ORACLE
FINANCIAL SERVICES

Oracle WebLogic Server Integration with Oracle Database 12c

ORACLE WHITE PAPER | OCTOBER 2015

ORACLE

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Table of Contents

"Active GridLink is a key feature for us. With this solution manual management tasks are no longer necessary. WebLogic is completely aware of all the changes which are happening to the RAC and all manual maintenance is completely eliminated."

DMITRI TYLES

SENIOR DIRECTOR OF DEVELOPMENT

DELTEK

Introduction

Oracle WebLogic Server 11g introduced Active GridLink for Real Application Clusters (RAC). In conjunction with Oracle Database, this powerful software technology simplifies management, increases availability, and ensures fast connection failover, runtime connection, load balancing and affinity capabilities.

Tight integration between Oracle WebLogic Server 12c (12.1.2) and Oracle Database 12c enhances these capabilities with improved availability, better resource sharing, inherent scalability, ease of configuration and automated management facilities in a global cloud environment. Oracle WebLogic Server is the only application server with this degree of integration to Oracle Database 12c. All content covered in this paper applies not only to Oracle WebLogic Server 12.1.2 but also to Oracle WebLogic Server 12.1.3 and 12.2.1.

This white paper explains how these unique technologies database, clustering, and application server work together to enable higher availability, scalability and performance for your business. We start by introducing Oracle Active GridLink for RAC with attention to ease of configuration, manageability, and performance. Then we describe the impact of Oracle WebLogic server on several leading features of Oracle Database12c such as Pluggable Database (PDB), Database Resident Connection Pool (DRCP), Application Continuity, and Global Data Services (GDS), and a data source to integrate to Oracle Universal Connection Pool (UCP).

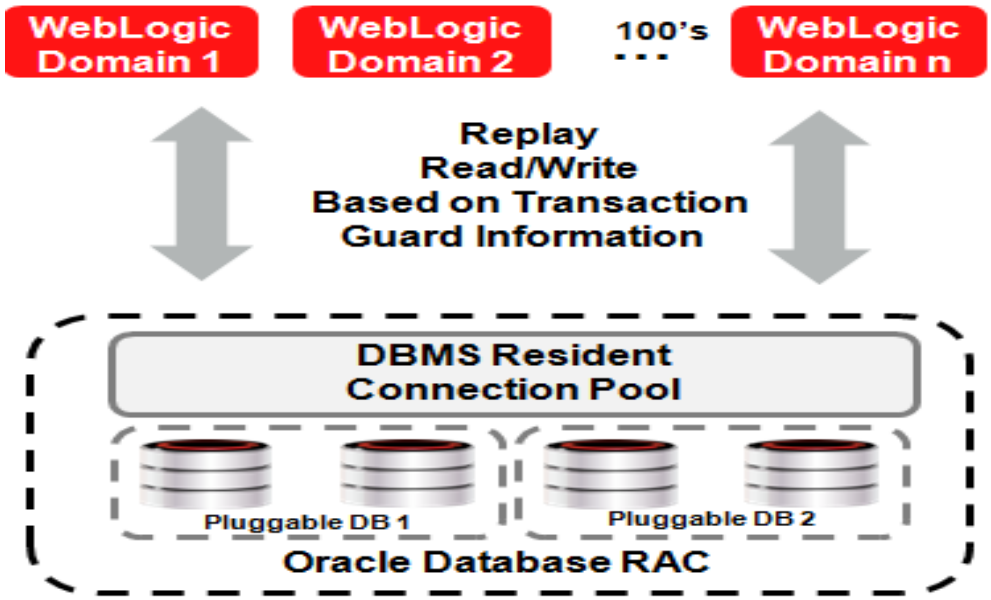

Figure 1: A visual depiction of how Oracle WebLogic Server 12c integrates with Oracle Database 12c

Active GridLink for RAC

Oracle WebLogic Server 11g (10.3.4) introduced a single data source implementation to support Oracle RAC called Active GridLink for RAC. This data source responds to Fast Application Notification (FAN) events to provide Fast Connection Failover (FCF), Runtime Connection Load-Balancing (RCLB), and graceful shutdown of RAC instances. XA affinity is supported at the global transaction level. Active GridLink for RAC is implemented as the GridLink data source within Oracle WebLogic Server.

Through deeper integration with Oracle RAC, this single data source implementation in Oracle WebLogic Server supports the full and unrestricted use of database services as the connection target for a data source. Active management of the connections in the pool is based on static settings configured on the connection pool itself (min/max capacity, timeouts, etc.) and real time information that the connection pool receives from the RAC Oracle Notification Service (ONS) subsystem, which advises the client of any state changes within the RAC cluster.

The Universal Connection Pool (UCP) Java library has been integrated with Oracle WebLogic Server. It is utilized by the WebLogic GridLink data source to enable Fast Connection Failover, Runtime Connection Load Balancing and Affinity features.

Simplified Configuration of Oracle RAC and WebLogic Server

This new implementation simplifies the integration of Oracle RAC database with Oracle WebLogic Server through the GridLink data source approach, which in turn reduces the configuration and management complexity required to use Oracle RAC. Oracle also supports WebLogic Multi Data Source configurations for RAC environments. Upgrades from WebLogic Multi Data Sources to Grid Link data sources are straightforward. They simply involve creating a single Grid Link data source with the same JNDI name as the Multi Data Source.

Active GridLink: Summary of Key Features

WebLogic GridLink data source has been integrated with Fast Connection Failover from the Universal Connection Pool to:

» Rapidly Detect Failures
» Abort and remove invalid connections from the connection pool.
» Perform graceful shutdown for planned and unplanned Oracle RAC node outages
» Adapt to changes in topology, such as adding or removing nodes.
» Distribute runtime work requests to all active Oracle RAC instances, including those rejoining a cluster

WebLogic GridLink data sources and JDBC connection pools leverage the runtime connection load balancing functionality provided by an Oracle RAC database through FAN notifications to improve throughput and more efficiently use resources.

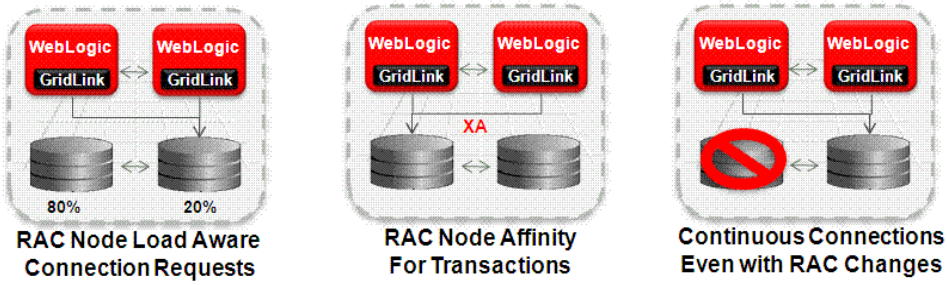

Figure 2: Active GridLink for RAC Load Balancing/XA Affinity/Failover

In Oracle WebLogic Server 11g (10.3.6), Active GridLink for RAC supports Oracle Database 11g features such as Application Continuity for read-only operations and WebSession Affinity.

Application Continuity is a general purpose, application-independent infrastructure that recovers work in the event of an outage and masks many system, communication, and hardware failures and hangs. This version only supports read-only operations.

Web Session Affinity is a new policy option for GridLink data sources that directs database operations made under a web session to the same RAC instance. This technique improves database utilization and boosts overall application performance.

Active GridLink: Supported JDBC URLS

As the supported topologies grow to include additional features like Global Database Services (GDS) and new features are added to the Oracle networking and database support, the complexity of the URL to access these has also gotten more complex.

Active GridLink data sources only support long format JDBC URLs. The supported long format pattern is basically the following (there are lots of additional properties, some of which are described below).

jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=[SCAN_VIP])(PORT=[SCAN_PORT])))(CONNECT_DATA=(SERVICE_NAME=[SERVICE_NAME])))

If not using SCAN, then the ADDRESS_LIST would have one or more ADDRESS attributes with HOST/PORT pairs. It's recommended to use SCAN if possible and it's recommended to use VIP addresses to avoid TCP/IP hangs.

Easy Connect (short) format URLs are not supported for Active GridLink data sources. The following is an example of an Easy Connect URL pattern that is not supported for use with AGL data sources:

jdbc:oracle:thin:[SCAN_VIP]:[SCAN_PORT]/[SERVICE_NAME]

General recommendations for the URL are as follows:

- Use a single DESCRIPTION. Avoid a DESCRIPTION_LIST to avoid connection delays.
- Use one ADDRESS_LIST per RAC cluster or DataGuard database.
- Put RETRY_COUNT, RETRY_DELAY, CONNECT_TIMEOUT at the DESCRIPTION level so that all ADDRESS_LIST entries use the same value.
- RETRY_DELAY specifies the delay, in seconds, between the connections retries. It is new in the 12.1.0.2 release.
- RETRY_COUNT is used to specify the number of times an ADDRESS list is traversed before the connection attempt is terminated. The default value is 0. When using SCAN listeners with FAILOVER = on, setting the RETRY_COUNT parameter to 2 means the three SCAN IP addresses are traversed three times each, such that there are nine connect attempts (3 * 3).
- CONNECT_TIMEOUT is used to specify the overall time used to complete the Oracle Net connect. Set CONNECT_TIMEOUT=90 or higher to prevent logon storms. Through the JDBC driver 12.1.0.2, CONNECT_TIMEOUT is also used for the TCP/IP connection timeout for each address in the URL. This second usage is preferred to be shorter and eventually a separate TRANSPORT_CONNECT_TIMEOUT will be introduced. Do not set the oracle.net.CONNECT_TIMEOUT driver property on the data source because it is overridden by the URL property.
- The service name should be a configured application service, not a PDB or administration service.
- Specify LOAD_BALANCE=on per address list to balance the SCAN addresses.

```
(DESCRIPTION=(CONNECT_TIMEOUT=90)(RETRY_COUNT=20)(RETRY_DELAY=3)(FAILOVER=on)

            (ADDRESS_LIST=(LOAD_BALANCE=on)
              (ADDRESS=(PROTOCOL=TCP)(HOST=scan1)(PORT=1525)))
             (ADDRESS_LIST=(LOAD_BALANCE=on)
              (ADDRESS=(PROTOCOL=TCP)(HOST=scan2)(PORT=1525)))

            (CONNECT_DATA=(SERVICE_NAME=read_write_service)))
```

Figure 3: Sample Active GridLink JDBC URL

Oracle JDBC Thin Driver Logging and Debug

In WebLogic Server 12.2.1 debugging of the Oracle JDBC thin driver has changed. It's necessary to configure Java Util Logging. Set the command line options -Djava.util.logging.config.file=file.properties -Doracle.jdbc.Trace=true.

For normal driver debugging, include a line of the form oracle.jdbc.level = FINEST.

Oracle WebLogic Server Integration with Oracle Database 12c

Delivering an integrated technology platform has always been a central component of Oracle's overall strategy. To that end, Oracle WebLogic Server 12c (12.1.2) supports and has been fully certified with Oracle Database 12c, particularly with respect to Application Continuity, Transaction Guard, Database Resident Connection Pool, Pluggable Database, Multi Tenancy, and Global Data Services. Oracle WebLogic Server is the only application server in the market that provides this level of the integration with Oracle Database. Table 1 shows which capabilities of Oracle Database 12c have been integrated with Oracle WebLogic Server 12.2.1, 12.1.3, 12.1.2, 12.1.1, and 10.3.6.

Oracle WebLogic server certification to DB 12c

Database Feature	WLS 10.3.6 and 12.1.1 – 12.2.1 11g driver 11gR2 DB	WLS 10.3.6 and 12.1.1 – 12.2.1 11g driver 12cDB	WLS 10.3.6 and 12.1.1 12c driver 11gR2 DB	WLS 12.1.2-12.2.1 12c driver 11gR2 DB	WLS 10.3.6 and 12.1.1 12c driver 12c DB	WLS 12.1.2-12.2.1 12c driver 12c DB
JDBC Replay (read/write)	No	No	No	No	Yes (read/write with Active GridLink no XA)	Yes (read/write with Generic data sources and Active GridLink no XA)
Pluggable Database (PDB)	No	Yes	No	No	Yes	Yes
Dynamic Switching between PDBs	No	No	No	No	No	Yes (no XA)
Database Resident Connection Pooling (DRCP)	*No*	*No*	*No*	*Yes*	*No*	*Yes*
Oracle Notification Service (ONS) Auto configuration	*No*	*No*	*No*	*No*	*No*	Yes (Active GridLink *Only)*
Global Database Services (GDS)	*No*	Yes (Active GridLink *Only)*	*No*	*No*	Yes (Active GridLink *Only)*	Yes (Active GridLink *Only)*
JDBC 4.1 (ojdbc7.jar files and JDK 7)	*No*	*No*	*Yes*	*Yes*	*Yes*	*Yes*

"The combinatorial solution with Application Continuity, Real Application Clusters, Data Guard, WebLogic Server Active GridLink and NEC hardware and middleware enables us to provide incredibly high available system for our Mission Critical customers. This solution will become our primary solution for cloud and big data areas."

YUKI MORIYAMA
SENIOR MANAGER
NEC CORPORATION

Application Continuity and Transaction Guard

As the key offering in Oracle Database 12c, Application Continuity is a general purpose, application-independent software utility that recovers work during an outage and masks system, communication, and hardware failures from users. Oracle Database is the first database management system to provide a generic infrastructure for at-most execution semantics in case of failures. This type of execution means that the operation must execute once, partially, or not at all. For example, adding or deleting an appointment from a calendar generally uses at-most-once semantics.

Application Continuity requires minimal effort to enable transaction replay with Oracle JDBC driver. The semantics assure that end-user transactions are executed on time and at-most-once. The only time an end user should see an interruption in service is when the outage is such that it is not possible to recover.

Database conversation interruptions are masked from users. Application Continuity allows requests to be retried safely elsewhere in the system without risk of duplication, increasing application availability, boosting developer productivity and improving the user experience.

Application Continuity will only replay recoverable errors, meaning those errors that occur following planned and unplanned outages of foregrounds, networks, nodes, storage devices, and databases. While applications sometimes receive error codes that don't reveal the status of the last operation submitted, Application Continuity reestablishes database sessions and resubmits the pending work for recoverable errors. It does not resubmit work following call failures due to non-recoverable errors such as when invalid data values are submitted.

Transaction Guard supplies a unique logical transaction identifier (LTXID) for each database transaction. This identifier can be used to query the Commit outcome of the transaction as well as to ensure that the transaction is applied only once. Transaction Guard is used by Application Continuity and automatically enabled by it and it can also be enabled independently. It prevents transactions that are replayed by Application Continuity from being applied more than once.

To use this feature in Oracle WebLogic Server, configure a GridLink data source with oracle.jdbc.replay.OracleDataSource as the connection factory. (This connection supports FAN/fast connection failover and runtime load balancing for fast error detection and smart rebalancing.) You must have a single pool data source to allow the driver to reconnect to a different instance following a failure. It does not support XA transactions and it cannot be used with proxy authentication.

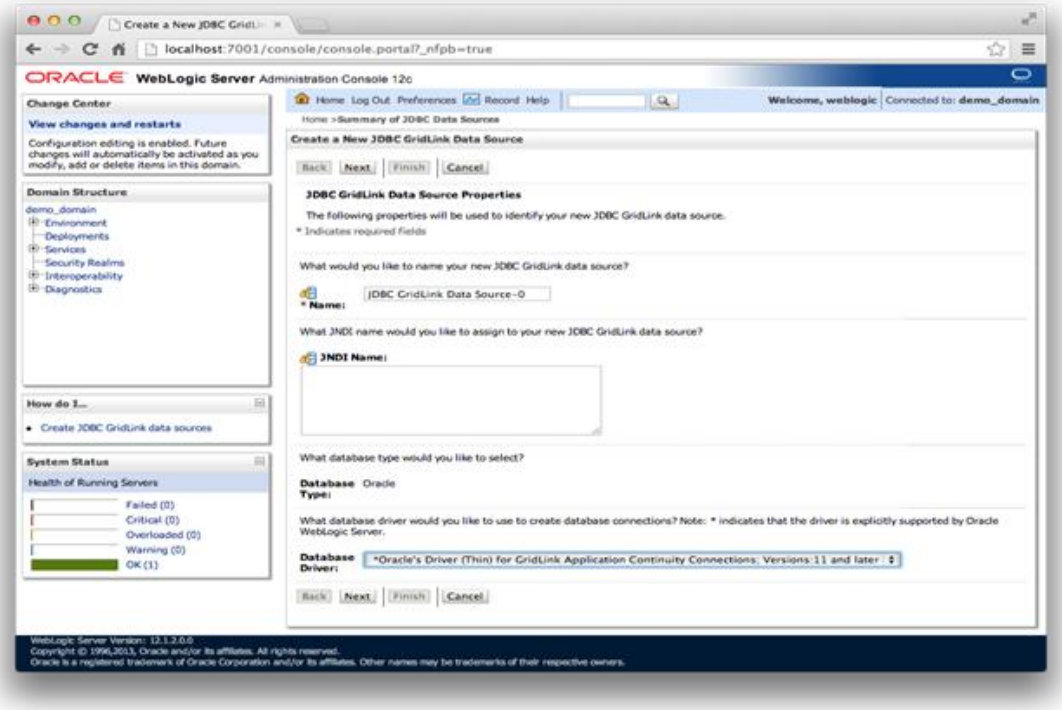

Figure 4: Configuring Application Continuity in Oracle WebLogic Server

Applying Application Continuity in Oracle WebLogic Server

When Oracle WebLogic Server first creates a connection it sets all the properties on that connection and enables Replay. Subsequent connections to the pool include a “begin” statement so that JDBC operations are remembered through final Commit. When the connection is returned to the pool an “end” statement is issued. Other important considerations are listed below:

- **Disabling Replay on a Connection-** Obtain a connection from the configured Oracle WebLogic Server Active GridLink data source, cast this connection to oracle.jdbc.replay.ReplayableConnection, and then call setReplayableEnabled(false) on this connection.
- **Using Reconnect Callback**- After a failure is aborted, a new connection request triggers a call to this registered callback, and the WebLogic implementation of the method getNewPhysicalConnection() will re-use the previous data source properties with the new connection. It will also trigger the pool to update the per-RAC instance statistics and possibly the affinity context.
- **Configuring Replay** - To configure the timeout use the replay-initiation-timeout MBean parameter. The timeout starts from the start of beginRequest to the end of Replay. When the value of replay-initiation-timeout is 0, no timeout is set. The default value is 3,600 seconds. The recommended setting is to match the HTTP timeout value.

JDBC Replay with Oracle Driver and Database

The JDBC Replay Driver stores JDBC operations that affect each JDBC object’s internal state, along with the arguments to those operations. For example, a Connection object stores all the Statement, PreparedStatement, and CallableStatement objects that it has created, along with all the ResultSet objects that they have created.

In order to conserve memory consumption, the JDBC Replay Driver purges recorded operations as necessary. Applications properly close ResultSets and Statements following use, via standard JDBC calls.

The JDBC Replay Driver purges the stored operation history and all replay-specific objects (such as checksums, cursor replay context) that are related to a Statement or a ResultSet. This purge includes PreparedStatement and CallableStatement when the Statement or ResultSet is closed and there is no bounding active transaction. These objects will also be purged if the connection is closed.

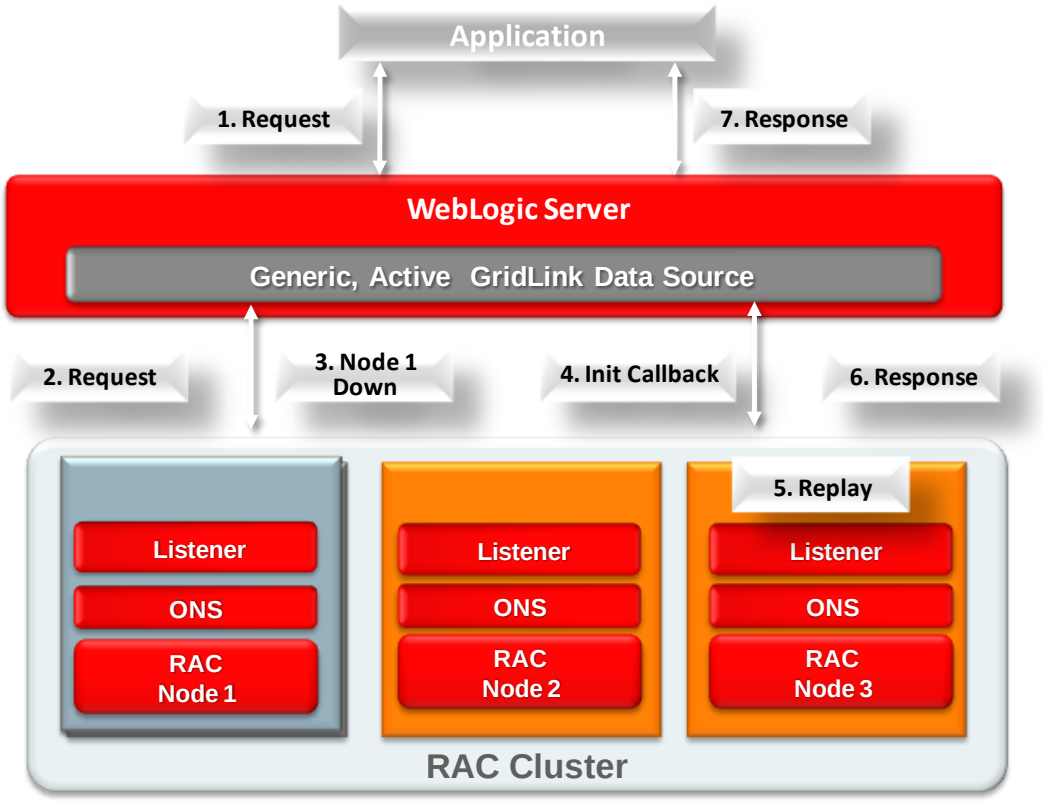

Figure 5: WebLogic Server integration to Application Continuity

As shown in Figure 5, client application requests are passed to Oracle WebLogic Server and then on to Oracle Database using the JDBC replay driver.

1. The JDBC replay driver issues each call in the request.

2. A FAN unplanned or planned interruption or recoverable error occurs. FAN/FCF then aborts the dead physical session.

3. Application Continuity begins the replay and does the following:

- Replaces the dead physical session with a new clean session and rewires FAN in case a later error occurs during or after replay
- Prepares for replay by using Transaction Guard to determine the outcome of the in-flight transaction, if one was open
- Optionally, calls back using a labeling callback or reconnect callback for the initial state
- Rebuilds the database session, recovering the transactional and non-transactional states and validating at each step that the data and messages seen by the client driver are the same as those that the client may have seen and potentially based a decision on
- Ends the replay and returns to runtime mode

» Submits the last queued call

4. This is the last call made when the outage was discovered. During replay, only this call can execute a commit. A commit made mid-way through rebuilding the session will cause replay to be aborted (excluding autonomous transactions).
5. The response is returned to the application.

If replay was successful, the application can continue with the problem masked. If replay was unsuccessful, the application will handle the original error.

Application Continuity Debug

Application Continuity debugging can be enabled in WebLogic Server, for replay debugging include a line of the form oracle.jdbc.internal.replay.level=FINEST.

Oracle Utility Tool: orachk

Application Continuity is unable to replay transactions that use oracle.sql deprecated concrete classes. These concrete classes are in the form of ARRAY, BFILE, BLOB, CLOB, NCLOB, OPAQUE, REF, or STRUCT as a variable type, a cast, the return type of a method, or calling a constructor. For Application Continuity to work correctly such applications need to be modified to use the new Oracle JDBC extension interfaces, for examples see Using API Extensions for Oracle JDBC Types.

Oracle has a utility program called orachk that can be used to validate various hardware, operating systems, and software attributes associated with the Oracle database and more. Version is 12.1.0.2.4 has some checks available for applications running with Application Continuity.

There are three values that control the Application Continuity checking (called acchk in orachk) for Oracle concrete classes. They can be set either on the command line or via shell environment variable (or mixed). They are the following.

VALUES FOR APPLICATION CONTINUITY CHECKING FOR CONCRETE CLASSES

Command Line Argument	Shell Environment Variable	Usage
–asmhome jarfilename	RAT_AC_ASMJAR	This must point to a version of asm-all-5.0.3.jar that you download from http://asm.ow2.org/.
-javahome JDK8dirname	RAT_JAVA_HOME	This must point to the JAVA_HOME directory for a JDK8 installation.
-appjar dirname	RAT_AC_JARDIR	To analyze the application code for references to Oracle concrete classes like oracle.sql.BLOB, this must point to the parent directory name for the code. The program will analyze .class files, and recursively .jar files and directories. If you have J2EE .ear or .war files, you must recursively explode these into a directory structure with .class files exposed. This test works with software classes compiled for Oracle JDBC 11 or 12.

Example of orachk invocation on the command line

$ ./orachk -asmhome /tmp/asm-all-5.0.3.jar -javahome /tmp/jdk1.8.0_40 -appjar /tmp/appdir

Oracle WebLogic Server applications should take advantage of this tool to verify that their applications are ready to be protected with Application Continuity.

Application Continuity Statistics

Starting with Oracle JDBC thin driver 12.1.0.2, there are 11 statistics available for related to replay processing. WebLogic Server exposes this information to application users via a runtime MBeans. Sample statistics

```
AC Statistics:
=============================================
TotalRequests = 1
TotalCompletedRequests = 1
TotalCalls = 19
TotalProtectedCalls = 19
=============================================
TotalCallsAffectedByOutages = 3
TotalCallsTriggeringReplay = 3
TotalCallsAffectedByOutagesDuringReplay = 0
=============================================
SuccessfulReplayCount = 3
FailedReplayCount = 0
ReplayDisablingCount = 0
TotalReplayAttempts = 3
=============================================
```

Figure 6: Sample Application Continuity Statistics.

The statistics will be available via a new runtime MBean that is accessible via the data source runtime MBean.

- It is available for Generic and AGL data sources. It is not available (null is returned) for MDS, PROXY, and UCP data sources.
- It is available only if running with the 12.1.0.2 or later Oracle thin driver. It is not available (null is returned) for earlier driver versions.
- It is available only if the data source is configured to use the replay driver. It is not available (null is returned) for non-replay drivers (e.g. Oracle driver, XA driver).
- The runtime MBean will initially have no statistics set (they will be initialized to -1). The refreshStatistics() operation on the MBean must be called to update the statistics before getting them.
- Refreshing the statistics is a heavy operation - it locks the entire pool and runs through all reserved and unreserved connections aggregating the statistics. Running this operation often will impact the performance of the data source. The same is true for clearing the statistics.

The information on the runtime MBean can be access via the WLST scripting language or via any program that provides access to the MBean tree (e.g., jconsole or JRMC).

```
import sys, socket, os
hostname = socket.gethostname()
datasource='JDBC GridLink Data Source-0'
svr='myserver'
connect("weblogic","welcome1","t3://"+hostname+":7001")
serverRuntime()
cd('/JDBCServiceRuntime/' + svr + '/JDBCDataSourceRuntimeMBeans/' +
  datasource + '/JDBCReplayStatisticsRuntimeMBean/' +
  datasource + '.ReplayStatistics')
cmo.refreshStatistics()
ls()
total=cmo.getTotalRequests()
cmo.clearStatistics()
```

Figure 7: WLST script that prints the information on the MBean

```
import javax.naming.NamingException;
import javax.management.AttributeNotFoundException;
import javax.management.MBeanServer;
import javax.management.InstanceNotFoundException;
import javax.management.ReflectionException
 import javax.management.ObjectName;
import javax.management.MalformedObjectNameException;
import javax.management.MBeanAttributeInfo;
import javax.management.MBeanOperationInfo;
import javax.management.MBeanException;
import javax.management.MBeanParameterInfo;
import weblogic.management.runtime.JDBCReplayStatisticsRuntimeMBean;

 public void printReplayStats(String dsName) throws Exception {
   MBeanServer server = getMBeanServer();
   ObjectName[] dsRTs = getJdbcDataSourceRuntimeMBeans(server);
   for (ObjectName dsRT : dsRTs) {
     String name = (String) server.getAttribute(dsRT, "Name");
     if (name.equals(dsName)) {
       ObjectName mb = (ObjectName)server.getAttribute(dsRT,
         "JDBCReplayStatisticsRuntimeMBean");
       server.invoke(mb, "refreshStatistics", null, null);

      MBeanAttributeInfo[] attributes =
        server.getMBeanInfo(mb).getAttributes();
       for (int i = 0; i < attributes.length; i++) {
         if (attributes[i].getType().equals("java.lang.Long")) {
           System.out.println(attributes[i].getName()+"="+
             (Long) server.getAttribute(mb, attributes[i].getName()));
         }
       }
     }
   }
 }

 MBeanServer getMBeanServer() throws Exception {
   InitialContext ctx = new InitialContext();
   MBeanServer server = (MBeanServer) ctx.lookup("java:comp/env/jmx/runtime");
   return server;
 }

 ObjectName[] getJdbcDataSourceRuntimeMBeans(MBeanServer server)
     throws Exception {
   ObjectName service = new ObjectName(
     "com.bea:Name=RuntimeService,Type=weblogic.management.mbeanservers.runtime
.RuntimeServiceMBean");
   ObjectName serverRT = (ObjectName) server.getAttribute(service,
     "ServerRuntime");
   ObjectName jdbcRT = (ObjectName) server.getAttribute(serverRT,
     "JDBCServiceRuntime");
   ObjectName[] dsRTs = (ObjectName[]) server.getAttribute(jdbcRT,
     "JDBCDataSourceRuntimeMBeans");
   return dsRTs;
 }
```

Figure 8: Sample Java code to print Application Continuity Statistics

Planned and Unplanned Database Down Events

Active GridLink data source using an Oracle Database RAC environment, can handle database outages either planned (patch or upgrade database) or unplanned (sudden and unexpected database failure). Oracle WebLogic Server applications can run continuously without experiencing any errors.

Active GridLink Configuration for Database Outages

It is assumed that an Active GridLink data source is configured as described in the Oracle WebLogic Server documentation with:

- **FAN enabled**- FAN provides rapid notification about state changes for database services, instances, the databases themselves, and the nodes that form the cluster. It allows for draining of work during planned maintenance with no errors whatsoever returned to applications.
- **ONS** - Either auto-ONS or an explicit ONS configuration.
- **A dynamic database service**- Do not connect using the database service or PDB service – these are for administration only and are not supported for FAN.
- **Testing connections**- Depending on the outage, applications may receive stale connections when connections are borrowed before a down event is processed. This can occur, for example, on a clean instance down when sockets are closed coincident with incoming connection requests. To prevent the application from receiving any errors, connection checks should be enabled at the connection pool. This requires setting test-connections-on-reserve to true and setting the test-table (the recommended value for Oracle is “SQL ISVALID”).
- **Optimize SCAN usage**- As an optimization to force re-ordering of the SCAN IP addresses returned from DNS for a SCAN address, set the connection property oracle.jdbc.thinForceDNSLoadBalancing=true.

Planned Outage Operations

For a planned downtime, the goals are to achieve:

- **Transparent maintenance**- Make the maintenance process fast and transparent to applications for Continuous Availability.
- **Session Draining**- When the targeted instance is brought down for maintenance, ensure that all work completes. It is important to drain sessions without impacting in-flight work and also avoid logon storms on active instance(s) during the planned maintenance.

The goal is to manage a planned outage with no application interruption while patching rolls across RAC instances (hence the phrase “rolling upgrade”). For maintenance purposes (e.g., software upgrades), the Oracle Database instances can be gracefully shutdown one or several at a time without disrupting the operations and availability of the Oracle WebLogic Server applications. Upon “FAN DOWN” event, Active GridLink drains sessions away from the instance(s) targeted for maintenance. It is necessary to stop non-singleton services running on the target database instance (assuming that they are still available on the remaining running instances) or relocate singleton services from the target instance to another instance.

The following is a high level overview of how planned maintenance occurs.

- Detect “DOWN” event triggered by DBA on instances targeted for maintenance
- Drain sessions away from that (those) instance(s)
- Perform transparent maintenance
- Resume operations on upgraded instance(s)

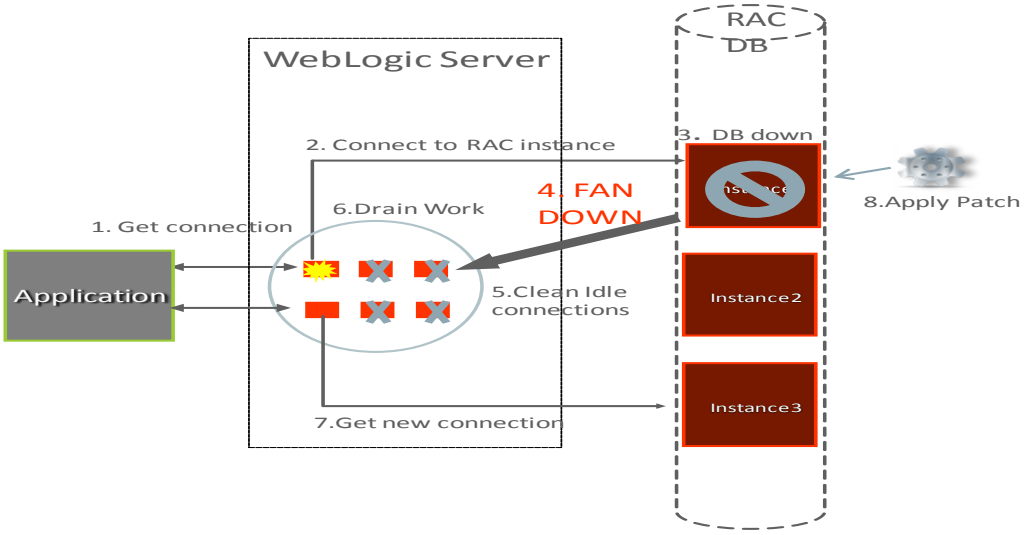

Figure 9: WebLogic Server Active GridLink Database Planned Down

Unlike Multi Data Source where operations need to be coordinated on both the database server and the mid tier, Active GridLink cooperates with the database so that all of these operations are managed from the database server, simplifying the process.

Unplanned Outage Operations

There are several differences when an unplanned outage occurs.

- A component at the database server may fail making all services unavailable on the instances running at that node. There is not stop or disable on the services because they have failed.
- The FAN unplanned "DOWN" event (reason=FAILURE) is delivered to the mid-tier.
- For an unplanned event, all sessions are closed immediately preventing the application from hanging on TCP/IP timeouts. Existing connections on other instances remain usable, and new connections are opened to these instances as needed.
- There is no graceful draining of connections. For those applications using services that are configured to use Application Continuity, active sessions are restored on a surviving instance and recovered by replaying the operations, masking the outage from applications. If not protected by Application Continuity, any sessions in active communication with the instance will receive a SQLException.

Database Resident Connection Pools

Mid-tiers create many idle connections to accommodate high user demand. The cost of creating and destroying these connections is expensive. Database Resident Connection Pooling (DRCP) allows multiple web-tier and mid-tier data sources to share Oracle Database server processes and sessions (together known as pooled servers), enabling better sharing of database resources and greater application scalability. It scales best when the database connections are not always in use.

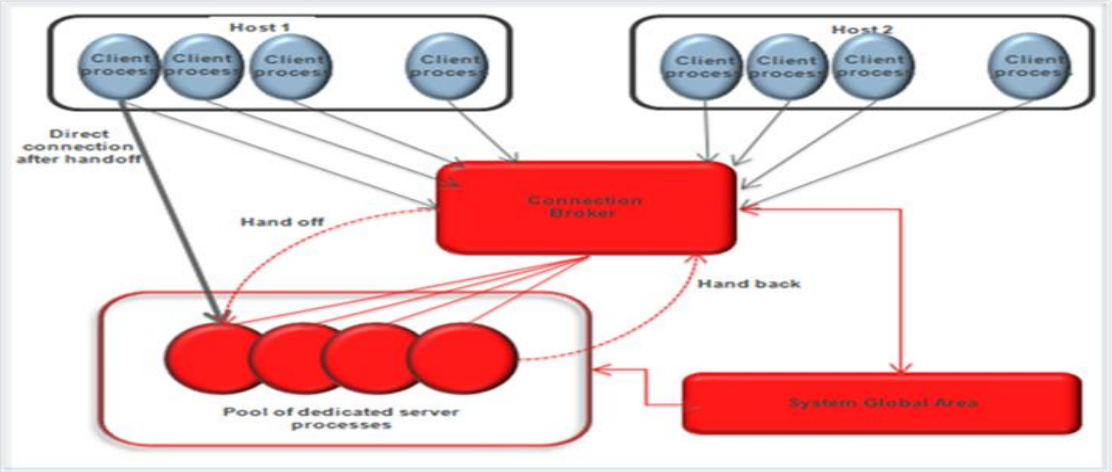

Figure 10: Database Resident Connection Pool

When the pooled connections are in use, they are equivalent to dedicated connections. Upon requesting a connection from the Oracle WebLogic Server data source, the appropriate pooled connection is handed-off. Oracle WebLogic Server directly communicates using the pooled connection for all database activity. The pooled connection is handed back when the data source releases it.

WebLogic Integration with DRCP works with Active GridLink for RAC and with Generic data sources using Oracle Database 12c and JDBC Driver 12c. Connections from the data source connection pool are placeholders in an “unattached” state. When a connection is given up to the application it is attached by calling attachConnection(). When the connection is returned to the pool it is detached by calling detachConnection().

By default when a WebLogic Server connection is attached, it does not actually reserve a session but returns and defers the reservation until the next database round-trip. While this may be slightly more efficient, it means that the subsequent database operation may fail because it can't reserve a session. Further, the attempt to reserve a connection blocks forever until a session becomes available. This can make it difficult for applications to add a non desirable timeout around each JDBC call. In WebLogic Server 12.2.1, when the call to attach a connection is made it is immediately followed by logic to set the network time, force a round-trip to the database (using an Oracle ping database operation), and then unset the network timeout. The default network timeout is 10,000 milliseconds. It can be re-set to another value by setting the system property "weblogic.jdbc.attachNetworkTimeout" (in milliseconds). This timeout waits for the attach to be done and the database round trip to return. If set to 0, then the additional processing around the server attach is not done.

When the pooled connection is used in an XA transaction a flag is enabled to indicate that the connection is in a transaction at enlistment time. It is disabled when the transaction commit or rollback is invoked. This mechanism assures that the same connection will be used for the life of the XA transaction.

DRCP guarantees that the pooled connections are never shared across different users. However, it does allow connections to be shared and pooled across different instances of the same application. Even for the same user, DRCP maintains a logical grouping of the pooled servers based on the “connection classes” chosen by the applications. A connection class is a logical name supplied by a client when a pooled connection is requested. It indicates that the client is willing to reuse a pooled connection that was used by other clients using the same logical name.

For example, if there are ten Oracle WebLogic Server data sources all pointing to the same database, and each data source has an initial capacity of 10 connections, then 100 connections/sessions will be created at startup. If they aren't all active all the time, then that is a waste of resources. With DRCP they can all share a single connection pool with a configurable number of connections. As with any type of virtualization, it only works if you have spare capacity and everyone doesn't try to use all of the resources at the same time.

DRCP boosts the scalability of Oracle Database and Oracle WebLogic Server since persistent connections to the database are held at minimal cost. Database resources are only used by the active connections. Administrators can scale the usage by controlling and setting the pool size.

Pluggable Database

Oracle Database implementations typically fall into the following categories:

- Small to medium size databases that each support a separate application
- Business databases in which each module of an application uses a separate database
- Multi tenant databases with multiple copies of the same database (one per tenant)
- Multiple tenant databases with different collections of schemas to support separate tenants

Pluggable Database implementations allow multiple distinct databases in a single larger database installation. The Container Database (CDB) feature in Oracle Database 12c minimizes the overhead of these multi-database configurations by consolidating them into a single database; multiple Pluggable Databases (PDB) in a single Container Database. The benefits of this type of "in-database virtualization" include the following:

- The ability to upgrade Oracle Database transparently to later versions within the context of a single Container Database
- The ability to run multiple versions of Oracle Database (tenant DBs) inside of a Container Database
- More efficient use of hardware resources
- Unified security management
- Greater density and scalability in the middle and data tiers, leading to better resource utilization
- Support for multiple tenants in a single database

When the client connects to a particular container (PDB) it issues ALTER SESSION SET CONTAINER. This new driver will make sure the container is a valid/existing one and that the current user has the correct privileges to connect to that container. If these checks are successful, the user data in the session is updated with the new PDB name and ID.

The usual connection to a PDB will be through a service. Every PDB will have a default service when it is created, much like the Database service that exists for a singleton database today. Other services can be created for the PDB explicitly. When an application connects using a service, the PDB attribute is used to set up the correct PDB context in the database session.

In Oracle WebLogic Server, you can configure a single data source per PDB. If a new PDB is added to the CDB a new data source needs to be configured in WebLogic Server. The benefits of this model are at the data level.

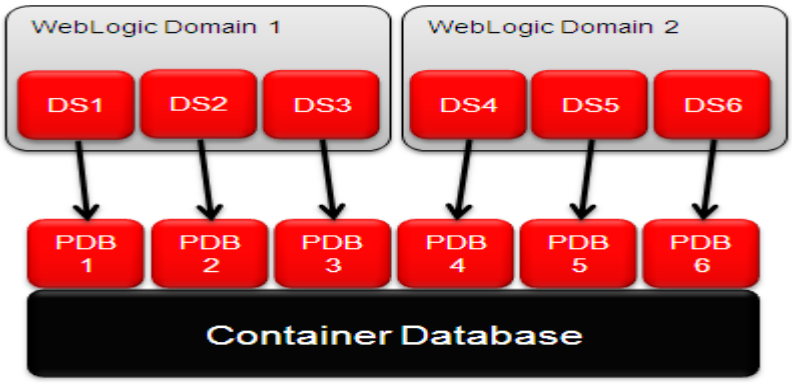

Figure 11: WebLogic Server configuration showing one data source per Pluggable Database.

Another valid Oracle WebLogic Server configuration is to configure a single data source to pool connections to multiple pluggable databases. However, connecting for the first time to a PDB, or switching between the PDBs is not transparent to the application. To enable switching between PDBs, a best practice is to register a callback that changes the connection with the ALTER SESSION SET CONTAINER, sets other properties, and then labels the connection. Any subsequent request for a connection to the same PDB would not need to be altered. This model would provide the true benefits of a pooled connection, including scalability and elasticity.

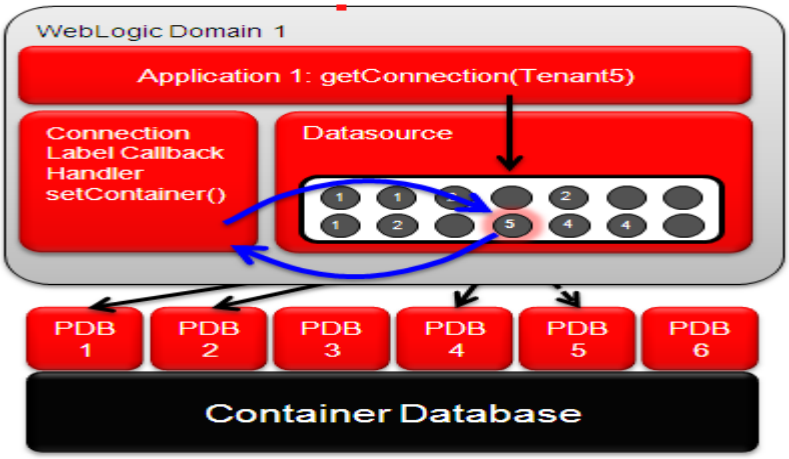

Figure 12: Oracle WebLogic Server and Pluggable Database

There are some limitations with PDB switching in Oracle Database 12.1. The following WebLogic Server limitations exist when using tenant switching.

- Oracle RAC Fast Application Notification (FAN) is not supported. Even though FAN is not supported, Active GridLink still provides the benefit of a single data source view of multiple RAC instances and the ability to reserve connections on new instances as they are available without reconfiguration using connection load balancing. If you want to use tenant switching with an Active GridLink data source, “FAN enabled” must be set to false. Generic data sources don’t use FAN so this restriction doesn’t apply.
- Database Resident Connection Pool (DRCP) is not supported
- Application Continuity is not supported.
- Proxy authentication is not supported.

» Recovery of XA transactions in single data source that is used for switching between multiple PDB's is not supported.

Auto ONS

With the initial version of Active GridLink data source with Oracle Database 11g, you must configure an ONS listener list when FAN is enabled. With Oracle Database 12c the ONS list is optional; the information is automatically provided from the database to the driver. The auto-ONS feature works only with Oracle Database 12c RAC or the new Global Database Service (GDS) feature. It does not work with a single-instance Oracle Database.

Previous versions of WebLogic Server, Admin Console requires ONS list for GridLink data source configurations when FAN is enabled. In Oracle WebLogic Server 12.1.2, the ONS list is optional if you are using the Oracle Database 12c and JDBC 12c driver.

ONS List Configuration

When configuring an Active GridLink data source ONS node list, it is recommended to take advantage of the 12c database feature auto-ONS. The ONS node list will remain blank; auto-ONS will configure ONS automatically, as described above. There are some cases where it is necessary to explicitly configure the ONS node list,

» When specifying a wallet file and password (this cannot be done with auto-ONS).

» To explicitly specify the ONS topology.

New in WebLogic Server 12.2.1 the ONS node list value must be configured either with a Single node list or a Property node list, but not both. If the Oracle WebLogic Server ONS node list contains an equals sign ('='), it is assumed to be a Property node list and not a Single node list.

Single Node List:

A comma separated list of ONS daemon listen addresses and ports pairs separated by colon.

Example:

rac1:6200,rac2:6200

Property Node List:

This string composed of multiple records, with each record consisting of a key=value pair and terminated by a newline ('\n') character. The following keys can be specified.

» **nodes.<id>** A list of nodes representing a unique topology of remote ONS servers. <id> specifies a unique identifier for the Node List -- duplicate entries are ignored. The list of nodes configured in any list must not include any nodes configured in any other list for the same client or duplicate notifications will be sent and delivered. The list format is a comma separated list of ONS daemon listen addresses and ports pairs separated by colon.

» **maxconnections.<id>** Specifies the maximum number of concurrent connections maintained with the ONS servers. <id> specifies the node list to which this parameter applies. The default is 3.

» **active.<id>** If true the list is active and connections will automatically be established to the configured number of ONS servers. If false the list is inactive and will only be used as a fail-over list in the event that no connections for an active list can be established. An inactive list can only serve as a fail-over for one active list at a time, and once a single connection is re-established on the active list, the fail-over list will revert to being inactive. Note that only notifications published by the client after a list has failed-over will be sent to the fail-over list. <id> specifies the node list to which this parameter applies. The default is true.

» **Remotetimeout** The timeout period, in milliseconds, for a connection to each remote server. If the remote server has not responded within this timeout period, the connection will be closed. The default is 30 seconds.

Note that although walletfile and walletpassword are supported in the string, WLS has separate configuration elements for these values, OnsWalletFile and OnsWalletPasswordEncrypted.

Example that is equivalent to the above single node list:

nodes.1=rac1:6200,rac2:6200

When using the administration console to configure an Active GridLink data source, it is not possible to specify a property node list during the creation flow. Instead, it is necessary to modify the ONS node value on the ONS tab after creation. The following figure shows a property node list with two groups (one uses VIP addresses and the other uses a SCAN address – this is for illustration only and would not be used in production).

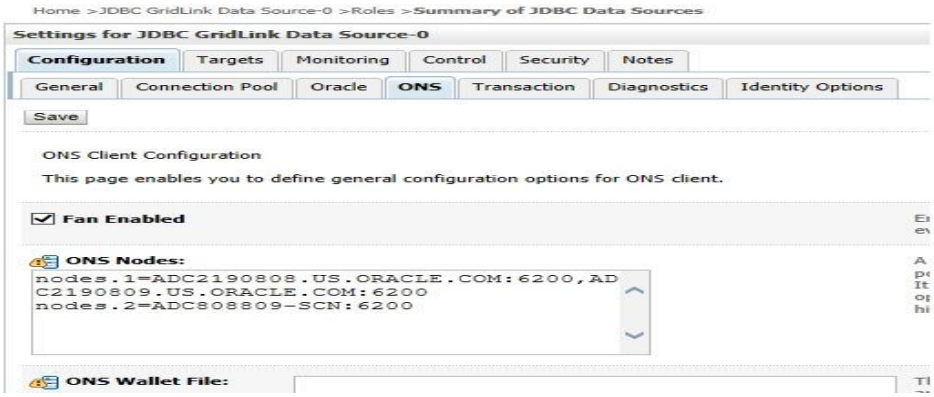

Figure 13: Active GridLink property Node List with two groups

You can also use WLST to create the ONS parameter. Multiple lines need to be separated by embedded newlines.

```
connect('weblogic','welcome1','t3://'+'localhost'+':7001')
edit()
startEdit()
cd('/')
dsName='myds'
cmo.createJDBCSystemResource(dsName)
cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName)
cmo.setName(dsName)
cmo.setDatasourceType('AGL')
cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDataSourceParams/' +
dsName )
set('JNDINames',jarray.array([String('jdbc/' + dsName )], String))
cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDriverParams/' + dsName )
cmo.setUrl('jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=rac
1)(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST=rac2)(PORT=1521)))(CONNECT_DATA=(SERVICE
_NAME=otrade)))')
cmo.setDriverName( 'oracle.jdbc.OracleDriver' )
cmo.setPassword('tiger')
cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCConnectionPoolParams/' +
dsName )
cmo.setTestTableName('SQL ISVALID')
cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDriverParams/' + dsName +
'/Properties/' + dsName )
cmo.createProperty('user')
cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDriverParams/' + dsName +
'/Properties/' + dsName + '/Properties/user')
cmo.setValue('scott')
cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDataSourceParams/' +
dsName )
cmo.setGlobalTransactionsProtocol('None')
cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCOracleParams/' + dsName)
cmo.setFanEnabled(true)
cmo.setOnsNodeList('nodes.1=rac1:6200,rac2:6200\nmaxconnections.1=3\n')
cd('/SystemResources/' + dsName )
set('Targets',jarray.array([ObjectName('com.bea:Name=' + 'myserver' + ',Type=Server')], ObjectName))
save()
activate()
```

Figure 14: Sample WLST configuration of ONS parameter in an Active GridLink data source

ONS Debugging

Enabling ONS debugging has changed starting in Oracle WebLogic Server 12.2.1. In addition to setting debug-jdbcons to true, it's necessary to configure Java Util Logging. Minimally, it is necessary to use the java.util.logging.config.file system property to point to a properties file and include a line of the form oracle.ons.level=FINEST.

Global Data Services

Global Data Services (GDS) streamline the delivery of database services on a global scale, which is key to deploying databases in large cloud architectures. These technologies oversee replication and failover while performing load balancing within and across data centers, optimizing resource utilization and streamlining database management practices in a distributed database environment. GDS works by enabling a Global Service across RAC

and single-instance Oracle databases interconnected via Oracle Data Guard, Oracle GoldenGate, or any other replication technology. Client access to this distributed infrastructure is completely transparent. GDS implementations are easy to apply to Oracle WebLogic Server with minimal changes.

The benefits of Oracle Database 12c Global Data Service include the following:

- Central management of database services across a distributed database cloud
- Dynamic migration of services based on load and availability
- Scalability by adding RAC clusters
- Automatic restart of failed services on an available database
- Easy integration with Oracle WebLogic Server via the Active GridLink d

When configuring the GridLink data source you simply specify a primary local region from which to access the global service, along with the addresses to each region. This configuration enables RAC-like failover for Oracle Database in the cloud. If a region loses its connection to the global database service the reconnection is based on FAN events. There is no need to restart mid-tier components on failure, ensuring business continuity.

GDS is designed to inherently balance the database load across global services. When there is a heavy load on a Read-Only service and the global service is available in another region, the GDS framework will notify the Active GridLink and a new connection will be made to the service in the other region. (The framework cannot load-balance across regions for Read-Write services. This type of processing can only be performed in the primary region.)

Oracle WebLogic Server administrators can configure a GDS data source via the Admin Console as follows:

- Specify connections by denoting:
    - Service name (Global Service Name)
    - Address/port pairs (for various Global Service Managers)
    - GDS Region (new)
- Listeners cannot be tested after configuring a GDS data source.

It is not possible to use a single SCAN address, instead of multiple GSM addresses.service name need to be configured. The following is a sample URL:

jdbc:oracle:thin:@(DESCRIPTION=

(ADDRESS_LIST=(LOAD_BALANCE=ON)(FAILOVER=ON)

(ADDRESS=(HOST=slc02wqh.us.oracle.com)(PORT=2711)(PROTOCOL=tcp))

(ADDRESS=(HOST=slc02wqh.us.oracle.com)(PORT=2709)(PROTOCOL=tcp)))

(CONNECT_DATA=(SERVICE_NAME=uniformdg1.pool1.oradbcloud)(REGION=EAST)))

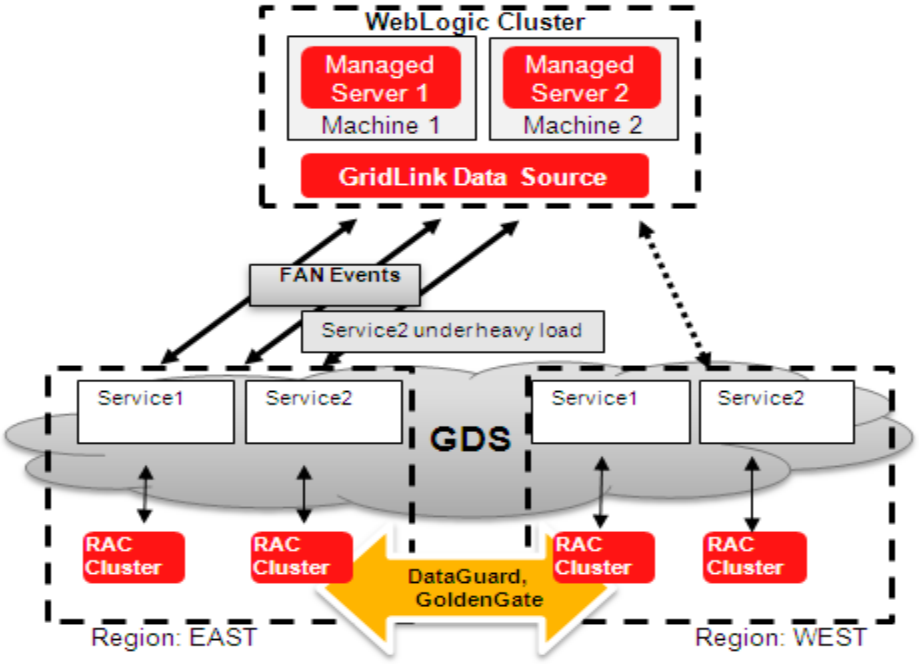

Figure 15: Oracle Oracle WebLogic Server and Global Data Services

In order for update operations to be handled correctly you must only define a service for updates on the primary database. Read-Only operations can be directed to another service that is defined on the primary and secondary databases. Since only a single service can be defined on a URL and a single URL on the data source configuration, one data source must be defined for the Update service and another data source must be defined for the Read-Only service. The application must be written so that Update operations go to the Update data source and Read-Only operations go to the Read-Only data source.

Universal Connection Pool in WebLogic Server

Oracle WebLogic Server 12.2.1 provides support for UCP data source as an option for users who wish to use Oracle Universal Connection Pooling (UCP) to connect to Oracle Databases. UCP provides an alternative connection pooling technology to Oracle WebLogic Server connection pooling.

Oracle generally recommends the use of Active GridLink, Multi Data Source, or Generic data source, and Oracle WebLogic Server connection pooling included in these data source implementations, to establish connectivity with Oracle Databases. Oracle WebLogic Server connection pooling is fully integrated with:

- Oracle WebLogic Server transaction processing,
- Oracle WebLogic Server security, based on WebLogic Server identity
- Oracle WebLogic Server threads management
- Oracle WebLogic high availability cluster features
- Oracle WebLogic Server logging and I18N
- Oracle WebLogic Server JMX, WLST, Console, REST, WebLogic Diagnostic Framework, and Fusion Middleware Control management tooling
- Oracle WebLogic Server data operations like JMS and EJB

Oracle WebLogic Server Active GridLink data sources takes advantage of real time information that the connection pool receives from the RAC Oracle Notification Service (ONS), enabling seamless integration with Oracle Database Real Application Clusters (RAC), including runtime connection load balancing, fast connection failover, automatic detection of RAC configuration changes, with transaction integrity/affinity and Web session affinity.

Together these features provide an integrated application infrastructure runtime experience. Usage of UCP data sources should be limited to scenarios which require UCP for some specific purpose that WebLogic connection pooling does not address.

UCP Data Source

The implementations of UCP data sources are loosely coupled, allowing the swapping of the ucp.jar to support the use of new UCP features by the applications. UCP data sources are not supported in an application-scoped/packaged or stand-alone module environment.

Oracle WebLogic Server provides the following support for UCP data source:

- Configuration as an alternative data source to Generic data source, Multi Data Source, or Active GridLink data source.
- Deploy and undeploy data source.
- Basic monitoring and statistics
    - ConnectionsTotalCount,
    - CurrCapacity,
    - FailedReserveRequestCount,
    - ActiveConnectionsHighCount,
    - ActiveConnectionsCurrentCount)
- Certify with the following drivers
    - Simple Driver
    - XA Driver
    - Application Continuity Driver

A UCP data source does not have support for:

- WebLogic Server Transaction Manager support (one-phase, LLR, JTS, JDBC TLog, determiner resource ...).
- Additional life cycle operations (suspend, resume, shutdown, forceshutdown, start, ...)
- Generic support for any connection pool.
- Oracle WebLogic Server Security options.
- Any drivers other than the Oracle drivers mentioned above.
- No Oracle WebLogic Server data operations like JMS, Leasing, EJB, etc.
- RMI access to a UCP data source.

Creation of a UCP Data Source

A UCP data source can be created using WLST, Admin Console, or Fusion Middleware Control. The following is a sample WLST script for creating a UCP data source.

```
import sys, socket
import os
hostname = socket.gethostname()
connect("weblogic","welcome1","t3://"+hostname+":7001")
edit()
startEdit()
serverName="AdminServer"
serverBean = getMBean('/Servers/'+serverName)
host='%s.us.oracle.com' %hostname
print 'Creating UCP datasource'
domain = getMBean("/")
startEdit()
resourceName='ucpDS'
print "Creating datasource ds in domain"
systemResource=domain.createJDBCSystemResource(resourceName)
systemResource.setName(resourceName)
jdbcResource=systemResource.getJDBCResource()
jdbcResource.setName(resourceName)
jdbcResource.setDatasourceType('UCP')
driverParams=jdbcResource.getJDBCDriverParams()
driverParams.setDriverName('oracle.ucp.jdbc.PoolDataSourceImpl')
driverParams.setUrl('jdbc:oracle:thin:@dbhost:1521/otrade')
properties = driverParams.getProperties()
properties.createProperty('user','dbuser')
properties.createProperty('ConnectionFactoryClassName', 'oracle.jdbc.pool.OracleDataSource')
driverParams.setPassword('MYPASSWD')
jdbcDataSourceParams=jdbcResource.getJDBCDataSourceParams()
jdbcDataSourceParams.addJNDIName(resourceName)
jdbcDataSourceParams.setGlobalTransactionsProtocol('None')
cd('/SystemResources/' + resourceName )
set('Targets',jarray.array([ObjectName('com.bea:Name=' + serverName + ',Type=Server')], ObjectName))
save()
activate()
```

Figure16 : Sample WLST script to create a UCP data source

The configuration elements for a UCP data source are as follows.

1. name

2. data source-type=UCP

3. jdbc-driver-params url

4. jdbc-driver-params property - user

5. jdbc-driver-params password-encrypted

6. jdbc-data-source-params jndi-name

7. jdbc-driver-params other properties

No other elements from the WLS data source descriptor are recognized. They may be specified but are ignored.

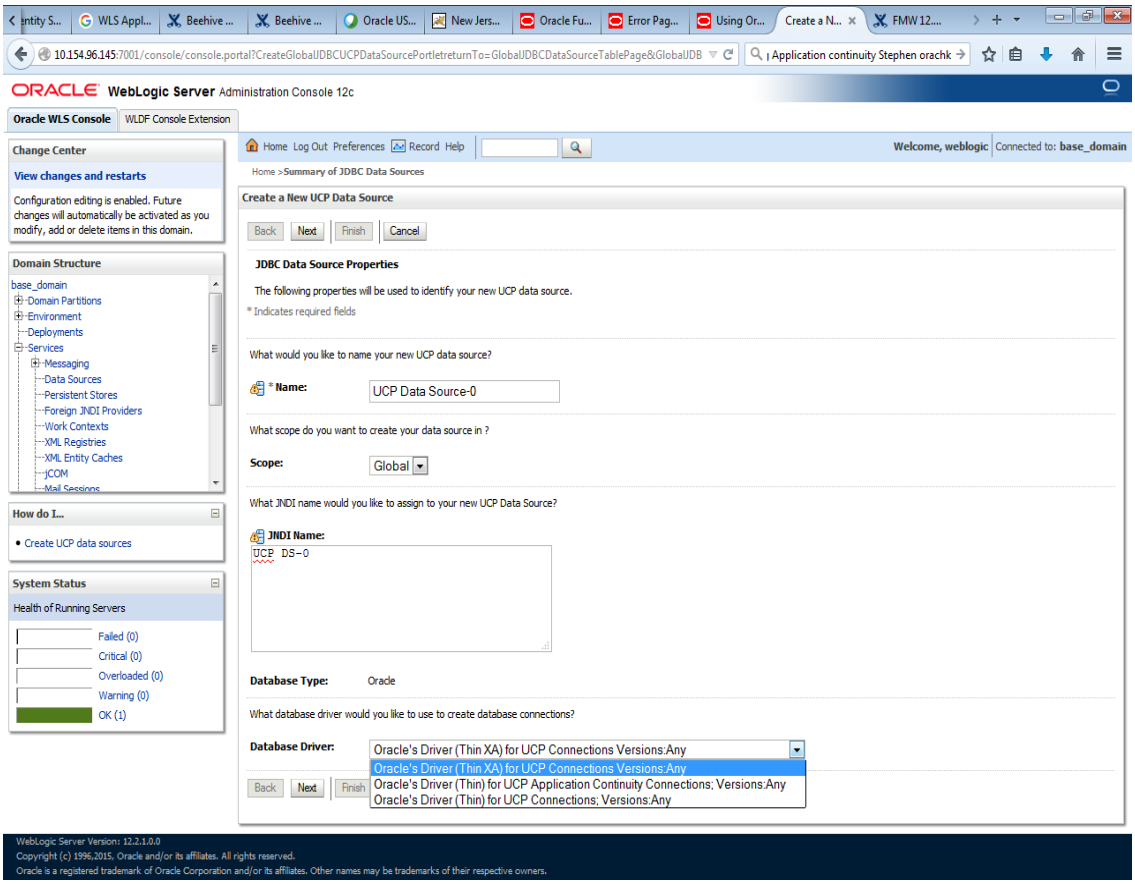

Figure17 : Oracle WebLogic Server Admin Server console configuration

Property handling

The application can configure values for any setters supported by oracle.ucp.jdbc.PoolDataSourceImpl except LogWriter (see oracle.ucp.PoolDaaSourceImpl). Please read documentation for a list of UCP properties Universal Connection Pool Properties.

System properties and encrypted properties are supported in addition to normal string literals. See Setting System Properties and Encrypt connection properties for more details.

If jdbc-driver-params URL is set, any URL property is ignored. If the encrypted-password is set, any password property is ignored. (Note: this is existing behavior with other data source types.)

Attributes ConnectionFactoryProperty, ConnectionFactoryProperties, ConnectionProperty, and ConnectionFactoryProperties accept values of the form "name1=value1,name2=value2...".

COMBINATIONS OF DRIVER AND CONNECTION FACTORY SUPPORTED

Driver	Factory (ConnectionFactoryClassName)
oracle.ucp.jdbc.PoolDataSourceImpl (default)	oracle.jdbc.pool.OracleDataSource
oracle.ucp.jdbc.PoolXADataSourceImpl	oracle.jdbc.xa.client.OracleXADataSource
oracle.ucp.jdbc.PoolDataSourceImpl	oracle.jdbc.replay.OracleDataSourceImpl

Oracle WebLogic Server will validate the driver Connection Factory pair, for example if a non-XA driver from the list above is specified with an XA factory from the list above, an exception is thrown. We allow additional values that are not in the table and do not validate them. If the "driver-name" is not specified in the "jdbc-driver-params", it will default to oracle.ucp.jdbc.PoolDataSourceImpl. If the ConnectionFactoryClassName connection property is not specified, the corresponding entry from the above table is used, assuming a supported driver name is used; otherwise it is an error.

Summary

Oracle WebLogic Server is tightly integrated with Oracle Database 12c to provide the highest level of availability, failover, multi tenancy, resource sharing, scalability, ease of configuration and management, all within the context of a cloud like infrastructure. It delivers a complete, best-of-breed data-processing platform to enable higher availability, scalability and performance for your business.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

ORACLE

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Integrated Cloud Applications & Platform Services

ORACLE WEBLOGIC SERVER INTEGRATION TO THE ORACLE DATABASE 12C
October 2015
Monica Riccelli