Oracle SOA Suite 10g XA and RAC Database Configuration Guide

An Oracle White Paper
April 2010

ORACLE

Oracle SOA Suite 10g XA and RAC Database Configuration Guide

Oracle SOA Suite 10g XA and RAC Database Configuration Guide

INTRODUCTION

This paper describes the requirements and configuration of Oracle SOA Suite components to provide consistency for distributed transactions (XA) during operation. The scope is specific to Oracle SOA Suite 10g and Oracle Application Server interaction and does not cover the broader areas of Oracle SOA Suite enterprise deployment, for which the reader is referred to the Oracle Application Server Enterprise Deployment Guide.

XA configuration provides support for coordinating distributed transactions across multiple resources such that these resources commit or rollback the transaction together.

This paper describes the steps to properly configure XA for both RAC and non-RAC environments. The required XA configurations are mostly the same for each environment, with only a few differences related to data source and database configuration for RAC instances.

XA configuration is applicable to both RAC and non-RAC deployments. There are special considerations for RAC-based deployment however, which are explicitly noted and described throughout this paper.

REQUIREMENTS

Oracle SOA Suite 10g is certified for XA using a specific set of required patches for Oracle SOA Suite, Oracle Application Server and JDBC. XA deployments are supported against Oracle Database 10.2.0.3+ and 11.1.0.6+, including RAC and its supported configurations with Oracle Clusterware. To find the list of the required patches, please search Oracle MetaLink for Note 738108.1 "Required Patching for XA Deployments of SOA Suite".

BPEL, ESB AND ADAPTER DATABASE CONNECTIONS

BPEL Process Manager, Enterprise Service Bus (ESB), and certain adapters use data source connection pools managed by the Oracle Application Server OC4J container to connect with their respective database schemas (Oracle Web Services Manager uses its own connection pooling mechanism described later in this paper).

Note the XA configurations described here are not RAC specific (with the obvious exception of the RAC database connection string), they are required in either RAC or non-RAC environments.

The configurations apply to the following component connection pools:

1. BPEL Process Manager (orabpel)
2. ESB Runtime (esb-rt)

3. ESB Design-Time (esb-dt)
4. Adapters for:
    - Oracle Streams Advanced Queuing (AQ)
    - JMS Adapter with AQ-JMS
    - Database
    - E-Business Suite

Data sources for BPEL Process Manager, ESB and database-backed adapters are the typical resources that must be configured for XA.

The connection pools are defined in *ORACLE_HOME/j2ee/<container-name>/config/data-sources.*xml. The BPEL Process Manager and ESB connection pools are named BPELPM_CONNECTION_POOL, ESBPool and ESBAQJMSPool. The adapter connection pools are named by the user when created and have a few extra notes outlined further below. For each connection pool (in every OC4J container), the following XA configurations are required.

XA Connection Factory

The connection pool should be configured with the XA connection factory class, oracle.jdbc.xa.client.OracleXADataSource, to provide the data source connection with support for distributed transactions.

```
<connection-factory
factory-class="oracle.jdbc.xa.client.OracleXADataSource"
```

Database Connection String

For deployments on RAC, the data source database connection string must be configured to use SERVICE_NAME (e.g. ORCLSVC), with the TNSNAMES connection description for RAC.

Note that using SERVICE_NAME *in the connection string is important, and is not the same as* SID. See the section later in this paper for more details on RAC configuration.

```
url="jdbc:oracle:thin:@(DESCRIPTION=
(ADDRESS_LIST=(LOAD_BALANCE=on)
(ADDRESS=(PROTOCOL=tcp)(HOST=host1-vip)(PORT=1521))
(ADDRESS=(PROTOCOL=tcp)(HOST=host2-vip)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME=ORCLSVC)))"
```

XA Recovery

Oracle SOA Suite uses the OC4J transaction manager within the Oracle Application Server as the transaction coordinator. When unexpected failures occur during transactions that span multiple data sources, it is the OC4J transaction manager that is responsible for proper recovery of those transactions that were not complete. The OC4J transaction manager uses its own file-based or database

transaction logs to store transaction information (that setup is described later in this paper) and manage the recovery.

As part of that recovery process for database resources, the recovery manager needs database user credentials with the correct privileges to access the database transaction state information and issue the appropriate commands, e.g. commit or rollback the resource transaction. To setup the database user for this purpose, first create a database user, e.g. `xauser`, and grant `CONNECT` and `RESOURCE` privileges. Then grant select privileges on `DBA_PENDING_TRANSACTIONS` and execute privileges on the `SYS.DBMS_SYSTEM` package.

```
SQL> CREATE USER xauser IDENTIFIED BY welcome1;
…
SQL> GRANT CONNECT, RESOURCE TO xauser;
…
SQL> GRANT SELECT ON DBA_PENDING_TRANSACTIONS TO xauser;
…
SQL> GRANT EXECUTE ON SYS.DBMS_SYSTEM TO xauser;
```

Now add the `<xa-recovery-config>` element inside the `connection-factory` element of the connection pool definition to specify the database username and password for the recovery manager.

The XA recovery username and password are part of the XA data source connection pool configuration. These user credentials are required for the OC4J transaction manager to access the database resource during recovery after a database crash.

```
<connection-factory …>
  <xa-recovery-config>
     <password-credential>
         <username>xauser</username>
         <password>welcome1</password>
     </password-credential>
  </xa-recovery-config>
…
</connection-factory>
```

To use password obfuscation for the recovery manager, create an OC4J JAAS (Java Authentication and Authorization Service) user with the username and password that matches the database user credentials.

In the below example, the user is created as part of the jazn.com realm with the username and encrypted password stored in the *`$ORACLE_HOME/j2ee/<container>/config/system-jazn-data.xml`* file. Note the `jazn.jar` utility is available within *`$ORACLE_HOME/j2ee/home`* directory, but must be run from the container directory (e.g. *`$ORACLE_HOME/j2ee/oc4j_soa`*) where the data sources are defined.

```
$OH/j2ee/oc4j_soa> java -jar ../home/jazn.jar -shell
AbstractLoginModule username: oc4jadmin
AbstractLoginModule password:
JAZN:> adduser jazn.com xauser welcome1
```

Once the user is created, update the data source `password` element with the notation for password indirection. OC4J will now lookup and utilize the encrypted password at runtime.

```
<connection-factory …>
  <xa-recovery-config>
    <password-credential>
      <username>xauser</username>
      <password>->xauser</password>
    </password-credential>
  </xa-recovery-config>
…
</connection-factory>
```

Restart the container for the new settings to take effect. Then repeat these steps for each container and `data-sources.xml` file. See the Oracle Containers for J2EE Security section OracleAS JAAS Provider Admintool Reference for more information on password obfuscation.

Full documentation for configuring the OC4J Transaction Manager can be found in chapter 3 of the Oracle Containers for J2EE Services Guide.

Adapter Data Sources

Certain adapters utilize user-defined data source connections that must be configured with the same XA recovery and connection pool settings as described above. These adapters are:

1. Oracle Streams Advanced Queuing (AQ)
2. JMS Adapter with AQ-JMS
3. Database
4. E-Business Suite

These adapter endpoints and corresponding data sources are created by users after installation of Oracle SOA Suite as part of BPEL Process Manager or ESB deployments that use adapters. Thus these steps are only necessary when such a deployment occurs.

Adapter data sources should be defined in data-sources.xml to utilize the full data source management capabilities of OC4J.

It's recommended to configure the data sources for these adapters in `data-sources.xml`, although they can be configured directly within the adapter configuration file at $*ORACLE_HOME/j2ee/<container-name>/application-deployments/default/<adapter>/oc4j-ra.xml*. By configuring the data source within `data-sources.xml`, it uses the full data source management implementation available in OC4J.

Note chapter 4 of the Oracle Application Server Adapters for Files, FTP, Databases, and Enterprise Messaging User's Guide has a useful diagram depicting where adapters derive their database connection information.

It is also recommended to use separate schemas and data sources for custom application data and AQ messaging topics. Either storing custom application data or creating custom AQ topics in the BPEL Process Manager or ESB schemas can cause performance or even deadlock issues.

Database and AQ Adapter Notes

Database and AQ adapter configurations must use the xADataSourceName property to specify the JNDI location of the relevant JDBC data source.

After configuring the data source within `data-sources.xml`, edit the Database or AQ adapter `oc4j-ra.xml` file and set the connector factory config property for `xADataSourceName` property to the data source JNDI name. In addition, verify the non-XA `dataSourceName` property has no value. This is an example from the Database Adapter `oc4j-ra.xml` file:

```
<connector-factory location="eis/DB/dbAdapter" …/>
   <config-property
      name="xADataSourceName" value="jdbc/dbAdapterDS"/>
   <config-property
      name="dataSourceName" value=""/>
   …
</connector-factory>
```

JMS and AQ-JMS Adapter Notes

JMS adapters that participate in global transactions need to set the `isTransacted` property to `false` within their `oc4j-ra.xml` file definition. This changes the default behavior that allows only one transaction session per JMS operation (when the property is set to `true`). The same `connector-factory` definition must specify an XA `connectionFactoryLocation` config property with either `XAQueueConnectionFactories` or `XATopicConnectionFactories` in the property value, per the example below.

```
<connector-factory location="eis/jms/myQueue" …/>
   …
<config-property name="connectionFactoryLocation"
value="java:comp/resource/ojmsdemo/XAQueueConnectionFacto
ries/myQCF"/>
<config-property name="isTransacted" value="false"/>
   …
</connector-factory>
```

Note "ojmsdemo" from the connectionFactoryLocation property must match the resource provider name defined for the JMS resource in $*ORACLE_HOME/j2ee/<container-name>/config/application.xml*. For more documentation on OJMS and OC4J JMS adapter configuration, see section 5.2.1.6 Configuring for OJMS or 5.2.1.7 Configuring for OC4J JMS of the Oracle Application Server Adapters for Files, FTP, Databases, and Enterprise Messaging User's Guide.

In addition to the above, JMS adapters operating against AQ-JMS (also called OJMS) within XA transactions have the following recommended configurations:

1. Configure separate connector factories within oc4j-ra.xml for inbound and outbound messaging to the same schema. This requires the creation of two different resource providers (each pointing to its own data source) within $*ORACLE_HOME/j2ee/<container-name>/config/application.xml*. For reference, see Oracle Application Server Release Notes 10g Release 3, under JMS adapter in an XA scenario against AQ-JMS (OJMS).

   Note this requirement is only necessary for XA configurations.

2. Data sources for AQ-JMS must be configured with the property tx-level set to global, and manage-local-transactions to false. Note tx-level defaults to global if not explicitly set, but manage-local-transactions defaults to true.

```
<managed-data-source name="AQSAMPLE_DS1"
     connection-pool-name="AQSAMPLE_POOL1"
     jndi-name="jdbc/aqSample1"
     tx-level="global"
     manage-local-transactions="false"
     login-timeout="0"/>
```

   This setting for manage-local-transactions allows the AQ-JMS provider to issue its non-transactional DDL statements (e.g. creating durable subscribers) without causing conflicts with global transactions. See the Oracle Containers for J2EE Services Guide, Managed Data Source Settings for details on these properties.

3. Configure the BPEL partnerlink or ESB endpoint for the JMS adapter with a new endpoint property `cacheConnections` that is set to `false`.

Fast Connection Failover

Fast Connection Failover (FCF) for Oracle RAC Databases is not supported with the 10g XA JDCB driver, thus FCF configuration is not applicable for these SOA Suite XA deployments.

Fast Connection Failover (FCF) is a well known RAC specific client feature provided by the JDBC Implicit Connection Cache (ICC), a feature of Oracle 10g JDBC drivers.

Fast Connection Failover is not supported by the JDBC ICC in 10g XA JDBC drivers. FCF only works with non-XA 10g JDBC drivers, and thus FCF configuration is not appropriate for the setup described in this paper.

OWSM DATABASE CONNECTIONS

OWSM uses its own database connection pooling mechanism, separate from OC4J's managed data sources for BPEL Process Manager and ESB. The only required configuration for using a RAC database is to modify the database connection strings with the appropriate TNSNAMES descriptor. These steps assume OWSM was previously installed with the Oracle Universal Installer.

First, edit the `$ORACLE_HOME/owsm/bin/coresv.properties` file and update the `dataload.messagelog.db.url` property to use the RAC database connection string, e.g.

```
jdbc:oracle:thin:@(DESCRIPTION=
(ADDRESS_LIST=(LOAD_BALANCE=on)
(ADDRESS=(PROTOCOL=tcp)(HOST=host1-vip)(PORT=1521))
(ADDRESS=(PROTOCOL=tcp)(HOST=host2-vip)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME=ORCLSVC)))
```

Then use the OWSM `wsmadmin` command line utility, located at `ORACLE_HOME\owsm\bin\wsmadmin`, to update and redeploy all the OWSM modules.

```
> wsmadmin copyDBConfig
…
> wsmadmin deploy all
```

Finally, restart the Oracle Application Server container and log into OWSM. Go to Policy Management and update the `cfluent.messagelog.db.url` component property with the same database connection string.

Refer to the Oracle Web Services Manager Deployment Guide for more information and descriptions of the `wsmadmin` commands.

OC4J TRANSACTION MANAGER LOGGING

The OC4J transaction manager requires logging to maintain information regarding the state of transactions. The logging type can be configured for filed-based or database persistence via the $*ORACLE_HOME/j2ee/<container-name>/config/transaction-manager.xml* file. The following steps must be done for each container with BPEL Process Manager or ESB installed.

The OC4J transaction manager uses its own logs to persist information regarding the distributed transactions it is coordinating. This enables the transaction manager to recover those transactions after a container crash.

Both database and file-based logging configurations are available.

To utilize file-based persistence, the file-system should be on shared-disk storage to maintain availability of the transaction logs if the Oracle Application Server node fails. The default file-based log location is the local file system ($ORACLE_HOME/j2ee/home/persistence/txlogs), use the location attribute to specify a different shared-disk directory location.

```
<commit-coordinator retry-count="4">
  <middle-tier>
    <log type="file"
         location="/net/server1/txlogs" />
    <recovery retry-interval = "30"/>
  </middle-tier>
</commit-coordinator>
```

Note the transaction manager automatically creates a specific subdirectory for each OC4J container configured with this directory location.

Alternatively, to configure database-persisted transaction logging, first create a native data source connection within the container `data-source.xml` file.

```
<native-data-source name="xaLogging"
   jndi-name="jdbc/xaLogging"
   data-source-class="oracle.jdbc.pool.OracleDataSource"
   url="jdbc:oracle:thin:@host:1521:orcl" />
```

Then edit the `transaction-manager.xml` file to enable database-persisted transaction logging (for simplicity, consider using the same database user configured for the XA recovery manager).

```
<commit-coordinator retry-count="4">
  <middle-tier>
    <log type="database" location="jdbc/xaLogging">
      <identity user="xauser" password="welcome1"/>
      <database-logging-performance …/>
    </log>
  </middle-tier>
</commit-coordinator>
```

To complete the database logging setup, log into the database as this user and execute the SQL script at $*OH/j2ee/home/database/j2ee/jta/oracle/2pc_jdbcstore.sql* to create the transaction manager schema. See Database Store Attributes within the OC4J Transaction Support documentation for more information.

To obfuscate the password, follow the steps described earlier in this paper and utilize the same notation for password indirection.

```
<commit-coordinator retry-count="4">
  <middle-tier>
    <log type="database" location="jdbc/xaLogging">
      <identity user="xauser" password="->xauser"/>
      <database-logging-performance …/>
    </log>
  </middle-tier>
</commit-coordinator>
```

Distributed Transactions with Non-XA Resources

The Last Resource Commit Optimization of the OC4J Transaction Manager allows exactly 1 non-XA resource to participate in a distributed transaction.

The OC4J transaction manager has support for coordinating XA resources with 1 non-XA resource in a distributed transaction. The transaction manager will automatically employ its Last Resource Commit Optimization in this scenario, no explicit configuration is necessary. This allows exactly 1 non-XA resource to participate in a distributed transaction, as described in OC4J Support for Last-Resource-Commit Optimization section of Oracle Containers for J2EE Resource Adapter Administrator's Guide 10g.

Without transaction manager logging enabled, multiple non-XA resources are allowed to enlist in a distributed transaction, but this is not recommended.

Note that if transaction logging is *not* enabled (be careful, this is the default upon installation), the transaction manager will allow any number of non-XA resources to participate in a global transaction. This is not recommended since XA recovery after a container crash isn't possible without these logs. Refer to Unsupported Transaction Scenarios in the same Oracle Containers for J2EE Resource Adapter Administrator's Guide 10g.

If transaction logging *is* enabled, a second non-XA resource attempting to enlist in the distributed transaction will cause an exception.

Post-Configuration Notes

Once the transaction manager persistence is configured as above and the container is restarted, the data sources defined in data-sources.xml will have a new attribute, `commit-record-table-name`. By default, the attribute value is an empty string. This attribute is part of an optional configuration to enable Recoverable Last Resource Commit (RLRC), another feature of the OC4J transaction manager. It is safe to remove the attribute or leave the value empty if not using this feature.

RLRC is related to the Last Resource Commit feature of OC4J transaction manager that allows for optimizing performance of global transactions with 1 non-XA database resource. To learn more about LRC and RLRC, see Oracle Containers for J2EE Services Guide 10g, Last Resource Commit and Oracle Application Server Release Notes 10g Release 3 (10.1.3.2), Support for Recoverable Last Resource Commit.

Full documentation for configuring the OC4J Transaction Manager can be found in chapter 3 of the Oracle Containers for J2EE Services Guide.

ENTERPRISE SERVICE BUS AQ TOPICS

Recreate the internal AQ topics for Enterprise Service Bus, with an updated compatible version parameter.

Within a clustered setup, ESB uses AQ messaging topics for internal communication and asynchronous interaction with external endpoints. These topics are created using the SQL script $*OH/integration/esb/sql/oracle/create_esb_topics.sql*. Modify this script to use database compatibility mode of `'10.0'` and run the script as the `ORAESB` user to create (or recreate) the AQ topics.

```
dbms_aqadm.create_queue_table(
     Queue_table => qtablename,
     Queue_payload_type => 'SYS.AQ$_JMS_TEXT_MESSAGE',
     multiple_consumers => true,
     compatible => '10.0');
```

The correct compatible value must also be used for user-defined AQ topics that participate in the transaction as well (e.g. those connected to AQ adapters or JMS adapters for AQ-JMS).

Verify the compatible value with the below query:

```
SQL> SELECT QUEUE_TABLE, COMPATIBLE FROM
USER_QUEUE_TABLES;

QUEUE_TABLE                    COMPAT
------------------------------ ------
ESB_CONTROL                    10.0.0
ESB_ERROR                      10.0.0
ESB_ERROR_RETRY                10.0.0
ESB_JAVA_DEFERRED              10.0.0
ESB_MONITOR                    10.0.0
```

JDBC DRIVERS

The JDBC related configuration for this XA setup includes an upgraded and patched set of JDBC drivers. Refer to Oracle MetaLink Note 738108.1 for the specific JDBC version and patch information. Important notes regarding the JDBC drivers:

1. The JDBC configuration described here should be used in SOA Suite-only application server environments. Custom OC4J containers with non-SOA J2EE applications that require a different JDBC version or applications that use JDBC-OCI are not supported.

2. The upgrade and fixes are recommended for RAC and XA behavior with BPEL and ESB. These JDBC drivers are not required for OWSM and in fact OWSM needs to continue using the existing / original JDBC drivers for compatibility reasons. The following sections describe (1) how to separate the JDBC drivers for OWSM and (2) upgrade the JDBC drivers for everything else.

3. The only drivers tested and verified for SOA Suite XA and RAC behavior are Thin JDBC drivers. OCI drivers were not tested in this setup and are not recommended.

OWSM JDBC drivers

These steps will create a separate directory to hold the original JDBC drivers, and configure OWSM to use them.

1. Create a new folder, e.g. *$OH/jdbc/lib_owsm*
2. Copy the original (10.1.0.5 version) *ojdbc14.jar* and *orai18n.jar* files from *$OH/jdbc/lib* into the new folder

```
$ ls -l jdbc/lib_owsm/
  total 6356
  -rwx------  1 dba 1378346 Feb 24 16:22 ojdbc14.jar
  -rw-------  1 dba 5110629 Feb 24 16:23 orai18n.jar
```

3. Update all OWSM shared libraries in *$OH/j2ee/<oc4j>/server.xml* to reference these jars as below, for each OC4J container where OWSM components are installed (e.g. oc4j_soa, oc4j_wsm, or oc4j_gateway).

   Note the `oracle.wsm.agent` library is created only after the 1st OWSM agent is deployed, and in that shared library the orai18n.jar reference must be explicitly added as a new code source element.

```
<shared-library name="oracle.wsm.ccore" …>
  …
  <code-source path="<OH>\jdbc\lib_owsm\ojdbc14.jar"/>
  <code-source path="<OH>\jdbc\lib_owsm\orai18n.jar"/>
  …
</shared-library>

<shared-library name="oracle.wsm.coreman" …>
  …
  <code-source path="<OH>\jdbc\lib_owsm\ojdbc14.jar"/>
  <code-source path="<OH>\jdbc\lib_owsm\orai18n.jar"/>
  …
</shared-library>

<shared-library name="oracle.wsm.gateway" …>
  …
  <code-source path="<OH>\jdbc\lib_owsm\ojdbc14.jar"/>
  <code-source path="<OH>\jdbc\lib_owsm\orai18n.jar"/>
  …
</shared-library>

<shared-library name="oracle.wsm.policymanager" …>
  …
  <code-source path="<OH>\jdbc\lib_owsm\ojdbc14.jar"/>
  <code-source path="<OH>\jdbc\lib_owsm\orai18n.jar"/>
  …
</shared-library>

<shared-library name="oracle.wsm.agent" …>
  …
  <code-source path="<OH>\jdbc\lib_owsm\ojdbc14.jar"/>
  <code-source path="<OH>\jdbc\lib_owsm\orai18n.jar"/>
  …
</shared-library>
```

BPEL, ESB and Adapters JDBC Drivers

Patch the JDBC drivers described in MetaLink Note 738108.1. This will be done on the Oracle Database via OPatch.

The next step is to copy the updated JDBC files to the application server. Copy the Database $*ORACLE_HOME/jdbc/lib/** files to the Application Server

$*ORACLE_HOME/jdbc/lib/* directory. Do this for each Application Server instance.

If copying the updated JDBC libraries per the above step does not include the corresponding 10.2.0.4 version of orai18n.jar, there remains 1 more step. If the orai18n.jar is found in $*ORACLE_HOME/jdbc/lib/* then this step is not necessary. Download orai18n.jar version 10.2.0.4 from the Oracle Database 10g Release 2 (10.2.0.4) JDBC Drivers OTN site and use this version to overwrite the same file in the $*ORACLE_HOME/jdbc/lib/* directory.

ORACLE RAC DATABASE

Within the data tier, the Oracle RAC Database 10g must be configured for distributed transaction processing (DTP). The DTP configuration can specify a preferred RAC instance for a particular database service, meaning that database service will be available on one and only one node at a time. All data source connections to that service are thereby routed to that node. If this preferred instance fails, those connections will fail over to the specified secondary preferred instance.

For SOA Suite XA deployments on Oracle RAC Database 10g, it's necessary to configure database services for Distributed Transaction Processing (DTP).

In Oracle RAC Database 10g, DTP is required to maintain transactional consistency when different data source connections create multiple transaction braches as part of the same global transaction. These transaction branches are not visible across different RAC nodes and therefore all transaction branches must be contained within the same RAC node to ensure consistency. Since BPEL Process Manager and ESB each use their own data source connections, it is necessary to use this configuration whenever these two components interact as part of the same global transaction. Similarly, data source connections for inbound or outbound adapter endpoints from BPEL Process Manager or ESB can introduce separate data source connections and require DTP.

Oracle RAC Database 11g *does not* require configuration of DTP services.

Note that Oracle RAC Database 11g *does not* require DTP, since transactions on one node are visible across the cluster. See the Services and Distributed Transaction Processing in Oracle Real Application Clusters section in Oracle Real Application Clusters Administration and Deployment Guide 11g Release 1 (11.1) for more information.

Distributed Transaction Processing

Create a RAC service using the command line `srvctl` utility available with RAC database installations, and specify an available RAC node as the preferred instance, with one or more RAC nodes as the secondary preferred instance(s).

```
> srvctl add service -d ORCL -s ORCLSVC -r ORCL1 -a ORCL2
```

In this example, the parameter values are as follows:

```
ORCL          Database name
ORCLSVC       DTP service name
ORCL1         Preferred instance
ORCL2         Secondary instance
```

Execute the following database command as SYS user to enable the service to support distributed transactions.

```
SQL> EXECUTE DBMS_SERVICE.MODIFY_SERVICE
    (service_name => 'ORCLSVC', dtp => true);
```

For a deeper review of DTP and XA with Oracle RAC Database, please refer to the Best Practices for Using XA with RAC.

Load Balancing with Distributed Transaction Processing

The preferred instance and secondary instance(s) for RAC DTP services affect how load-balancing is achieved across RAC nodes. For environments with a cluster of SOA Suite nodes, it is possible to achieve RAC load-balancing using multiple RAC services with different preferred instances.

```
> srvctl add service -d ORCL -s ORCLSVC1 -r ORCL1 -a ORCL2
…
> srvctl add service -d ORCL -s ORCLSVC2 -r ORCL2 -a ORCL1
```

Data-tier load-balancing with DTP services can be achieved by "pinning" mid-tier SOA Suite nodes to different RAC nodes.

This example creates two services, each with its own preferred instance. The SOA Suite data source connections for a specific node can then be configured to use either service, ORCLSVC1 or ORCLSVC2, thus balancing the data tier load across the two RAC instances. In the simple case of a two-node SOA Suite cluster, node 1 data sources should point to ORCLSVC1 and node 2 data sources should point to ORCLSVC2.

This model can naturally be expanded to more Oracle Application Server instances and RAC instances. The more DTP services available (each configured with its own unique preferred instance) the more Oracle Application Servers can connect to their own DTP service and better distribute load-balancing against the database tier.

AQ Processing and RAC

The ESB makes use of an internal, database-based AQ messaging queue for its asynchronous routing to ESB endpoints. Within a RAC environment, AQ enqueue /dequeue processing is performed in parallel on multiple nodes, but at high volumes AQ enqueue /dequeue processing performs best if done on one node at a time. This introduces a performance consideration when load-balancing data sources using DTP services as described above.

Note the default recommendation is to allow AQ enqueue / dequeue processing to operate across RAC nodes. Adopting the configuration described below in this section will configure all SOA Suite nodes to use the same RAC node – this should only be considered if enqueue / dequeue performance slows due to change propagation delay across RAC nodes and becomes a significant bottleneck.

The Oracle Streams Advanced Queuing and Real Application Clusters: Scalability and Performance Guidelines best practices document for AQ in a RAC multi-node environment recommends setting instance affinity for a particular queue table and restricting all clients to access the queue table on a specified instance. Instance affinity for the queue table ensures that background processing of messages, like propagating messages and implementing timing properties like delay and expiration, run in the primary or failover instance of the queue. The AQ instance affinity and the preferred instances of the DTP services should be consistent. Therefore to adopt this configuration, the data sources for each SOA Suite node should point to a single DTP service (e.g. `ORCLSVC1`).

To implement the configuration that ties SOA Suite data sources and AQ processing to the same RAC node, follow these steps:

- Configure all SOA Suite data sources to use the same DTP service, e.g. `ORCLSVC1`.
- Determine the appropriate RAC instance number for the primary and secondary instance values that correlate to the primary and secondary nodes of the DTP service.

```
SQL> SELECT INSTANCE_NUMBER, INSTANCE_NAME FROM V$INSTANCE;
```

- Edit the file `ESB_HOME/sql/oracle/create_esb_topics.sql` per the below example (using the primary and secondary instance numbers from the previous step), then connect to the ORAESB schema and execute the SQL script. This will recreate the ESB AQ topics and configure them for processing affinity to a specific RAC node. See Oracle Streams Advanced Queuing User's Guide section 8.1.2 for more information

```
dbms_aqadm.create_queue_table( … );

dbms_aqadm.alter_queue_table(
      queue_table        => qtablename,
      primary_instance   => 1,
      secondary_instance => 2);

dbms_aqadm.create_queue ( … );
```

Optionally, you can execute the `alter_queue_table` without recreating the topics caused by running the script. Simply execute the

`alter_queue_table` SQL command directly for each topic listed in the `create_esb_topics.sql` script.

While this may improve performance, it restricts database processing to one node of the RAC cluster and therefore limits scalability at the data tier. It is advised to consider this scalability vs. performance aspect when designing your deployment topology.

Commit Propagation Delay

The speed of propagating changes between RAC instances is configurable via the `MAX_COMMIT_PROPAGATION_DELAY` database property. For more information on the property, refer to the Oracle RAC Administrator's Guide.

The following scenarios describe when this property may require its value set to zero:

- A BPEL process initiates an asynchronous invoke of an external service that immediately sends a response message
- A BPEL process initiates a one-way invoke and uses a pick activity to process a correlated response message that is immediately received

In these scenarios, the BPEL system automatically hydrates the process instance data to the database at the point it expects to receive a response message. When that message arrives, the BPEL system will rehydrate the process instance. If the response message arrives before the instance data has propagated to the RAC instance from which the BPEL process is rehydrated, an error will occur. Thus to ensure proper behavior of the BPEL system, in environments where asynchronous response messages can return quickly (e.g. sub-second), the `MAX_COMMIT_PROPAGATION_DELAY` property value should be set to zero.

Database Patches

The following patches are also recommended for the 10.2.0.3 or 10.2.0.4 database:

For 10.2.0.3

- 5941601 - TRANSACTION DEADLOCK / OJMS AND DBMS_AQADM_SYS.ALTER_SUBSCRIBER()

For 10.2.0.3 and 10.2.0.4

- 6445864 - ORA-02049: TIMEOUT: DISTRIBUTED TRANSACTION WAITING FOR LOCK USING AQ
- 5843814 - SERIOUS PERFORMANCE ISSUE ON GV$GLOBAL_TRANSACTION WITH PENDING TRANSACTION

For 10.2.0.3 and 10.2.0.4 (DTP service failover)

These fixes are related to DTP service failover. Note at the time of this writing, these are not available for 10.2.0.3 and backport requests may not be approved.

- 8373286 - MERGE OF SERVICE LAYER FIXES FOR 10.2.0.4
- 5348308 - INSTANCE HUNG WHILE RUNNING DESTRUCTIVE TEST
- 8565972 - K2G_DTP_STOP_SVC(): ERROR 30006 (RET -1) AFTER BUG#7508090 IS APPLIED.

Oracle SOA Suite and RAC Database XA Configuration Guide
April 2010
Author: Kevin Clugage

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com