_ DEFINITION_ID	NUMBER	Not Null	FK to `PF_PLACEHOLDER_DEFINITION`.
PORTLET_INSTANCE_ID	NUMBER	Null	FK to `PF_PORTLET_INSTANCE`.
BOOK_INSTANCE_ID	NUMBER	Null	FK to `PF_BOOK_INSTANCE`.
DESKTOP_INSTANCE_ID	NUMBER	Null	FK to `PF_DESKTOP_INSTANCE`. If this placement grouping is an administrator- or end-user-customization, the value is non null and points to the administrator's or user's desktop. If this field is null, it represents the library's view.

PF_PORTAL Database Table

This table identifies a `PORTAL` application library resource or template that can be associated with a `DESKTOP` definition.

Table 9-50 PF_PORTAL Table Metadata

Column Name	Data Type	Null Value	Description
PORTAL_PATH	VARCHAR(40)	Not Null	PK – Partial primary key and partial URL to this portal.
WEBAPP_NAME	VARCHAR(80)	Not Null	PK – Name of the J2EE web application to which the portal resource is scoped.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to `L10N_INTERSECTION`.
CONTENT_URI	VARCHAR(255)	Null	Defines an optional URI to be forwarded to when only the portal portion of the URL is supplied. You can use this URL (JSP or `.portal`) to forward to a default desktop or to display a list of desktops available under this portal.

PF_PORTLET_CATEGORY Database Table

This table associates a `PORTLET CATEGORY` resource with a `PORTLET DEFINITION`.

Table 9-51 PF_PORTLET_CATEGORY Table Metadata

Column Name	Data Type	Null Value	Description
PORTLET_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
PORTLET_CATEGORY_ DEFINITION_ID	NUMBER	Not Null	FK to `PF_PORTLET_CATEGORY_ DEFINITION`.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.

PF_PORTLET_CATEGORY_DEFINITION Database Table

This table identifies a PORTLET CATEGORY and PORTLET CATEGORY hierarchy resource or template for association with a PORTLET resource.

Table 9-52 PF_PORTLET_CATEGORY_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
PORTLET_CATEGORY_ DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to L10N_INTERSECTION.
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application to which the portal resource is scoped.
PARENT_CATEGORY_ DEFINITION_ID	NUMBER	Null	FK to PF_PORTLET_CATEGORY_ DEFINITION that identifies the parent portlet category. NULL if this is a top level category.

PF_PORTLET_DEFINITION Database Table

This table identifies the characteristics of a PORTLET library resource or template that can used as the user interfaces for a web application.

Table 9-53 PF_PORTLET_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
PORTLET_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
MARKUP_DEFINITION_ID	NUMBER	Not Null	FK to PF_MARKUP_DEFINITION.

Table 9-53 PF_PORTLET_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
IS_PUBLIC	NUMBER	Not Null	A boolean indicating that the portlet definition is public. Only public portlet definitions are ever exposed to site visitors.
IS_FORKABLE	NUMBER	Not Null	A boolean indicating that the portlet supports multi-threading.
FORK_RENDER	NUMBER	Not Null	A boolean indicating whether multi-threading is being used for this portlet; this value can be true only if `IS_FORKABLE` is true.
IS_CACHEABLE	NUMBER	Not Null	A boolean indicating whether this portlet can use render caching.
CACHE_EXPIRES	NUMBER	Not Null	Indicates whether this portlet is using caching and if so, gives the ttl: -1 indicates off; 0..n indicates a ttl for the cache. Can have a value other than -1 only if `IS_CACHEABLE` is true.
WSRP_PROPERTIES_MODE	NUMBER	Not Null	Indicates required user properties. for WSRP user profile propagation. Possible values are: 0 = None 1 = All 2 = Specified properties 3 = User properties are defaulted to the web application's default. This is the default value.

Table 9-53 PF_PORTLET_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
IS_PORTLET_FILE_DELETED	NUMBER	Not Null	A boolean that indicates whether the `PORTLET_FILE` associated with this object has been removed from the file system. This flag is set to true only when the `.portlet` file is deleted and the portlet is still in use. You can either return the `.portlet` file and this flag is automatically reset, or remove the portlet in the Administration Console.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
PORTLET_LABEL	VARCHAR(80)	Not Null	A moniker used to reference this portal resource for development purposes.
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application to which the portal resource is scoped.
CONTENT_URI	VARCHAR(255)	Not Null	The content URI for this portlet (JSP, HTML). This value can be null for Java (JSR168) portlets.
EDIT_URI	VARCHAR(255)	Null	The Edit mode URI (JSP) for this portlet (if the portlet supports edit mode).
HELP_URI	VARCHAR(255)	Null	The Help mode URI (JSP) for this portlet (if the portlet supports help mode).
BACKING_FILE	VARCHAR(255)	Null	The optional backing file (Java class name) for this portlet. Backing classes must implement JspBacking or extend AbstractJspBacking.

Table 9-53 PF_PORTLET_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
PORTLET_FILE	VARCHAR(255)	Null	The (`*.portlet`) file describing the controls that make up the portlet. The field is null if no portlet file exists.

PF_PORTLET_INSTANCE Database Table

This table identifies a customized or localized instance of a portlet

Note: To view the WSRP tables that are a subset of the PF_PORTLET_INSTANCE table, see "WSRP Portal Framework Database Objects" on page 9-61.

Table 9-54 PF_PORTLET_INSTANCE Table Metadata

Column Name	Data Type	Null Value	Description
PORTLET_INSTANCE_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
PORTLET_DEFINITION_ID	NUMBER	Not Null	FK to `PF_PORTLET_DEFINITION`.
DEFAULT_MINIMIZED	NUMBER	Not Null	A boolean that indicates whether the portlet is to be displayed in the minimized state by default.
INSTANCE_TYPE	NUMBER	Not Null	Type codes for the portlet instance. Valid values: 1=Primary, 2=Library, 3=Admin, 4=User.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
TITLE_BAR_ORIENTATION	NUMBER	Null	A hint to the skeleton file to display this portlet's title bar in either the top, left, right, or bottom location. Not all skeletons may implement this and therefore may not have any effect.
INTERSECTION_ID	NUMBER	Not Null	FK to `L10N_INTERSECTION`.

Table 9-54 PF_PORTLET_INSTANCE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
PORTLET_LABEL	VARCHAR(80)	Not Null	A moniker used to reference this portal resource for development purposes.
THEME_DEFINITION_ID	NUMBER	Null	FK to `PF_THEME_DEFINITION`.
PARENT_PORTLET_ INSTANCE_ID	NUMBER	Null	FK to `PF_PORTLET_INSTANCE` that identifies the parent portlet instance. the value is null if this is a top-level portlet instance.

PF_PORTLET_PREFERENCE Database Table

This table identifies preference values for the portlet instance.

Table 9-55 PF_PORTLET_PREFERENCE Table Metadata

Column Name	Data Type	Null Value	Description
PORTLET_INSTANCE_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
PREFERENCE_NAME	VARCHAR(40)	Not Null	An optional name associated with the preference values.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
IS_MODIFIABLE	NUMBER	Not Null	A boolean, indicating whether the name/value of this preference can be modified by portlets.
IS_MULTIVALUED	NUMBER	Not Null	A boolean, indicating whether a preference can have more than one value.
PREFERENCE_DESCRIPTION	VARCHAR(255)	Null	An optional description of the portlet preferences.

PF_PORTLET_PREFERENCE_VALUE Database Table

This table maintains values of portlet preferences. There is a one-to-many correspondence between the records in the PF_PORTLET_PREFERENCE table and this table.

Table 9-56 PF_PORTLET_PREFERENCE_VALUE Table Metadata

Column Name	Data Type	Null Value	Description
PORTLET_PREFERENCE_VALUE_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
PORTLET_INSTANCE_ID	NUMBER	Not Null	FK to `PF_PORTLET_PREFERENCE`.
PREFERENCE_NAME	VARCHAR(40)	Not Null	FK to `PF_PORTLET_PREFERENCE`.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
PREFERENCE_VALUE	VARCHAR(255)	Null	The actual value for this preference.

PF_PORTLET_USER_PROPERTIES Database Table

This table maintains values of portlet user properties for WSRP user profile propagation.

This table was added in WebLogic Portal 9.2.

Table 9-57 PF_PORTLET_USER_PROPERTIES Table Metadata

Column Name	Data Type	Null Value	Description
USER_PROPERTY_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the property ID.
PORTLET_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the portlet definition ID.
PROPERTY_STRING	VARCHAR(255)	Not Null	String value.

PF_SHELL_DEFINITION Database Table

This table represents a shell definition. There is a one-to-one correspondence between records in this table and .shell files.

Table 9-58 PF_SHELL_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
SHELL_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
MARKUP_DEFINITION_ID	NUMBER	Not Null	FK to `PF_MARKUP_DEFINITION`.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to `L10N_INTERSECTION`.
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application to which the portal resource is scoped.
IS_SHELL_FILE_DELETED	NUMBER	Not Null	A boolean indicating that the file associated with this shell was removed from the file system. If the shell is not being used, then the record is deleted outright. This flag is set to true only when the `.shell` file is deleted and the shell is still in use. You can either return the `.shell` file and this flag is automatically reset, or remove the shell with a replacement in the Administration Console.
SHELL_FILE	VARCHAR(255)	Not Null	The name of the `.shell` file contained in the application's `framework/markup/shell` directory backing this shell definition.

PF_THEME_DEFINITION Database Table

This table represents a theme definition. There is a one-to-one correspondence between records in this table and `.theme` files.

Table 9-59 PF_THEME_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description
THEME_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
INTERSECTION_ID	NUMBER	Not Null	FK to `L10N_INTERSECTION`.
MARKUP_DEFINITION_ID	NUMBER	Not Null	`FK to PF_MARKUP_DEFINITION`.
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application to which the portal resource is scoped.
IS_THEME_FILE_DELETED	NUMBER	Not Null	A boolean indicating that the file associated with this theme was removed from the file system. If the theme is not being used, then the record is deleted outright. This flag is set to true only when the `.theme` file is deleted and the theme is still in use. You can either return the `.theme` file and this flag is automatically reset, or remove the theme using the Administration Console.
THEME_FILE	VARCHAR(255)	Not Null	The name of the `.theme` file contained in the application's `framework/markup/theme` directory backing this theme definition.

PF_URL_COMPRESSION Database Table

This table was added in WebLogic Portal version 9.2. It maintains URL compression values.

Table 9-60 PF_URL_COMPRESSION Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
URL_COMPRESSION_ID	INTEGER	Not Null	PK – A unique, system-generated number to use as the URL Compression ID.	9.2
EXPANDED_URL_HASH	INTEGER	Not Null	Hashed URL used to look up the compressed URL; hashes are not guaranteed to be unique, so a secondary search is performed on EXPANDED_URL.	9.2
EXPANDED_URL	VARCHAR(1000)	Not Null	The full uncompressed URL value.	9.2
PAGE_LABEL	VARCHAR(80)	Null	A name used to reference this portal resource for development purposes.	9.2

Localization Dictionary Tables

This section documents the database objects for the WebLogic Portal framework. The following tables support localization:

- L10N_INTERSECTION Database Table
- L10N_LOCALE Database Table
- L10N_RESOURCE Database Table
- L10N_RESOURCE_TYPE Database Table

L10N_INTERSECTION Database Table

This table ties an application resource (menu, portlet, and so on) to a localized title and description.

Table 9-61 L10N_INTERSECTION Table Metadata

Column Name	Data Type	Null Value	Description
INTERSECTION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
RESOURCE_TYPE	VARCHAR(80)	Not Null	FK to `L10N_RESOURCE_TYPE`.

L10N_LOCALE Database Table

This table defines the characteristics of a locale that are needed to localize an application.

Table 9-62 L10N_LOCALE Table Metadata

Column Name	Data Type	Null Value	Description
LOCALE_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
ENCODING	VARCHAR(20)	Not Null	The encoding that is used by the locale. The default encoding is UTF-8.
LANGUAGE	CHAR(2)	Not Null	Lowercase two-letter ISO-639 language code that is used by the locale; for example, en, au.

Table 9-62 L10N_LOCALE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description
COUNTRY	CHAR(2)	Null	Uppercase two-letter ISO-3166 country code that is used by the locale; for example, US, UK.
VARIANT	VARCHAR(40)	Null	Vendor- and browser-specific code variant code that is used by the locale; for example, WIN, MAC, UNIX.

L10N_RESOURCE Database Table

This table defines the localized title and description of a localized resource.

Table 9-63 L10N_RESOURCE Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
LOCALE_ID	NUMBER	Not Null	PK and FK to `L10N_LOCALE`.	
INTERSECTION_ID	NUMBER	Not Null	PK and FK to `L10N_INTERSECTION`.	
CREATION_DATE	DATE	Not Null	The date and time the row was created.	
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.	
TITLE	VARCHAR(80)	Not Null	A localized title for the object, typically used for display purposes; for example, the name of the portal or portlet.	
DESCRIPTION	VARCHAR(500)	Null	A localized description of the object.	

Table 9-63 L10N_RESOURCE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
TITLE_CK	VARCHAR(160)	Null	Collation key column for the title; an ASCII-sortable string that will sort correctly in the given language.	9.2
DESCRIPTION_CK	VARCHAR(1000)	Null	Collation key column for the description; an ASCII-sortable string that will sort correctly in the given language.	9.2

L10N_RESOURCE_TYPE Database Table

This table defines portal resource types for localization.

Table 9-64 L10N_RESOURCE _TYPE Table Metadata

Column Name	Data Type	Null Value	Description
RESOURCE_TYPE	VARCHAR(80)	Not Null	PK – Type of resource to be localized; for example, `BOOK, DESKTOP, DESKTOP CATEGORY`.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.
APPLICATION_NAME	VARCHAR(100)	Not Null	The name of the application to which the resource belongs. `APPLICATION_NAME` is currently set to `PORTAL` for all types of resources to be localized.

WSRP Portal Framework Database Objects

Click to view, and use the magnifying tool to inspect individual tables in the logical entity-relation diagram for the WSRP Portal Framework tables.

The following WSRP-related tables are a subset of the PF_PORTLET_INSTANCE database table:

- PF_CONSUMER_PORTLETS Database Table
- PF_CONSUMER_PROPERTIES Database Table
- PF_CONSUMER_REGISTRY Database Table
- PF_PRODUCER_PROPERTIES Database Table
- PF_PRODUCER_REGISTRY Database Table
- PF_PORTLET_INSTANCE Database Table
- PF_PROXY_BOOK Database Table
- PF_PROXY_PAGE Database Table

PF_CONSUMER_PORTLETS Database Table

The PF_CONSUMER_PORTLETS table associates consumer IDs and portlet instance IDs so that when a consumer unregisters from a producer, the producer can clean up any portlets that it created for the consumer.

Table 9-65 PF_CONSUMER_PORTLETS Database Table

Column Name	Data Type	Null Value	Description
CONSUMER_ID	NUMBER	Not Null	PK/FK to `PF_PRODUCER_REGISTRY`.
PORTLET_ID	NUMBER	Not Null	PK/FK to `PF_PORTLET_INSTANCE`.

PF_CONSUMER_PROPERTIES Database Table

This table contains optional registration properties. You can set up the consumer to ask for these during registration.

Table 9-66 PF_CONSUMER_PROPERTIES Database Table

Column Name	Data Type	Null Value	Description
PRODUCER_ID	INTEGER	Not Null	PK/FK to `PF_PRODUCER_REGISTRY`.
PROPERTY_NAME	VARCHAR(80)	Not Null	PK – The name of the property.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was modified. The column's data is maintained using a database trigger.
PROPERTY_VALUE	VARCHAR (80)	Null	The value associated with the property name.

PF_CONSUMER_REGISTRY Database Table

This table contains registration handles that are assigned by the producer during registration by a consumer.

Table 9-67 PF_CONSUMER_REGISTRY Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
PRODUCER_ID	INTEGER	Not Null	PK – A unique, system-generated number to use as the record ID.	
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application (as defined in `config.xml`) to which the portal application is scoped.	
CREATION_DATE	DATE	Not Null	The date and time the row was created.	

Table 9-67 PF_CONSUMER_REGISTRY Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
MODIFIED_DATE	DATE	Not Null	The date and time the row was modified. The column's data is maintained using a database trigger.	
PRODUCER_HANDLE	VARCHAR(40)	Not Null	Uniquely identifies the producer to the consumer.	
SERVICE_DESCRIPTION_PORT_URL	VARCHAR(255)	Null	Optional. URL to the description service port offered by the producer.	9.2
MARKUP_PORT_URL	VARCHAR(255)	Null	Optional. URL to the markup service port offered by the producer.	9.2
COOKIE_PROTOCOL	NUMBER	Not Null	The cookie protocol. Values: 0 = None, 1 = Per User, 2 = Per Group.	
WSDL_URL	VARCHAR(255)	Not Null	URL to the WSDL offered by the producer.	
REQUIRES_REGISTRATION	NUMBER	Not Null	A boolean indicating that registration is required.	
REGISTRATION_PORT_URL	VARCHAR(255)	Null	URL to the registration service port offered by the producer (if offered).	
PORTLET_MANAGEMENT_PORT_URL	VARCHAR(255)	Null	URL to the portlet management service port offered by the producer (if offered).	
REGISTRATION_HANDLE	VARCHAR(255)	Null	Registration handle returned by the producer after registration.	

Table 9-67 PF_CONSUMER_REGISTRY Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
VENDOR_NAME	VARCHAR(255)	Null	Name of the vendor of the producer implementation.	
DESCRIPTION	VARCHAR(255)	Null	A description of the portlet.	
REGISTRATION_STATE	BLOB	Null	Registration state returned by the producer after registration.	

PF_PRODUCER_PROPERTIES Database Table

This table contains optional registration properties. The producer might be asked for these during registration.

Table 9-68 PF_PRODUCER_PROPERTIES Database Table

Column Name	Data Type	Null Value	Description
CONSUMER_ID	INTEGER	Not Null	PK/FK to `PF_CONSUMER_REGISTRY`.
PROPERTY_NAME	VARCHAR(80)	Not Null	PK – The name of the property.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was modified. This column's data is maintained using a database trigger.
PROPERTY_VALUE	VARCHAR(80)	Null	The value associated with the `PROPERTY_NAME`.

PF_PRODUCER_REGISTRY Database Table

This table contains producer-generated registration handles stored for each consumer during registration.

Table 9-69 PF_PRODUCER_REGISTRY Database Table

Column Name	Data Type	Null Value	Description
CONSUMER_ID	INTEGER	Not Null	PK – A unique system-generated number to use as the record ID.
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the J2EE web application to which the portal application is scoped.
CREATION_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was modified. This column's data is maintained using a database trigger.
CONSUMER_NAME	VARCHAR(80)	Null	A unique name that identifies the consumer. For the producer to assert user identity, the consumer name must correspond to the alias of the consumer's public key deployed in the producer's key store.
CONSUMER_AGENT	VARCHAR(80)	Null	Name and version of the consumer's vendor. The value must start with `productName.majorVersion.minorVersion` where `productName` identifies the product that the consumer installed for its deployment, and `majorVersion` and `minorVersion` are vendor-defined indications of the version of its product.

PF_PORTLET_INSTANCE Database Table

The consumer manages remote portlet-specific data from here. The framework inserts data into this table whenever a proxy portlet instance is created (including successors). When portlet instances are deleted, the `IS_SET_FOR_DESTROY` flag is set for subsequent cleanup.

Table 9-70 PF_PROXY_PORTLET_INSTANCE Database Table

Column Name	Data Type	Null Value	Description
PROXY_PORTLET_ INSTANCE_ID	INTEGER	Not Null	PK – A unique system-generated number to use as the record ID.
PRODUCER_ID	INTEGER	Not Null	FK to `PF_CONSUMER_REGISTRY`.
CREATED_DATE	DATE	Not Null	The date and time the row was created.
MODIFIED_DATE	DATE	Not Null	The date and time the row was modified. This column's data is maintained using a database trigger.
REQUIRES_URL_TEMPLATES	NUMBER	Not Null	A boolean indicating that URL templates are required by the producer.
TEMPLATES_STORED_ IN_ SESSION	NUMBER	Not Null	A boolean indicating whether the consumer should send templates with every request. The default is 1 = True.
PORTLET_STATE_CHANGE	NUMBER	Not Null	A flag that indicates how the consumer handles customizations of remote portlets. Based on the value of this flag, the consumer may clone remote portlets. Possible values include: 0 = Readonly (default) 1 = CBW(CloneBeforeWrite) 2 = Read/Write.
IS_PRODUCER_OFFERED	NUMBER	Not Null	Identifies the portlet as producer-offered. The defaults is 1 = True. If `IS_PRODUCER_OFFERED` is True, and the `PORTLET_INSTANCE_ID` is null, this `PF_PROXY_PORTLET_INSTANCE` is removed from the database during data cleanup processing.

Table 9-70 PF_PROXY_PORTLET_INSTANCE Database Table (Continued)

Column Name	Data Type	Null Value	Description
PORTLET_INSTANCE_ID	INTEGER	Null	FK to `PF_PORTLET_INSTANCE` for the proxy portlet. If the associated `PF_PORTLET_INSTANCE` row is deleted, the value of this column is set to null.
PORTLET_HANDLE	VARCHAR(255)	Null	The handle to the remote portlet as it is specified by the producer. The consumer uses portlet handles throughout the communication to address and interact with portlets using the producer.
DELETE_ERROR_CAUSE	VARCHAR(255)	Null	A description of the cause of the error, if an error is encountered while trying to delete the counter part of this proxy portlet on the producer.
PORTLET_STATE	BLOB	Null	Portlet state as returned by the producer after implicit/explicit cloning.
IS_PRIMARY	NUMBER	Not Null	A boolean indicating whether the proxy portlet is primary or not. When multiple versions of the same proxy portlet are being used on a consumer, operations that involve either using or removing the proxy portlet will be done on the primary version (unless specified that the operation affects all versions). The defaults is 0 (False).

PF_PROXY_BOOK Database Table

This table maintains information on proxy books for the WebLogic Portal Administration Console. If a user is adding new remote resources to the portal, this data can be used to let the user know if a particular proxy book is already in the library.

This table was added in WebLogic Portal 9.2.

Table 9-71 PF_PROXY_BOOK Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
PROXY_BOOK_ID	INTEGER	Not Null	PK – A unique, system-generated number to use as the property ID.	9.2
PRODUCER_HANDLE	VARCHAR(40)	Not Null	Uniquely identifies the producer to the consumer.	9.2
ORIGINAL_NAME	VARCHAR(80)	Not Null	Original name of the book on the producer; this name is tracked in order to find duplicates.	9.2
BOOK_DEFINITION_ID	INTEGER	Not Null	FK to PF_BOOK_DEFINITION	9.2
CREATION_DATE	DATE	Not Null	The date and time the row was created.	9.2
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.	9.2
IS_PRIMARY	NUMBER	Not Null	A boolean indicating whether the proxy book is primary or not. When multiple versions of the same proxy book are being used on a consumer, operations that involve either using or removing the proxy book will be done on the primary version (unless specified that the operation affects all versions). The defaults is 0 (False).	9.2

PF_PROXY_PAGE Database Table

This table maintains information on proxy pages for the WebLogic Portal Administration Console. If a user is adding new remote resources to the portal, this data can be used to let the user know if a particular proxy page is already in the library.

This table was added in WebLogic Portal 9.2.

Table 9-72 PF_PROXY_PAGE Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
PROXY_PAGE_ID	INTEGER	Not Null	PK – A unique, system-generated number to use as the property ID.	9.2
PRODUCER_HANDLE	VARCHAR(40)	Not Null	Uniquely identifies the producer to the consumer.	9.2
ORIGINAL_NAME	VARCHAR(80)	Not Null	Original name of the book on the producer; this name is tracked in order to find duplicates.	9.2
PAGE_DEFINITION_ID	INTEGER	Not Null	FK to PF_PAGE_DEFINITION	9.2
CREATION_DATE	DATE	Not Null	The date and time the row was created.	9.2
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.	9.2

Table 9-72 PF_PROXY_PAGE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
IS_PRIMARY	NUMBER	Not Null	A boolean indicating whether a proxy page is primary or not. When multiple versions of the same proxy page are being used on a consumer, operations that involve either using or removing the proxy page will be done on the primary version (unless specified that the operation affects all versions). The defaults is 0 (False).	9.2

Communities Framework Database Objects

Communities are a new feature in WebLogic Portal version 9.2. Communities are WebLogic Portal desktops that let users with common goals and interests work together in—and administer—a web-based environment. Whether for specific events, work groups, partners, or for any other groups that need to share information, communities provide a dedicated, secure, self-managed place to collaborate.

Click to view, and use the magnifying tool to inspect individual tables in the logical entity-relation diagram for the Communities Portal Framework tables.

Communities Portal Framework Data Dictionary Tables

The Communities Portal Framework system has the following tables:

- PF_COMMUNITY_DEFINITION Database Table
- PF_COMMUNITY_MEMBER Database Table
- PF_COMMUNITY_MEMBERSHIP Database Table
- PF_COMMUNITY_PROPERTY Database Table

- PF_COMMUNITY_PROPERTY_VALUE Database Table
- PF_FILE_TEMPLATE Database Table
- PF_INVITATION Database Table
- PF_INVITEE Database Table
- PF_INVITEE_PROPERTY Database Table
-
- PF_NOTIFICATION Database Table
- PF_NOTIFICATION_PAYLOAD Database Table

PF_COMMUNITY_DEFINITION Database Table

The PF_COMMUNITY_DEFINITION table contains community definition data.

Table 9-73 PF_COMMUNITY_DEFINITION Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
COMMUNITY_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the community definition ID.	9.2
INTERSECTION_ID	NUMBER	Not Null	FK to L10N_RESOURCE for the localization resource.	9.2
DESKTOP_PATH	VARCHAR(40)	Not Null	FK to PF_DESKTOP_DEFINITION.	9.2
PORTAL_PATH	VARCHAR(40)	Not Null	FK to PF_DESKTOP_DEFINITION.	9.2

Table 9-73 PF_COMMUNITY_DEFINITION Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
WEBAPP_NAME	VARCHAR(80)	Not Null	FK to PF_DESKTOP_DEFINITION. Name of the web application that hosts the desktop.	9.2
TIMEZONE	VARCHAR(80)	Not Null	Display TimeZone identifier. This can be used by Communities applications to format dates and times in a time zone specific to a community instance.	9.2
IS_ACTIVE	NUMBER	Not Null	Indicates whether the community is active; default = 0 (inactive).	9.2
IS_PUBLIC	NUMBER	Not Null	Indicates whether the community is public; default = 0 (not public).	9.2
IS_TEMPLATE	NUMBER	Not Null	Indicates whether the community is a template; default = 0 (not a template).	9.2
IS_ACCESS_TRACKING_ENABLED	NUMBER	Not Null	Indicates whether the community has access tracking enabled; default = 0 (not enabled).	9.2
IS_PERSONAL_PAGES_ENABLED	NUMBER	Not Null	Indicates whether the community has personal pages enabled; default = 0 (not enabled).	9.2

Table 9-73 PF_COMMUNITY_DEFINITION Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
IS_SELF_REG_ENABLED	NUMBER	Not Null	Indicates whether self-registration is enabled; default = 0 (not enabled).	9.2
IS_GLOBAL	NUMBER	Not Null	Indicates whether the community is global; default = 0 (not global).	9.2
CREATION_DATE	TIMESTAMP	Not Null	The date and time the row was created.	9.2
MODIFIED_DATE	TIMESTAMP	Not Null	The date and time the row was modified. This column's data is maintained using a database trigger.	9.2
EXPIRATION_DATE	TIMESTAMP	Not Null	The date and time this community will expire.	9.2
PARENT_COMMUNITY_ DEFINITION_ID	NUMBER	Null	FK - References parent community.	9.2
CALLBACK_CLASS_NAME	VARCHAR(255)	Null	The class name for the CommunityCallback implementation that is configured for this community, if one exists.	9.2

Table 9-73 PF_COMMUNITY_DEFINITION Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
PRIMARY_MESSAGING_ADDR	VARCHAR(80)	Null	The MessagingAddress name that identifies the default MessagingAddressType that is configured for use with this community. If this is null, the default MessagingAddressType for all community operations will be the type configured in `communities-config.xml`.	9.2
SELF_REG_URI	VARCHAR(255)	Null	URI pointing to the registration page for this community instance.	9.2
DEACTIVATED_URI	VARCHAR(255)	Null	URI to page when community is either deactivated or the membership of the user has been deactivated.	9.2

PF_COMMUNITY_MEMBER Database Table

The PF_COMMUNITY_MEMBER table contains user information for community members.

Table 9-74 PF_COMMUNITY_MEMBER Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
COMMUNITY_MEMBER_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the community member ID.	9.2
USER_NAME	VARCHAR(200)	Not Null	The WebLogic Server user name for the community member.	9.2
MEMBER_TYPE	NUMBER	Not Null	Identifies the community member as internal (1) or external (0).	9.2
MEMBER_STATUS	NUMBER	Not Null	Setting for member's community status. Values are: 0 = Disabled, 1 = Active.	9.2
CREATION_DATE	TIMESTAMP	Not Null	The date and time the row was created.	9.29.2
MODIFIED_DATE	TIMESTAMP	Not Null	The date and time the row was modified. This column's data is maintained using a database trigger.	9.2

PF_COMMUNITY_MEMBERSHIP Database Table

The PF_COMMUNITY_MEMBERSHIP table contains information regarding the membership status of community members.

Table 9-75 PF_COMMUNITY_MEMBERSHIP Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
COMMUNITY_MEMBERSHIP_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the community membership ID.	9.2
COMMUNITY_DEFINITION_ID	INTEGER	Not Null	FK - A foreign key to PF_COMMUNITY_DEFINITION.	9.2
COMMUNITY_MEMBER_ID	INTEGER	Not Null	FK - A foreign key to PF_COMMUNITY_MEMBER.	9.2
MEMBERSHIP_STATUS	NUMBER	Not Null	Setting to determine membership status. Values are: 1 = active, 0 = Disabled	9.2
CREATION_DATE	TIMESTAMP	Not Null	The date and time the row was created.	9.2
MODIFIED_DATE	TIMESTAMP	Not Null	The date and time the row was modified. This column's data is maintained using a database trigger.	9.2
LAST_ACCESS_DATE	TIMESTAMP	Null	If access tracking is enabled, this field contains a timestamp that is updated at each login event by the user associated with this CommunityMembership to the linked Community desktop.	9.2

PF_MEMBERSHIP_CAPABILITY Database Table

The PF_MEMBERSHIP_CAPABILITY table contains information about the capabilities of a member within the community.

Table 9-76 PF_COMMUNITY_MEMBERSHIP Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
COMMUNITY_MEMBERSHIP_ID	INTEGER	Not Null	PK/FK to COMMUNITY_MEMBERSHIP.	9.2
CAPABILITY	VARCHAR(80)	Not Null	PK - Text label applied to the community member. The only default security functionality with capabilities is that a community "creator" can delete a community and a community "owner" can manage a community.	9.2

PF_COMMUNITY_PROPERTY Database Table

The PF_COMMUNITY_PROPERTY table contains information regarding custom community properties, used to uniquely identify the community and its characteristics.

Table 9-77 PF_COMMUNITY_PROPERTY Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
COMMUNITY_DEFINITION_ID	INTEGER	Not Null	PK/FK – A foreign key to PF_COMMUNITY_DEFINITION.	9.2
PROPERTY_NAME	VARCHAR(100)	Not Null	PK – The property name.	9.2

Table 9-77 PF_COMMUNITY_PROPERTY Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
CREATION_DATE	TIMESTAMP	Not Null	The date and time the row was created.	9.2
MODIFIED_DATE	TIMESTAMP	Not Null	The date and time the row was modified. This column's data is maintained using a database trigger.	9.2
IS_MULTI_VALUED	NUMBER	Not Null	Boolean value.	9.2
DESCRIPTION	VARCHAR(255)	Null	Descriptive text to for this property.	9.2

PF_COMMUNITY_PROPERTY_VALUE Database Table

The PF_COMMUNITY_PROPERTY_VALUE table contains information regarding the actual value of the custom community property.

Table 9-78 PF_COMMUNITY_PROPERTY_VALUE Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
COMMUNITY_PROPERTY_VALUE_ID	INTEGER	Not Null	PK – A system-generated unique number to use as the community property value ID.	9.2
COMMUNITY_DEFINITION_ID	INTEGER	Not Null	A foreign key to PF_COMMUNITY_PROPERTY.	9.2
PROPERTY_NAME	VARCHAR(100)	Not Null	A foreign key to PF_COMMUNITY_PROPERTY.	9.2

Table 9-78 PF_COMMUNITY_PROPERTY_VALUE Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
CREATION_DATE	TIMESTAMP	Not Null	The date and time the row was created.	9.2
MODIFIED_DATE	TIMESTAMP	Not Null	The date and time the row was modified. This column's data is maintained using a database trigger.	9.2
PROPERTY_VALUE	VARCHAR(400)	Null	The value associated with the property name.	9.2

PF_FILE_TEMPLATE Database Table

The PF_FILE_TEMPLATE table records the location of a file type template. This includes the web application and the URI within the web application to the file, as well as an intersection ID that can be used to retrieve a localized name for the file template.

Table 9-79 PF_FILE_TEMPLATE Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
TEMPLATE_ID	INTEGER	Not Null	PK – A system-generated unique identifier for the file template.	9.2
INTERSECTION_ID	INTEGER	Not Null	A foreign key to L10N_RESOURCE for the Localization Resource.	9.2
TEMPLATE_URI	VARCHAR(255)	Not Null	URI pointing to the `.portal` file that this FileTemplate is based on.	9.2

Table 9-79 PF_FILE_TEMPLATE Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the web application that contains the `.portal` file that this FileTemplate is based.	9.2

PF_INVITATION Database Table

The PF_INVITATION table stores data related to community invitations.

Table 9-80 PF_INVITATION Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
INVITATION_ID	INTEGER	Not Null	PK – A system-generated unique identifier for the invitation ID.	9.2
COMMUNITY_DEFINITION_ID	INTEGER	Not Null	A foreign key to PF_COMMUNITY_ DEFINITION.	9.2
EXPIRATION_DATE	TIMESTAMP	Null	The expiration date for the Invitation, after which the Invitation is no longer active and cannot be consumed.	9.2

Table 9-80 PF_INVITATION Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
IS_SELF_DESTRUCT_ENABLED	NUMBER	Not Null	Indicates whether the framework should automatically clean up database information about the invitation after it has been consumed or timed out, or stored for possible later use in queries. Default = 0 (not enabled/false).	9.2
CREATION_DATE	DATE	Not Null	The date and time the row was created.	
INVITATION_PAYLOAD	VARCHAR(200)	Null	Application-specific text data that can be attached to an invitation, for use by a communities application during member registration based on this Invitation.	9.2
FROM_WLS_USER_NAME	VARCHAR(200)	Null	The WebLogic Server user name from which the invitation originated	9.2

PF_INVITEE Database Table

The PF_INVITEE table contains information regarding the target of the invitation.

Table 9-81 PF_INVITEE Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
INVITEE_ID	INTEGER	Not Null	PK – A system-generated unique identifier for the invitee ID.	9.2
INVITATION_ID	INTEGER	Not Null	A foreign key to PF_INVITATION.	9.2
INVITEE_ADDRESS	VARCHAR(255)	Null	The messaging address (usually an e-mail address) used to send invitations to external users.	9.2
INVITATION_STATUS	NUMBER	Not Null	The status of the invitation. Values are: 10 - Sent (default) 20 -Accepted 30 - Rejected 40 - Revoked 50 - Expired	9.2
MODIFIED_DATE	TIMESTAMP	Not Null	Date the status was updated.	9.2
USER_NAME	VARCHAR(200)	Null	The WebLogic Server user name if the invitation is for an existing user, but can be null when inviting external invitees who do not yet have a WebLogic Server user name.	9.2

Table 9-81 PF_INVITEE Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
INVITATION_VALIDATION_TEXT	VARCHAR(400)	Null	Validation text that is sent as part of the notification, which must then be provided by the invitee when accepting or declining an invitation. The format of this text is developer- definable.	9.2

PF_INVITEE_PROPERTY Database Table

The PF_INVITEE_PROPERTY table contains information regarding persistent invitee properties such as invitee attributes and capabilities

Table 9-82 PF_INVITEE_PROPERTY Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
INVITEE_ID	INTEGER	Not Null	PK/FK - A foreign key to PF_INVITEE.	9.2
PROPERTY_TYPE	NUMBER	Not Null	PK - Property type. Possible values are: 10 - Persisted Attribute 20 - Persisted Capability	9.2
PROPERTY_NAME	VARCHAR(100)	Not Null	PK - Property name.	9.2
PROPERTY_VALUE	VARCHAR(400)	Not Null	Value of the persisted attribute or capability.	9.2

PF_MESSAGING_ADDR Database Table

The PF_MESSAGING_ADDR table is reserved for future use.

PF_NOTIFICATION Database Table

The PF_NOTIFICATION table stores information about event notifications. Notifications for the Portal Framework are a mechanism for publishing availability, and subscribing to and delivering notification events within a portal. A notification event is a sibling to other types of portlet events as defined in the current inter-portlet eventing design, in that portlets can be authored in a declarative manner to listen for notifications using an event subscription tag within the portlet or portlet instance tag. The primary difference between notification events and normal portlet events is that the source of the notification comes from outside the user's portal desktop. Instead, notifications can come from other users, web applications, or even the framework itself.

Table 9-83 PF_NOTIFICATION Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
NOTIFICATION_ID	INTEGER	Not Null	PK – A system-generated unique identifier for the notification ID.	9.2
USER_NAME	VARCHAR(200)	Not Null	User name of the notification recipient.	9.2
PAYLOAD_ID	INTEGER	Not Null	A foreign key to PF_NOTIFICATION_PAYLOAD.	9.2
IS_CONSUMED	NUMBER	Not Null	Indicates whether the notification has been consumed; the notification remains active until it is explicitly consumed or until the expiration time occurs. Default = 0 (not consumed).	9.2

PF_NOTIFICATION_PAYLOAD Database Table

The PF_NOTIFICATION_PAYLOAD table stores information about notifications for individual users. Because a notification can be sent to groups of users and each user must individually acknowledge (remove) the notification, the notification table contains an entry for each user for each notification. Because the *payload* and other characteristics of a single notification batch are identical for each user, this information is separated and stored only once per notification to minimize redundancy in the database.

For portal desktops that are enabled for notification delivery, WebLogic Portal performs a query against the active notifications for a user prior to executing the portal lifecycle for each request.

Table 9-84 PF_NOTIFICATION_PAYLOAD Database Table

Column Name	Data Type	Null Value	Description	New or Changed Version
PAYLOAD_ID	INTEGER	Not Null	PK – A system-generated unique identifier for the payload ID.	9.2
NOTIFICATION_NAMESPACE	VARCHAR(80)	Not Null	Denotes the namespace for this Notification, which is should be generated as a unique identifier by the application that constructed the Notification. This is meant to help in avoiding collisions due to duplicate NOTIFICATION_NAMEs used by applications. For example, this could be the company name of the application developer that makes use of the Notifications framework.	9.2

Table 9-84 PF_NOTIFICATION_PAYLOAD Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
NOTIFICATION_NAME	VARCHAR(40)	Not Null	Name of the notification.	9.2
IS_SELF_DESTRUCT_ENABLED	NUMBER	Not Null	Indicates whether the notification framework should automatically clean up database information about the notification after it has been consumed or timed out, or stored for possible later use in queries. Default = 0 (not enabled/false).	9.2
IS_ACTIVE	NUMBER	Not Null	Indicates whether the notification is active. When a notification is first sent, its status is set to active for each user. Default = 1 (active).	9.2
EXPIRATION_DATE	TIMESTAMP	Not Null	Expiration time for the notification; the notification remains active until it is explicitly consumed or until the expiration time occurs.	9.2
SOURCE_WEBAPP_NAME	VARCHAR(80)	Not Null	Name of the source web application.	9.2
PORTAL_PATH	VARCHAR(40)	Null	Path for the portal.	9.2
DESKTOP_PATH	VARCHAR(40)	Null	Path for the desktop.	9.2

Table 9-84 PF_NOTIFICATION_PAYLOAD Database Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
DESKTOP_INSTANCE_TITLE	VARCHAR(20)	Null	Title of the desktop instance.	9.2
PORTLET_DEFINITION_LABEL	VARCHAR(80)	Null	Definition label of the portlet.	9.2
PORTLET_INSTANCE_LABEL	VARCHAR(80)	Null	Instance label of the portlet.	9.2
TARGET_WEBAPP_NAME	VARCHAR(80)	Null	Name of the target web application.	9.2
PAYLOAD	VARCHAR(4000)	Null	The content of the notification.	9.2

Content Management Database Objects

WebLogic Portal's content management system allows you to store content, track its progress, and incorporate content in your portal applications. It provides an easy integration between creating content and delivering that content to your users. Content creators can use WebLogic Portal's repositories to create content and portal developers use the content API and JSP tools to deliver content to portal visitors.

Click to view, and use the magnifying tool to inspect individual tables in the logical entity-relation diagram for the content management tables.

Content Management Data Dictionary Tables

The Content Management system has the following tables:

- CM_NODE Database Table
- CM_COLLABORATION_API_METADATA Database Table
- CM_PROPERTY Database Table
- CM_PROPERTY_CHOICE Database Table

- CM_PROPERTY_DEFINITION Database Table
- CM_WORKFLOW Table

CM_NODE Database Table

In the `CM_NODE` table, a node represents an element in a hierarchy that can either be a "Hierarchy Node" or a "Content Node." A hierarchy node can contain both other hierarchy and content nodes while a content node can contain only other content nodes. Nodes can contain properties based on the `ObjectClass` (schema) defined for it.

Both Content Nodes and Hierarchy Nodes can contain an ObjectClass and properties. Each node has a path that uniquely identifies it within the repository.

Table 9-85 CM_NODE Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
NODE_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.	
PARENT_NODE_ID	NUMBER	Null	FK – The node's parent record ID (`NODE_ID`).	
CREATION_DATE	DATE	Not Null	The date and time the row was created.	
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.	
OBJECT_CLASS_ID	NUMBER	Null	FK – The object class ID associated to the node.	
NODE_NAME	VARCHAR(50)	Not Null	The name of the node. The name is unique relative to its siblings. The name must not contain forward or backward slashes.	

Table 9-85 CM_NODE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
NODE_TYPE	NUMBER	Not Null	The node type. Possible values are: 0 = Unpopulated value 1 = Hierarchy Node 2 = Content Node	
NODE_STATUS	VARCHAR(40)	Null	The status of the node. The available values are defined by the application as property definition choices.	
CREATED_BY	VARCHAR(100)	Not Null	ID of the user who created the node.	
MODIFIED_BY	VARCHAR(100)	Null	ID of the user who last modified the node.	
CM_CREATION_DATE	TIMESTAMP	Not Null	The date and time the row was created. Maintained by the application.	
CM_MODIFIED_DATE	TIMESTAMP	Not Null	The date and time the row was last modified. Maintained by the application.	

Table 9-85 CM_NODE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
FULL_PATH	VARCHAR(882)	Null	AK – Each node has a path that uniquely identifies it within the repository. The path is defined in a UNIX-like format such as /a/b/c where "/" is the root and "a" ("a" is the Nodes NODE_NAME) is the root's child. The path must always begin with "/" and never end with it. So neither of the following are valid: a/b/c/d or /a/b/d/d/.	9.2
LIFECYCLE_STATUS	INTEGER	Null	The specific workflow status that the node version has been assigned:	
WORKFLOW_ID	INTEGER	Null	FK – The Workflow ID associated with the node.	
TITLE	VARCHAR(254)	Not Null	*Applies only to a GroupSpace repository.* Title for GroupSpace content, used for all GroupSpace content types.	
DESCRIPTION	VARCHAR(254)	Null	*Applies only to a GroupSpace repository.* Description for GroupSpace content; can be used by all GroupSpace content types.	

Table 9-85 CM_NODE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
VISIBILITY	VARCHAR(254)	Not Null	*Applies only to a GroupSpace repository.* Scopes viewing of GroupSpace data; used for GroupSpace content types. Valid visibility values are: • Personal – only viewable by a specific user but viewable in all GroupSpace Communities • Private – only viewable by a specific user and only viewable in the GroupSpace Community in which it was created • Community – viewable by all users within a GroupSpace Community	
OWNER	VARCHAR(254)	Not Null	*Applies only to a GroupSpace repository.* User who owns the GroupSpace content.	
FEED_ENTRY_ PUBLICATION_DATE	TIMESTAMP	Not Null	*Applies only to a GroupSpace repository.* Date that a Feed Entry was published. This is used only by content of type Feed_Entry.	

CM_COLLABORATION_API_METADATA Database Table

This table is used to store information about WebLogic Portal's collaboration portlets.

Table 9-86 CM_COLLABORATION_API_METADATA Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
NODE_ID	INTEGER	Not Null	PK - A unique, system-generated number to use as the record ID.	9.2
ICONTAINER_SUBCONTAINER_COUNT	INTEGER	Not Null	The number of shallow child containers for the current node.	9.2
IGWARE_CONTR_TOTAL_ITEM_COUNT	INTEGER	Not Null	The number of shallow child items for the current node.	9.2
IGWARE_CONTR_UNREAD_ITEM_COUNT	INTEGER	Not Null	The number of shallow child items that are marked as UNREAD for the current node.	9.2
IFORUM_CATEGORY_FORUM_COUNT	INTEGER	Not Null	The number of shallow forums beneath this category.	9.2
ITOPIC_SUPPRT_DEEP_TOPIC_COUNT	INTEGER	Not Null	The number of topics, infinitely deep, below this node.	9.2
ITOPIC_SUPPRT_TOPIC_COUNT	INTEGER	Not Null	The number of shallow topics below this node.	9.2

CM_OBJECT_CLASS Database Table

The ObjectClass is the schema for a Node (a content type). It has both an ID and a name that uniquely identifies it within a content repository. An ObjectClass can have PropertyDefinitions associated with it that define the shape of Properties required for a Node. This does not mean that the Property must contain a value, but simply that the Property must exist for the Node.

The ObjectClass may have a primary PropertyDefinition that defines the primary content Property for a Node. This allows for the definition of content in the schema since the schema does not distinguish between content and meta-content. A Node is considered valid in the repository only if its Properties conform to its ObjectClass PropertyDefinitions.

Table 9-87 CM_OBJECT_CLASS Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
OBJECT_CLASS_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.	
CREATION_DATE	DATE	Not Null	The date and time the row was created.	
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.	
OBJECT_CLASS_NAME	VARCHAR (100)	Not Null	AK – A unique name for the object class.	
PRIMARY_PROPERTY_ DEFINITION_ID	NUMBER	Null	FK – The `PROPERTY_DEFINITION_ID` for the primary `CM_PROPERTY_ DEFINITION` table row that defines the content for a node associated to the object class.	
IS_ABSTRACT	NUMBER	Not Null	A flag indicating if this type (object class) is abstract. An abstract type cannot be used as the object class of a node, but can be used as a base type for others using type inheritance or as a nested type. Possible values are: 0 = false 1 = true	9.2

Table 9-87 CM_OBJECT_CLASS Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
IS_SEARCHABLE	NUMBER	Not Null	A flag indicating if this property is searchable.	9.2
PARENT_OBJECT_CLASS_ID	INTEGER	Null	FK – The OBJECT_CLASS_ID of the parent type if this type is inheriting from a parent type.	9.2
PATH	VARCHAR (882)	Not Null	The string path of the object class. This will take the form of `/object_class_name` for most object classes. Those that use inheritance will take the form `/parent/child` or `/grandparent/parent/child`, and so on.	9.2
OBJECT_CLASS_DESCRIPTION	VARCHAR (254)	Null	Optional description for the content type.	9.2
WORKFLOW_ID	INTEGER	Null	The workflow ID associated with the type.	9.2

CM_PROPERTY Database Table

The `CM_PROPERTY` table identifies a property. The property consists of a name value pair; the name is unique relative to the `CM_NODE`, and the value is either a Date, BLOB, Boolean, Number, Float, or Varchar.

Only one value should be set on a given row; if the value is a `BLOB`, then all of the `BLOB_` columns can be set. If the `IS_MULTIVALUED` column is set to 1, then there will be multiple rows with the same property name and same `NODE_ID`. A property can represent both the content and meta-content for a Node.

Table 9-88 CM_PROPERTY Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
PROPERTY_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.	
CREATION_DATE	DATE	Not Null	The date and time the row was created.	
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.	
NODE_ID	NUMBER	Not Null	FK – The ID of the node that contains the property.	
PROPERTY_NAME	VARCHAR(894)	Not Null	The name of the property. It must be unique relative to its node.	9.2
PROPERTY_TYPE	NUMBER	Not Null	The type of the property: 0 = Boolean 1 = Number 2 = Float 3 = Varchar 4 = Date 5 = BLOB 6 = Nested 7 = Linked	
PROPERTY_DEFINITION_ID	NUMBER	Null	FK – The ID of the property definition to which this property must conform.	
BOOLEAN_VALUE	NUMBER	Null	True (1) for the Property if the `PROPERTY_TYPE` is Boolean (`PROPERTY_TYPE=0`).	

Table 9-88 CM_PROPERTY Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
DATETIME_VALUE	TIMESTAMP	Null	The datetime value for the property if the PROPERTY_TYPE is DATE (PROPERTY_TYPE=4).	
LONG_VALUE	NUMBER	Null	The long number or integer value for the Property if the PROPERTY_TYPE is NUMBER(PROPERTY_TYPE=1).	
DOUBLE_VALUE	FLOAT	Null	The floating point decimal number value for the Property if the PROPERTY_TYPE is FLOAT(PROPERTY_TYPE=2).	
TEXT_VALUE	VARCHAR(254)	Null	The textual property value for the Property if the PROPERTY_TYPE is VARCHAR(PROPERTY_TYPE=3).	
BLOB_VALUE	BLOB	Null	The binary large object for the Property if the PROPERTY_TYPE is BLOB(PROPERTY_TYPE=5).	
BLOB_VALUE_CHECKSUM_HEX	VARCHAR(32)	Null	The HEX representation of the MD5 checksum for each binary value in the table.	9.2
BLOB_FILE_NAME	VARCHAR(50)	Null	Name of the file associated with the BLOB_VALUE.	

Table 9-88 CM_PROPERTY Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
BLOB_FILE_SIZE	NUMBER	Null	The size of the file in bytes associated with the BLOB_VALUE.	
BLOB_CONTENT_TYPE	VARCHAR(100)	Null	The content type (mime type and character set) for the BLOB_VALUE. For example: "text/html;charset=iso8859-1"	
LINKED_NODE_ REPOSITORY_NAME	VARCHAR(254)	Null	For a property of type link, the name of the repository to which the link property refers.	9.2
LINKED_NODE_UID	VARCHAR(254)	Null	For a property of type link, the repository-specific node UID to which the link property refers.	9.2
NESTED_GROUP_ID	VARCHAR(254)	Null	An ID that acts as a grouping mechanism to relate a set of multi-valued nested properties together.	9.2

CM_PROPERTY_CHOICE Database Table

This table identifies the valid values or choices for a PropertyDefinition (row in the CM_PROPERTY_DEFINITION table). A property choice can identify a default choice (DEFAULT_PROERTY=1); if the creator of a Property does not choose different values, it is set as a Property value.

If the PropertyChoice value is defined as NULL (no value is supplied for the PROPERTY_TYPE), it allows for an empty choice. For example, a Property that has a String type (or TEXT_VALUE) could have three PropertyChoices - "blue," "red," "*," and null.

Table 9-89 CM_PROPERTY_CHOICE Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
PROPERTY_CHOICE_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.	
CREATION_DATE	DATE	Not Null	The date and time the row was created.	
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.	
PROPERTY_DEFINITION_ID	NUMBER	Not Null	FK – The ID of the property definition that contains the property choice.	
DEFAULT_PROPERTY	NUMBER	Not Null	Set to `1` if the property choice is a default, or `0` if it is not.	
BOOLEAN_VALUE	NUMBER	Null	True (1) for the Property if the `PROPERTY_TYPE` is `BOOLEAN (PROPERTY_TYPE=0)`.	
DATETIME_VALUE	TIMESTAMP	Null	The date/time value for the Property if the `PROPERTY_TYPE` is `DATE (PROPERTY_TYPE=4)`.	
LONG_VALUE	NUMBER	Null	The long number or integer value for the Property if the `PROPERTY_TYPE` is `NUMBER(PROPERTY_TYPE=1)`.	

Table 9-89 CM_PROPERTY_CHOICE Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
DOUBLE_VALUE	FLOAT	Null	The floating point decimal number value for the Property if the PROPERTY_TYPE is FLOAT(PROPERTY_TYPE =2).	
TEXT_VALUE	VARCHAR(254)	Null	The textual property value for the Property if the PROPERTY_TYPE is VARCHAR(PROPERTY_TY PE=3).	
BLOB_VALUE	BLOB	Null	The binary large object for the Property if the PROPERTY_TYPE is BLOB(PROPERTY_TYPE= 5).	
BLOB_VALUE_CHECKSUM _HEX	VARCHAR (32)	Null	The HEX representation of the MD5 checksum for each binary value in the table.	9.2
BLOB_FILE_NAME	VARCHAR(50)	Null	The name of the file associated with the BLOB_VALUE.	
BLOB_FILE_SIZE	NUMBER	Null	The size of the file in bytes associated with the BLOB_VALUE.	
BLOB_CONTENT_TYPE	VARCHAR(100)	Null	The content type (mime type and characterset) for the BLOB_VALUE. For example: “text/html;charset=iso8859 -1”	

CM_PROPERTY_DEFINITION Database Table

The `PropertyDefinition` table defines the shape of a property. It describes the property type (BLOB, Boolean, Varchar, Float, Date, Number), whether it is required, whether it is editable, the default value, and restricted values, if applicable. A `PropertyDefinition` can have 0..n PropertyChoices.

This is a list of values that you can select for a Property's values. Rules for a `PropertyDefinition` are as follows:

- If the `PropertyDefinition` contains a reference, it cannot be multi-valued or binary.
- If the `PropertyDefinition` is binary, it cannot be multi-valued or restricted and can have only one PropertyChoice.
- If the `PropertyDefinition` is boolean, it cannot be multi-valued. If the `PropertyDefinition` is restricted, then the Property's value(s) must be contained in the `PropertyChoice` list, or be null.

For example: consider a PropertyDefinition named *color*. It has PropertyChoices *blue*, *green*, and *red*. If the PropertyDefinition is restricted, then the value of a property defined by this PropertyDefinition cannot have a value other than *green*, *red*, *blue*, or null.

Table 9-90 CM_PROPERTY_DEFINITION Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
PROPERTY_DEFINITION_ID	NUMBER	Not Null	PK – A unique, system-generated number to use as the record ID.	
CREATION_DATE	DATE	Not Null	The date and time the row was created.	
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.	

Table 9-90 CM_PROPERTY_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
OBJECT_CLASS_ID	NUMBER	Not Null	FK – The OBJECT_CLASS_ID of the property definitions CM_OBJECT_CLASS.	
PROPERTY_NAME	VARCHAR(100)	Not Null	The name associated with the property definition. The combination of PROPERTY_NAME and OBJECT_ CLASS_ID for an Alternate Key for the CM_PROPERTY_DEFINI TION table.	
PROPERTY_TYPE	NUMBER	Not Null	The type of the property: 0 = Boolean 1 = Number 2 = Float 3 = Varchar 4 = Date 5 = BLOB	
IS_MANDATORY	NUMBER	Not Null	True if the value of a property must be set.	
IS_READ_ONLY	NUMBER	Not Null	True if the value of a property should not be set by an end user.	
IS_RESTRICTED	NUMBER	Not Null	True if the value of a property should come from the property choice values.	
IS_MULTI_VALUED	NUMBER	Not Null	True if there can be multiple rows with the same property name, node_id, but different property IDs.	

Table 9-90 CM_PROPERTY_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
COLUMN_NAME	VARCHAR(30)	Null	The name of a column added to the CM_NODE table that defines an explicit property.	
DESCRIPTION	VARCHAR(254)	Null	A description of the property definition.	
IS_SEARCHABLE	NUMBER	Not Null	Defines whether values of this property definition should be indexed by the full text search engine. Possible values are: 0 = Not searchable (the default) 1 = Searchable	9.2
PROPERTY_DEFINITION_TYPE	NUMBER	Not Null	The type of the property definition. Native definitions are defined on the containing object class. Inherited definitions are populated through type inheritance, and overridden definitions are also populated through type inheritance, but changed by the inheriting (child) type. Possible values are: 1 = Native (the default) 2 = Inherited 3 = Overridden	9.2

Table 9-90 CM_PROPERTY_DEFINITION Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
NESTED_OBJECT_CLASS_ID	INTEGER	Not Null	FK – The OBJECT_CLASS_ID of the nested type. An object class can "nest" the set of definitions as defined by another object class.	9.2

CM_WORKFLOW Table

This table was added in WebLogic Portal version 9.2. It stores information for the Content Management workflow.

Table 9-91 CM_WORKFLOW Table

Column Name	Data Type	Null Value	Description	New or Changed Version
WORKFLOW_ID	INTEGER	Not Null	PK – A unique, system-generated number to use as the workflow ID.	9.2
WORKFLOW_NAME	VARCHAR (50)	Not Null	The name of the workflow.	9.2
WORKFLOW_DOCUMENT	BLOB	Not Null	The binary representation of the workflow document.	9.2
CREATION_DATE	DATE	Not Null	The date and time the row was created.	9.2
MODIFIED_DATE	DATE	Not Null	The date and time the row was last modified. This column's data is maintained using a database trigger.	9.2
WORKFLOW_COMMENT	VARCHAR(254)	Null	Optional comment for the workflow.	9.2

Content Management Virtual Database Objects

The BEA repository provides automatic versioning.

Click to view, and use the magnifying tool to inspect individual tables in the logical entity-relation diagram for Virtual Content Repository (versioning) object tables.

Content Management Virtual Object Data Dictionary Tables

The Content Management system has the following tables for virtual objects:

- CMV_NODE Table
- CMV_NODE_ASSIGNED_ROLE Table
- CMV_NODE_VERSION Table
- CMV_PROPERTY Table
- CMV_VALUE Table
- CMV_NODE_VERSION_PROPERTY Table

CMV_NODE Table

This table uniquely identifies a content-managed node from a BEA repository (that is, CM_NODE table) that has been versioned and is being edited within the Content Management Virtual Repository.

Table 9-92 CMV_NODE Table

Column Name	Data Type	Null Value	Description
NODE_ID	VARCHAR (254)	Not Null	PK – A unique, system-generated number to use as the record ID.
NODE_NAME	VARCHAR (50)	Not Null	The name of the node.
REPOSITORY_NAME	VARCHAR (254)	Not Null	The name of the repository where the node was created and published.
IS_LOCKED	NUMBER	Not Null	Flag to determine if the record is locked.
OBJECT_CLASS_ID	VARCHAR (254)	Null	The object class ID that is associated with the node.

Table 9-92 CMV_NODE Table (Continued)

Column Name	Data Type	Null Value	Description
ASSIGNED_TO_USER_NAME	VARCHAR(200)	Null	Username to which the node is assigned.

CMV_NODE_ASSIGNED_ROLE Table

This table uniquely identifies all roles for a given node that have authorization to view or alter the node.

Table 9-93 CMV_NODE_ASSIGNED_ROLE Table

Column Name	Data Type	Null Value	Description
NODE_ID	VARCHAR(254)	Not Null	PK/FK – The ID of the node that roles are associated with. Foreign key relationship to CMV_NODE table.
ROLE_NAME	VARCHAR(254)	Not Null	PK – The name of the role.

CMV_NODE_VERSION Table

This table uniquely identifies all the versions of a mode within the Virtual Content Repository.

Table 9-94 CMV_NODE_VERSION Table

Column Name	Data Type	Null Value	Description
NODE_ID	VARCHAR(254)	Not Null	PK/FK – The ID of the node for which versions have been created. Foreign key relationship to CMV_NODE table.
NODE_VERSION_ID	VARCHAR(254)	Not Null	PK – The unique version ID for the node.
CM_MODIFIED_DATE	TIMESTAMP	Not Null	Time the node version was last edited.
MODIFIED_BY	VARCHAR(100)	Not Null	Username of the person who last edited the node version.

Table 9-94 CMV_NODE_VERSION Table

Column Name	Data Type	Null Value	Description
VERSION_COMMENT	VARCHAR(254)	Not Null	Comment added to a node version when saving.
LIFECYCLE_STATUS	INTEGER	Null	Specific life cycle status that the node version has been assigned (for example, In Progress, Published, and so on).

CMV_PROPERTY Table

This table uniquely identifies a property that can be associated with a node version. For example, some properties of a book might be author, title, and subject.

Table 9-95 CMV_PROPERTY Table Metadata

Column Name	Data Type	Null Value	Description	New or Changed Version
PROPERTY_ID	VARCHAR(254)	Not Null	PK – A unique, system-generated number to use as the record ID.	
CREATION_DATE	TIMESTAMP	Not Null	The date and time the row was created; the default is the current timestamp.	
MODIFIED_DATE	TIMESTAMP	Not Null	Date and time the row was last modified; the default is the current timestamp. This column's data is maintained using a database trigger.	
PROPERTY_NAME	VARCHAR(894)	Not Null	The name of the property. It must be unique relative to its node.	9.2

Table 9-95 CMV_PROPERTY Table Metadata (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
PROPERTY_TYPE	NUMBER	Not Null	The type of the property: 0 = Boolean 1 = Number 2 = Float 3 = Varchar 4 = Date 5 = BLOB	

CMV_VALUE Table

This table uniquely identifies a value for a given property. For example, a property `SUBJECT` for a `BOOK` might have a value of `FINANCE`.

Only one value is set on a given record. If the value is `BLOB`, then all the `BLOB_` columns can be set.

Table 9-96 CMV_VALUE Table

Column Name	Data Type	Null Value	Description	New or Changed Version
PROPERTY_ID	VARCHAR(254)	Not Null	PK/FK – ID of the property with which the values are associated. Foreign key relationship to CMV_PROPERTY.	
VALUE_ID	VARCHAR(254)	Not Null	PK – A unique, system-generated number to use as the value ID.	
CREATION_DATE	TIMESTAMP	Not Null	The date and time the row was created; the default is the current timestamp. This column's data is maintained using a database trigger.	

Table 9-96 CMV_VALUE Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
MODIFIED_DATE	TIMESTAMP	Not Null	Date and time the row was last modified; the default is the current timestamp. This column's data is maintained using a database trigger.	
BOOLEAN_VALUE	NUMBER	Not Null	Flag to determine if property is a Boolean value: 1= True, 0 = False.	
DATETIME_VALUE	TIMESTAMP	Null	The datetime value for the property if the PROPERTY_TYPE is DATE.	
LONG_VALUE	NUMERIC (20)	Null	The long number or integer value for the property if the PROPERTY_TYPE is NUMBER.	
DOUBLE_VALUE	FLOAT	Null	The floating point decimal number value for the property value if the PROPERTY_TYPE is FLOAT.	
TEXT_VALUE	VARCHAR(254)	Null	The textual property value if the PROPERTY_TYPE is VARCHAR.	
BLOB_VALUE	BLOB	Null	The binary large object for the property value if the PROPERTY_TYPE is BLOB.	
BLOB_VALUE_CHECKSUM_HEX	VARCHAR(32)	Null	The HEX representation of the MD5 checksum for each binary value in the table.	9.2

Table 9-96 CMV_VALUE Table (Continued)

Column Name	Data Type	Null Value	Description	New or Changed Version
BLOB_FILE_NAME	VARCHAR (50)	Null	The name of the file associated with BLOB_VALUE.	
BLOB_FILE_SIZE	INTEGER	Null	The size of the file (in bytes) associated with BLOB_VALUE.	
BLOB_CONTENT_TYPE	VARCHAR (100)	Null	The content type (MIME and character set) for the BLOB_VALUE. For example: "text/html;charset=iso8859-1"	
LINKED_NODE_ REPOSITORY_NAME	VARCHAR(254)	Null	For a property of type link, the name of the repository to which the link property refers.	9.2
LINKED_NODE_UID	VARCHAR(254)	Null	For a property of type link, the repository-specific node UID to which the link property refers.	9.2
NESTED_GROUP_ID	VARCHAR (254)	Null	An ID that acts as a grouping mechanism to relate a set of multi-valued nested properties together.	9.2

CMV_NODE_VERSION_PROPERTY Table

This table uniquely identifies a relationship between a CMV_NODE_VERSION and CMV_PROPERTY.

Table 9-97 CMV_NODE_VERSION_PROPERTY Table

Column Name	Data Type	Null Value	Description
NODE_ID	VARCHAR(254)	Not Null	PK/FK – ID of the node with which the properties are associated. Foreign key relationship to CMV_NODE_VERSION.
NODE_VERSION_ID	VARCHAR(10)	Not Null	PK/FK – ID of the node version with which the properties are associated. Foreign key relationship to CMV_NODE_VERSION.
PROPERTY_ID	VARCHAR(254)	Not Null	PK/FK – ID of the property with which the node versions are associated. Foreign key relationship to CMV_PROPERTY.

APPENDIX A

Scripts and Properties Files

This appendix describes the SQL scripts and properties files that specify the settings for database creation and upgrade, and perform Database Definition Language (DDL) commands for WebLogic Portal and personalization data structures. It includes the following sections:

- WebLogic Portal DDL Modules
- Personalization DDL Modules
- The database.properties File
- Scripts to Create the GroupSpace Repository Database
- Scripts to Create an Additional Content Management Repository
- Scripts to Manually Upgrade from Version 8.1 SP4 or SP5
- Scripts to Drop Deprecated Compoze Database Tables
- Scripts to Drop Deprecated RDBMS Authenticator Tables

WebLogic Portal DDL Modules

WebLogic Portal DDL modules (SQL scripts) are provided in the following directories:

WL_HOME\portal\db\db2

WL_HOME\portal\db\oracle

WL_HOME\portal\db\pointbase

WL_HOME\portal\db\sql_server

WL_HOME\portal\db\sybase

Each of these directories, except for `pointbase`, has an `admin` subdirectory, which contains the scripts that create the database user or database, and the appropriate database objects (such as tablespaces, bufferpools, and so on, depending on the database requirements).

Insert commands for bootstrap data that must be inserted into tables in each WebLogic Portal database are provided in the following directory:

WL_HOME\portal\db\data\required\xx_insert_system_required_data.sql

WebLogic Portal DDL is provided in files named as follows:

*xx*_create_fkeys.sql

*xx*_create_indexes.sql

*xx*_create_tables.sql

*xx*_create_triggers.sql

*xx*_create_views.sql

*xx*_drop_constraints.sql

*xx*_drop_fkeys.sql

*xx*_drop_indexes.sql

*xx*_drop_tables.sql

*xx*_drop_views.sql

In these file names, *xx* is one of the prefixes listed Table A-1.

Table A-1 WebLogic Portal DDL Module File Prefixes

Prefix	Description
au	Anonymous user
cm and cm9	Content management and GroupSpace
cmv and cmv9	Content management
comm	Communities
groupspace	GroupSpace

Table A-1 WebLogic Portal DDL Module File Prefixes

Prefix	Description
dep9	Deprecated Compoze/Collaboration portlets
pf and pf9	Framework and localization
wlcs	WebLogic Commerce Services
wps	WebLogic Portal Services

Tip: The scripts with the `comm` and `groupspace` prefixes, and those with a `9` suffix, contain new and updated DDL for WebLogic Portal version 9.2.

Personalization DDL Modules

Personalization DDL modules (SQL scripts) are provided in the following directories:

WL_HOME\common\p13n\db\db2

WL_HOME\common\p13n\db\oracle

WL_HOME\common\p13n\db\pointbase

WL_HOME\common\p13n\db\sql_server

WL_HOME\common\p13n\db\sybase

Insert commands for bootstrap data that must be inserted into tables in each personalization database are provided in the following directory:

WL_HOME\common\p13n\db\data\required\xx_insert_system_required_data.sql

Personalization DDL is provided in files named as follows:

*xx*_create_fkeys.sql

*xx*_create_indexes.sql

*xx*_create_tables.sql

*xx*_create_triggers.sql

*xx*_drop_constraints.sql

*xx*_drop_fkeys.sql

*xx*_drop_indexes.sql

*xx*_drop_tables.sql

In these file names, *xx* is one of the prefixes listed Table A-2.

Table A-2 Personalization DDL Module File Prefixes

Prefix	Description
bt	Behavior tracking
dep9	Deprecated WebLogic Portal RDBMS Authenticator
er	Visitor entitlements and delegated administration
p13n & p13n9	Users, groups, and user profiles
seq	Sequencer

Tip: The scripts with a 9 suffix contain new and updated DDL for WebLogic Portal version 9.2.

The database.properties File

Database scripts use the database.properties file for the following purposes:

- To connect to the database
- To drop, create, or alter database objects
- To perform data inserts

You can find the database.properties file in any domain directory that contains WebLogic Portal. Database scripts, such as create_db.cmd and create_db.sh, use the database.properties file that is located in the same directory from which you start the script.

Setting the Database Parameters in the Properties File

Use the database.properties file to specify the database you plan to use with WebLogic Portal.

Note: The `groupspace_database.properties` file, used to create the GroupSpace repository database, is described in the next section.

The following is the section in the file where you choose your database vendor.

```
# Set database= to a valid database.
# Valid databases are:  pointbase, oracle, sql_server, sybase and db2
#-------------------------------------------------------------------------
database=pointbase
```

In addition to modifying the database name, you must specify the appropriate values for the database user, password, host, and so on. You can use an encrypted password, as described in "Encrypting Passwords" on page 3-2. The following shows the PointBase and Oracle sections of the properties file.

```
#-------------------------------------------------------------------------
#  For the database specified above, fill in the appropriate @DB_USER@
#  and @DB_PASSWORD@ settings and complete the url parameters by
#  setting:  @DB_HOST@, @DB_PORT@, @DB_NAME@
#  Note: @DB_NAME@ for sql_server and sybase is often the same as user
#
#  To use an encrypted password for any database use:
#    weblogic.security.Encrypt to obtain the encrypted password.
#  The saltFile used to encrypt/decrypt passwords is the domains
#    <DOMAIN_HOME>/security/SerializedSystemIni.dat saltFile.
#
#-------------------------------------------------------------------------
pointbase.user=WEBLOGIC
pointbase.password=WEBLOGIC
pointbase.driver=com.pointbase.jdbc.jdbcUniversalDriver
pointbase.url=jdbc:pointbase:server://localhost:9093/weblogic_eval
#-------------------------------------------------------------------------
oracle.user=@DB_USER@
```

```
oracle.password=@DB_PASSWORD@

oracle.driver=weblogic.jdbc.oracle.OracleDriver

oracle.url=jdbc:bea:oracle://@DB_HOST@:@DB_PORT@;SID=@DB_NAME@

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

You can specify a log file other than the default.

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

# logFile= the file that output will be logged to

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

logFile=create_db.log
```

The `files=` setting indicates which SQL scripts to execute. The default setting is appropriate for creating the main WebLogic Portal database.

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

# files= points to the jdbc.index files, which points to the .sql

#   files, to be executed by create_db

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

files=${env.WL_HOME}/common/p13n/db/${database}/jdbc_index/jdbc.index,
${env.WL_HOME}/portal/db/${database}/jdbc_index/jdbc.index
```

You can also change the debug setting or send all SQL to a file rather than executing it, as described in the comments in the properties file. You can then execute the SQL file against the database.

Scripts to Create the GroupSpace Repository Database

You create the GroupSpace database repository with the `create_db.cmd/.sh` command, but using the `groupspace_database.properties` properties file instead of the `database.properties` properties file. The format is the same as the `database.properties` file, described in the previous section. The jdbc.index `files=` setting specifies a content management repository database with the appropriate SQL scripts to support GroupSpace:

```
files=${env.WL_HOME}/portal/db/${database}/jdbc_index/CM/jdbc.index,
${env.WL_HOME}/portal/db/${database}/jdbc_index/GROUPSPACE/jdbc.index
```

When you create the GroupSpace database, you must use the `-database.properties=` parameter to indicate the correct properties file to use, for example:

```
create_db.cmd -database.properties=groupspace_database.properties
```

Scripts to Create an Additional Content Management Repository

If you need to create an additional content management repository, you use `create_db.cmd/.sh`. However, you must use a properties file that you have customized instead of the `database.properties` properties file. You must also update the jdbc.index `files=` setting to specify the content management SQL scripts to run in the file properties file you create. For more information, see the Content Management Guide.

Scripts to Manually Upgrade from Version 8.1 SP4 or SP5

If you did not use the WebLogic Upgrade Wizard to perform the database upgrade from 8.1 SP4 or SP5, you must upgrade manually. See the Upgrade Guide for details on the upgrade process.

Scripts to Drop Deprecated Compoze Database Tables

After you have upgraded to version 9.2, you can drop the tables associated with Compoze/Collaboration using the following script:

WL_HOME\portal\db*dbms_name*\dep9_drop_tables.sql

Scripts to Drop Deprecated RDBMS Authenticator Tables

If you do not upgrade your user store using the WebLogic Upgrade Wizard during the domain upgrade process, you can perform a manual upgrade later. The script to upgrade from the WebLogic Portal-specific RDBMS Authenticator to the WebLogic SQL Authenticator is:

WL_HOME\common\p13n\db*dbms_name*\upgrade_fromdbmsauth_tosqlauth.sql

After you have upgraded to the WebLogic SQL Authenticator, you can drop the tables associated with the WebLogic Portal RDBMS Authenticator using the following script:

WL_HOME\common\p13n\db*dbms_name*\dep9_drop_tables.sql

Sample WLST Scripts

The sample `build.xml`, `oraracconf.py` and `oraracconf.py.properties` listed in this appendix provide an example of how to configure an existing WebLogic Portal domain to point to an Oracle RAC database. For more information, see "Using WebLogic Portal with Oracle RAC" on page 6-9.

Listing B-1 oraracconf.py

```
from java.lang import Exception
from jarray import array
from com.bea.plateng.domain.script.jython import WLSTException
import re


class SkipConfError(Exception):
  def __init__(self, msg):
    self.msg = msg

  def __str__(self):
    return repr(self.msg)

class OracleRAC:
  instance = None
```

```
  def __init__(self):
    try:
      self.sid = oracle_rac_sid
    except NameError, ne:
      raise SkipConfError, "Oracle RAC disabled, skipping. "

    print "Re-configuring data sources to target Oracle RAC: " + self.sid
    self.initClusterSize()
    self.vips=[]
    self.ports=[]
    self.sids=[]
    for i in range(self.size):
      index = i + 1
      self.vips.append( eval("oracle_rac_vip_" + str(index)) )
      self.ports.append( eval("oracle_rac_port_" + str(index)) )
      self.sids.append( eval("oracle_rac_sid_" + str(index)) )
    self.initDriverURL()
    self.xa_driver = oracle_xa_driver
    self.nonxa_driver = oracle_nonxa_driver
    OracleRAC.instance = self

  def initClusterSize(self):
    self.size = 0
    while true:
      try:
        eval('oracle_rac_vip_' + str(self.size+1))
        self.size += 1
      except NameError:
        return

  def initDriverURL(self):
    self.driver_url = 'jdbc:oracle:thin:@(DESCRIPTION =(ADDRESS_LIST ='
    for i in range(self.getSize()):
     self.driver_url += '(ADDRESS = (PROTOCOL = TCP)(HOST = ' + self.vips[i]
+ ')(PORT = ' + self.ports[i] + '))'
    self.driver_url +=
'(FAILOVER=on)(LOAD_BALANCE=load_balancing_var))(CONNECT_DATA =(SERVER =
DEDICATED)(SERVICE_NAME = ' + self.sid + ')))'
```

```
  def getSize(self):
    if self.size < 0:
      self.initClusterSize()
    return self.size

  def getDriverDSURL(self, loadbal):
    return self.driver_url.replace('load_balancing_var',loadbal)

  def getMultiDSURL(self, node):
    index = node - 1
    return 'jdbc:oracle:thin:@'+ self.vips[index] +':'+ self.ports[index]
+':'+ self.sids[index]


class DataSource:

  def __init__(self, dsName, user, passwd):
    self.dsName = dsName
    self.user = user
    self.passwd = passwd

    cd('/JDBCSystemResource/' + dsName + '/JdbcResource/' + dsName +
'/JDBCDataSourceParams/NO_NAME_0')
    jarray_jndi_names=get('JNDINames')
    self.jndi_names=[]
    for jname in jarray_jndi_names:
      self.jndi_names.append(jname)

  def deleteDataSource(self, dsName):
    try:
      WLDomain.instance.all_datasources.index(dsName)
      cd('/')
      print "Deleting datasource: " + dsName
      delete(dsName,'JDBCSystemResource')
      print dsName + " deleted!"
    except ValueError:
      print dsName + " does not exist!"
```

```
  def createPhysicalDataSource(self, dsName, jndiName, xaProtocol, url,
xa_driver, user, passwd):
    print 'Creating Physical DataSource ' + dsName
    self.deleteDataSource(dsName)

    cd('/')

    sysRes = create(dsName, "JDBCSystemResource")

    cd('/JDBCSystemResource/' + dsName + '/JdbcResource/' + dsName)
    dataSourceParams=create('dataSourceParams','JDBCDataSourceParams')
    dataSourceParams.setGlobalTransactionsProtocol(xaProtocol)
    cd('JDBCDataSourceParams/NO_NAME_0')
    print "Setting JNDI Names: "
    print jndiName
    set('JNDIName',jndiName)

    cd('/JDBCSystemResource/' + dsName + '/JdbcResource/' + dsName)
    connPoolParams=create('connPoolParams','JDBCConnectionPoolParams')
    connPoolParams.setMaxCapacity(20)
    connPoolParams.setInitialCapacity(5)
    connPoolParams.setCapacityIncrement(1)
    connPoolParams.setTestConnectionsOnReserve(true)
    connPoolParams.setTestTableName('SQL SELECT 1 FROM DUAL')

    cd('/JDBCSystemResource/' + dsName + '/JdbcResource/' + dsName)
    driverParams=create('driverParams','JDBCDriverParams')
    driverParams.setUrl(url)
    if xa_driver == "true":
      driverParams.setDriverName(OracleRAC.instance.xa_driver)
    else:
      driverParams.setDriverName(OracleRAC.instance.nonxa_driver)
    driverParams.setPasswordEncrypted(passwd)
    cd('JDBCDriverParams/NO_NAME_0')
    create(dsName,'Properties')
    cd('Properties/NO_NAME_0')
```

```
    create('user', 'Property')
    cd('Property/user')
    cmo.setValue(user)

    if xaProtocol != "None":
      cd('/JDBCSystemResource/' + dsName + '/JdbcResource/' + dsName)
      XAParams=create('XAParams','JDBCXAParams')
      XAParams.setKeepXaConnTillTxComplete(true)
      XAParams.setXaRetryDurationSeconds(300)
      XAParams.setXaTransactionTimeout(120)
      XAParams.setXaSetTransactionTimeout(true)
      XAParams.setXaEndOnlyOnce(true)


assign('JDBCSystemResource',dsName,'Target',WLDomain.instance.targetServer
)
    print dsName + ' successfully created.'

class MultiDataSource(DataSource):
  def __init__(self, dsName, user, passwd):
    DataSource.__init__(self, dsName, user, passwd)
    self.xa_protocol = eval(dsName.replace('-','_') + '_xa_protocol')
    self.xa_driver = eval(dsName.replace('-','_') + '_xa_driver')
    self.mp_algorithm = eval(dsName.replace('-','_') + '_mp_algorithm')

  def configure(self):
    print 'Creating Multi DataSource ' + self.dsName
    self.deleteDataSource(self.dsName)

    ds_list = ''
    for i in range(OracleRAC.instance.size):
      index = i + 1
      physical_ds_name = self.dsName + '-' + str(index)
      self.createPhysicalDataSource(physical_ds_name, physical_ds_name,
self.xa_protocol, OracleRAC.instance.getMultiDSURL(index), self.xa_driver,
self.user, self.passwd)
      if i > 0:
        ds_list += ','
```

```
      ds_list += physical_ds_name

    cd('/')

    sysRes = create(self.dsName, "JDBCSystemResource")

    cd('/JDBCSystemResource/' + self.dsName + '/JdbcResource/' +
self.dsName)
    dataSourceParams=create('dataSourceParams','JDBCDataSourceParams')
    dataSourceParams.setAlgorithmType(self.mp_algorithm)
    dataSourceParams.setDataSourceList(ds_list)
    cd('JDBCDataSourceParams/NO_NAME_0')
    print "Setting JNDI Names: "
    print self.jndi_names
    set('JNDINames',self.jndi_names)
    set('GlobalTransactionsProtocol',self.xa_protocol)


assign('JDBCSystemResource',self.dsName,'Target',WLDomain.instance.targetS
erver)
    print 'Multi DataSource '+ self.dsName + ' successfully created.'

class PhysicalDataSource(DataSource):
  def __init__(self, dsName, user, passwd):
    DataSource.__init__(self, dsName, user, passwd)
    self.loadbalance = eval(dsName.replace('-','_') + '_loadbalance')

  def configure(self):
    print "Re-configuring existing datasource: " + self.dsName
    self.createPhysicalDataSource(self.dsName, self.jndi_names, 'None',
OracleRAC.instance.getDriverDSURL(self.loadbalance), 'false', self.user,
self.passwd)
    print self.dsName + ' successfully re-configured.'

class WLDomain:
  instance = None
  def __init__(self):
```

```
    try:
      self.db_user = database_user
      self.db_passwd = database_passwd
    except NameError, ne:
      print "No databse user/password specified.  Will not configure
portalDataSource, etc."

    try:
      self.gs_user = groupspace_user
      self.gs_passwd = groupspace_passwd
    except NameError, ne:
      print "No groupspace cmrepo user/password specified.  Will not
configure appGroupSpaceDataSource"


    try:
      readDomain(domain_dir)
      server=ls('Server').splitlines()
      self.targetServer = re.split('\s+', server[0])[1]
    except IndexError, ie:
      raise SkipConfError, "No valid domain found at: " + domain_dir + ".
Skipping RAC configuration. "
    except NameError, ne:
      print "Required parameter domain_dir not specified! "
      sys.exit(-1)

    self.all_datasources = ls('/JDBCSystemResource').splitlines()
    for i in range(len(self.all_datasources)):
      self.all_datasources[i]= re.split('\s+', self.all_datasources[i])[1]

    WLDomain.instance = self

  def configure(self):
    for dsName in self.all_datasources:
      print "Processing datasource: " + dsName + " ..."

      try:
        is_mp=eval(dsName.replace('-','_') + '_is_mp')
```

```
        except NameError:
          print "Skipping unknown datasource: " + dsName
          continue

        try:
          if dsName == 'appsGroupSpaceDataSource':
            user = self.gs_user
            passwd = self.gs_passwd
          else:
            user = self.db_user
            passwd = self.db_passwd
        except AttributeError, ae:
          print "Skipping un-used datasource: " + dsName
          continue

        if is_mp=="true":
          MultiDataSource(dsName, user, passwd).configure()
        else:
          PhysicalDataSource(dsName, user, passwd).configure()

    updateDomain()
    print "Successfully re-configured domain for Oracle RAC! "


for i in range(1,len(sys.argv)):
  exec sys.argv[i]

try:
  OracleRAC()
  WLDomain().configure()
except SkipConfError, sce:
  print sce
```

Listing B-2 build.xml

```
<?xml version="1.0"?>
<project name="WLP_AND_RAC" default="configServer">
  <target name="configServer">
      <property name="domain_dir" value="."/>
     <wlst  debug="false"
            failOnError="false"
            fileName="./oraracconf.py"
            properties="./oraracconf.py.properties"
            arguments="domain_dir='${domain_dir}'">
     </wlst>
  </target>
</project>
```

Listing B-3 oraracconf.py.properties

```
database_user=WEBLOGIC_8
database_passwd=WEBLOGIC_8
groupspace_user=WEBLOGIC_9
groupspace_passwd=WEBLOGIC_9

oracle_rac_sid=DBSRAC10
oracle_xa_driver=oracle.jdbc.xa.client.OracleXADataSource
oracle_nonxa_driver=oracle.jdbc.OracleDriver

oracle_rac_vip_1=rnhp380-c11-23-vip
oracle_rac_port_1=1521
oracle_rac_sid_1=DBSRAC101

oracle_rac_vip_2=rnhp380-c11-25-vip
oracle_rac_port_2=1521
oracle_rac_sid_2=DBSRAC102

p13nDataSource_is_mp=true
p13nDataSource_xa_protocol=None
```

```
p13nDataSource_xa_driver=false
p13nDataSource_mp_algorithm=Load-Balancing

appsGroupSpaceDataSource_is_mp=true
appsGroupSpaceDataSource_xa_protocol=None
appsGroupSpaceDataSource_xa_driver=false
appsGroupSpaceDataSource_mp_algorithm=Load-Balancing

portalDataSourceNeverXA_is_mp=true
portalDataSourceNeverXA_xa_protocol=OnePhaseCommit
portalDataSourceNeverXA_xa_driver=false
portalDataSourceNeverXA_mp_algorithm=Failover

portalDataSourceAlwaysXA_is_mp=true
portalDataSourceAlwaysXA_xa_protocol=TwoPhaseCommit
portalDataSourceAlwaysXA_xa_driver=true
portalDataSourceAlwaysXA_mp_algorithm=Load-Balancing

portalDataSource_is_mp=true
portalDataSource_xa_protocol=OnePhaseCommit
portalDataSource_xa_driver=false
portalDataSource_mp_algorithm=Load-Balancing
```