BEA WebLogic Platform™

Managing WebLogic Platform Database Resources

Version 8.1 Service Pack 6
Revised:June 2006

Contents

CHAPTER 1

Configuring Database Resources

BEA WebLogic Platform delivers a set of integrated administration tools for managing your database resources. This section provides the following guidelines, procedures, and references:

- Configuring Database Resources in a Domain
- Important Recommendations for Configuring WebLogic JDBC
- Switching Databases
- Preparing Your Database for Production
- Related Topics

Configuring Database Resources in a Domain

You can configure your database resources as part of the process of creating your domain and/or while your application is running in an existing domain:

If you configure database resources when . . .	Use the . . .	For more information, see . . .
Creating your domain	Configuration Wizard	"Configuring a Database Service" in *Creating WebLogic Platform Configurations Using the Configuration Wizard*
Your application is running in an existing domain	WebLogic Server Administration Console	"Configuring Database Connectivity" in *WebLogic Server System Administration*

Important Recommendations for Configuring WebLogic JDBC

The following table provides some important recommendations for configuring WebLogic JDBC using distributed and local transactions.

If you . . .	Then . . .
Plan to use an XA driver	Configure only one Tx Data Source per XA-compliant JDBC connection pool. Note that you can specify multiple JNDI names per Tx Data Source, separated by semicolons.
Plan to use distributed transactions with non-XA drivers	Configure all Tx Data Sources on the same connection pool.
Plan to use only local transactions	Use a Data Source. Otherwise, use a Tx Data Source, but ensure that you suspend and subsequently resume any global transaction currently associated with your thread around your local transaction boundary.
Start a local transaction	Always commit the transaction. Do not assume autocommit.

Plan to use a JMS JDBC Store	Configure WebLogic JMS on a non-XA JDBC connection pool.
Have created multiple domains	Each domain must reference unique database schemas.

For more information about configuring WebLogic JDBC to use distributed and local transactions, see:

- FAQs: WebLogic JDBC
- FAQs: JTA
- Guidelines for Changing to an XA Configuration in *Creating WebLogic Configurations Using the Configuration Wizard*

Switching Databases

Note: Before proceeding, review the databases and drivers supported for WebLogic Platform 8.1, as described in "Supported Database Configurations" in *Supported Configurations*.

When you install WebLogic Platform, all sample domains, and all the default configuration templates used to create domains, are preconfigured to use the PointBase database. To configure a domain with a database other than PointBase, such as Oracle, see the appropriate instructions.

- For information that applies to all domains, see the following information in *Creating WebLogic Configurations Using the Configuration Wizard*:
  - Guidelines for Changing to an XA Configuration
  - How Do I: Create a Domain Using a Different Database?
- For information specific to WebLogic Portal domains, see:
  - Using a Microsoft SQL Server Database in *WebLogic Portal Database Administration Guide*
  - Using an Oracle Database in *WebLogic Portal Database Administration Guide*
  - Using a Sybase Database in *WebLogic Portal Database Administration Guide*
  - Using a DB2 Database in *WebLogic Portal Database Administration Guide*
  - How Do I: Create a Domain Using a Database with an XA Driver? in *Creating WebLogic Configurations Using the Configuration Wizard*

- For information specific to WebLogic Integration domains, see:

  How Do I: Create a Domain Using a Database with an XA Driver? in *Creating WebLogic Configurations Using the Configuration Wizard*

Optimally, the same enterprise-quality database management system (or an equivalent system) should be used for development and production deployment.

Preparing Your Database for Production

Databases used with the WebLogic Integration, WebLogic Portal, and WebLogic Workshop components of WebLogic Platform require some preparation before they can be delivered for production. For instructions on preparing a database for production, see the following documentation:

- WebLogic Integration — See "Configuring a Production Database" in *Managing WebLogic Integrations Solutions*.
- WebLogic Portal — See the chapter about your database type in *Database Administration Guide*.
- WebLogic Workshop — See "To Manually Create Required Resources on the Production Server" in How Do I: Deploy a WebLogic Workshop Application to a Production Server? in the *WebLogic Workshop Online Help*.
- WebLogic Platform — See "Configuring the Production Database" in *Deploying WebLogic Platform Applications*.

Related Topics

The following table provide references to information that may help you manage your database resources.

To learn more about . . .	See . . .
Administration environment: default environment for WebLogic Portal 8.1	The following sections in the *WebLogic Portal Database Administration Guide*: • Overview of Database Administration for WebLogic Portal • Data Dictionary

Bulk Loader: how to import, export, and delete trading partner management (TPM) data	Using the Trading Partner Bulk Loader in *Managing WebLogic Integration Solutions*
Database configuration: DB2, MS SQL Server, Oracle9i RAC, Oracle 10g RAC, Sybase	Configuring Databases
JDBC configuration: default	Default JDBC Configurations for WebLogic Platform
Adapters: DBMS sample adapters (J2EE-compliant adapters, each of which includes a JSP-based GUI)	Learning to Develop Adapters Using the DBMS Sample Adapters
CMP entity beans	How Do I: Generate an Entity Bean from a Database Table?
Database controls: how to simplify access to a relational database from your application by using database controls	Database Control
JDBC applications	*Programming WebLogic JCBC*
	dev2dev: Java Database Connectivity (JDBC)
JDBC database stores for WebLogic JMS: how to regenerate existing stores	JDBC Database Utility in *Programming WebLogic JMS*

Configuring Databases

This section presents guidelines and caveats about the following:

- Configuring DB2
- Configuring MS SQL Server
- Configuring Sybase
- For information on Configuring WebLogic Platform with Oracle RAC in 8.1 SP5, go to: "Configuring WebLogic Platform to Use Oracle9i or 10g RAC" on page 3-2.

Note: Descriptions of how to configure a WebLogic Platform 8.1 SP4 domain with Oracle 9i RAC, and WebLogic Server 8.1 SP4 with Oracle9i and 10g RAC are provided in *Important Usage Notes for Previous Releases of WebLogic Platform 8.1*. This document can be found at: `http://e-docs.bea.com/platform/docs81/sp_notes/sp_notes.html`.

Configuring DB2

If your configuration includes DB2:

- Sample scripts for creating DB2 buffer pools and tablespaces are located in the following directory: `WL_HOME/portal/db/db2/8/admin`.
- Before you start creating database objects for DB2, make sure an 8K-buffer pool and 8K-tablespaces are available to the DB2 database user. For example, to create tablespaces in a DB2 database and grant them to a user, you can use the following SQL statements:

```
CONNECT to <database_name> user <DB2admin> using <DB2admin_password>
;
CREATE BUFFERPOOL BP8K
    ALL NODES
    SIZE 500
    PAGESIZE 8K
    NOT EXTENDED STORAGE
;
CREATE REGULAR TABLESPACE WEBLOGIC_DATA_4K
    PAGESIZE 4K
    MANAGED BY SYSTEM
    USING('<data_container>')
    BUFFERPOOL IBMDEFAULTBP
;
CREATE REGULAR TABLESPACE WEBLOGIC_DATA_8K
    PAGESIZE 8K
    MANAGED BY SYSTEM
    USING('<data_container>')
    BUFFERPOOL BP8K
;
CREATE TEMPORARY TABLESPACE TEMPSPACE_8K
    IN NODEGROUP IBMTEMPGROUP
    PAGESIZE 8K
    MANAGED BY SYSTEM
    USING('<data_container>')
    BUFFERPOOL BP8K
;
GRANT USE OF TABLESPACE WEBLOGIC_DATA_4K TO USER <user_name>
;
GRANT USE OF TABLESPACE WEBLOGIC_DATA_8K TO USER <user_name>
;
```

Note that the DB2 database needs to be stopped and restarted after the buffer pool is created. This procedure enables the tablespaces to use the newly created buffer pool.

- To make sure that Portal applications can be run successfully on UDB 8.1 in WebLogic Portal domains, configure your DB2 database, with the following parameters as guidelines for minimums:

  – `Dynamic Sections`: 20,000

  – `pplheapsz`: 24,000

  – `ckcachesz`: 2500

  If the values shown here are not set as the minimum values, heavy Portal activity may exceed database capacity.

- For WebLogic Platform 8.1 SP4, BEA provides a patch for the IBM DB2 database. BEA recommends that you apply this patch to your system if you are using DB2 with WebLogic Platform 8.1 SP4. The patch is available at the following URL:

  `http://dev2dev.bea.com/products/wlplatform81/patch/wlplat81sp4_db2_patch.jsp`

  Additional recommendations about using DB2 with WebLogic Platform are available in the README file provided with the patch.

Configuring MS SQL Server

If your configuration includes MS SQL Server:

1. To use JDBC distributed transactions through JTA, your system administrator should install Microsoft SQL Server JDBC XA procedures using the procedure described at the following URL:

   `http://e-docs.bea.com/wls/docs81/jdbc_drivers/mssqlserver.html#install_jta`

2. Check to see whether you have MS SQL Server version 8.00.818. To check the version number, enter the following commands:

   ```
   OSQL -Usa -P -S
   1>select @@version
   2>go
   ```

   If MS SQL Server is already at version 8.00.818, proceed to step 3. If SQL Server version is at 8.00.810 or below, obtain the hot fix from the following link. This hot fix upgrades MS SQL Server to 8.00.818.

   `http://www.microsoft.com/downloads/results.aspx?productID=&freetext=ms03-031&DisplayLang=en`

3. Apply one of the patches identified in this section to fix a problem in Windows that will enable MS SQL Server to run properly. These patches are available from Microsoft TechNet and published with the Microsoft Security Bulletin MS04-012 at the following URL:

   `http://www.microsoft.com/technet/security/bulletin/MS04-012.mspx`

   To install this patch on Windows 2000 systems with Service Pack 2, 3, or 4, complete the following steps:

   a. Download Microsoft Security Bulletin MS04-012 from the following URL:

      `http://www.microsoft.com/downloads/details.aspx?FamilyId=FBD38C36-D1D3-47A2-A5D5-6C8F27FDCC40`

b. Click Download under *Security Update for Windows 2000 (KB828741)* to download the file `Windows2000-KB828741-x86-ENU.EXE`.

c. Shut down all applications, the MS SQL Server, and the DTC services.

d. Run the extracted executable file `Windows2000-KB828741-x86-ENU.EXE`.

e. Reboot the system.

f. Start the MS SQL Server and DTC services, if they are not started automatically.

To install this patch on a Windows XP or Windows XP Service Pack 1 system, complete the following steps:

a. Download Microsoft Security Bulletin MS04-012 from the following URL:

http://www.microsoft.com/downloads/details.aspx?FamilyId=D488BBBB-DA77-448D-8FF0-0A649A0D8FC3

b. Click Download under *Security Update for Windows XP (KB828741)* to download the file `WindowsXP-KB828741-x86-ENU.EXE`.

c. Shut down all applications, the MS SQL Server, and the DTC services.

d. Run the extracted executable file `WindowsXP-KB828741-x86-ENU.EXE`.

e. Reboot the system.

f. Start the MS SQL Server and DTC services, if they are not started automatically.

Configuring Sybase

If your configuration includes Sybase:

- When you are creating the domain using the Configuration Wizard, and specifying values in the Configure JDBC Connection Pools page in the Configuration Wizard, enter `SelectMethod=cursor` in the Additional Properties field. (This step is not required for the `portalPool` connection pool.)
- Enable DDL in transactions for the Sybase database. Log in to isql as the Sybase 'sa' user, and execute the following commands. (In these commands, `WEBLOGIC` is used as an example of the Sybase database name.)

```
exec master.dbo.sp_dboption WEBLOGIC, 'ddl in tran' ,true
go
use WEBLOGIC
go
```

```
checkpoint
go
```

- If you are configuring your domain for XA, complete the following steps:

  a. Install the license for Distributed Transaction Management (DTM), and configure the Sybase server as follows:

  ```
  sp_configure "enable DTM",1
  go

  sp_configure "enable xact coordination",1
  go
  ```

  b. Copy the sample `xa_config` file from the `SYBASE_INSTALL\OCS-12_0\sample\xa-dtm` subdirectory up three levels to the `SYBASE_INSTALL` directory, where `SYBASE_INSTALL` is the directory of your Sybase server installation. For example:

  ```
  $ SYBASE_INSTALL\xa_config
  ```

  c. Edit the `xa_config` file. In the first `[xa]` section, modify the sample server name to indicate the correct server name.

For more information about creating XA domains, see "Creating XA Domains Using Configuration Templates" in *Creating WebLogic Configurations Using the Configuration Wizard*, at the following URL:

`http://e-docs.bea.com/platform/docs81/confgwiz/examples.html`

Using WebLogic Platform with Oracle RAC

This section presents:

Overview of Oracle Real Application Clusters

The Oracle Real Application Clusters (RAC) feature is a software option you can add to an Oracle 9i or 10g database to enable multiple database instances to access the same database (storage) simultaneously via cluster technology. The benefits of using Oracle RAC are:

- High availability—Because all instances access the same database, access to stored data is not interrupted if a single instance fails.
- Load balancing—Oracle RAC makes it possible to distribute the load to multiple database nodes within a cluster.
- Scalability—Oracle RAC enables you to accommodate increasing numbers of users by adding database nodes to a cluster, increasing storage, or doing both.

Configuring WebLogic Platform to Use Oracle9i or 10g RAC

BEA supports Oracle9i and 10g Real Application Clusters (RAC) for use with WebLogic Platform 8.1 SP5. A WebLogic Platform domain configured with Oracle9i or 10g RAC supports the use of WebLogic JDBC MultiPools and global (XA) transactions with connection pool failover and load balancing.

To use WebLogic Platform with Oracle RAC (9i or 10g), you must meet specific configuration requirements for both systems. The configuration requirements for WebLogic Platform are provided in this section. The configuration requirements for Oracle RAC include clustering software and a shared storage solution. To find out how to set up hardware and software required for an environment that supports Oracle RAC, see "Using WebLogic Server with Oracle RAC" in *Programming WebLogic JDBC*:

`http://e-docs.bea.com/wls/docs81/jdbc/oracle_rac.html`

Configuration Options

The following configuration options are provided:

- MultiPools
- Connect-Time Failover
- Global Transactions

MultiPools

WebLogic Platform 8.1 SP5 supports the use of JDBC MultiPools with an Oracle RAC database. A MultiPool is a group of connection pools, but to an application, the characteristics of a MultiPool appear identical to those of a single basic connection pool. Each connection pool in a MultiPool is assigned to a different instance of the same database. If an application cannot obtain a connection from a particular connection pool because database connectivity from the pool is down, WebLogic Platform attempts to obtain a connection from the next connection pool in the MultiPool. Optionally, a MultiPool can be configured to provide load balancing, too.

Depending on whether a MultiPool is configured for load balancing or failover, it may have the attributes defined in the following table.

Table 3-1 MultiPool Attributes

Attribute	Description
`AlgorithmType= "High-Availability" or "Load-Balancing"`	With the High-Availability option, connection requests are served by the first available pool in the list. When a connection pool becomes defunct, connection requests are served by the next connection pool in the list. With the Load-Balancing option, connection requests are distributed among available connection pools.
`FailoverRequestIfBusy= "true"`	When invoked with the High-Availability algorithm, this attribute enables failover when all connections in a connection pool are in use.
`HealthCheckFrequencySeconds ="300"`	Controls how often WebLogic Platform checks automatically disabled connection pools in a MultiPool to determine whether connections can be recreated and whether a connection pool can be re-enabled. The default value is 300 seconds.

To connect WebLogic Platform to multiple Oracle RAC nodes using MultiPools, you must:

1. Configure a JDBC connection pool with the Oracle Thin driver for each RAC instance in your RAC cluster.
2. Configure a MultiPool using the algorithm for load balancing or the algorithm for high availability.
3. Add all your connection pools to the MultiPool.

A detailed version of this procedure is available in "Setting Up MultiPools with Global Transactions" on page 3-4.

Note: WebLogic Platform does not support the use of MultiPools with JMS JDBC Stores. If your application makes use of JMS JDBC Stores, you must configure your JMS JDBC Store to use Oracle RAC with connect-time failover. See "Using Connect-Time Failover" on page 3-11.

Connect-Time Failover

If MultiPools cannot be used in your application (for example, when you use a JMS JDBC Store, which does not support the use of MultiPools) and if load balancing is not required, you can configure your connection pools to use connect-time (driver-level) failover. If, under such circumstances, a RAC instance becomes unavailable, you can use the connect-time failover feature of the Oracle Thin driver to handle connection failover. In such a configuration, whenever WebLogic Platform tests a connection and the connection fails, the driver replaces the failed connection with a new one. Then the driver determines which RAC instance to use, based on instance availability.

To connect WebLogic Platform to multiple Oracle RAC nodes using connection pools configured for connect-time failover, configure a JDBC connection pool for each RAC instance in your RAC cluster with the Oracle Thin driver. A detailed version of this procedure is available in "Using Connect-Time Failover" on page 3-11.

Global Transactions

WebLogic Platform supports global (XA) transactions and the two-phase commit protocol for enterprise applications. A global transaction updates multiple resources, such as databases, in a coordinated manner.

The two-phase commit protocol is a method of coordinating a single transaction across two or more resources. It guarantees data integrity by ensuring that transactional updates are either committed in all of the participating resources, or fully rolled back out of all resources. When updates are rolled back, the state of the relevant resources reverts to the state in which those resources were running before the transaction. In other words, either all the participating resources are updated, or none of them are updated.

In contrast, a local (non-XA) transaction is one that is committed to a single resource.

Setting Up MultiPools with Global Transactions

Note: WebLogic Platform 8.1 SP5 is certified to support Multipools with XA only on Oracle RAC.

This section provides information about the following topics:

- Methods of Configuring MultiPools with Global Transactions
- WebLogic JDBC Components that Must Be Configured for Oracle RAC
- Rules for Connection Pools within a MultiPool
- Required Attributes of Connection Pools within a MultiPool
- Sample config.xml Code

Methods of Configuring MultiPools with Global Transactions

WebLogic Platform gives you a choice of several methods for configuring Oracle RAC to use global transactions. Table 3-2 summarizes the methods for which instructions are provided later in this chapter.

Table 3-2 Configuration Procedures

To...	Use...
Create a domain that uses Oracle RAC	One of the following methods: • Edit and run a WLST Offline script. This method is recommended. For more information, see "Configuring a WebLogic Platform Domain with WLST Offline" on page 3-13. • Configure some parameters with the Configuration Wizard, and then manually edit the resulting `config.xml` file. For more information, see "Creating XA Domains that Use Oracle RAC" in *Creating WebLogic Configurations Using the Configuration Wizard*.
Upgrade an existing Platform XA domain for Oracle RAC use	WLST Offline script—If you already have a platform domain set up with XA resources, you can use a WLST Offline script to RAC-enable that domain. For information about configuring an XA domain, see the XA guidelines in "Creating XA Domains Using Configuration Templates" in *Creating WebLogic Configurations Using the Configuration Wizard*. For information about updating a domain to use Oracle RAC, see "Updating an Existing XA Domain to Use Oracle RAC" on page 3-16.

In addition, you may configure these components through other tools, such as the Administration Console, the `weblogic.Admin` command-line utility, or a JMX program. Instructions for these methods are not provided here.

WebLogic JDBC Components that Must Be Configured for Oracle RAC

WebLogic Platform gives you the option of configuring Oracle RAC to use global transactions by defining and configuring the following WebLogic JDBC components:

- Six XA Connection Pools — `portalPool1`, `portalPool2`, `cgPool1`, `cgPool2`, `bpmArchPool1`, `bpmArchPool2`
- One non-XA Connection Pool — `cgJMSPool-nonXA`
- Three MultiPools — `portalMultiPool`, `cgMultiPool`, `bpmArchMultiPool`

- Three predefined data sources set up for XA — `portalFrameworkPool`, `cgDataSource`, `bpmArchDataSource`
- Three predefined data sources set up for non-XA — `p13n_trackingDataSource`, `p13nDataSource`, `cgDataSource-nonXA`
- One JMS JDBCStore — `cgJMSStore`

The following procedure describes the configuration parameters you must define in order to connect WebLogic Platform to multiple Oracle RAC nodes using MultiPools with global transactions:

1. Define XA connection pools. For each connection pool, specify values for the connection pool name, driver name, DBMS name, DBMS host name, DBMS port, user name, and user password. To make sure the values you specify for *`dbname1`*, *`dbname2`*, *`dbhost1`*, *`dbhost2`*, *`user_name`*, and *`user_password`* are appropriate for your Oracle database setup, ask your database administrator to supply them.

   Table 3-3 shows some sample values. Here *`dbname1`* and *`dbhost1`* represent the name of instance 1 of the database and the name of its host machine, respectively. *`dbname2`* and *`dbhost2`* represent the name of instance 2 of the database and the name of its host machine.

Table 3-3 Sample Values for Connection Pools

Name	Driver Name	DBMS Name	DBMS Host	DBMS Port	User Name	User Password
`portalPool1`	`oracle.jdbc.xa.client.OracleXADataSource`	*`dbname1`*	*`dbhost1`*	1521	*`user_name`*	*`user_password`*
`portalPool2`	`oracle.jdbc.xa.client.OracleXADataSource`	*`dbname2`*	*`dbhost2`*	1521	*`user_name`*	*`user_password`*
`cgPool1`	`oracle.jdbc.xa.client.OracleXADataSource`	*`dbname1`*	*`dbhost1`*	1521	*`user_name`*	*`user_password`*
`cgPool2`	`oracle.jdbc.xa.client.OracleXADataSource`	*`dbname2`*	*`dbhost2`*	1521	*`user_name`*	*`user_password`*

Table 3-3 Sample Values for Connection Pools

Name	Driver Name	DBMS Name	DBMS Host	DBMS Port	User Name	User Password
bpmArchPool1	oracle.jdbc.xa.client.OracleXADataSource	*dbname1*	*dbhost1*	1521	*user_name*	*user_password*
bpmArchPool2	oracle.jdbc.xa.client.OracleXADataSource	*dbname2*	*dbhost2*	1521	*user_name*	*user_password*

2. Assign connection pools to MultiPools, as shown in Table 3-4.

Table 3-4 Assignments of Connection Pools to MultiPools

Assign these Connection Pools...	To this MultiPool...
portalPool1, portalPool2	portalMultiPool
cgPool1, cgPool2	cgMultiPool
bpmArchPool1, bpmArchPool2	bpmArchMultiPool

3. Assign JDBC data sources to a MultiPool or a connection pool, as shown in Table 3-5.

Table 3-5 Assignment of Data Sources

Assign this data source...	To this MultiPool or Connection Pool...
portalFrameworkPool	portalMultiPool
cgDataSource	cgMultiPool
bpmArchDataSource	bpmArchMultiPool
p13n_trackingDataSource	cgJMSPool-nonXA
p13nDataSource	cgJMSPool-nonXA
cgDataSource-nonXA	cgJMSPool-nonXA

Rules for Connection Pools within a MultiPool

The following rules apply to the XA connection pools within a MultiPool:

- All connection pools must be homogeneous; make sure that either all of them or none of them are XA connection pools.
- If you choose to specify all XA-related attributes, you must set all of them to the same values for each connection pool. The following XA attributes are available:
  - `XARetryDurationSeconds`
  - `SupportsLocalTransaction`
  - `KeepXAConnTillTxComplete`
  - `NeedTxCtxOnClose`
  - `XASetTransactionTimeout`
  - `XATransactionTimeout`
  - `NewXAConnForCommit`
  - `RollbackLocalTxUponConnClose`
  - `RecoverOnlyOnce`
  - `KeepLogicalConnOpenOnRelease`

Required Attributes of Connection Pools within a MultiPool

For each connection pool within a MultiPool, the following attributes must be set:

- Oracle JDBC Thin XA driver. For example:

  ```
  DriverName="oracle.jdbc.xa.client.OracleXADataSource"
  URL="jdbc:oracle:thin:@dbhost1:1521:dbname1"
  ```

- `KeepXAConnTillTxComplete="true"`
  - Forces the connection pool to reserve a physical database connection and maintain a connection to an application throughout transaction processing, that is, until the distributed transaction is complete.
  - Required for proper transaction processing with Oracle RAC.

- `XARetryDurationSeconds="300"`

  Specifies the maximum amount of time in which the WebLogic Platform transaction manager can retry XA recover, commit, and rollback calls.

- `CountOfTestFailuresTillFlush="1"`
  - Specifies the number of test failures allowed before WebLogic Platform closes all connections in the connection pool to minimize the delay caused by further database testing. For more information about this attribute, see "JDBC Connection Pool Testing Enhancements" in *Programming WebLogic JDBC*.
  - Minimizes the amount of time allowed for failover when an Oracle RAC node fails.

- `CountOfRefreshFailuresTillDisable= "1"`

  Specifies the number of test failures allowed before WebLogic Platform disables the connection pool to minimize the delay in handling the connection request caused by a database failure.

- `TestConnectionsOnReserve="true"`
  - Enables testing of a database connection when an application reserves a connection from the connection pool. For more details about this attribute, see "Testing Connection Pools and Database Connections" in *Programming WebLogic JDBC*.
  - Required to enable failover to another RAC node.

- `TestTableName="`*`name_of_small_table`*`"`
  Specifies the table used to test a physical database connection. For details about `TestTableName`, see "JDBCConnection Pool →Configuration →Connections" in the *WebLogic Server Administration Console Online Help*.

Sample config.xml Code

The following code provides an example of how two connection pools, a MultiPool, and an associated data source are configured in a `config.xml` file:

```
<JDBCConnectionPool
        CapacityIncrement="1"
        CountOfRefreshFailuresTillDisable= "1"
        CountOfTestFailuresTillFlush="1"
        DriverName="oracle.jdbc.xa.client.OracleXADataSource"
        InitialCapacity="5"
        KeepXAConnTillTxComplete="true"
        MaxCapacity="100"
        Name="cgPool1"
        PasswordEncrypted="{3DES}lBifoTsg8fc="
        Properties="user=user_name"
        RefreshMinutes="1"
        SupportsLocalTransaction="true"
        Targets="cgServer"
        TestConnectionsOnReserve="true"
        TestTableName="dual"
        URL="jdbc:oracle:thin:@dbhost1:1521:dbname1"
        XARetryDurationSeconds="300"
        XASetTransactionTimeout="true"
        XATransactionTimeout="302"/>
<JDBCConnectionPool
        CapacityIncrement="1"
        CountOfRefreshFailuresTillDisable= "1"
        CountOfTestFailuresTillFlush="1"
        DriverName="oracle.jdbc.xa.client.OracleXADataSource"
        InitialCapacity="5"
        KeepXAConnTillTxComplete="true"
        MaxCapacity="100"
        Name="cgPool2"
        PasswordEncrypted="{3DES}lBifoTsg8fc="
        Properties="user=user_name"
        RefreshMinutes="1"
        SupportsLocalTransaction="true"
        Targets="cgServer"
        TestConnectionsOnReserve="true"
        TestTableName="dual"
        URL="jdbc:oracle:thin:@dbhost2:1521:dbname2"
        XARetryDurationSeconds="300"
        XASetTransactionTimeout="true"
        XATransactionTimeout="302"/>
<JDBCMultiPool
        Name="cgMultiPool"
        PoolList="cgPool1,cgPool2"
#The following attribute is valid only for high availability.
        FailoverRequestIfBusy="true"
        HealthCheckFrequencySeconds="300"
        Targets="cgServer"
        AlgorithmType="High-Availability"/>
<JDBCTxDataSource
        JNDIName="cgDataSource"
        Name="cgDataSource"
        EnableTwoPhaseCommit="true"
        PoolName="cgMultiPool"
        Targets="cgServer"/>
```

Note: Line breaks are included only for readability.

Using Connect-Time Failover

If MultiPools cannot be used in your application (for example, when you use a JMS JDBC Store, which does not support the use of MultiPools) and if load balancing is not required, you can configure your connection pools to use connect-time failover.

Note: Connection pools cannot be configured with connect-time load balancing in Release 8.1 SP5.

To connect WebLogic Platform to multiple Oracle RAC nodes using connection pools configured for connect-time failover, set up a JDBC connection pool with the Oracle Thin driver for each RAC instance in your RAC cluster. Then configure each connection pool to use connect-time failover, as described in the sections that follow.

When connections are created in the connection pool, the Oracle Thin driver determines which Oracle RAC instance to use. When an application needs a connection, it looks up a data source on the JNDI tree and requests a connection from the data source. In response, the underlying connection pool delivers one of the available connections from the pool.

The following sections describe a configuration in which Oracle RAC's connect-time failover feature is used to handle connection failures. Keep in mind that in some situations, the connect-time failover feature is slow; the amount of time allowed for failover can be equivalent to the amount of time allowed for a TCP time-out. Depending on your environment, this time period may be as long as several minutes.

Attributes of a Connect-Time Failover Configuration

To use this configuration, create JDBC connection pools in your WebLogic domain with the following attributes:

- Oracle JDBC Thin driver configured for connect-time failover. For example:

  ```
  DriverName="oracle.jdbc.OracleDriver"

  URL="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=
  (ADDRESS=(PROTOCOL=TCP)(HOST=dbhost1)(PORT=1521))
  (ADDRESS=(PROTOCOL=TCP)(HOST=dbhost2)(PORT=1521))
  (FAILOVER=on)(LOAD_BALANCE=off))(CONNECT_DATA=(SERVER=DEDICATED)
  (SERVICE_NAME=dbservice)))"
  ```

- `CountOfTestFailuresTillFlush="1"`
  - Specifies the number of test failures allowed before WebLogic Platform closes all connections in the connection pool to minimize the delay caused by further database testing. For more information about this attribute, see "JDBC Connection Pool Testing Enhancements" in *Programming WebLogic JDBC.*
  - Minimizes the failover time when an Oracle RAC node fails.
- `TestConnectionsOnReserve="true"`
  - Enables testing of a database connection when an application reserves a connection from the connection pool. For more details about this attribute, see "Testing Connection Pools and Database Connections" in *Programming WebLogic JDBC*.
  - Required to enable failover to another RAC node.
- `TestTableName="`*`name_of_small_table`*`"`
  Specifies the table used to test a physical database connection. For details about `TestTableName`, see "JDBCConnection Pool →Configuration →Connections" in the *WebLogic Server Administration Console Online Help*.

Sample config.xml Code

```
<JDBCConnectionPool Name="cgJMSPool-nonXA"
        Targets="cgServer"
        DriverName="oracle.jdbc.OracleDriver"
        URL="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=
        (ADDRESS=(PROTOCOL=TCP)(HOST=dbhost1)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=dbhost2)(PORT=1521))
        (FAILOVER=on)(LOAD_BALANCE=off))(CONNECT_DATA=(SERVER=DEDICATED)
        (SERVICE_NAME=dbservice)))"
        InitialCapacity="10"
        MaxCapacity="100"
        CapacityIncrement="1"
```

```
        Password="user_password"
        Properties="user=user_name"
        PreparedStatementCacheSize="15"
        ConnLeakProfilingEnabled="true"
        TestTableName="dual"
        TestConnectionsOnReserve="true"
        CountOfTestFailuresTillFlush="1" />
<JDBCDataSource Name="cgDataSource-nonXA"
        Targets="cgServer"
        JNDIName="cgDataSource-nonXA"
        PoolName="cgJMSPool-nonXA" />
```

Note: Line breaks are included only for readability.

Configuring a WebLogic Platform Domain with WLST Offline

To configure a WebLogic Platform domain that includes Oracle RAC, BEA recommends using the WebLogic Server Scripting Tool (WLST) Offline. (The Configuration Wizard GUI does not completely support the configuration of domains that include Oracle RAC.) WLST Offline is a command-line scripting interface for configuring a domain. It consists of two components:

- A script template that you customize to create your own script
- A program that you run on your script to have your configuration settings implemented

To access WLST Offline, along with usage instructions and examples, go to:

`https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97`

The configuration options available in the WLST script are shown in Table 3-6.

Table 3-6 WLST Script Configuration Options

Configuration	Connection Pools	Drivers	MultiPools	Data Sources
`MultiPools with XA`	`portalPool1, portalPool2`	`oracle.jdbc.xa.client.OracleXADataSource`	`portalMultiPool`	`portalFrameworkPool`
	`cgPool1, cgPool2`	`oracle.jdbc.xa.client.OracleXADataSource`	`cgMultiPool`	`cgDataSource`
	`bpmArchPool1, bpmArchPool2`	`oracle.jdbc.xa.client.OracleXADataSource`	`bpmArchMultiPool`	`bpmArchDataSource`
`Connect-Time Failover`	`cgJMSPool-nonXA`	`oracle.jdbc.OracleDriver`	Not applicable	`p13n_trackingDataSource` `p13nDataSource` `cgDataSource-nonXA`

To configure your WebLogic Platform domain by using WLST Offline, complete the following procedure:

1. In a supported browser, display the WLST Offline script template `rac10g_sample_script.py`. This script is available at `http://e-docs.bea.com/platform/docs81/db_mgmt/scripts/rac10g_sample_script.py`.
2. In the popup window that is displayed, choose **File→Save As...**
3. Save the contents of the script in a text file.

   Note: This script is also available in Appendix A, "WebLogic Server Scripting Tool (WLST) Offline File."
4. In your text file, make the changes listed in the following table.

Replace . . .	**With the . . .**	**Notes**
All occurrences of *`BEA_HOME`*	Pathname of the directory in which the product is installed	For example: `bea/home`
Value for `cmo.setUserName` (which is defined in the sample script as `user_name`)	Appropriate name for your database account	Ask your database administrator to provide the name.
Value for `cmo.setPassword` (which is defined in the sample script as `user_password`)	Appropriate password for your database account	Ask your database administrator to provide the password.
All occurrences of the names of RAC database instances specified in the sample script (see `HOST=dbname1` and `HOST=dbname2`)	Names of the RAC database instances configured in your domain	Ask your database administrator to provide the RAC configuration settings that are appropriate for your environment.
All occurrences of the names of RAC instances specified in the sample script (see `HOST=dbhost1` and `HOST=dbhost2`)	Names of the RAC instances configured in your domain	
Value of the `SERVICE_NAME` parameter (which is defined in the sample script as `SERVICE_NAME=dbservice`)	Name of the database service configured in your environment	
All occurrences of the port numbers	Port numbers configured in your domain	
Values of • `setListenAddress` • `setListenPort` which are defined in the sample script as follows: `cmo.setListenAddress('10.61.6.134')` `cmo.setListenPort(9101)` `cmo.setListenPort(9102)`	The listener address and port numbers for the administration server	

You have now finished making all required changes to the script.

5. Review all the parameter settings in your completed script and change any settings for which you do not want to use the default values. For example, you may want to use your own password instead of the default password, `weblogic`.
6. Run the completed script, following the instructions for WLST Offline at BEA's dev2dev codes samples site:

   `https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97`

Configuring Your Component-Specific Domain to Use Oracle RAC

You can also use the WLST Offline script described in the previous section to create a domain for a component of WebLogic Platform, such as WebLogic Portal, as long as you:

- Specify the template appropriate for the desired component, instead of the Platform template specified in the script, `platform.jar`. For example, to create a domain for WebLogic Portal, specify the `wlp.jar` template.
- Make all other modifications appropriate for your domain, including those required to set up the JDBC connection pools. How you configure these pools depends on whether you want them to handle distributed transactions:
  - For information about configuring JDBC connection pools that handle distributed transactions, see:

    `http://e-docs.bea.com/platform/docs81/confgwiz/examples.html#creating_XA_domains`
  - For information about configuring JDBC connection pools that do not handle distributed transactions, see:

    `http://e-docs.bea.com/platform/docs81/db_mgmt/default_JDBC_configs.html`

Updating an Existing XA Domain to Use Oracle RAC

This procedure is designed to help you upgrade an existing basic (that is, non-Oracle RAC) XA Platform domain created with the Configuration Wizard. To upgrade an existing domain so it can accommodate Oracle RAC, BEA recommends that you run a script created with a tool called WebLogic Server Scripting Tool (WLST) Offline.

Note: The Configuration Wizard GUI cannot be used for this type of upgrade because it does not completely support the configuration of domains that include Oracle RAC.

WLST Offline is a command-line scripting interface for configuring a domain. It consists of two components:

- A script template that you customize to create your own script
- A program that you run on your script to have your configuration settings implemented

To access WLST Offline, along with usage instructions and examples, go to:

`https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97`

The configuration options available in the WLST script are shown in Table 3-7.

Table 3-7 WLST Script Configuration Options

Configuration	Connection Pools	Drivers	MultiPools	Data Sources
`MultiPools with XA`	`portalPool1, portalPool2`	`oracle.jdbc.xa.client.OracleXADataSource`	`portalMultiPool`	`portalFrameworkPool`
	`cgPool1, cgPool2`	`oracle.jdbc.xa.client.OracleXADataSource`	`cgMultiPool`	`cgDataSource`
	`bpmArchPool1, bpmArchPool2`	`oracle.jdbc.xa.client.OracleXADataSource`	`bpmArchMultiPool`	`bpmArchDataSource`
`Connect-Time Failover`	`cgJMSPool-nonXA`	`oracle.jdbc.OracleDriver`	Not applicable	`p13n_trackingDataSource` `p13nDataSource` `cgDataSource-nonXA`

To configure your WebLogic Platform domain by using WLST Offline, complete the following procedure:

1. In a supported browser, display the WLST Offline script template `update_rac10g_sample_script.py`. This script is available at `http://e-docs.bea.com/platform/docs81/db_mgmt/scripts/update_rac10g_sample_script.py`.

2. In the popup window that is displayed, choose **File→Save As...**
3. Save the contents of the script in a text file.

   Note: This script is also available in Appendix B, "WebLogic Server Scripting Tool (WLST) Offline Update File."

4. In your text file, make the changes listed in the following table.

Replace . . .	With the . . .	Notes
All occurrences of *`Domain_Path`*	Pathname of the domain directory	For example: `C:/bea/home/user_projects/domains/platform`
Value for `cmo.setUserName` (which is defined in the sample script as `user_name`)	Appropriate name for your database account	Ask your database administrator to provide the name.
Value for `cmo.setPassword` (which is defined in the sample script as `user_password`)	Appropriate password for your database account	Ask your database administrator to provide the password.
All occurrences of the names of RAC database instances specified in the sample script (see `HOST=dbname1` and `HOST=dbname2`)	Names of the RAC database instances configured in your domain	Ask your database administrator to provide the RAC configuration settings that are appropriate for your environment.
All occurrences of the names of RAC instances specified in the sample script (see `HOST=dbhost1` and `HOST=dbhost2`)	Names of the RAC instances configured in your domain	
Value of the `SERVICE_NAME` parameter (which is defined in the sample script as `SERVICE_NAME=dbservice`)	Name of the database service configured in your environment	
All occurrences of the port numbers	Port numbers configured in your domain	

You have now finished making all required changes to the script.

5. Run the completed script, following the instructions for WLST Offline at BEA's dev2dev code samples site:

   https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97

Limitations on Using Oracle RAC with Your WebLogic Domain

For information about known limitations on the use of Oracle RAC with WebLogic Platform, see "Using WebLogic Server with Oracle RAC" in *Programming WebLogic JDBC*:

http://e-docs.bea.com/wls/docs81/jdbc/oracle_rac.html

Default JDBC Configurations for WebLogic Platform

The following sections describe the default JDBC configurations for the prepackaged WebLogic Platform configuration templates used to create domains with the Configuration Wizard Topics include:

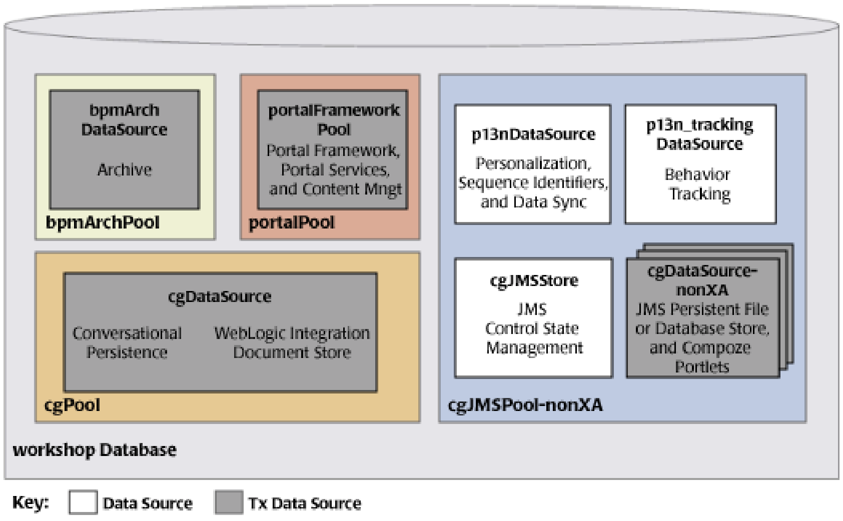
- Default JDBC Configuration for the Basic WebLogic Platform Domain
- Summary of Default JDBC Configurations for All Configuration Templates

Note: If you are using WebLogic Platform 8.1 SP2 or earlier, see "Default JDBC Configurations in SP2 or Earlier" in *Important Usage Notes for Previous Releases of WebLogic Platform 8.1* at the following URL: `http://e-docs.bea.com/platform/docs81/sp_notes/sp_notes.html#sp2_defaultJDBC`

Default JDBC Configuration for the Basic WebLogic Platform Domain

The following figure shows the default JDBC configuration for a domain created using the Basic WebLogic Platform Domain configuration template.

Figure 4-1 JDBC Configuration for the Basic WebLogic Platform Domain (Default)


bpmArch DataSource
Archive
bpmArchPool
portalFramework Pool
Portal Framework, Portal Services, and Content Mngt
portalPool
p13nDataSource
Personalization, Sequence Identifiers, and Data Sync
p13n_tracking DataSource
Behavior Tracking
cgDataSource
Conversational Persistence
WebLogic Integration Document Store
cgPool
cgJMSStore
JMS Control State Management
cgDataSource-nonXA
JMS Persistent File or Database Store, and Compoze Portlets
cgJMSPool-nonXA
workshop Database
Key:
Data Source
Tx Data Source

As shown in the figure, the Basic WebLogic Platform Domain configuration template configures the following JDBC components, by default:

- One database, `workshop`: Provided by WebLogic Workshop and used by all components as default PointBase database.
- Four connection pools
  - `bpmArchPool`: Provided and used by WebLogic Integration for business process management archiving.
  - `cgJMSPool-nonXA`: Provided by WebLogic Workshop and used by WebLogic Workshop, WebLogic Portal, and WebLogic Integration for WebLogic JMS persistent file or database store.
  - `cgPool`: Provided by WebLogic Workshop and used by WebLogic Workshop and WebLogic Integration for multiple functions.
  - `portalPool`: Provided and used by WebLogic Portal for portal framework, services, and content management.
- Three data sources (Data Source)
  - `cgJMSStore`: Provided and used by WebLogic Workshop for WebLogic JMS persistent file or database store.

  - `p13n_trackingDataSource`: Provided and used by WebLogic Portal for behavior tracking.
  - `p13nDataSource`: Provided and used by WebLogic Portal for personalization, sequence identifiers, and data synchronization.
- Four transactional data sources (Tx Data Source)
  - `bpmArchDataSource:` Provided and used by WebLogic Integration for business process management archiving.
  - `cgDataSource`: Provided and used by WebLogic Workshop for conversational persistence and WebLogic Integration document store.
  - `cgDataSource-nonXA`: Provided and used by WebLogic Workshop for JMS control state management and third-party Compoze portlets. See a list of downloadable Compoze portlets at the following URL:

    `http://dev2dev.bea.com/products/wlportal/psc/Compoze.jsp`
  - `portalFrameworkPool`: Provided and used by WebLogic Portal for portal framework, services, and content management.

Summary of Default JDBC Configurations for All Configuration Templates

The following table lists the default JDBC configuration settings for all of the WebLogic Platform prepackaged configuration templates.

Notes: The Basic WebLogic Server Domain template does not have preconfigured JDBC settings. The default domain configurations do not support XA.

Table 2 Default JDBC Configuration Template Settings (Non-XA)

Configuration Template	JDBC Resource Type	Resource Name	JDBC Configuration Setup
Basic WebLogic Workshop Domain	Database	workshop	PointBase
	Connection Pool	cgJMSPool-nonXA	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	cgPool	Connects to a PointBase database using a PointBase Type 4 Driver

Table 2 Default JDBC Configuration Template Settings (Non-XA)

Configuration Template	JDBC Resource Type	Resource Name	JDBC Configuration Setup
	Tx Data Source	cgDataSource	Bound to the cgPool connection pool
	Tx Data Source	cgDataSource-nonXA	Bound to the cgJMSPool-nonXA connection pool
	JMS Store	cgJMSStore	Bound to the cgJMSPool-nonXA connection pool
Basic WebLogic Integration Domain	Database	workshop	PointBase
	Connection Pool	bpmArchPool	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	cgJMSPool-nonXA	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	cgPool	Connects to a PointBase database using a PointBase Type 4 Driver
	Tx Data Source	bpmArchDataSource	Bound to the bpmArchPool connection pool
	Tx Data Source	cgDataSource	Bound to the cgPool connection pool
	Tx Data Source	cgDataSource-nonXA	Bound to the cgJMSPool-nonXA connection pool
	JMS Store	cgJMSStore	Bound to the cgJMSPool-nonXA connection pool
Basic WebLogic Portal Domain	Database	workshop	PointBase
	Connection Pool	cgJMSPool-nonXA	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	cgPool	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	portalPool	Connects to a PointBase database using a PointBase Type 4 Driver

Table 2 Default JDBC Configuration Template Settings (Non-XA)

Configuration Template	JDBC Resource Type	Resource Name	JDBC Configuration Setup
	Data Source	p13n_trackingDataSource	Bound to the cgJMSPool-nonXA connection pool
	Data Source	p13nDataSource	Bound to the cgJMSPool-nonXA connection pool
	Tx Data Source	cgDataSource	Bound to the cgPool connection pool
	Tx Data Source	cgDataSource-nonXA	Bound to the cgJMSPool-nonXA connection pool
	Tx Data Source	portalFrameworkPool	Bound to the portalPool connection pool
	JMS Store	cgJMSStore	Bound to the cgJMSPool-nonXA connection pool
Basic WebLogic Platform Domain	Database	workshop	PointBase
	Connection Pool	bpmArchPool	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	cgJMSPool-nonXA	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	cgPool	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	portalPool	Connects to a PointBase database using a PointBase Type 4 Driver
	Data Source	p13n_trackingDataSource	Bound to the cgJMSPool-nonXA connection pool
	Data Source	p13nDataSource	Bound to the cgJMSPool-nonXA connection pool
	Tx Data Source	bpmArchDataSource	Bound to the bpmArchPool connection pool

Table 2 Default JDBC Configuration Template Settings (Non-XA)

Configuration Template	JDBC Resource Type	Resource Name	JDBC Configuration Setup
	Tx Data Source	cgDataSource	Bound to the cgPool connection pool
	Tx Data Source	cgDataSource-nonXA	Bound to the cgJMSPool-nonXA connection pool
	Tx Data Source	portalFrameworkPool	Bound to the portalPool connection pool
	JMS Store	cgJMSStore	Bound to the cgJMSPool-nonXA connection pool
WebLogic Server Examples Domain	Database	demo	PointBase
	Connection Pool	demoPool	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	demoXAPool	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	oraclePool	Connects to an Oracle database using an Oracle Thin DriverTx
	Data Source	examples-dataSource-demoPool	Bound to the demoPool connection pool
	Tx Data Source	examples-dataSource-demoXAPool	Bound to the demoXAPool connection pool
	Tx Data Source	examples-dataSource-oracleXAPool	Bound to the oraclePool connection pool
	JMS Store	exampleJDBCStore	Bound to the demoPool connection pool
Avitek Medical Records Sample Domain	Database	demo	PointBase
	Connection Pool	MedRecPool-PointBase	Connects to a PointBase database using a PointBase Type 4 Driver
	Connection Pool	MedRecXAPool-PointBase	Connects to a PointBase database using a PointBase Type 4 Driver

Table 2 Default JDBC Configuration Template Settings (Non-XA)

Configuration Template	JDBC Resource Type	Resource Name	JDBC Configuration Setup
	Connection Pool	MedRecPool-Oracle	Connects to an Oracle database using an Oracle Thin Driver
	Tx Data Source	MedRecTxDataSource	Bound to the MedRecXAPool-PointBase JDBC connection pool
	JMS Store	MedRecJMSJDBCStore	Bound to the MedRecPool-PointBase JDBC connection pool

WebLogic Server Scripting Tool (WLST) Offline File

This appendix contains the WebLogic Server Scripting Tool (WLST) Offline file that you can use to configure a WebLogic Platform domain. For more information about this file and how to download it using a browser, see "Configuring a WebLogic Platform Domain with WLST Offline" on page 3-13. The file is listed in this appendix for the convenience of those who are unable to download it using a browser.

Listing A-1 WLST Offline File

```
#==============================================================================
# Read Template

#==============================================================================


print "opening template 'platform'"
readTemplate(r'c:/BEA_HOME/weblogic81/common/templates/domains/platform.jar')


#==============================================================================
# Create JDBC Connection Pools and MultiPools

#==============================================================================


#Create JDBCConnectionPool-cgPool1
print "creating cgPool1"
create('cgPool1','JDBCConnectionPool')
cd('JDBCConnectionPool/cgPool1')
```

```
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost1:1521:dbname1')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)


#Create JDBCConnectionPool-cgPool2
print "creating cgPool2"
cd ('/')
create('cgPool2','JDBCConnectionPool')
cd('JDBCConnectionPool/cgPool2')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set(’RefreshMinutes’,1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost2:1521:dbname2')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)
```

```
#Create JDBCConnectionPool-portalPool1
print "creating portalPool1"
cd ('/')
create('portalPool1','JDBCConnectionPool')
cd('JDBCConnectionPool/portalPool1')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost1:1521:dbname1')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)


#Create JDBCConnectionPool-portalPool2
print "creating portalPool2"
cd ('/')
create('portalPool2','JDBCConnectionPool')
cd('JDBCConnectionPool/portalPool2')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost2:1521:dbname2')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)

#Create JDBCConnectionPool-bpmArchPool1
print "creating bpmArchPool1"
cd ('/')
create('bpmArchPool1','JDBCConnectionPool')
cd('JDBCConnectionPool/bpmArchPool1')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost1:1521:dbname1')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)

#Create JDBCConnectionPool-bpmArchPool2
print "creating bpmArchPool2"
cd ('/')
create('bpmArchPool2','JDBCConnectionPool')
cd('JDBCConnectionPool/bpmArchPool2')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost2:1521:dbname2')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)

#Create JDBCMultiPool-cgMultiPool
print "creating cgMultiPool"
cd ('/')
create('cgMultiPool','JDBCMultiPool')
cd('JDBCMultiPool/cgMultiPool')
set('PoolList','cgPool1,cgPool2')
set('Targets','cgServer')
#The following attribute is valid only for high availability.
set('FailoverRequestIfBusy','true')
set('HealthCheckFrequencySeconds',300)

#============================================================================
#  This MultiPool is configured for High Availability. To configure for
#  Load Balancing, comment out the following line and uncomment the line
#  after it.
#============================================================================

set('AlgorithmType','High-Availability')
#set('AlgorithmType','Load-Balancing')

#Create JDBCMultiPool-portalMultiPool
print "creating portalMultiPool"
cd ('/')
create('portalMultiPool','JDBCMultiPool')
cd('JDBCMultiPool/portalMultiPool')
set('PoolList','portalPool1,portalPool2')
set('Targets','cgServer')
#The following attribute is valid only for high availability.
set('FailoverRequestIfBusy','true')
set('HealthCheckFrequencySeconds',300)

#============================================================================
#  This MultiPool is configured for High Availability. To configure for
#  Load Balancing, comment out the following line and uncomment the line
#  after it.
#============================================================================

set('AlgorithmType','High-Availability')
#set('AlgorithmType','Load-Balancing')

#Create JDBCMultiPool-bpmArchMultiPool
print "creating bpmArchMultiPool"
```

```
cd ('/')
create('bpmArchMultiPool','JDBCMultiPool')
cd('JDBCMultiPool/bpmArchMultiPool')
set('PoolList','bpmArchPool1,bpmArchPool2')
set('Targets','cgServer')
#The following attribute is valid only for high availability.
set('FailoverRequestIfBusy','true')
set('HealthCheckFrequencySeconds',300)

#==============================================================================
# Modify Existing Datasources
#==============================================================================
print "Modifying existing datasources"
cd('/')
cd('JDBCTxDataSources/portalFrameworkPool')
cmo.setPoolName('portalMultiPool')
set('Targets','cgServer')
cd('/')
cd('JDBCTxDataSources/bpmArchDataSource')
cmo.setPoolName('bpmArchMultiPool')
set('Targets','cgServer')

cd('/')
cd('JDBCTxDataSources/cgDataSource')
cmo.setPoolName('cgMultiPool')
set('Targets','cgServer')

#==============================================================================
# Domain

#==============================================================================


cd ('/')
cmo.setName('platformDomain_rac10')


#==============================================================================
# User

#==============================================================================
cd ('/')
cd('Security/platformDomain_rac10/User/weblogic')
cmo.setPassword('weblogic')


#==============================================================================
# Administration Server

#==============================================================================
cd('/')
cd('Server/cgServer')
cmo.setListenAddress('10.61.6.134')
cmo.setListenPort(9101)
cd('SSL/cgServer')
cmo.setEnabled(1)
cmo.setListenPort(9102)
cmo.setHostnameVerificationIgnored(1)




#==============================================================================
# Create JDBC connection pools for Connect-Time Failover
#==============================================================================
cd ('/')
cd ('JDBCConnectionPool')
nonXApools = 'cgJMSPool-nonXA',
for pool in nonXApools:
  print "configuring non-XA pool '" + pool + "'"
  cd (pool)
  cmo.setDriverName('oracle.jdbc.OracleDriver')
  cmo.setUserName('user_name')
  cmo.setPassword('user_password')
  cmo.setURL('jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=
(PROTOCOL=TCP)(HOST=dbhost1)(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST=dbhost2)
(PORT=1521))(FAILOVER=on)(LOAD_BALANCE=off))(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=dbservice)))')
  cmo.setTestConnectionsOnReserve(1)
  cmo.setCountOfTestFailuresTillFlush(1)
  cmo.setTestTableName('dual')
  cmo.setInitialCapacity(5)
  cmo.setMaxCapacity(5)
  cd ('..')

#==============================================================================
# Options
#==============================================================================
setOption('OverwriteDomain', 'true')
setOption('CreateStartMenu','false')
setOption('ServerStartMode','prod')
setOption('JavaHome',r'c:/BEA_HOME/jrockit81sp5_142_08')
```

```
#==============================================================================
# Write Domain and Close Template
#==============================================================================
writeDomain(r'c:/BEA_HOME/user_projects/domains/platformDomain_rac10')
closeTemplate()
```

WebLogic Server Scripting Tool (WLST) Offline Update File

This appendix contains the WebLogic Server Scripting Tool (WLST) Offline file that you can use to update a WebLogic Platform domain. For more information about this file and how to download it using a browser, see "Updating an Existing XA Domain to Use Oracle RAC" on page 3-16. The file is listed in this appendix for the convenience of those who are unable to download it using a browser.

Listing B-1 WLST Offline File

```
#==============================================================================
# Read Domain

#==============================================================================

print "opening domain 'platform'"
readDomain('Domain_Path')
setOption('ReplaceDuplicates','false')
cd('Server/cgServer')
cmo.setStdoutDebugEnabled(1)
cmo.setStdoutSeverityLevel(64)

#==============================================================================
# Create JDBC Connection Pools and MultiPools

#==============================================================================

#Create JDBCConnectionPool-cgPool1
print "creating cgPool1"
create('cgPool1','JDBCConnectionPool')
```

```
cd('JDBCConnectionPool/cgPool1')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost1:1521:dbname1')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)

#Create JDBCConnectionPool-cgPool2
print "creating cgPool2"
cd ('/')
create('cgPool2','JDBCConnectionPool')
cd('JDBCConnectionPool/cgPool2')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set(’RefreshMinutes’,1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost2:1521:dbname2')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)
```

```
#Create JDBCConnectionPool-portalPool1
print "creating portalPool1"
cd ('/')
create('portalPool1','JDBCConnectionPool')
cd('JDBCConnectionPool/portalPool1')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost1:1521:dbname1')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)


#Create JDBCConnectionPool-portalPool2
print "creating portalPool2"
cd ('/')
create('portalPool2','JDBCConnectionPool')
cd('JDBCConnectionPool/portalPool2')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost2:1521:dbname2')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)

#Create JDBCConnectionPool-bpmArchPool1
print "creating bpmArchPool1"
cd ('/')
create('bpmArchPool1','JDBCConnectionPool')
cd('JDBCConnectionPool/bpmArchPool1')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost1:1521:dbname1')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)

#Create JDBCConnectionPool-bpmArchPool2
print "creating bpmArchPool2"
cd ('/')
create('bpmArchPool2','JDBCConnectionPool')
cd('JDBCConnectionPool/bpmArchPool2')
cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
cmo.setInitialCapacity(5)
set('KeepXAConnTillTxComplete','true')
cmo.setMaxCapacity(50)
cmo.setUserName('user_name')
cmo.setPassword('user_password')
set('CountOfTestFailuresTillFlush',1)
set('CountOfRefreshFailuresTillDisable',1)
set('RefreshMinutes',1)
set('ShrinkPeriodMinutes',15)
set('ShrinkingEnabled','true')
set('SupportsLocalTransaction','true')
set('Targets','cgServer')
set('TestConnectionsOnRelease','false')
set('TestConnectionsOnReserve', 'true')
cmo.setURL('jdbc:oracle:thin:@dbhost2:1521:dbname2')
set('XARetryDurationSeconds',300)
set('XASetTransactionTimeout','true')
set('TestConnectionsOnCreate', 'true')
set('XATransactionTimeout',302)

#Create JDBCMultiPool-cgMultiPool
print "creating cgMultiPool"
cd ('/')
create('cgMultiPool','JDBCMultiPool')
cd('JDBCMultiPool/cgMultiPool')
set('PoolList','cgPool1,cgPool2')
set('Targets','cgServer')
#The following attribute is valid only for high availability.
set('FailoverRequestIfBusy','true')
set('HealthCheckFrequencySeconds',300)

#============================================================================
#  This MultiPool is configured for High Availability. To configure for
#  Load Balancing, comment out the following line and uncomment the line
#  after it.
#============================================================================

set('AlgorithmType','High-Availability')
#set('AlgorithmType','Load-Balancing')

#Create JDBCMultiPool-portalMultiPool
print "creating portalMultiPool"
cd ('/')
create('portalMultiPool','JDBCMultiPool')
cd('JDBCMultiPool/portalMultiPool')
set('PoolList','portalPool1,portalPool2')
set('Targets','cgServer')
#The following attribute is valid only for high availability.
set('FailoverRequestIfBusy','true')
set('HealthCheckFrequencySeconds',300)

#============================================================================
#  This MultiPool is configured for High Availability. To configure for
#  Load Balancing, comment out the following line and uncomment the line
#  after it.
#============================================================================

set('AlgorithmType','High-Availability')
#set('AlgorithmType','Load-Balancing')

#Create JDBCMultiPool-bpmArchMultiPool
print "creating bpmArchMultiPool"
```

```
cd ('/')
create('bpmArchMultiPool','JDBCMultiPool')
cd('JDBCMultiPool/bpmArchMultiPool')
set('PoolList','bpmArchPool1,bpmArchPool2')
set('Targets','cgServer')
#The following attribute is valid only for high availability.
set('FailoverRequestIfBusy','true')
set('HealthCheckFrequencySeconds',300)

#==============================================================================
# Modify Existing Datasources
#==============================================================================
print "Modifying existing datasources"
cd('/')
cd('JDBCTxDataSources/portalFrameworkPool')
cmo.setPoolName('portalMultiPool')
set('Targets','cgServer')

cd('/')
cd('JDBCTxDataSources/bpmArchDataSource')
cmo.setPoolName('bpmArchMultiPool')
set('Targets','cgServer')

cd('/')
cd('JDBCTxDataSources/cgDataSource')
cmo.setPoolName('cgMultiPool')
set('Targets','cgServer')

#==============================================================================
# Create JDBC connection pools for Connect-Time Failover
#==============================================================================
cd ('/')
cd ('JDBCConnectionPool')
nonXApools = 'cgJMSPool-nonXA',
for pool in nonXApools:
  print "configuring non-XA pool '" + pool + "'"
  cd (pool)
  cmo.setDriverName('oracle.jdbc.OracleDriver')
  cmo.setUserName('user_name')
  cmo.setPassword('user_password')
  cmo.setURL('jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=
(PROTOCOL=TCP)(HOST=dbhost1)(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST=dbhost2)
(PORT=1521))(FAILOVER=on)(LOAD_BALANCE=off))(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=dbservice)))')
  cmo.setTestConnectionsOnReserve(1)
  cmo.setCountOfTestFailuresTillFlush(1)
  cmo.setTestTableName('dual')
  cmo.setInitialCapacity(5)
  cmo.setMaxCapacity(5)
  cd ('..')

#=======================================================================
# Write and Close Domain
#=======================================================================

cd ('/')
updateDomain()
closeDomain()
exit()
```