Oracle Fusion Middleware 11g Release 1 SOA High Availability Assessment

Oracle Maximum Availability Architecture White Paper
January 2011

Maximum Availability Architecture

Oracle Best Practices For High Availability

ORACLE

Executive Summary

This document is intended to provide a description of the expected behaviors in an Enterprise Deployment when runtime failures occur in the most relevant components of the Oracle Fusion Middleware 11g Service-Oriented Architecture (SOA) Suite. It is also intended to provide some best practices extending those currently provided in the *Oracle Fusion Middleware Enterprise Deployment Guide for SOA Suite*.

Given the large number of components included in Oracle Fusion Middleware 11*g* SOA Suite, this document presents a first analysis covering only Oracle Fusion Middleware Technology Adapters in the context of BPEL and Mediator service engines. Through a series of tests and verifications, the document describes the results when different types of failures are injected in a SOA system using these components. As a result of the tests and failures injected, average failover metrics for simple composites are provided. You can use these metrics to extrapolate expected failover latencies for real production systems. Additionally, the white paper includes guidance on how to verify behaviors and what tools to use for the verifications. The results, best practices, and recommendations provided in this document are based on Oracle SOA Suite release 11.1.1.3.0, patch set 2 (PS2).

The document provides the following examples to analyze behaviors:

- JMS Adapter Example
- Database Adapter Example
- File Adapter Example

For each area, a description of the fundamental high availability characteristics of the components, description of the tests performed, expected behaviors and best practices are provided. Note that although the tests seem primarily focused on Technology Adapters, the verifications involved, in all cases, BPEL and/or Mediator composites. The failover characteristics of the service engines are implicitly described in the use cases verified for the adapters. Metrics and useful SOA database queries are provided to obtain relevant information about BPEL and Mediator instances.

Introduction to Service-Oriented Architecture High Availability

Service Oriented architecture (SOA) is a software architecture that enables rapid response to changes in competition, market dynamics, and regulatory mandates, with timely information that is critical for the effective

functioning and overall success of businesses.. Oracle Fusion Middleware SOA Suite enables discrete functions contained in enterprise applications to be organized as layers of interoperable, standards-based shared "services" that can be combined and reused in composite applications and processes. These services are reusable business functions that can be discovered and invoked using open standard protocols across networks. A single SOA service can be used by many different systems both inside and outside a company (for interactions with partners, clients and third-party services).

In a SOA system, an outage in a single component or a service can affect multiple applications. Hence, a single service's availability can have a huge impact on a variety of consumers. To maximize availability and security in SOA systems, Oracle provides the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle SOA Suite*. An enterprise deployment is the Oracle MAA best practices blueprint based on proven Oracle high availability and security technologies and recommendations for Oracle Fusion Middleware. An Oracle Fusion Middleware enterprise deployment provides the following benefits:

- Considers various business service-level agreements (SLA) to make high availability best practices as widely applicable as possible
- Leverages database grid servers and storage grid with low-cost storage to provide highly resilient, lower cost infrastructure
- Uses results from performance impact studies for different configurations to ensure that the high availability architecture is optimally configured to perform and scale to business needs
- Enables control over the length of time to recover from an outage and the amount of acceptable data loss from a natural disaster
- Evolves with each Oracle version and it is completely independent of the hardware and operating system

For more information about the Oracle Fusion Middleware 11*g* SOA Suite Enterprise Deployment topology, see the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle SOA Suite* at http://download.oracle.com/docs/cd/E14571_01/core.1111/e12036/toc.htm.

Topology and System Configuration Used for the Tests

The topology used in this analysis verified availability and reaction to failures. Hence, some aspects, such as those related to security that are included in *Oracle Fusion Middleware Enterprise Deployment Guide for SOA* [1], were obviated from testing as a way to minimize the system's complexity. The most relevant pieces that were excluded from the topology include the following:

- A central single sign-on system
- An Oracle LDAP directory for credentials/policies
- Host name verification for the communication between Node Manager and the Managed Servers

Figure 1 shows the topology used for testing:

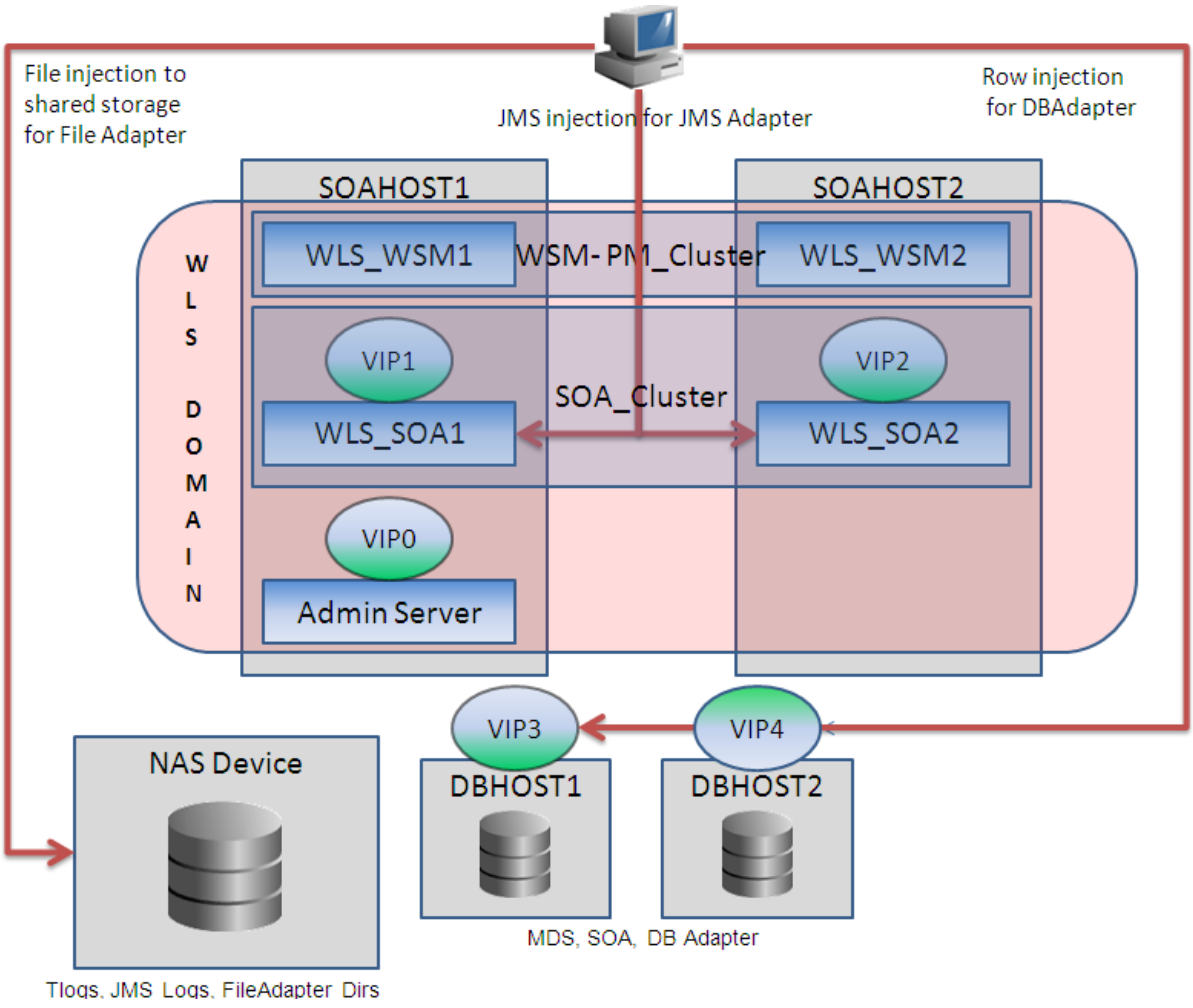


Figure 1: Topology Used for the Verifications

The tests were performed using four different nodes to host the Oracle Fusion Middleware and Oracle Database tiers with the following characteristics:

- Oracle Fusion Middleware servers: 2x Dell 2650 OP Linux 2.6.9-78.0.0.0.1.ELs, 6GB Ram, Intel® Xeon™ CPU 3.40GHzx4
- Oracle RAC database servers: 2x Dell 2650 OP Linux 2.6.9-78.0.0.0.1.ELs, 6GB Ram, Intel® Xeon™ CPU 3.40GHzx4

The Oracle Fusion Middleware system uses the recommendations provided in the *Oracle Fusion Middleware Enterprise Deployment Guide for SOA* [1]. The following list highlights some of the recommendations from the documentation:

- The Oracle Fusion Middleware SOA servers are configured for server migration using database-based leasing. (With Oracle RAC database used for the leasing data source).
- The Oracle RAC data sources are configured with the standard timeouts and pool configuration recommended by the *Oracle Fusion Middleware Enterprise Deployment Guide for SOA* [1]and *the Oracle Fusion Middleware High Availability Guide 11.1.1.2* [2].
- Shared storage is configured for JMS and TX logs: A NAS device is mounted using these mount options: type nfs rw,bg,hard,nointr,tcp,nfsvers=3,timeo=300,rsize=32768,wsize=32768.
- Oracle WebLogic Uniform Distributed Destinations are used by the SOA infrastructure (UMS, BPEL, and so on) and also for the custom destinations used in the tests.
- The client node stressing the different SOA components has the following characteristics: 2.6.9-89.0.0.0.1.ElxenU (OVM virtual machine) 4Gb RAM, CPU 3.0GHzx2).

Mediator and JMS Adapter Analysis

Brief Introduction to the JMS Adapter

The Oracle JMS Adapter enables Oracle SOA Suite and Oracle Fusion Middleware to communicate with JMS end points. It enables the interaction of both Oracle BPEL processes and Oracle Mediator components. The JMS Adapter in Oracle Fusion Middleware 11*g* Release 1 (11.1.1.3/PS2) is based on JMS version 1.0.2b and provides an interface that can work with any JMS provider. It has been certified against AQ JMS, TIBCO JMS, IBM MQSeries, Weblogic JMS and Apache Active MQ. The tests executed for this assessment used the Oracle WebLogic Server (WLS) JMS provider.

High Availability Considerations for Oracle JMS Adapter

Transaction control is important for consistency and reliability purposes when using the Oracle JMS Adapter in a highly available environment. The Oracle JMS Adapter supports global transactions based on the JCA 1.5 XA contracts that leverage the underlying application server transaction manager.

Transaction Control

In the context of the JMS Adapter, a transaction enables an application to coordinate a group of messages for production and consumption, treating messages sent or received as a single unit, as follows:

- If an application commits a transaction, then all messages it received within the transaction are removed by the JMS provider. The messages it sent within the transaction are delivered as one unit to all JMS consumers.
- If an application rolls back a transaction, then the messages it received within the transaction are returned to the messaging system and the messages it sent are discarded.

When using Oracle WebLogic JMS provider, the adapter supports XA transactions by using an XA-enabled connection factory (by default and out of the box, the `weblogic.jms.XAConnectionFactory` is used by the adapter).

To verify the type of connection factory used by the adapter login to the Oracle Weblogic Admin Console:

1. Select **→Deployments→JMSAdapter→Configuration→Outbound Connection pools**
2. Expand **oracle.tip.adapter.jms.mjsConnectionFactory**
3. Click the **eis/wls/Queue** link to show the properties for the specific connection factory, as shown in Image 2:

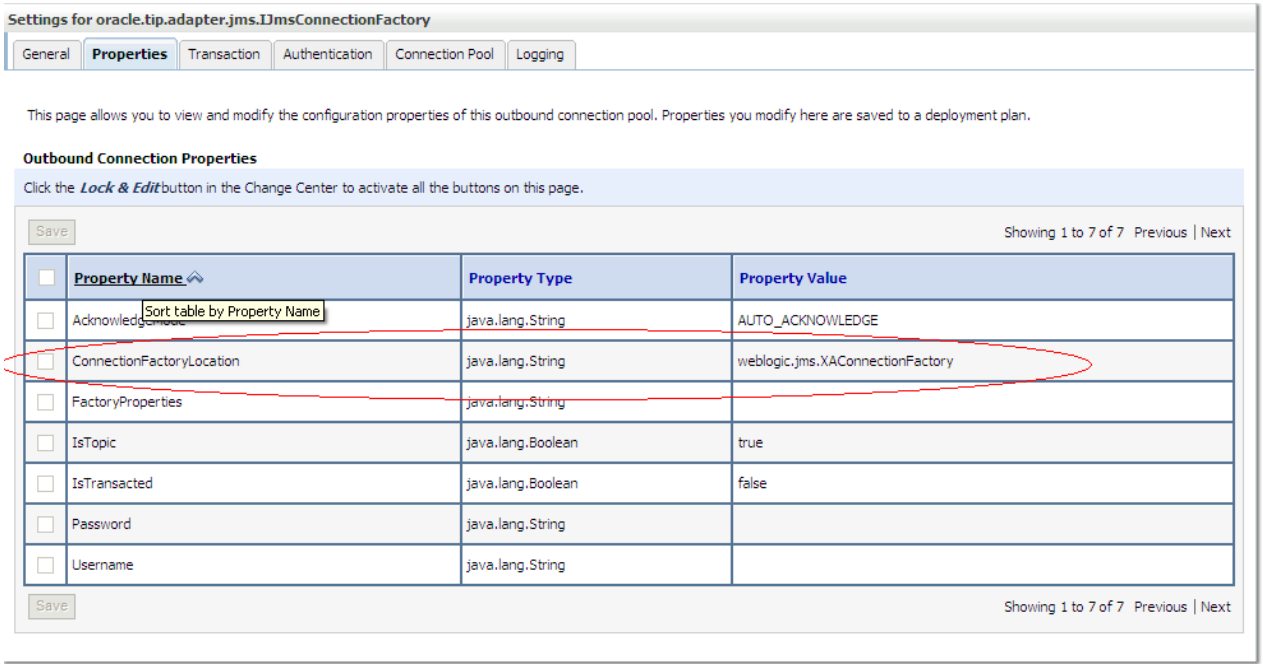


Image 2: Type of Connection Factory specified in Oracle WebLogic Administration Console for the JMS Adapter

Behavior for Inbound and Outbound Transactions

From the transaction control perspective, the behavior for inbound and outbound transactions when using Oracle JMS Adapter is as follows:

- **For inbound transactions**:

  When the adapter does not participate in an existing JTA transaction, it initiates a global transaction before sending an inbound message to a composite/service engine. When control returns to the adapter, it commits the JTA transaction, executing the following set of actions as an atomic unit of work:

  - Commit the removal of the message from the inbound adapter endpoint (JMS destination).
  - Commit the execution of the composite instance.

  If anything fails during this process, both of these actions are rolled back based on XA guarantees.

- **For outbound transactions**:

  For the JMS Adapter, outbound JCA interactions (invoke activities) are scoped with the global JTA transaction of the Composite instance. This means that all composite activities, including Oracle JMS adapter invocations, are part of a global transaction, and as such all activities are either committed or rolled back if an error occurs.

Therefore, the adapter can guarantee exactly-once message delivery when both inbound and outbound adapters are transactional and the connection factories have been configured to support XA global transactions. (Note: Some restrictions and configuration recommendations need to be considered for full guaranteed delivery. These are explained in the "Best Practices for JMS Adapter" section)

Message Redelivery and Redelivery Delay

When using the Oracle WLS JMS provider for the adapter, the high availability behavior is tied to the configuration of the Queues and Topics that are accessed by the adapter. Message redelivery is an important aspect in this respect.

In Oracle WLS JMS, you can delay the redelivery of messages when a temporary, external condition prevents an application from properly handling a message. This allows an application to temporarily inhibit the receipt of *poison* messages that it cannot currently handle. When a message is rolled back or recovered, the **redelivery delay** is the amount of time a message is put aside before an attempt is made to redeliver the message.

- If JMS immediately redelivers the message, then the error condition may not be resolved and the application may still be unable to handle the message.
- If JMS is configured for a redelivery delay, then when it rolls back or recovers a message, the message is set aside until the redelivery delay has passed, at which point the messages are made available for redelivery.

  All messages consumed and subsequently rolled back or recovered by a session receive the redelivery delay for that session at the time of rollback or recovery. Messages consumed by multiple sessions as part of a single user transaction receive different redelivery delays as a function of the session that consumed the individual messages. Messages that are left unacknowledged or uncommitted by a client (either intentionally or as a result of a failure) are not assigned a redelivery delay.

A session in JMS context inherits the redelivery delay from its connection factory when the session is created. The `RedeliveryDelay` attribute of a connection factory is configured using the Oracle WebLogic Administration Console. To modify the redelivery delay of the destinations used by the adapter, access the JMS Module where the destination resides and click the connection factory used to connect to the destinations. The redelivery options are specified in the "Default delivery" tab, as shown in Image 3.

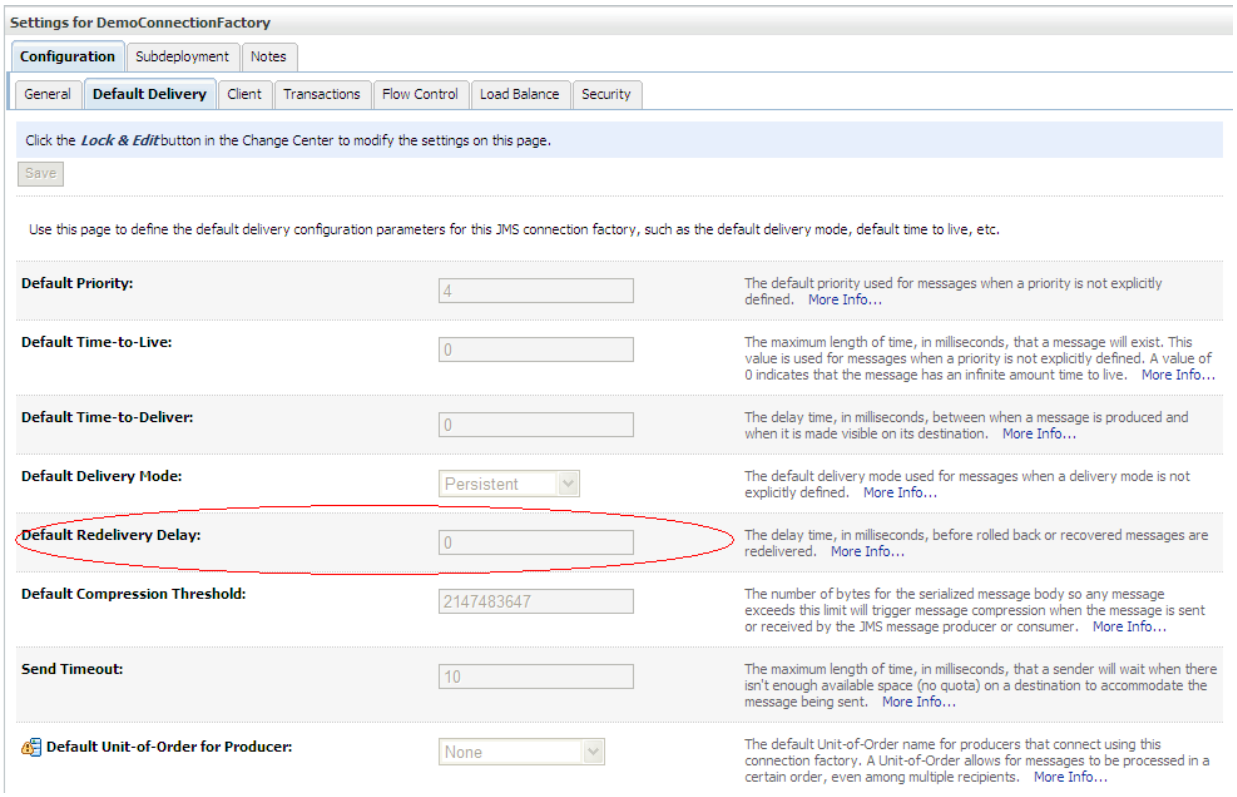


Image 3: Location of the redelivery setting in Oracle WebLogic Administration Console

WebLogic JMS defines two default connection factories that can be instantiated using the following JNDI names:

- `weblogic.jms.ConnectionFactory`
- `weblogic.jms.XAConnectionFactory`

The JMS Adapter uses `weblogic.jms.XAConnectionFactory` by default when configured for the WLS JMS Provider. Therefore, it uses the same redelivery settings as the XAConnectionFactory. The XAConnectionfactory uses a redelivery delay of 0 seconds, which is the expected delay between retries for the adapter.

In addition to the JMS layer redelivery, the JMS Adapter's infrastructure retries invocations in different ways. The behavior for retries in outbound interactions is configured through the `jca.retry.*` parameters (see the chapter about "Adapter Life-Cycle Management" in the *Oracle Fusion Middleware User's Guide for Technology Adapters* documentation at http://download.oracle.com/docs/cd/E14571_01/integration.1111/e10231/life_cycle.htm#BABBAIJJ).

- For inbound invocations, the value of `jca.retry.count` indicates the maximum number of retries before rejection.
- For outbound invocations, the value of `jca.retry.count` indicates the maximum number of retries before throwing a error that can be retried back to the invoking service engine. If an exception occurs in an outbound invocation involving a global transaction and the exception is of type `XARetriableResourceException`,

then the `jca.retry` setting is ignored. In this case, the entire transaction at composite level is rolled back and it is the service engine's responsibility to replay it. For all other (local) exceptions the value specified by the `jca.retry` takes effect.

These properties are orthogonal to the retry logic for getting appropriate connections to the destinations. Set the `adapter.jms.retry.interval` to specify the interval between retries when the adapter poller thread is initializing and trying to get a connection to a destination. If the adapter is unable to create a connection during initialization, it enters a connection recovery loop until a connection is established (i.e. it will attempt to reconnect forever or until the composite is stopped or undeployed). All of these settings affect the behavior of the adapter when a failure is injected in a cluster.

Use Case for Verifying the Availability of the JMS Adapter

For a description of the composite, the failures injected in the system, the type of client, and the tools and scripts used to verify the High Availability behavior of the JMS Adapter, see Appendix A. These details can be used to verify appropriate failover and load balancing in a JMS Adapter Composite.

Summary of Behaviors and Detailed Results

Behaviors Observed During Failure Injection

Table 1 summarizes the results of the failure injection tests run for the JMS Adapter.

Table 1: Results of the different failures injected in an Oracle Fusion Middleware SOA cluster using JMS Adapter

TYPE OF FAILURE	EXPECTED BEHAVIOR/IMPACT	ASSOCIATED EXCEPTIONS, MESSAGES IN LOGS	POSSIBLE RECOVERY
WLS Server Failure	• NO message loss nor duplications takes place • All messages injected in DemoIn are pushed correctly to DemoOut • Non high performance impact of failures on failover node • Non high resource-consumption penalty for restarts • No re-loadbalancing of JMS Destinations on restarted WLS Managed servers	**Node Manager Output:** <INFO> <Successfully removed X.X.X.X from eth0:Y.> … … <Server failed so attempting to restart (restart count = 1)> … <Sleeping for 30 seconds before attempting to restart server> .. <WARNING> <Successfully brought X.X:X:X with netmask 255.255.255.0 online on eth0:Z>	N/A
WLS Server migration due to successive failures in restarting the server in the local node	• NO message loss nor duplications takes place • All messages injected in DemoIn are pushed correctly to DemoOut • Non high performance impact of failures on failover node • Non high resource-consumption penalty for restarts • No re-loadbalancing of JMS Destinations on restarted WLS	**Local-Node Manager output for failed restarts:** <INFO> <Successfully removed X.X.X.X from eth0:Y.> … <Server failed so attempting to restart (restart count = 1)> … <Sleeping for 30 seconds before attempting to restart server> .. INFO: Server failed during startup so will not be restarted -Failover-node Node Manager output: WARNING> <Successfully brought X.X:X:X with netmask	N/A

Table 1: Results of the different failures injected in an Oracle Fusion Middleware SOA cluster using JMS Adapter

TYPE OF FAILURE	EXPECTED BEHAVIOR/IMPACT	ASSOCIATED EXCEPTIONS, MESSAGES IN LOGS	POSSIBLE RECOVERY
	Managed servers	255.255.255.0 online on eth0:Z>	
WLS Server migration due to Node failure	• NO message loss nor duplications takes place • All messages injected in DemoIn are pushed correctly to DemoOut • Non high performance impact of failures on failover node • Non high resource-consumption penalty for restarts • -No re-loadbalancing of JMS Destinations on restarted WLS Managed servers • Watch for possible file lock issues in JMS and TX stores when using NAS device for logs (Please refer to FMWOracle Fusion Middleware 11g release notes here on potential NFS issues for details)	**Failover-node Node Manager output:** WARNING> <Successfully brought X.X:X:X with netmask 255.255.255.0 online on eth0:Z> …	N/A
Database Instance Failure (hard)	• NO message loss nor duplications takes place • All messages injected in DemoIn are pushed correctly to DemoOut • Non high performance impact of failures • Non high resource-consumption penalty for restarts	**SERVER.out file:** java.sql.SQLRecoverableException: No more data to read from socket at oracle.jdbc.driver.SQLStateMapping.newSQLException(SQLStateMapping.java:105) at oracle.jdbc.driver.DatabaseError.newSQLException(DatabaseError.java:135) Followed by <Test "SELECT 1 FROM DUAL" set up for pool "EDNDataSource-rac0" failed with exception: "oracle.jdbc.xa.OracleXAException".>	N/A
Database Node Failure (hard)	• NO message loss nor duplications takes place • All messages injected in DemoIn are pushed correctly to DemoOut • Non high performance impact of failures • Non high resource-consumption penalty for restarts • NO reload balancing for restarted database instances	**SERVER.out file:** <Warning> <JDBC> <BEA-001129> <Received exception while creating connection for pool "SOADataSource-rac1": The Network Adapter could not establish the connection>	N/A
File System Failures (JMS and	If pending transactions exist in the lost/removed TX, JMS log, then those	**NO messages in out file** Server .log file:	Restore TX, JMS logs to

Table 1: Results of the different failures injected in an Oracle Fusion Middleware SOA cluster using JMS Adapter

TYPE OF FAILURE	EXPECTED BEHAVIOR/IMPACT	ASSOCIATED EXCEPTIONS, MESSAGES IN LOGS	POSSIBLE RECOVERY
TX logs)	may cause underprocessing/loss of messages in outbound destinations.	####<DATE> <Warning> <JTA> <strasha02> <WLS_SOA2> <[STUCK] ExecuteThread: '4' for queue: 'weblogic.kernel.Default (self-tuning)'> <<*WLS Kernel*>> <> <> <1288703109088> <BEA-110484> <The JTA health state has changed from HEALTH_OK to HEALTH_WARN with reason codes: Resource WLStore_soaedg_domain_SOAJMSFileStore_auto_2 declared unhealthy.> For the regular persistency of files, hanged operations (like ungraceful drop of the NFS connection) the WLS servers blocked without reporting any exceptions until write was reported successful or failed. Persistent store marked as unhealthy in server's log.	latest PIT

Test Results

The tests showed a memory-intensive behavior (rather than disk-intensive behavior) during failovers. There was practically no variation in the vmstat[1] bi/bo metrics or CPU during the failover windows. The short payloads were quickly persisted by the WLS JMS infrastructure. Because the JMS adapter is file-system intensive (by means of its use of JTA and JMS resources), it was especially interesting to analyze the effect of failures in the storage used for transactions and JMS logs while the client was feeding the DemoInQueue. The following different behaviors were observed depending on the type of failure:

- Deletion of the transaction or JMS logs while insertions were taking place did not cause any run-time issues.

  This behavior was due to the default file system and NFS behavior, where file system's related inodes continue to exist as long there's already an existing user for them. For example, even if the file is deleted by another session, write I/O's continue until all users and processes close the file, at which point the file is deleted.

- I/O calls that were blocked for a long period of time (the default timeout is 30 seconds) but that did not necessarily implied a write failure (like drops of TCP connections to the Network Attached Storage device (NAS)) caused the JTA subsystem to time out its internal requests to the store, forced a roll back of ongoing transactions and, caused the store to be marked as `UNHEALTHY` by the JTA manager.

  This occurs whenever the store fails to respond to JTA requests within two minutes). This behavior prevented new transactions from being initiated but it did not force the shutdown of the server.

[1] Virtual memory statistics (vmstat)

- Effective/confirmed write failures in the NAS device caused the affected file stores to shut down, which in turn flagged its dependent resources (such as JMS or JTA) into an `UNHEALTHY` state.

  At this point, WebLogic Server was locally restarted by Node Manager and migrated if the failure persisted. If the NAS device was unavailable in the failed-over node as well, the servers are shut down completely and the system comes to a halt (including blocking the client making the invocations).

However, for all of the above behaviors, no data loss took place. The "Possible Recovery" column in Table 1 above reflects the scenarios when the stores are *completely gone* and need to be restored from a backup.

Table 2 summarizes the results of a set of initial tests that were performed to identify possible duplication and under-processing scenarios. .

TABLE 2. FAULTS, PROCESSING TIMES AND MESSAGE COUNTS FOR JMS ADAPTER FAILURE SCENARIO

TEST	DESCRIPTION	INSERTS	FAULTS	# MESSAGES DEMOOUT QUEUE (WLS_SOA1+ WLS_SOA2)	AVERAGE DURATION PER INSTANCE (SECS)	MAX INSTANCE TIME (SECS)	MIN INSTANCE TIME (SECS)	AVERAGE PROCESSING TIME (DEQUEUE+ MEDIATOR+ ENQUEUE)(SECS)
T1	Client feeding the system with composite stopped with failure injection: sent messages to DemoInQueue.	59997 (3x19999) (received 31389 in WLS_SOA2, 28608 in WLS_SOA1)	0	N/A	N/A	N/A	N/A	N/A
T2	Test the consumption by the composite without failure injection when client has fed all message (for example: first push all messages to DemoInQueue, then start composite).	59997 (3x19999)	0	59997 (31389+28608)	0.036	1.453	0.019	0,097
T3	Test the consumption with failure injection when client has fed all (i.e. first push all messages to DemoIn, and start composite. While the composite is consuming messages, inject failures).	59997 (3x19999)	0	59997 (29997+30000)	0,0307	2.045	0,010	0,079
T4	Test the consumption with failure injection when client is feeding.	79996 (4x19999)	0	79996 (37084+42912)	0.0308	2.145	0.003	0,0838
T5	Test the consumption with failure injection while the client is feeding.	59997 (3x19999)	0	59997 (28379+31618)	0,0293	2.224	0.004	0,0723
T6	Test the consumption with failure injection while the client is feeding.	59997 (3x19999)	0	59997 (30910+29087)	0,0305	1,78	0.003	0,0668

The overall metrics (performed with `WLST` queries and the `WLSTATS`[2] utilities) showed a balanced consumption and throughput in both nodes. As explained above, when a distributed queue member is down (such as during the server migrations performed in the tests), the JMS Adapter reattaches and is load balanced to another member. However, when the failed member comes up again, the adapter does *not* load balance back to the original member. Because the destinations were deployed in the same cluster as the JMS Adapter, failovers (restarts) guaranteed that at least one adapter was consuming from the restarted destination. See the "Best Practices for Mediator and JMS Adapter" section for more details.

Tests T4, T5, and T6 ran with the same configuration for the adapter. These tests were intended to identify possible windows for transaction loss or submission of the same message more than once. None of the server migrations that took place generated any sort of faults, duplications, or errors. Transactions were properly recovered in all cases and the same number of messages that were inserted in the inbound queues, were pushed by the composite to the outbound queues. **This confirms that appropriate delivery and correct behavior (one-and-only-once delivery) is guaranteed for systems using the JMS adapter and Oracle WebLogic Uniform Distributed Queues as configured in the example. This is an effective supportability statement for Oracle JMS Adapter configuration with Uniform Distributed Queues. (**See the "Best Practices for Mediator and JMS Adapter" section for more details.)

Analysis of Server Migration on the Node's Performance (Process Failures)

Given that server migration is used especially in JMS Adapter scenarios (where UDD[3] producers and consumers rely on server migration for transaction protection in case of node failures), a series of failover latency measurements were performed for the JMS Adapter example. The JMS Adapter example represents a case where engine failover takes place. At the same time, transaction logs and JMS stores are recovered from a different node (using a common mount location to both SOAHOST1 and SOAHOST2). Other adapters and BPEL and Mediator service engines by themselves do not make so intensively use of JMS.

The following screenshots and graphs show the effect of failures and server migration on the available resources in the nodes. This part of the analysis does ***not*** provide any performance metrics per se, but compares the behavior of each node while server migration is taking place. During the analysis, the number of processes waiting, the free memory, the cache memory, and the idle CPU was obtained every three seconds from both nodes.

The following images show the results for the following server migrations:

- WLS_SOA2 from SOAHOST2 to SOAHOST1
- WLS_SOA2 from SOAHOST1 to SOAHOST2

[2] The wlstats program is a command-line utility that uses standard JMX mbean calls to print publicly documented WebLogic JMX runtime mbean statistics to the java console. It is not intended for production systems due the high number of Mbean invocations it performs.

[3] Uniform Distributed Definitions (UDD).

The following graphs compare the effect of server migration on the system's resources when the same WLS server is migrated from one node to another. Random windows of time were used for the analysis (some graphs reflect the state at 9:00 am, others at 4:00 pm, and so on) to eliminate ramp-up values and use points in time where the clients had already been running for awhile.

Note the following:

- The hardware configurations for SOAHOST1 and SOAHOST2 are the same. The relevant information to this respect is the comparison for the nodes more than the values per se. For example, this analysis is not intended to provide expected CPU and memory values that can be extrapolated to other systems.
- Data is shown for windows of time starting when server migration is triggered. Failures/server migration are triggered at HH:00, HH:15, HH:30 and HH:45 (see Appendix A for details). The period of time shown starts 3 minutes before server migration is triggered and continues until the system stabilizes, which is approximately 10 minutes after failures are introduced.
- This server migration analysis was run for the JMS Adapter because it represents a case where engine failover take place and at the same time transaction logs and JMS stores are recovered from a different node (using a common mount location to both SOAHOST1 and SOAHOST2).

Failover Node (SOAHOST1):

Server Migration at HH:15 (transaction rolled back in client)

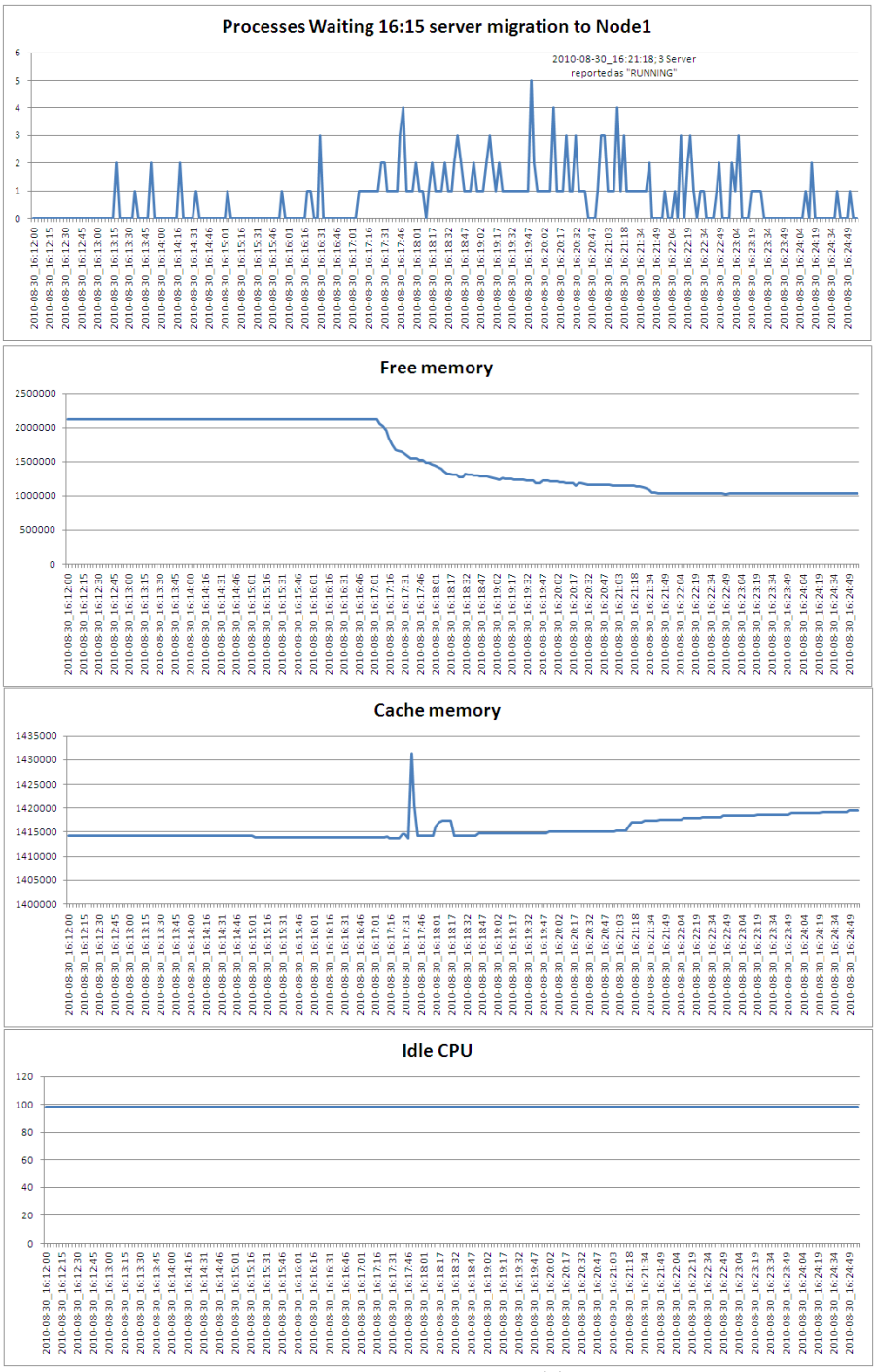


Image 2: Effect of Server Migration on SOAHOST1 Resources

Average # of process waiting since 16:15 till 16:25 (stabilized system) = 0,84

Failure Node (SOAHOST2)

Server Migration at HH:15 (transaction rolled back in client)

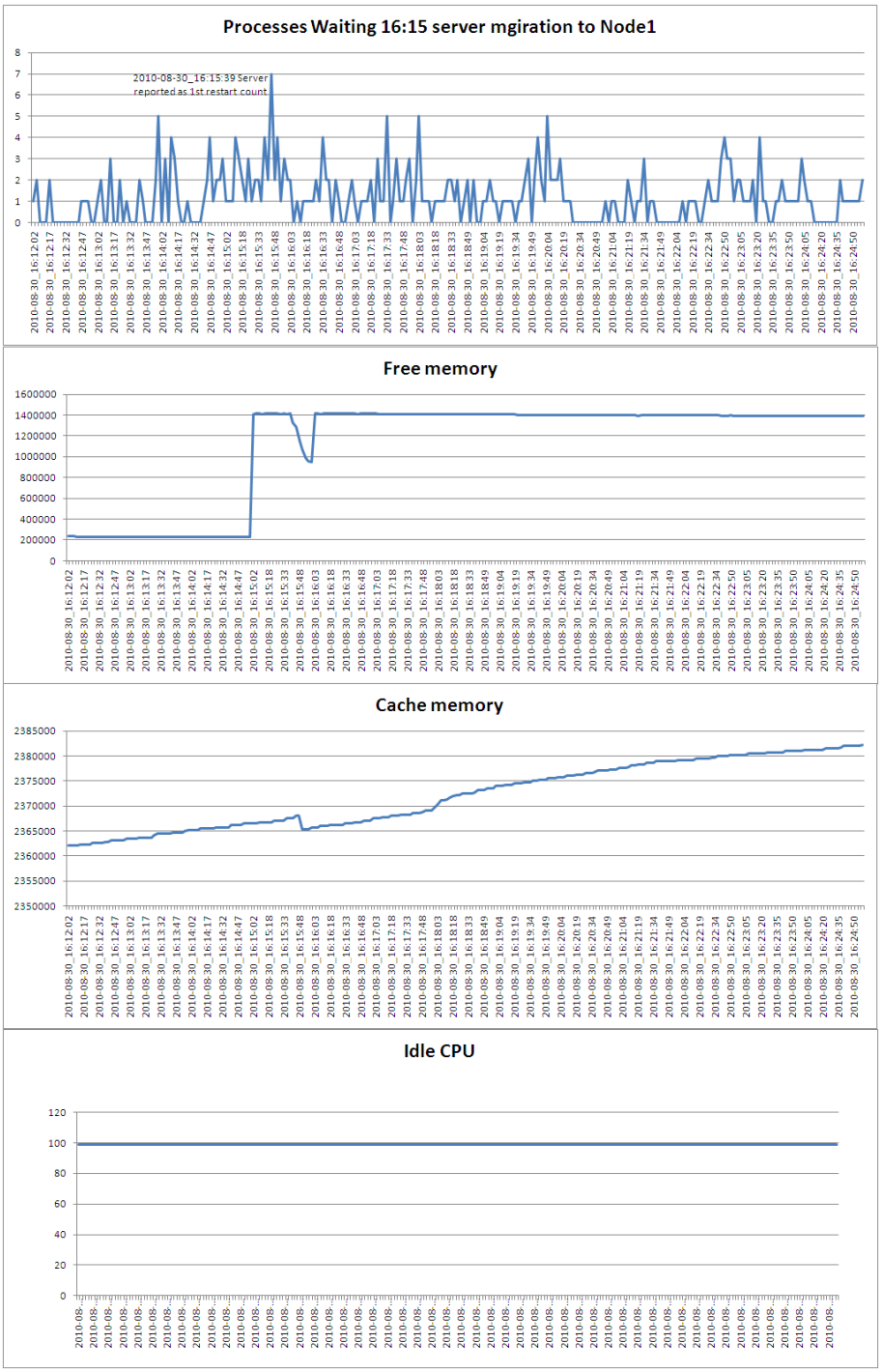


Image 3: Effect of Server Migration on SOAHOST2 Resources

Average # of process waiting since 16:15 till 16:25 (stabilized system) = 1,23

Failover Node (SOAHOST2)

Server Migration at HH:45 (transaction rolled back in client)

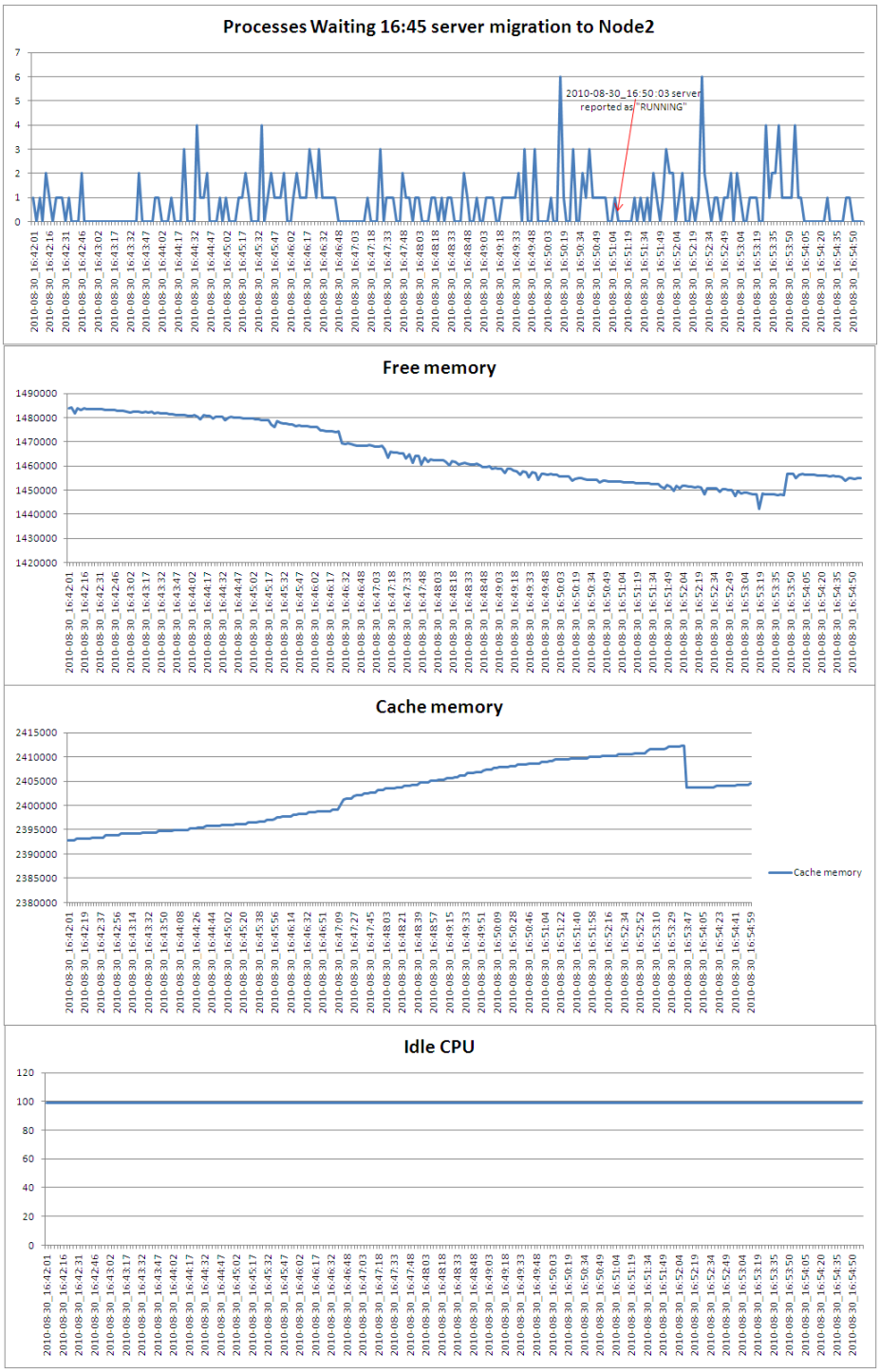


Image 4: Effect of Server Migration on SOAHOST2 Resources

Average # of process waiting since 16:45 till 16:55 (stabilized system) = 0,745

Failure Node (SOAHOST1)

Server Migration at HH:45 (transaction rolled back in client)

Image 4: Effect of Server Migration on SOAHOST1 Resources

Average # of process waiting since 16:45 till 16:55(stabilized system) = 0,726

The most relevant variations in resources when failures were injected happened for the number of processes waiting. Table 3 shows the average number of process during the 10-minute period, starting from the failure injection.

TABLE 3. NUMBER OF PROCESSES WAITING DURING SERVER MIGRATION

FAILURE INJUECTION TEST / PIT	AVERAGE # PROCESSES WAITING FOR FAILOVER NODE (FO)	AVERAGE # PROCESSES WAITING IN FAILURE NODE (FN)
T1: 9:15	0,844 (SOAHOST1)	0,795 (SOAHOST2)
T4:11:15	0,395 (SOAHOST1)	0,340 (SOAHOST2)
T7:16:30	1,361 (SOAHOST1)	0,435 (SOAHOST2)
T5:11:30	1,343 (SOAHOST1)	0,160 (SOAHOST2)
T8:16:45	0,829 (SOAHOST2)	0,779 (SOAHOST1)
T2:9:45	1,145 (SOAHOST2)	0,919 (SOAHOST1)
T3:11:00	0,575 (SOAHOST2)	0,884 (SOAHOST1)
T6:12:00	0,601 (SOAHOST2)	0,803(SOAHOST1)

The following list describes the conclusions inferred from these tests and the metrics obtained:

1. **For process failures, server migration has a considerable resource impact on the server experiencing the failure**.
   The average number of process waiting on the **failover** node is not much higher than the average number of processes waiting in the server that is experiencing the failure. (Average FO=0,886) vs. (Average FN = 0,634). The way server migration is triggered justifies this: After the first failure, the local NM will try to restart the failed server. To a great extent, this triggers CPU and RAM consumption. The impact on process wait times is higher if the node experiencing the server failure runs another server at that point in time (as is the case for tests T1, T4, T8, and T2).

2. **Highest resource impact takes place for servers migrating to nodes that already contain running servers.**
   (Average FO [T7,T5, T3,T6]=0,97) > (Average FO [T1, T4, T8,T2]=0,80)

   This is expected and reflects the high impact of migrating a server to a node that *already* has another server running. It is also interesting to note that the number of processes waiting in the failure node is higher for the case when the failure node does contain a running instance:

   Average FN [T7,T5, T3,T6]=0,57) > (Average FN [T1,T4, T8,T2]=0,71

3. **The worst failover scenario and maximum resource contention case takes place for the node hosting the administration server.**

   (Average FO[SOAHOST1]=0,987 > Average FO[SOAHOST2]=0,785))

   This behavior sets also the limits for the appropriate capacity planning (see the Best Practices for JMS Adapter section).

4. **The effect of restarts on the system throughout is not too high for long-running systems.**

   The wlstat utility showed relatively small differences in the throughput for higher number of server migrations/local restarts).

   This qualifies rolling restarts as a plausible approach for reload balancing the system post server failures or scale out scenarios. (See the "Best Practices for JMS Adapter" section.)

Best Practices for JMS Adapter

The following items describe the best practices for Mediator and JMS Adapter inferred from the tests performed:

Analyze JMS Client Failover Separately From the References/Endpoint Behavior

For any SOA system using Uniform Distributed Queues and the JMS Adapter, prepare and analyze JMS client failover separately from the references/endpoint behavior. Before trying a failure injection on the system (SOA infrastructure), make sure your external producers and consumers are transitionally safe and highly available. For example, verify failure injection without composites started so that it can be guaranteed that destinations are receiving the expected messages. After no message loss or duplication is guaranteed in the perimeter, then verify the composite and the adapter itself.

Collocate Consumers and Producers In the Same Cluster

Collocate consumers and producers in the same cluster (UDD destinations and JMS Adapter) as much as possible, until a more granular scale out/up is required. When destinations are deployed in a separate cluster from the JMS Adapter, server migration and local restarts may cause the servers to be totally inactive from JMS consumption point of view (no JMS Adapter consumes from them). If at least one JMS Adapter is collocated with the destination, then failovers and restart at the same time as the destination guarantee that at least one adapter will consume from the restarted destination. Alternatively, if a more distributed system is used (where destinations are remote to the JMS Adapter Cluster), rolling restarts of both the adapter and destination servers may be plausible to guarantee a balanced system. (**Note:** Oracle Fusion Middleware 11g Release 1 Patch Set 3 enhancements in the JMS stack eliminate the need for rolling restarts and collocation of consumers and producers.

Watch for Stuck Messages Post Server Migration and In Scale Up/Out Use Cases

The JMS Adapter (Oracle Fusion Middleware 11g Release 1 and Release 1 patch set 1 and patch set 2) cannot create new consumers after it is already attached to UDD members. The JMS Adapter also maintains the corresponding threads attached to a destination member once consumption is initiated and during its life time. If the server that experienced the failure does not restart, then the messages in the corresponding failed destination are not consumed, which is referred to as being *stuck*. Similarly, the JMS Adapter cannot add new members/servers dynamically (members that join after the initial destination's attachment is performed will not be consumed from). It is recommended to perform rolling restarts of the servers holding the composite—one at a time—so that load is correctly distributed post failures and when new servers are added.

Increase the Default Value for JMS Receiving Threads

The `adapter.jms.receive.threads` parameter controls the number of thread to be used for consuming from different destination members. This parameter default value is 1. This means that adapter instances consume

from one unique member in a UDD configuration. This has performance and availability implications. To guarantee continuous service through failures in the destination members and to improve performance and scalability (so that each member of a Distributed Destination is serviced by an adapter poller thread), increase the value of the property according to the number of members in the UDD system. This is a binding property that is set in each composite:

```
<property name="adapter.jms.receive.threads" type="xs:string" many="false">4</property>
```

BPEL and Oracle Database Adapter Analysis

Brief Introduction to the Adapter

Oracle Database Adapter enables Oracle SOA Suite and Oracle Fusion Middleware to communicate with database end points. These include Oracle Database servers and any relational databases that comply with ANSI SQL and provide JDBC drivers. Oracle Database Adapter works in conjunction with Oracle BPEL Process Manager and Oracle Mediator. The adapter relies on an underlying WLS JDBC driver to communicate with the database.

When Oracle Database Adapter polls for database events (usually an `INSERT` operation on an input table) and initiates a process in a Mediator component or an SOA composite, it is referred to as an *exposed service*. In an Oracle BPEL process it is referred to as *partner link tied to a receive activity*. The expression *inbound* (from database into SOA) is commonly used. When you use Oracle Database Adapter to invoke a one-time DML statement such as `INSERT` or `SELECT` in a Mediator component or an SOA composite, it is called a *service reference*. In an Oracle BPEL process, it is referred to as "*partner link tied to an Invoke activity."* The expression *outbound* from SOA out to the database, is commonly used to refer to the Invoke activity.

High Availability Considerations for Oracle Database Adapter

You can configure Oracle Database Adapter in both Active-Active and Active-Passive set-ups for high availability. Although Active-Passive may be a plausible approach depending on the type of composite, the Active-Active implementation provides higher scalability and better overall availability rates than the Active-Passive solution.

Configure Retry Invocations

Oracle Database Adapter can survive database failures without data loss, and you can configure Database Adapter to retry invocations in different ways. The database adapter is configured by default to retry a number of times when a retryable exception takes place on outbound invokes (four by default when creating references in the JDeveloper Configuration wizard) and indefinitely on retryable faults on the inbound side. The adapter classifies the most common exceptions as auto-retryable or not.

Additionally, the database adapter allows you to define "custom" or uncommon remote exceptions as auto-retryable (as opposed to facing binding faults), by specifying the codes of the exception in the `weblogic-ra.xml` file. Additionally for time outs in the connections to the database, the deployed composites can be configured to automatically retry the operation (whether inbound or outbound). The number of retries, intervals between retries, and the back-off factor for retries (the increasing periods of time between retries) are all configurable through the corresponding JDeveloper configuration wizard. The parameter settings for retries are part of the `adapater.jca` file).

To configure the behavior for retries in outbound interactions, you set the `jca.retry.*` parameters:

- For inbound invocations, the jca.retry.count parameter indicates the maximum number of retries before rejection.
- For outbound invocations, the jca.retry.count parameter indicates the maximum number of retries before throwing a retryable error back to the invoking service engine.

For more information, see the chapter about "*Adapters Life-Cycle Management" in the Oracle Fusion Middleware User's Guide for Technology Adapters documentation at* http://download.oracle.com/docs/cd/E14571_01/integration.1111/e10231/life_cycle.htm#BABBAIJJ).

Configure a Distributed Polling Strategy

When multiple Oracle Database Adapter process instances are deployed to multiple Oracle Fusion Middleware SOA servers, you need to configure a Distributed Polling strategy using the Adapter's Configuration Wizard. The `MaxTransactionSize` and `MaxRaiseSize`parameters are used to adjust the number of rows per read and number of documents raised to the corresponding service engines, respectively. Concurrency in active-active configurations is increased by setting the adapter binding property `ActivationInstances` or the number of threads for the service (NumberofThreads property in `adapter.jca` file).

> **Note**: In Oracle Fusion Middleware 11g Release 11.1.1.3 (patch set 2, PS2), both the `ActivationInstances` and `NumberofThreads` parameters can be used to increase the effective number of polling instances in a distributed polling configuration.

Using Distributed Polling implies the use of `SELECT FOR UPDATE SKIP LOCKED` syntax to perform database queries. Concurrent threads each try to select and lock the available rows for the adapter insertions, but the locks are only obtained on fetch. If a row that is about to be fetched is already locked, the next unlocked row is locked and fetched instead. `SELECT FOR UPDATE SKIP LOCKED` locks are released when the database session dies or is stopped. Messages to BPEL are returned in the Database Adapter's transaction used for polling, so *once-and-only-once* delivery is guaranteed.

> **Note:** The use of Distributed Polling is not configured in the JCA file for the adapter itself. Instead, it is specified as one of the attributes in the mappings file for the `adapter.jca`, as follows:

To disable distributed polling:

```
<query name="InboundServiceSelect" xsi:type="read-all-query">
   <reference-class>InboundService.ReceiverDept2</reference-class>
   <lock-mode>none</lock-mode>
   <container xsi:type="list-container-policy">
      <collection-type>java.util.Vector</collection-type>
   </container>
</query>
```

To enable distributed polling:

```
<query name="InboundServiceSelect" xsi:type="read-all-query">
   <reference-class>InboundService.ReceiverDept2</reference-class>
   <refresh>true</refresh>
   <remote-refresh>true</remote-refresh>
   <lock-mode>lock-no-wait</lock-mode>
   <container xsi:type="list-container-policy">
      <collection-type>java.util.Vector</collection-type>
   </container>
</query>
```

Take care to correctly set the `Polling Strategy`, the `Number of Instances`, the `MaxTransactionSize`, and the `MaxRaiseSize` properties because these parameters impact on failover latency and the overall performance of the system.

Use Case for Verifying the Availability of the Database Adapter

See Appendix B for a description of the composite, the failures injected in the system, the type of client, and the tools and scripts used to verify the high availability behavior of the Database Adapter. These details are useful to verify appropriate failover and load balancing in a Database Adapter composite.

Summary of Behaviors and Detailed Results

Behaviors Observed During Failure Injection

Table 4 summarizes the results of the failure injection tests run for the Database Adapter.

TABLE 4. RESULTS OF THE DIFFERENT FAILURES INJECTED IN AN ORACLE FUSION MIDDLEWARE SOA CLUSTER USING ORACLE DATABASE ADAPTER

TYPE OF FAILURE	EXPECTED BEHAVIOR AND IMPACT	ASSOCATED EXCEPTIONS, MESSAGES IN LOGS	POSSIBLE RECOVERY
WLS Server failure (Automatic restart by local NM)	• No message loss nor duplications takes place. • All messages injected in EMP2_RECEIVER table are pushed correctly to EMP2_SENDER table. • Non high performance impact of failures on failover node. • Non high-resource consumption penalty for restarts. • Re-load balancing for restarted WLS Managed Server.	**Node Manager Output:** <INFO> <Successfully removed X.X.X.X from eth0:Y.> … … <Server failed so attempting to restart (restart count = 1)> … <Sleeping for 30 seconds before attempting to restart server> .. <WARNING> <Successfully brought X.X:X:X with netmask 255.255.255.0 online on eth0:Z>	N/A
WLS Server migration due to successive failures in restarting the server in the local node	• No message loss nor duplications takes place. • All messages injected in EMP2_RECEIVER table are pushed correctly to EMP2_SENDER table. • Non high performance impact of failures on failover node. • Non high resource-consumption penalty for restarts. • Re-load balancing for restarted WLS Managed Server.	**Local-Node Manager output for failed restarts:** <INFO> <Successfully removed X.X.X.X from eth0:Y.> … <Server failed so attempting to restart (restart count = 1)> … <Sleeping for 30 seconds before attempting to restart server> .. INFO: Server failed during startup so will not be restarted **Local-Node Manager output for node failure:** N/A **Failover-node Node Manager output:** WARNING> <Successfully brought X.X:X:X with netmask 255.255.255.0 online on eth0:Z>	N/A
WLS Server migration due to node failure	• No message loss nor duplications takes place. • All messages injected in EMP2_RECEIVER table are pushed correctly to EMP2_SENDER table.	**Failover-node Node Manager output:** WARNING> <Successfully brought X.X:X:X with netmask 255.255.255.0 online on eth0:Z>	

TABLE 4. RESULTS OF THE DIFFERENT FAILURES INJECTED IN AN ORACLE FUSION MIDDLEWARE SOA CLUSTER USING ORACLE DATABASE ADAPTER

TYPE OF FAILURE	EXPECTED BEHAVIOR AND IMPACT	ASSOCATED EXCEPTIONS, MESSAGES IN LOGS	POSSIBLE RECOVERY
	• Non high-performance impact of failures on failover node. • Non high-resource consumption penalty for restarts. • Re-load balancing for restarted WLS managed server.		
Database Instance Failure (hard)	• No message loss nor duplications takes place. • All messages injected in EMP2_RECEIVER table are pushed correctly to EMP2_SENDER table.	java.sql.SQLRecoverableException: No more data to read from socket at oracle.jdbc.driver.SQLStateMapping.newSQLException(SQLStateMapping.java:105) at oracle.jdbc.driver.DatabaseError.newSQLException(DatabaseError.java:135) .. Caused by: oracle.net.ns.NetException: Listener refused the connection with the following error: `ORA-12514, TNS:listener does not currently know of service requested in connect descriptor`	NA
Database Node Failure (hard)	• No message loss nor duplications takes place. • All messages injected in EMP2_RECEIVER table are pushed correctly to EMP2_SENDER table.	java.sql.SQLRecoverableException: No more data to read from socket … java.sql.SQLException: The Network Adapter could not establish the connection at oracle.jdbc.driver.SQLStateMapping.newSQLException(SQLStateMapping.java:74)	NA
File System Failures (TX logs)	If pending transactions existed in the lost/removed TX log, those may cause underprocessing or loss of rows in outbound writes.	**No messages in out fileServer .log file:** ####<DATE> <Warning> <JTA> <strasha02> <WLS_SOA2> <[STUCK] ExecuteThread: '4' for queue: 'weblogic.kernel.Default (self-tuning)'> <<WLS Kernel>> <> <> <1288703109088> <BEA-110484> <The JTA health state has changed from HEALTH_OK to HEALTH_WARN with reason codes: Resource WLStore_soaedg_domain_SOAJMSFileStore_auto_2 declared unhealthy.> For the regular persistency of files, hanged operations (like ungraceful drop of the NFS connection) the WLS servers blocked without reporting any exceptions until write was reported successful or failed. Persistent store marked as unhealthy in server's log	Restore transaction logs to latest point in time

Behavior When Distributed Polling is Disabled

For the purpose of describing the expected errors in a distributed/clustered system where the DB adapter is used and distributed poling has NOT been configured, the system was stressed for 12 hours, using the client described (NOTE: This test was just intended to verify the expected behavior more than to describe a realistic configuration). After a few iterations, the Oracle Enterprise Manager FMW Control showed a considerable amount of faults:

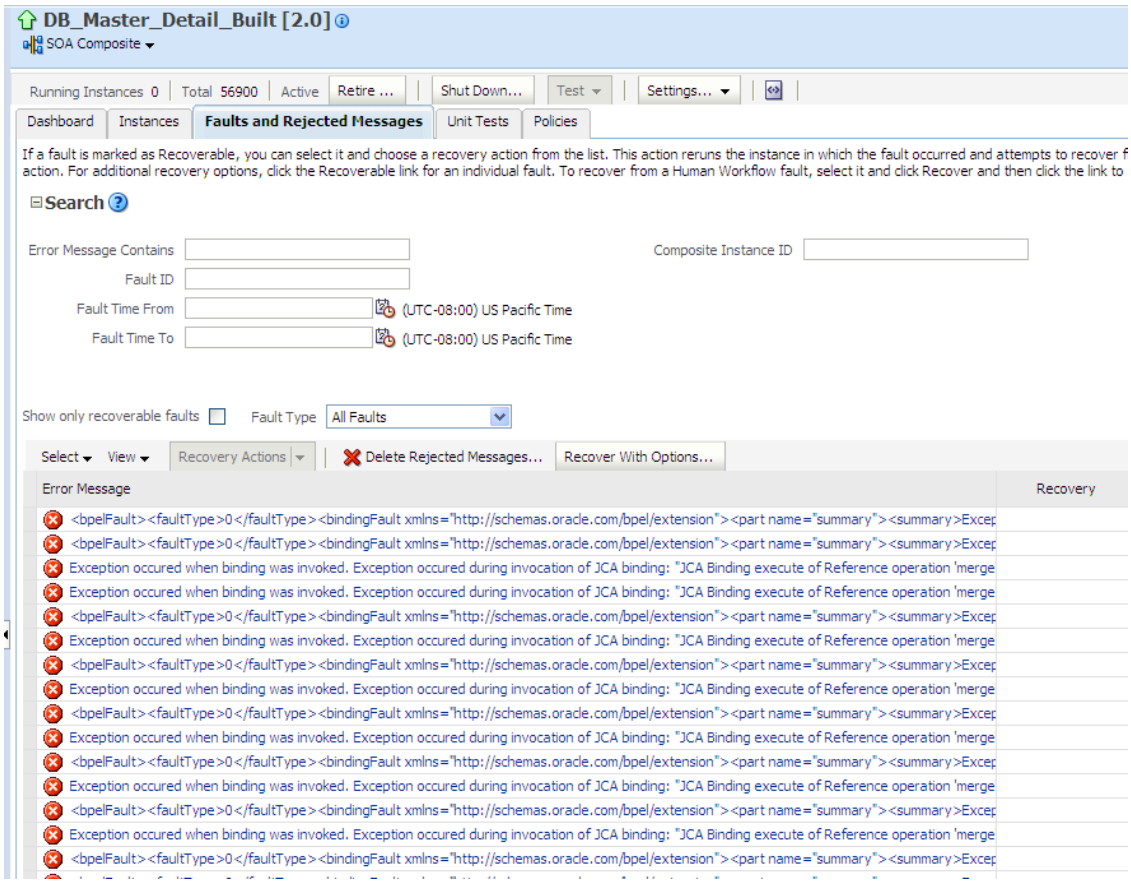


Image 4: Typical exceptions reported in Oracle Enterprise Manager Fusion Middleware Control for a Database Adapter cluster processing with Distributed Polling Disabled

As expected, the errors that were reported indicated issues with concurrency and primary key violations:

```
<bpelFault>
        <faultType>0</faultType>
        <bindingFault xmlns="http://schemas.oracle.com/bpel/extension"><part
        name="summary"><summary>Exception occured when binding was invoked. Exception occured
        during invocation of JCA binding: "JCA Binding execute of Reference operation 'merge'
        failed due to: DBWriteInteractionSpec Execute Failed Exception. merge failed. Descriptor
        name: [OutboundService.SenderDept]. Caused by java.sql.BatchUpdateException: ORA-00001:
        unique constraint (SCOTT.SENDER_PK_DEPT) violated . ". The invoked JCA adapter raised a
        resource exception. Please examine the above error message carefully to determine a
        resolution. </summary></part><part name="detail"><detail>ORA-00001: unique constraint
        (SCOTT.SENDER_PK_DEPT) violated </detail></part><part name="code"><code>1</code></part>
        </bindingFault>
</bpelFault>
```

These issues were also traced in the WLS_SOA1 and WLS_SOA2 output files:

```
Caused By: com.oracle.bpel.client.BPELFault: faultName:
{{http://schemas.oracle.com/bpel/extension}bindingFault}
parts: {{
summary=<summary>Exception occured when binding was invoked.
Exception occured during invocation of JCA binding: "JCA Binding execute of Reference operation
'merge' failed due to: DBWriteInteractionSpec Execute Failed Exception.
merge failed. Descriptor name: [OutboundService.SenderDept].
Caused by java.sql.BatchUpdateException: ORA-00001: unique constraint (SCOTT.SENDER_PK_DEPT)
violated
.
".
The invoked JCA adapter raised a resource exception.
Please examine the above error message carefully to determine a resolution.
</summary>
,detail=<detail>ORA-00001: unique constraint (SCOTT.SENDER_PK_DEPT) violated
</detail>
…
```

The above are the exceptions that can be expected in a Database Adapter system when multiple instances of the adapter in a SOA Cluster are concurrently processing the same tables without Distributed Polling strategy enabled.

Results for multiple instances without distributed polling:

- Number of Insertions: 50000
- Number of Composite Instances: 69595
- Faulted Instances: 5911
- Number of rows in SENDER_EMP: 48969
- Number of WLS_SOA servers restarts: 69

The tests showed a10% chance of getting PK violations and the results worsen with an increasing number of threads and smaller polling intervals. Because each polling thread accesses the database with higher frequency, it increases the chance of concurrently processing the same row.

Behavior When Distributed Polling is Enabled

Use Distributed Polling when multiple Oracle Database Adapter process instances are deployed to multiple Oracle BPEL PM or Mediator nodes. Duplicate reads (and PK violations on the outbound operation in the MasterDetail example shown in Appendix B) are directly caused by parallel reads performed by multiple SOA servers.

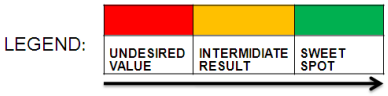
In a typical production environment, information in the inbound tables is typically inserted at random intervals, and not at a fixed or established frequency. Thus, the polling interval, which is the sleep time between polls for new records, is not easily tunable. It is also impossible to determine the timeline separating the polling of rows between the multiple WLS_SOA instances in a cluster, because the initial poll may happen on different points in time (depending on restart periods, workload on the node, and so on). Different tests were carried out, with each test varying the `PollingFrequency`, `MaxTransactionSize`, `MaxRaiseSize`, and `Number of Instances` parameters.

Table 5 describes the results obtained when Distributed Polling was enabled in the test environment. **Note:** In reality, each row in the table reflects the average of three tests executed with the same configuration. The tests were repeated to obtain more meaningful metrics and eliminate point conditions during the tests.

TABLE 5 FAILOVER LATENCY AND THROUGHPUT FOR DATABASE ADAPTER CLUSTER

TEST	POLLING INTERVAL	NUMBER OF THREADS	MAX RAISESIZE	MAX TRANSACTIONSIZE	ACTIVATION INSTANCES	OUTBOUND THROUGHPUT (*)	AVERAGE FAILOVER LATENCY (SECS)
T2	5	1	4	40(**)	DEFAULT (1)	19,83X4=79,32	246.00
T4	30	1	2	20	DEFAULT (1)	27,28X2=55,64	251.38
T5	30	1	2	20	5	28,51X2=57,02	240.13
T7	30	1	2	20	20	30,23X2=60,46	248.50
T8	30	1	5	5	4	14,16X5=70,8	247,16
T9	30	1	5	20	50	60,36X5=301,8	254,38

(*) Number of BPEL Instances per minute × Messages per instance
(**)Polling frequency limits TransactionSize's effective value



Analysis of results:

The tests did not prove any direct relationship between the rate of BPEL instances being created and the failover latency. Rather, testing verified that:

- The larger the number of rows read by the adapter, the higher the failover times.
- Transaction recovery and message redelivery seem to be the major cause of restart delay.
- The highest failover latency was measured for the lowest number of activation instances with the highest number of transactions raised.

It needs to be understood that the polling frequency may become a limiting factor for `MaxTransactionSize` if the injection rates are not too high (such as shown in the case of T2 in Table 5 (**)). For example, whether `MaxTransactionSize` is 10 or 100, if there are not enough `pending` rows to read in each polling interval, then the maximum number of transactions will not increase:

- Setting the `MaxTransactionSize` parameter

  For a polling interval of 30 seconds with an injection rate of 1 row/sec , setting the `MaxTransactionSize` parameter to a value of 20 produced a higher failover time. The effect on failover is diminished when you increase the number of adapter instances being activated, because parallel processing improves the message flow and the chances of having the system resume large pending operations are reduced. At the same time, if the number of activation instances is too high, then the system's restart becomes a heavier operation and degrades the recovery.

- Setting the `MaxRaiseSize` parameter

  The results also showed that the value of the `MaxRaiseSize` parameter does not seem to have a big impact on recovery period (recovery in the sense of server failover). Provided that the tests show that the number of instances improves the throughput of the system, it is advised to set the number of activation instances to an intermediate value according to the processing power of each node. Consider the number of documents and the size of each document that can be concurrently processed on each server.

Best Practices for BPEL and Oracle Database Adapter

Use a Separate Data Source for the Database Adapter

As explained in the configuration for the data sources used by the adapter (see Appendix B.), for performance and availability isolation purposes, the best practice is to use separate pools for the Oracle Fusion Middleware SOA Infrastructure and Oracle Database Adapter. This allows you to optimize the configuration of each and operate on each pool separately without affecting the other. Configure a multi data source according to the following recommendations:

1. **XA requirements:** The database adapter participates in distributed transactions and requires the back-end database setup for XA recovery by Oracle WebLogic Transaction Manager. Ensure that XA pre-requisites are met. For this, log on to SQL*Plus as a system user, for example, enter `sqlplus "/ as sysdba"` and then grant the following privileges:

```
Sql>Grant select on sys.dba_pending_transactions to public.
Sql>Grant execute on sys.dbms_xa to public.
Sql>Grant force any transaction to public;
```

2. **Server-side load balancing:** If the server-side load balancing feature has been enabled for the Oracle RAC back end (using remote_listeners) used by the Database Adapter, the JDBC URL used in the data sources of a multi data source configuration should include the `INSTANCE_NAME`. (i.e. when creating the data sources, select **Oracle's Driver (ThinXA) for RAC Service-Instance connections** database driver.)
3. **Multi data source settings:** Configure the multi data source used by the Database Adapter with a test frequency of five seconds and specify "load-balancing" for the algorithm type. These settings are specified in the general properties for the multi data source. **Note:** Depending on the specific database and application using the adapter these settings may vary.
4. **Data source settings:** For the data sources in the multi data source, use the parameter settings shown in the following table:

PARAMETER	CONFIGURED VALUE
initial-capacity	0
Property command	<property> <name>oracle.net.CONNECT_TIMEOUT</name> <value>10000</value> </property>(*)
connection-creation-retry-frequency-seconds	10

PARAMETER	CONFIGURED VALUE
test-frequency-seconds	300
test-connections-on-reserve	true
test-table-name	SQL SELECT 1 FROM DUAL (default)
seconds-to-trust-an-idle-pool-connection	0
global-transactions-protocol	TwoPhaseCommit(**)
keep-xa-conn-till-tx-complete	True (default)
xa-retry-duration-seconds	300
xa-retry-interval-seconds	60

(*) to add this property, include in the Connection Pool->Properties text field both the user and connect_timeout property in separate lines, as shown in Image 5.

(**)TwoPhaseCommit is chosen automatically when creating the data source using Thin XA for Oracle RAC Service-Instance connection.

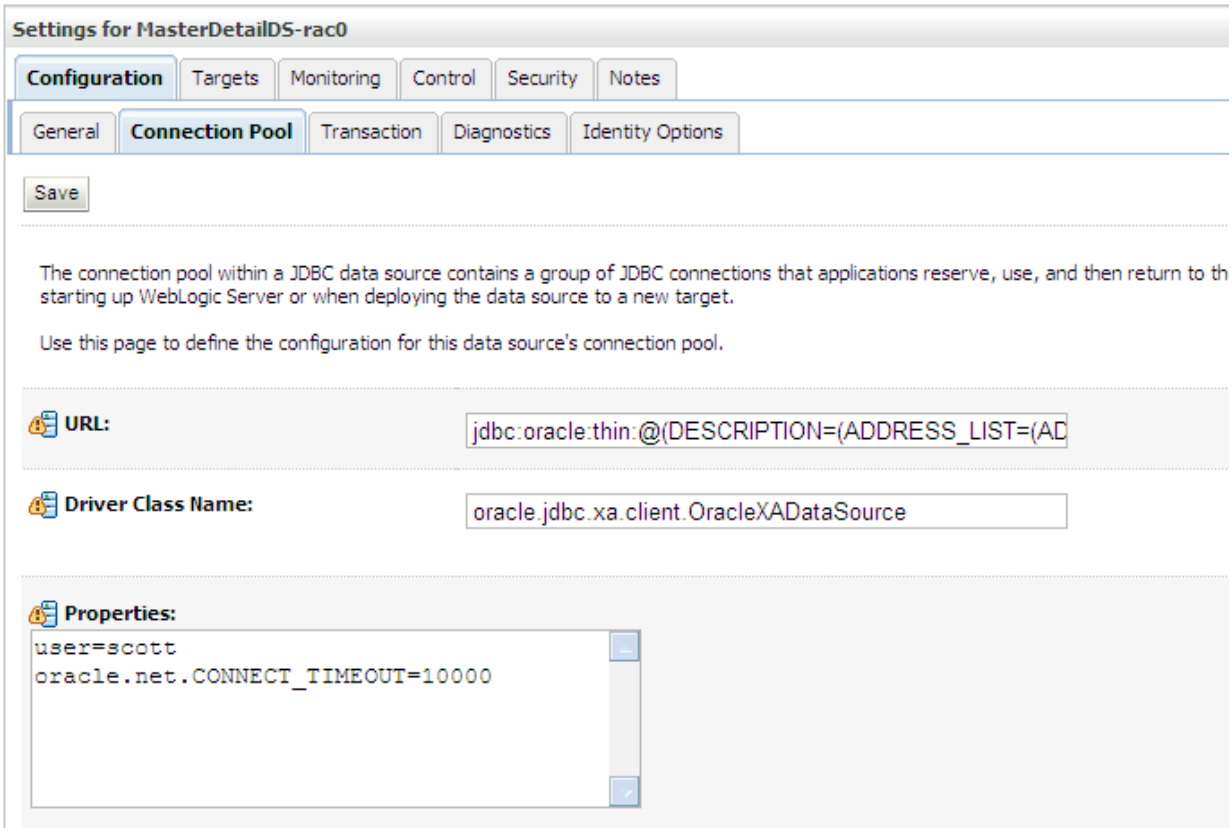


Image 5: Properties specified in the Oracle WebLogic Administration Console for the Database Adapter's Data Source

Adjust the Value of the MaxTransactionSize Parameter for Improved Failover Latency

Adjust the Database Rows per Transaction (`MaxTransactionSize`) based on the expected business process throughput but also keeping failover and load balancing in mind.

The `MaxTransactionSize` parameter defines the number of rows to process per polling interval. For example, if the database contains 1000 rows at the start of a polling interval and `MaxTransactionSize` is set to 100, then a cursor is used to read and process 100 rows at a time. When it is finished, the adapter instance releases the cursor and leaves 900 rows to be processed for the next polling interval or for other instances to consume the next 100 rows. Consider the following when setting the `MaxTransactionSize` parameter:

- Set the `MaxTransactionSize` parameter to a value that is close to the throughput of the entire business process (what the service engine will process as output), and relatively low to minimize failover latency. When set using these guidelines, no document buffering takes place and in-memory batching-processing is reduced.
- Do not set the `MaxTransactionSize` too low in a distributed environment, because it forces the creation of a cursor for each iteration.
- Do not set `MaxTransactionSize` too high because it:
  - Can lead to transaction time outs and because, in the event of a server migration, it can force a large number of transactions to be blocked until failover completes.

Can lead to memory issues derived from maintaining too big payloads in each polling instance.

Adjust XML Records to be Raised to Service Engines Based on the Inbound Processing Rates

Adjust the number of XML records to be raised to service engines (`MaxRaiseSize`) based on the number of rows expected to be read per poll interval (driven by the injection rate, polling interval and `MaxTransactionSize`). The number of rows per XML document determines the batch size between the database adapter and its consumer: Oracle BPEL Process Manager or Oracle Mediator. For maximum throughput, pass a collection of rows as a single event. This reduces the overhead per instance and allows batching on the other end. That is, the database adapter `MERGE` or `INSERT` command can leverage batch database writing if it receives multiple rows as part of one invoke.

Adjust the Activationinstances or Numberofthreads Parameter for Improved Failover Latency

Set either the `ActivationInstances` or the `NumberofThreads` parameter to the desired throughput but also keeping recovery in mind. Either the `Activation instances` or `NumberofThreads` parameter can be used to adjust the number of polling threads in a SOA cluster. Activation instances control the throughput of the endpoint in each node, but, similarly to how it happens with the `MaxTransactionSize` parameter, too high values may have adverse effects in the event of a server migration (server recovery is delayed).

For example: Consider a system where an injection rate of 30 rows per minute is expected. Payloads are small and it is not expected to reach the available resource's limits (in terms of memory) even for the highest load peaks. A polling interval of 30 seconds is set. (Configure the system so that instances in different nodes alternatively grab rows. If a too large polling interval is set, it is likely one single server will grab all rows in one single poll). With a constant and regular flow of injections to the inbound rows (for example, assume a message arrives every 2 seconds), the maximum number of rows that an instance can pick up on every read is approximately 15. Hence, you should set the `MaxTransactionSize` parameter to 15. You could then set the `MaxRaiseSize`

parameter to 15 to eliminate parsing overhead (passing each row as one single doc to BPEL). This would set the optimum throughput. However, to reduce failover latency due to message recovery during migrations, consider lowering the setting for the `MaxTransactionSize` parameter and increase the number of instances. Doing so will allow parallel processing and reduce the recovery periods. Using 2 instances per node, with a `MaxTransactionSize=5` provides the following benefits:

- A better throughput (Read Throughput=2nodes ×[activationInstances × MaxTransactionSize / PollingInterval])
- A more scalable system because both servers are actively processing requests
- A lower restart latency for the system because recovery operation is lighter

BPEL and File Adapter Analysis

Brief Introduction to the Adapter

The Oracle File Adapter enables a BPEL process or a Mediator composite to exchange (read and write) files on a local file system based on a JCA 1.5 architecture. The file contents can be both XML and non-XML data formats. The Oracle File Adapter supports the following operations when integrating in a SOA composite (Mediator or BPEL service engines).

- Read file (inbound operation)
- Write file (outbound operation)
- Synchronous read file (outbound operation)
- List files (outbound operation)

Adapter High Availability Considerations

The Oracle File Adapter picks up a file from inbound directories and/or sends processed files to an output directory. These operations are non transactional in nature. Specifically, the invocation of outbound operations by service engines is not performed in a transaction with the effective write of the file to the file system. This means that write operations may result in duplication of files when failures take place in the service engines (failures that occur once the write operation to the file system has been triggered). Consider the following scenario:

1. A message/entry gets scheduled in BPEL for a write I/O operation using the File Adapter.
2. The file write I/O occurs before the BPEL invoke/instance is committed.
3. The BPEL instance fails.
4. When recovery is performed, it starts from the receive activity.
5. The BPEL engine sends the same message again for a file write I/O, resulting in a duplicate file.

When the File Adapter processes files for inbound operations in a clustered SOA system, it requires:

- A shared directory (on shared storage) that is concurrently mounted by the nodes where the SOA cluster runs.

- Shared storage locations for the folder where the control files for the adapter are stored. This is referred to as a control directory, which holds the following information used by the file adapter:
  1. De-batching information (last record that was processed in de-batchable configuration).
  2. Files that have been processed (as a record system so that they are not processed again) for read-only scenarios.
  3. Staging files for outbound batching.
  4. Sequence numbers when the database is not used for locking.
  5. The error-archival directory that is placed (by default) under the control directory. The error-archival directory is used as the location where the adapter copies files that could not be processed.

Thus, ensure that the appropriate file system protection and backup and restore mechanisms are in place to protect the control directory from failures, accidental deletions, and file corruptions.

In clustered environments using the File Adapter, a coordinator is required to guarantee that only one service processes a particular file in a distributed topology. At the same time, the system needs to ensure that if multiple references write to the same directory; these do not overwrite each other. Database-based mutex and locks are used to coordinate these operations in a File Adapter clustered topology. Other coordinators are available but Oracle recommends using the Oracle Database.

This requires configuration of the required data sources for the `eis/HAFileAdapter` connection factory. The connection factory uses by default the JDBC SOA Data Source. Hence, it can leverage the already configured high availability properties for accessing the database. To configure a database as a mutex coordinator for outbound operation, follow the steps in the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle SOA Suite*. Given the possible corruptions and duplications that may be caused by the lack of an efficient coordinator for the adapter operations, it becomes very important to guarantee the database's availability for the File Adapter processing to work properly.

When you use Oracle Database as coordinator, rows in the database are used to mark the different states of the processing cycles. Files are successively marked in these rows as "found" or "claimed for processing" and "processed." Table 6 describes these values.

TABLE 6. DIFFERENT VALUES FOR THE STATE OF A FILE AS PROCESSED BY THE FILE ADAPTER

VALUE INSERTED IN THE FILE ADAPTER IN TABLE	MEANING FOR PROCESSING LOGIC
0	File found
1	File claimed for processing by an instance
2	File already processed

The adapter performs these status updates in separate transactions from the boundaries of BPEL/Mediator. For services using the File Adapter, the following sequence of events occurs when a new file is detected in the file-system,

1. A thread inserts the appropriate mutex row into the file_processed table.

2. After the operation in step 1 is committed, then the adapter enqueues the work-item in an in-memory queue for the processors.
3. The processors dequeue from the queue, and then claim that the file is under processing by updating the status to "claimed for processing" in a new transaction.
4. After processing, the status is updated to "2" in a new transaction.
5. The entries that are marked as processed are cleaned by the poller threads at the end of each polling iteration in a separate transaction.
6. An expiration time is used for dangling entries in "claimed for processing" or "found" status. When this timer expires, other threads can pick the row for processing. This enables the system to continue processing after server failures involving long recovery periods
    - A 15-minute timer is used for records in status 1.
    - One hour is used to expire records in status 0.
    - Any thread can clean up all status 2 records, as soon as the record is read

The following configuration properties for JCA adapters are especially relevant in a failure scenario for a File Adapter system:

- `MaxRaiseSize`

  Set the `MaxRaiseSize` parameter to specify the maximum number of XML records that can be raised at a time to the BPEL/Mediator engine.

  For example, if you set `MaxRaiseSize=10`, then 10 documents are raised simultaneously. From the high availability perspective, note that larger values for the `MaxRaiseSize` parameter imply a higher recovery time when failover scenarios take place. By increasing the `MaxRaiseSize` parameter, the adapter is forced to "block" a larger number of documents to be processed. This results in a higher recovery latency. When the poller thread comes up post recovery, it needs to claim all the rows that were "pending" from a previous run. The `MaxRaiseSize` parameter may also impact performance negatively if its value is too high (higher than the injection rate × polling frequency), because it may unbalance the system by pushing all the files to a single BPEL/Mediator instance.

- `MaxRaiseSize` and `PollingInterval`

  The `MaxRaiseSize` and the `PollingInterval` values are, in general, very business case dependent. For example, some use cases may require an immediate processing of many incoming files (thus requiring low poll intervals and high `MaxRaiseSize`), while others may require large payloads to be processed without immediate copy being a concern, and thus benefiting from a high `PollingInterval`). As explained in the "Best Practices for BPEL and File Adapter" section, once the business cases adjustments are done, the values can be modified to the "business allowed window" to improve availability.

Use Case for Verifying the Availability of the File Adapter

See Appendix C for a description of the composite, the failures injected in the system, the type of client, and the tools and scripts used to verify the high availability behavior of the File Adapter. You can use these details to verify appropriate failover and load balancing in a File Adapter Composite.

Summary of Behaviors and Detailed Results

Table 5 summarizes the results of the failure injection tests run for the file adapter.

TABLE 5. RESULTS OF THE DIFFERENT FAILURES INJECTED IN AN ORACLE FUSION MIDDLEWARE SOA CLUSTER USING FILE ADAPTER

TYPE OF FAILURE	EXPECTED BEHAVIOR/IMPACT	ASSOCIATED EXCEPTIONS, MESSAGES IN LOGS	POSSIBLE RECOVERY
WLS Server Failure	• No message loss • Message duplication can take place if file writes by outbound adapter completes before returning to BPEL and failure occurs (BPEL will resubmit from receive operation) • Performance impact of failures on failover node depends on MaxRaiseSize • Non high resource-consumption penalty for restarts • Re-loadbalancing for restarted WLS Managed server	**Node Manager Output:** <INFO> <Successfully removed X.X.X.X from eth0:Y.> … … <Server failed so attempting to restart (restart count = 1)> … <Sleeping for 30 seconds before attempting to restart server> .. <WARNING> <Successfully brought X.X:X:X with netmask 255.255.255.0 online on eth0:Z>	N/A
WLS Server migration (either node failure or successive failure in restarting the server in the local node)	• No message loss • Message duplication can take place if file writes in outbound adapter complete before returning to BPEL and failure occurs (BPEL will resubmit from receive operation) • Performance impact of failures on failover node depends on MaxRaiseSize • Non high resource-consumption penalty for restarts • Re-loadbalancing for restarted WLS Managed server	**Local-Node Manager output for failed restarts:** <INFO> <Successfully removed X.X.X.X from eth0:Y.> … <Server failed so attempting to restart (restart count = 1)> … <Sleeping for 30 seconds before attempting to restart server> .. INFO: Server failed during startup so will not be restarted • **Local-Node Manager output for node failure:** N/A • **Failover-node Node Manager output:** WARNING> <Successfully brought X.X:X:X with netmask 255.255.255.0 online on eth0:Z>	N/A
SOAHOST (node) failure	• No message loss • Message duplication can take place if file writes in outbound adapter complete before returning to BPEL and failure occurs (BPEL will resubmit from receive operation) • Performance impact of failures on failover node depends on MaxRaiseSize • Non high resource-consumption penalty for restarts • Re-loadbalancing for restarted WLS Managed server	**Failover-node Node Manager output:** WARNING> <Successfully brought X.X:X:X with netmask 255.255.255.0 online on eth0:Z>	N/A

Database Failure (hard)	Instance	• No message loss. Failures in the database can prevent BPEL instance creation during database connection failover which in turn may cause messages being 'stuck' temporarily. These messages can be recovered using the EM console (BPEL recovery) • Message duplication can take place if file writes in outbound adapter completes before returning to BPEL and failure occurs (BPEL will resubmit from receive operation)	**SERVER.out** <Sep 15, 2010 11:30:12 AM PDT> <Warning> <JDBC> <BEA-001129> <Received exception while creating connection for pool "SOALocalTxDataSource-rac0": Socket read timed out> [TopLink Warning]: 2010.09.15 11:30:13.649--ClientSession(322222607)--Exception [TOPLINK-4002] (Oracle TopLink - 11g Release 1 (11.1.1.3.0) (Build 100323)): oracle.toplink.exceptions.DatabaseException Internal Exception: java.sql.SQLException: weblogic.common.ResourceException: **Followed by --SERVER.out file:** java.sql.SQLRecoverableException: No more data to read from socket at oracle.jdbc.driver.SQLStateMapping.newSQLException(SQLStateMapping.java:105) at oracle.jdbc.driver.DatabaseError.newSQLException(DatabaseError.java:135) **Followed by** <Test "SELECT 1 FROM DUAL" set up for pool "EDNDataSource-rac0" failed with exception: "oracle.jdbc.xa.OracleXAException".> <Sep 15, 2010 11:30:13 AM PDT> <Error> <oracle.soa.adapter> <BEA-000000> <JCABinding=> FlatStructureIn FlatStructureInAdapter Service FlatStructureIn was unable to perform delivery of inbound message to the composite default/FlatStructure!30.0*soa_0ac85e65-c93e-43e0-9112-95e10238b398 due to: Exception [TOPLINK-4002] (Oracle TopLink - 11g Release 1 (11.1.1.3.0) (Build 100323)): oracle.toplink.exceptions.DatabaseException Internal Exception: java.sql.SQLException: weblogic.common.ResourceException: No good connections available.	Use Oracle Enterprise Manager Console for recovery
Database Node Failure (hard)		• No message loss. Failures in the database can prevent BPEL instance creation during database connection failover which in turn may cause messages being 'stuck' temporarily. These messages can be recovered using the EM console (BPEL recovery) • Message duplication can take place if file writes in outbound adapter completes before returning to BPEL and failure occurs (BPEL will resubmit from receive operation)	**SERVER.out file:** <Warning> <JDBC> <BEA-001129> <Received exception while creating connection for pool "SOADataSource-rac1": The Network Adapter could not establish the connection>	N/A

File System Failures (Files and Transaction Logs)	• No duplications • Message loss can occur if the transaction logs for SOA server are lost while the file adapter is processing files. i.e. The invocation to outbound file adapter can be lost if failure happens after a TX has been rolled back for file processing	**NO messages in out file** **Server .log file:** ####<DATE> <Warning> <JTA> <strasha02> <WLS_SOA2> <[STUCK] ExecuteThread: '4' for queue: 'weblogic.kernel.Default (self-tuning)'> <<WLS Kernel>> <> <> <1288703109088> <BEA-110484> <The JTA health state has changed from HEALTH_OK to HEALTH_WARN with reason codes: Resource WLStore_soaedg_domain_SOAJMSFileStore_auto_2 declared **unhealthy**.> For the regular persistency of files, hanged operations (like ungraceful drop of the NFS connection) the WLS servers blocked without reporting any exceptions until write was reported successful or failed. Persistent store marked as unhealthy in server's log.	Restore Transaction logs to latest point in time
File System Failures (Files and transaction logs)	• No duplications • Message loss can occur if the transaction logs for SOA server are lost while the file adapter is processing files. i.e. The invocation to outbound file adapter can be lost if failure happens after a TX has been rolled back for file processing	**NO messages in out file** **Server .log file:** ####<DATE> <Warning> <JTA> <strasha02> <WLS_SOA2> <[STUCK] ExecuteThread: '4' for queue: 'weblogic.kernel.Default (self-tuning)'> <<WLS Kernel>> <> <> <1288703109088> <BEA-110484> <The JTA health state has changed from HEALTH_OK to HEALTH_WARN with reason codes: Resource WLStore_soaedg_domain_SOAJMSFileStore_auto_2 declared **unhealthy**.> For the regular persistency of files, hanged operations (like ungraceful drop of the NFS connection) the WLS servers blocked without reporting any exceptions until write was reported successful or failed. Persistent store marked as unhealthy in server's log	Restore Transaction logs to latest point in time

Middle-Tier Failure Results

Table 6 summarizes the results for different tests run with failures being injected in the middle tiers.

Note: In reality, each row in the table reflects the average of three tests executed with the same configuration. The tests were repeated to obtain more meaningful metrics and eliminate point conditions during the tests.

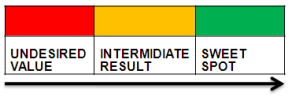
TABLE 6. FAILOVER LATENCY, PROCESSING RESULTS, AND TRANSLATION FAULTS FOR MIDDLE-TIER FAILURES IN A FILE ADAPTER CLUSTER

TEST	PAYLOAD	POLLING FREQUENCY	MAXRAISESIZE	MIN AGE	OVER PROCESSING	UNDER PROCESSING	TRANSLATE.FAULTS	FAILOVER LATENCY
T14	176B	3	Def (1000)	0	0	0	0	243
T15	176B	1	Def (1000)	0	2	0	0	246

TABLE 6. FAILOVER LATENCY, PROCESSING RESULTS, AND TRANSLATION FAULTS FOR MIDDLE-TIER FAILURES IN A FILE ADAPTER CLUSTER

TEST	PAYLOAD	POLLING FREQUENCY	MAXRAISESIZE	MIN AGE	OVER PROCESSING	UNDER PROCESSING	TRANSLATE.FAULTS	FAILOVER LATENCY
T16	176B	60	Def (1000)	0	3	0	0	240
T17	176B	60	15	0	0	0	0	234
T19	176B	5	15	0	0	0	1	234
T20	176B	1	15	0	0	0	0	239
T22	37KB	30	15	15	0	0	2	239
T24	37KB	5	15	15	0	0	39	247
T29	37KB	5	Def (1000)	10	0	0	2	248
T36	37KB	30	100	10	0	0		251

LEGEND:



The results in Table 6 demonstrate that higher polling frequencies generate fewer translation errors. A translation error is typically generated when the `MinAge` value is not correctly adjusted for large payloads. In such scenarios, a file takes a long time to get copied, so the adapter tries to process it before it is complete. The `MinAge` property specifies a delay between the file detection and the file processing. Translation errors of this type result in the file not being processed and being rejected, and in an effective under-processing scenario—where the number of files resulting in the output directory is less than the number of files processed and archived—requiring recovery (reposting the file). These errors are not related to failover events per se, but the errors are more likely to happen in a SOA cluster where more threads compete for the same files and the payloads are large. The points in time at which the translation errors took place did not show any direct relationship with a failover being performed.

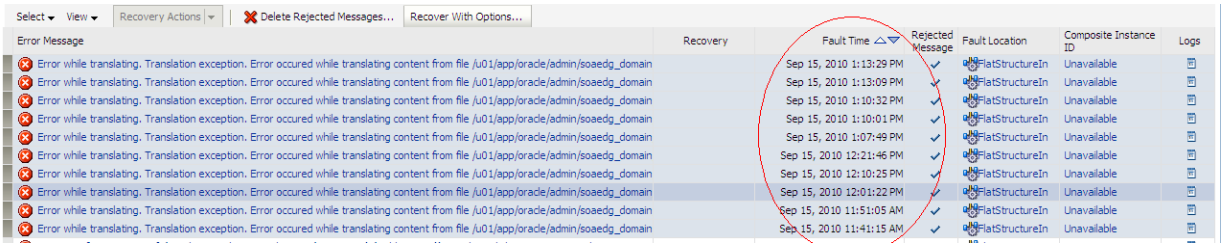


Image 6: Point in time at which translation errors took place

Once the appropriate `MinAge` value was set for the inbound adapter, the number of translation errors was reduced. However, even after adjusting `MinAge`, errors of this type were still taking place, even for small payloads. The issue was caused by the type of locks acquired by the polling threads. After processing, the files get deleted but before the deletion completes, other poller threads may list them (because the lock table has already

been cleaned up). Poller threads try to acquire an exclusive FileLock on the file and end up creating a blank file. The error is more likely to happen with larger payloads and in lower performance storage where the latency for delete operations is higher. Setting higher values for both the polling frequencies and the `MinAge` parameter will improve this behavior. For more information, see the "Best Practices for BPEL and Oracle Database Adapter" section.

The translation exceptions generate messages such as the following:

```
ORABPEL-11117

minOccurs not satisfied.
minOccurs="1" not satisfied for the node "<element name="Root-Element">". Loop terminated with
cardinality "0". Insufficient or invalid data.
Please correct the NXSD schema.
```

Note: These faults are *harmless* in the sense that they do not cause duplications and the original message is correctly processed). They cause the above exceptions and the copy of a zero byte file to the default archive directory. However, rejection handlers should account for this properly: if a rejection handler reposts the file, the system may generate more and more translation errors in a loop. A fix for the problem is available in Oracle Fusion Middleware 11*g* Release 1 Patch Set 1 and Patch Set 2, which you can obtain through the regular Oracle Support channels.

For translation errors caused because file copies were not completed, the tests proved that the exceptions are not directly related to the load in the system. Rather, they are related to the performance of the storage used for input files and happen more commonly with large payloads. The OS metrics for the period of time during which the translation exceptions took place demonstrate that there is no direct relation between the load on the system and the error. That is, for average loads, a translation issue is not necessarily a sign of a system being overloaded. This is a random issue that can only be prevented by increasing the MinAge value for the files.

In Tables 7 and 8, the rows highlighted in yellow show the exact point in time when a translation error occurred.

TABLE 7. VMSTAT METRICS DURING THE TIME WINDOW AT WHICH A TRANSLATION ERROR TOOK PLACE (NODE1)

DATE	R	B	SWAP	FREE	BUFF	CACHE	SI	SO	BI	BO	IN	CS	US	SYS	ID	WA
2010-09-13_11:51:46	1	0	224	1262548	118340	1155140	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:51:49	0	0	224	1262612	118340	1155140	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:51:52	0	0	224	1262708	118340	1155140	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:51:55	0	0	224	1262644	118340	1155140	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:51:58	0	0	224	1262452	118340	1155140	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:01	0	0	224	1262324	118340	1155140	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:04	0	0	224	1262324	118340	1155140	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:07	0	0	224	1262196	118340	1155140	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:10	0	0	224	1261812	118340	1155660	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:13	1	0	224	1261876	118340	1155660	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:16	0	0	224	1261876	118340	1155660	0	0	2	10	2	0	5	0	95	0

TABLE 7. VMSTAT METRICS DURING THE TIME WINDOW AT WHICH A TRANSLATION ERROR TOOK PLACE (NODE1)

DATE	R	B	SWAP	FREE	BUFF	CACHE	SI	SO	BI	BO	IN	CS	US	SYS	ID	WA
2010-09-13_11:52:19	0	0	224	1262068	118340	1155660	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:22	0	0	224	1262068	118340	1155660	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:25	1	0	224	1260436	118340	1155660	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:28	0	0	224	1262100	118340	1155660	0	0	2	10	2	0	5	0	95	0
2010-09-13_11:52:31	0	0	224	1262100	118340	1155660	0	0	2	10	2	0	5	0	95	0

TABLE 8. VMSTAT METRICS DURING THE TIME WINDOW AT WHICH A TRANSLATION ERROR TOOK PLACE (NODE2)

DATE	R	B	SWAP	FREE	BUFF	CACHE	SI	SO	BI	IN	CS	US	SYS	ID	WA
2010-09-13_11:34:04	1	0	1424	1205960	158720	2396572	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:07	1	0	1424	1205960	158720	2396572	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:10	1	0	1424	1205896	158720	2396572	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:13	1	0	1424	1205896	158720	2396572	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:16	1	0	1424	1205896	158720	2396572	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:19	1	0	1424	1205960	158720	2396572	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:22	1	0	1424	1206032	158720	2396572	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:25	1	0	1424	1205448	158720	2396572	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:28	1	0	1424	1205504	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:31	1	0	1424	1205504	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:34	1	0	1424	1205392	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:37	2	0	1424	1205584	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:40	1	0	1424	1205584	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:43	2	0	1424	1205712	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:46	1	0	1424	1205712	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:49	1	0	1424	1205520	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:52	1	0	1424	1205576	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:55	1	0	1424	1205632	158720	2396832	0	0	0	6	0	0	2	0	98
2010-09-13_11:34:58	1	0	1424	1205288	158720	2397092	0	0	0	6	0	0	2	0	98
2010-09-13_11:35:01	1	0	1424	1205376	158720	2397092	0	0	0	6	0	0	2	0	98
2010-09-13_11:35:04	1	0	1424	1205184	158720	2397092	0	0	0	6	0	0	2	0	98

TABLE 8. VMSTAT METRICS DURING THE TIME WINDOW AT WHICH A TRANSLATION ERROR TOOK PLACE (NODE2)

DATE	R	B	SWAP	FREE	BUFF	CACHE	SI	SO	BI	IN	CS	US	SYS	ID	WA
2010-09-13_11:35:07	1	0	1424	1205312	158720	2397092	0	0	0	6	0	0	2	0	98
2010-09-13_11:35:10	1	0	1424	1205312	158720	2397092	0	0	0	6	0	0	2	0	98
2010-09-13_11:35:13	1	0	1424	1205312	158720	2397092	0	0	0	6	0	0	2	0	98

The tests verified the following information:

- During failover, the database mutex rows remained in higher counts (rows are not updated until the failed server is restarted). You can issue the following SQL query to monitor the database mutex rows to determine the correct flow of processing and to identify server failures or server blocks:

  ```
  SQL> select count(*),file_processed from fileadapter_in group by file_processed
  order by file_processed;
  ```

  A constant count value in the mutex rows with `file_processed field=2` clearly indicates a server restart (mutex row is not being updated).

- There is a dependency between the `MaxRaiseSize` and the failover latency. Low `MaxRaiseSize` values involved higher recovery periods (restart of the failed server). The justification for this is that lower `MaxRaiseSize` values force a higher number of transaction recoveries because the parameter controls the number of documents presented to the service engine (BPEL in this case). I.e. the more documents per transaction the less number of transaction-recovery operations that need to take place when the system is restarting after a failure (when recovery needs to happen).

- Failures that happen immediately after the file is written by the outbound adapter, but before the BPEL transaction completes, can generate duplicates. The following scenario depicts a possible duplicate/overprocessing generation:

  1. The inbound adapter has published and deleted the file in the input directory.
  2. An entry gets scheduled at the BPEL end.
     The `File::write` occurs on outbound adapter.
  3. Immediately before the outbound adapter returns the call to BPEL (this is a blocking operation), the BPEL instance fails/server crashes.
  4. Because recovery starts from the receive, the same XML document may be posted again by BPEL to the outbound adapter, hence generating a duplicate.

- Larger payloads tend to generate less overprocessing than smaller ones. The justification is that write I/Os of smaller payloads complete quicker, and the probability of facing a BPEL instance failure immediately after a complete file write I/O but before committing from the receive operation is lower than for smaller write I/Os.

Database Failure Results

Table 9 summarizes the tests in which failures were injected in the database tier.

Note: In reality, each row in the table reflects the average of three tests executed with the same configuration. The tests were repeated to obtain more meaningful metrics and eliminate point conditions during the tests.

TABLE 9. TRANSLATION FAULTS AND PROCESSING RESULTS FOR DB FAILURES IN A FILE ADAPTER CLUSTER

	PAYLOAD	POLLING FREQUENCY	OVER PROCESSING	UNDER PROCESSING	MAXRAISESIZE	MINAGE	TRANSLATION FAULTS
T30	37Kb	5	3	N/A	Def (1000)	10	15
T38	37KB	5	0	0	15	10	35
T37	37KB	20	0	0	100	10	0
T39	176B	5	0	1 (recovery needed)	Def (1000)	0	0
T40	176B	20	2	NA	100	0	0
T42	176B	60	0	NA	100	0	0

The tests verified that, despite the fact that the file adapter uses the database scarcely, mostly for locking and mutex, higher polling frequencies resulted in lower database node's stress. Less frequent access to the database (which happens for lower poling frequencies and higher `MaxRaiseSize`) also minimize the effect of database failures in the system. The file adapter will retry the raise of the documents to the service engines, and in most cases the failed operation succeeds, provided that the data source for the SOA infrastructure is properly configured (see database configuration example in Appendix B). The number of retries, by default, is indefinite. This can be set as a service binding property. Thus, although the database failures resulted in failed BPEL instances, the composite completed successively, as shown in Image 7:

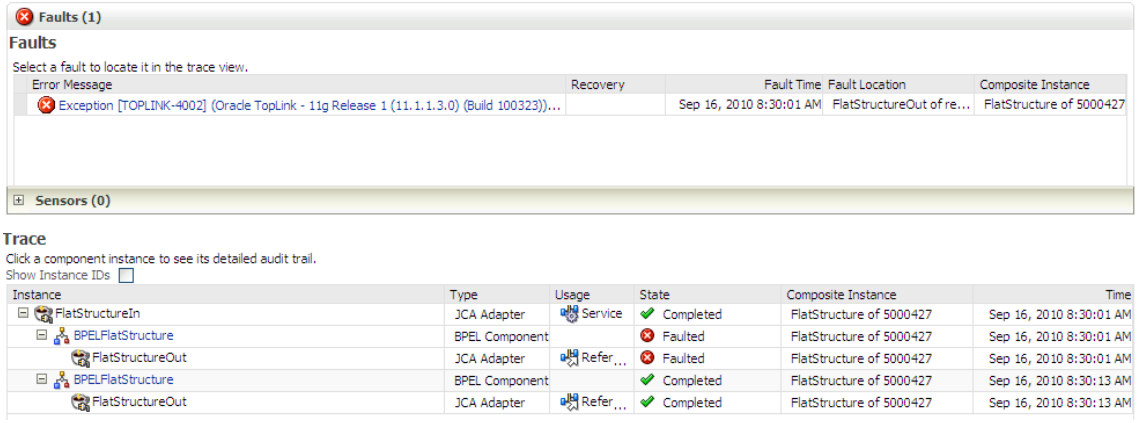


IMAGE 7: RETRY REPORTED IN ORACLE ENTERPRISE MANAGER FUSION MIDDLEWARE CONTROL

None of the tests resulted in messages being lost, although recovery from the Oracle Enterprise Manager FMW Control was required for a couple instances). However, a few tests showed overprocessing of files resulting from the effect of database connection failover on the BPEL instance after the file is written. There is no possible rollback of the write operation, as explained in the Middle-Tier Failures Results section.

Best Practices for BPEL and File Adapter

The following sections provide recommendations for configuring a SOA system using the file adapter and a service engine in a distributed highly available system. The recommendations are inferred from the results of the tests executed.

Configure a Control Directory on Shared Storage

Configure a control directory on shared storage and define an appropriate backup strategy for it. The control directory defines, among other things, the default location for the error-archival directory (where the adapter copies files that could not be processed due to errors, like "translation errors"). The directory will hold the files that are potentially pending to be processed. Losing this directory may have additional implications when de-batching and when not using the database as mutes for writes. It is highly recommended that you place this location in a highly available file system and ensure it is backed up.

Adjust the MinimumAge Parameter to the Expected Latency for File Injections

Stress the system with loads that are similar to what is expected in a production environment as a way to verify the `MinimumAge` that may be required for the type of payloads (size) that will be used in the system. Notice that larger `MinimumAge` values may result in poorer performance as reads will be delayed. You must make a compromise between consistency and performance. The effect of low `MinimumAge` values (or the default `MinimumAge` of zero) is increased by the use of higher frequencies in polling as shown in the data above (for example, very frequent polling with low `MinimumAge` values may result in more and more translation errors).

Verify that Files are Being Consumed By All Servers in a Cluster Before Testing Failover

Use the Oracle Enterprise Manager FMW Control as described in Appendix C to make sure that endpoints are being evenly activated in all the instances in the cluster. Queries to the database such as the following should show that different threads (identified by different end point GUIDs) are picking up files:

```
SQL>select count(*),file_endpoint_guid from fileadapter_in group by file_endpoint_guid;

"COUNT(*)"     "FILE_ENDPOINT_GUID"
1       "31e583d5-d7d5-41fb-a25b-ca3b927b52e8"
56      "b6d7fad9-9214-4eac-a511-fb8a1ccc7391"
```

If only one unique GUID is shown at all points in time, then that means that one of the endpoints is not working properly.

During Troubleshooting Phase, Tag Inbound Files with Dates and Sequence IDs

Using time stamps and IDs in the file names helps a lot to correlate inbound and outbound operations. It also facilitates the identification of failover latencies with groups of files.

Increase Transaction Timeouts When the Database Is Used as a Coordinator

The fact that each file's processing requires additional accesses to the database (for updating the database mutex/lock), may result in longer delays in the completion of instances. It is recommended to increase the global transaction time out for the system.

Best Practices for JMS, Database, and File Adapters Composites

The following generic best practices apply indistinctly to the different cases analyzed in the proceeding sections.

Watch Space and Autoextend in SOA Infrastructure Database

Although not all adapters use the database intensively, the service engines generate a considerable amount of data in the `CUBE_INSTANCE` and `MEDIATOR_INSTANCE` schemas. Lack of space in the database may prevent the adapter (JMS, File, or Database) composites from functioning. Watch for generic errors such as "`oracle.fabric.common.FabricInvocationException`" in the Oracle Fusion Middleware Control console. Also, search in the SOA server's logs for errors such as the following:

```
Error Code: 1691
[TopLink Warning]: 2010.09.16 10:06:24.818--ClientSession(301616889)--Exception [TOPLINK-4002]
(Oracle TopLink - 11g Release 1 (11.1.1.3.0) (Build 100323)):
oracle.toplink.exceptions.DatabaseException
Internal Exception: java.sql.BatchUpdateException: ORA-01691: unable to extend lob segment
MAAJ1_SOAINFRA.SYS_LOB0000108469C00017$$ by 128 in tablespace MAAJ1_SOAINFRA
```

These messages typically indicate space issues in the database that will likely require adding more data files or more space to the existing data files. The SOA database administrator must decide whether to add more data files or extend existing ones when adding space. Additionally, old composite instances can be purged to reduce space. It is not recommended to use the Oracle Fusion Middleware Control console for this type of operation, because in most case the operations will cause a transaction time out. There are specific packages provided with the Repository Creation Utility to purge instances.

For example:

```
DECLARE
 FILTER INSTANCE_FILTER := INSTANCE_FILTER();

 MAX_INSTANCES NUMBER;
 DELETED_INSTANCES NUMBER;
 PURGE_PARTITIONED_DATA BOOLEAN := TRUE;
BEGIN

 FILTER.COMPOSITE_PARTITION_NAME:='default';
 FILTER.COMPOSITE_NAME := 'FlatStructure';
 FILTER.COMPOSITE_REVISION := '10.0';
 FILTER.STATE := fabric. STATE_UNKNOWN;
 FILTER.MIN_CREATED_DATE := to_timestamp('2010-09-07','YYYY-MM-DD');
 FILTER.MAX_CREATED_DATE := to_timestamp('2010-09-08','YYYY-MM-DD');
 MAX_INSTANCES := 1000;

 DELETED_INSTANCES := FABRIC.DELETE_COMPOSITE_INSTANCES(
   FILTER => FILTER,
   MAX_INSTANCES => MAX_INSTANCES,
   PURGE_PARTITIONED_DATA => PURGE_PARTITIONED_DATA
 );

END;
```

This example deletes the first 1000 instances of the FlatStructure composite (version 10) created between '2010-09-07' and '2010-09-08' that are in "UNKNOWN" state. See chapter 8 in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite* and the *Oracle Business Process Management Suite* for more details on the possible operations included in the SQL packages provided. Always use the scripts provided for a correct purge. A deletion of rows in just the `COMPOSITE_DN` table could leave dangling references in other tables used by the Oracle Fusion Middleware SOA Infrastructure.

Perform Capacity Planning Based on Server Migration Scenario

As described in the different cases analyzed and especially for the JMS Adapter Use Case, the highest resource contention in the failover tests was detected when a server is migrated (while it is being restarted) to a node where the other available server is processing messages and contains the Admin Server. This should be the worst case scenario to be used when you are determining the appropriate sizing of the system.

Monitor Service Component Instances, not Composite Instance

Status for composites is not tracked by default for performance reasons. A value such as "?" in Oracle Fusion Middleware Control of Enterprise Manager for a composite's state means that state is not being tracked at composite level. You can still see the state of any of the components in the composite by going to the composite flow using the composite instance ID link, without affecting their system's performance. At the same time, the number of composite instances does not necessarily match the number of service engine instances. Use the service engine's Instance IDs as the reference for drilling down to latency and failure issues.

Detecting Issues in Failover Scenarios with the CUBE_INSTANCE and MEDIATOR_INSTANCE tables

See the examples in each of the scenarios for some SQL queries that may help to identify errors, rejections, and faults. In general, you can check for recoverable instances, activities and callbacks by running the following SQL statements:

- To report the recoverable activities:

```
SQL> select * from work_item where state = 1 and execution_type != 1;
```

- To report the recoverable invoke messages:

```
SQL> select * from dlv_message where dlv_type = 1 and state = 0;
```

- To report the recoverable callbacks:

```
SQL> select * from dlv_message where dlv_type = 2 and (state = 0 or state = 1);
```

- To check for rejected messages for a specific composite:

```
SQL> select count(*) from rejected_message where composite_dn like '%name%version%';
```

- To check data in the instance tracking table:

```
SQL> select ID, STATE from COMPOSITE_INSTANCE where composite_dn like '%name%version%';
```

Notice that adapters keep track of the message's status through the `COMPOSITE_INSTANCE` table. As soon as the adapter gets a message, it inserts an entry into the `COMPOSITE_INSTANCE` table with `STATE` set to 0. After the message is processed, the `STATE` is updated to 1. Any errors set `STATE >= 2`.

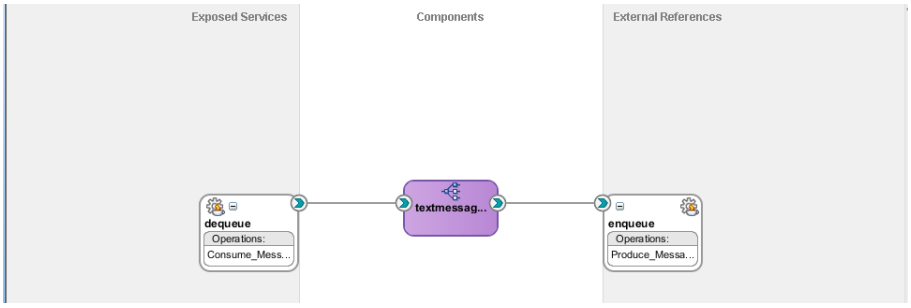
Appendix A: Use Case for Verifying JMS Adapter's HA Behavior

Description of the Composite Used

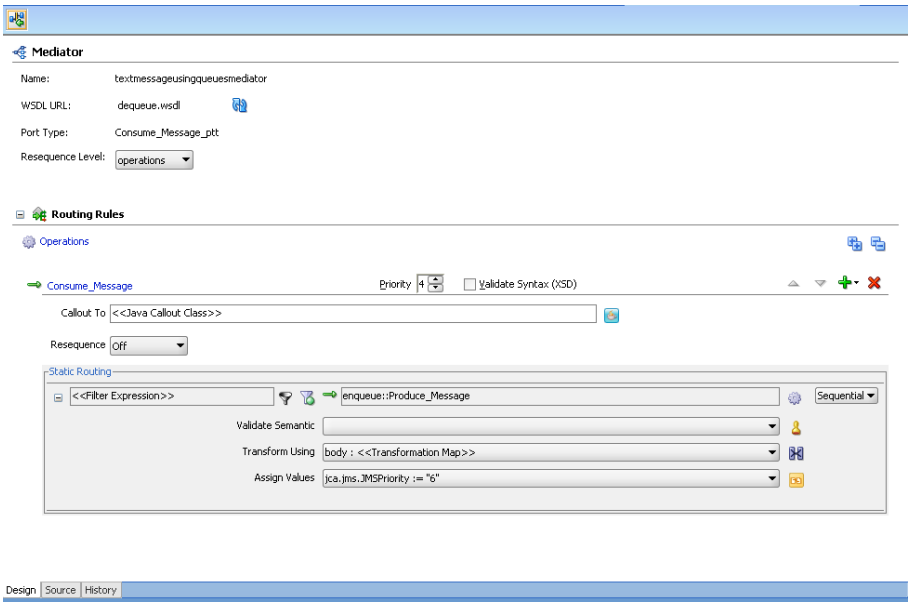
To verify the expected behaviors and identify best practices for Oracle's JMS Adapter, the "textmessageusingqueues" example was used. This example is available on the Oracle Technical Network (OTN) at http://download.oracle.com/technology/sample_code/products/soa/soasuite/samples.zip.

The texmessageusingqueues example shows a simple scenario where the JMS Adapter is used to poll from an incoming queue. Messages activate a Mediator composite and are routed using the JMS Adapter to an out queue.

The following graphic shows the JDeveloper composite diagram:



The following graphic shows the Mediator properties:



The composite was deployed to the SOA_Cluster directly from JDeveloper. All default configuration parameters were used for the JMS Adapter connection factories and the WLS JMS infrastructure. For initial verification, the

composite was tested with WLS_SOA1 and WLS_SOA2 that were running alternatively; one server was down while the other was being used to create instances of the composite. Both the DemoIn and the DemoOut queues were Uniform Distributed Destinations added to the exiting SOA JMS module.

Note: For performance and availability isolations purposes that may apply in a real production environment, sharing custom destinations in the same JMS module as the SOA infrastructure may not be ideal. In a production environment, it may be desired to isolate the custom destinations in a separate module and even a separate cluster. However, there are limitations in the WLS JMS infrastructure that may advise against this configuration. When using distributed queues, ensuring that all members are serviced by consumers under all conditions becomes a problem: When a distributed queue member is down (such as during a server migration), typical applications that are using the member (like the JMS Adapter) will try to reattach and will be load balanced to another member. However, when the member later comes back up, application consumers that are already attached will not be load balanced back to the original member. When the destinations are deployed in a separate cluster from the JMS Adapter, this may result in servers being totally inactive after a restart (no JMS Adapter consumes from them). If at least one JMS Adapter is collocated with the destination, failovers and restarts at the same time as the destination guarantee that at least one adapter will consume from the restarted destination. See the "Best Practices for JMS Adapter" section for more details.

Note: It is not the intention of this assessment to cover the behavior of JMS distributed destinations per se, but only to verify how the JMS Adapter reacts to failures in Oracle Fusion Middleware 11*g* Release 1 Patch Set 2.

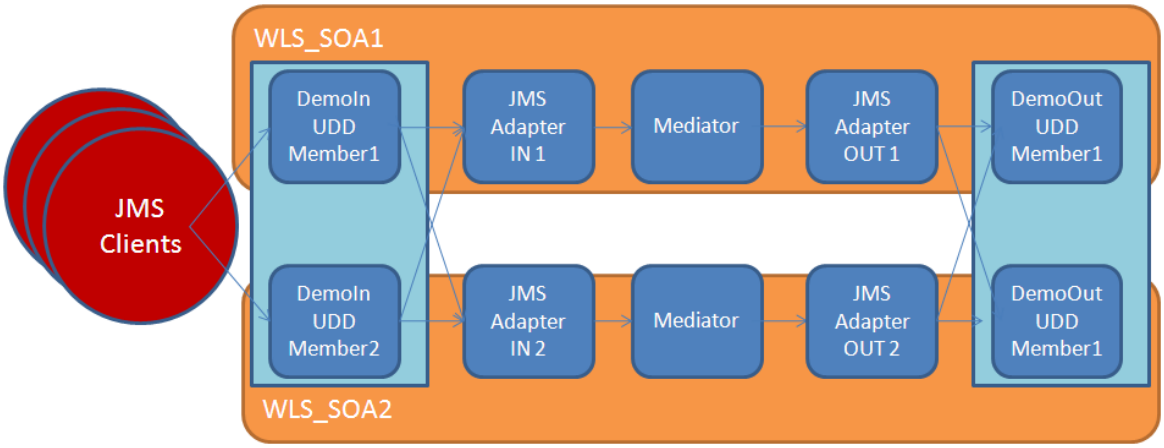
Description of the Client Stressing the System

A Java standalone program was used to constantly send messages to the DemoIn queue. The client allowed configuring the number of threads used to send messages to the queue, the number of messages to be sent, and the wait period between each two send operations.

The PAYLOAD used contained an ID for each message:

```
PAYLOAD=
<?xml version=\"1.0\" ?>
<ExpenseRecord xmlns=\"http://xmlns.oracle.com/pcbpel/samples/expense\">
      <EmpId>"+i+"
      </EmpId>
      <Item>Item2</Item>
      <Count>3</Count>
      <Cost>40</Cost>
</ExpenseRecord>";
```

Transactions were used on the client to guarantee once-and-only-once delivery (each JNDI initial context, connection factory, and JMS session were created inside the same transaction). Note that to guarantee the appropriate HA verifications on the server side, it was first verified that failure in the servers hosting the destinations did not imply any message loss or duplication. To guarantee this, the client was tested first and failures were injected on the servers (without the composite being active).



To obtain significant ratios and metrics form the tests, the client was configured to inject a high number (in the order of tens of thousands) of messages with wait periods of 100 msecs between each insertion. Each test period lasted from 6 to 9 hours.

Regarding failures in the file system used for JMS and transaction logs, all SOA JMS servers use paging and direct-write as the write I/O policy for the corresponding persistent stores by default. These were also the settings used for the queues used by the composite. When paging is enabled, WLS JMS uses memory to allocate messages and flushes batches of messages to disk. When the JMS server writes the message bodies to disk, it clears them from memory. The amount of memory (in bytes) that a JMS server can use to store message bodies before it writes them to disk is controlled with the `Message Buffer Size` property. This property is defaulted to -1. A value of -1 specifies that the server will automatically determine a size based on the maximum heap size of the JVM. This default will be set to either one-third the maximum heap size, or 512 megabytes, whichever is larger. Because the payload used in the verifications did not exceed a few Kbs and the tests never reached the order of 100,000 messages, its limits were never reached (for example, paging never occurred. Hence, all write I/Os took place to the persistent store directly). The intention was to use the default configuration in a node with the recommended hardware resources for a SOA System.

Description of the Failures Injected

Failures were injected in the system in the following ways:

- WLS Server Failures: Cron JOBS were created to ungracefully stop the servers present in the node every 30 minutes. To guarantee that least one server would be available for consuming messages, the jobs were alternated so that server migration would be triggered alternatively for WLS_SOA1 and WLS_SOA2:
  - At HH:00 minutes during each hour WLS_SOA1 is crashed in SOAHOST1. Because the default behavior is that NM will try to start the server again locally (restart count=1), a one minute period of time is granted for this restart and the server is stopped again at HH:02. Server migration is triggered then to SOAHOST2.
  - At HH:15 minutes during each hour, WLS_SOA2 is crashed in SOAHOST2. Because the default behavior is that NM will try to start the server again locally (restart count=1), a one-minute period of time is granted for this restart and the server is stopped again at HH:17. Server migration is triggered then to SOAHOST1.

- At HH:30 minutes during each hour, WLS_SOA1 is crashed in SOAHOST2 . Because the default behavior is that NM will try to start the server again locally (restart count=1,) a one-minute period of time is granted for this restart and the server is stopped again at HH:32. Server migration is triggered then to SOAHOST1.
- At HH:45 minutes during each hour, WLS_SOA2 is crashed in SOAHOST1 . Because the default behavior is that NM will try to start the server again locally (restart count=1), a one-minute period of time is granted for this restart and the server is stopped again at HH:476. Server migration is triggered then to SOAHOST2.

With this failure injection cycles, the system is reverted to its original state (WLS_SOA1 running in SOAHOST1 and WLS_SOA2 running in SOAHOST2) in every hour cycle

- **File system failures:** In both servers, removal of the JMS and TX logs was performed as well as a force unmount of the shared drive hosting these logs. Additional force drop of communications to the NAS device was performed by using iptables.
- **Database failures:** The composite used in the test does not use database resources other than for the SOA infrastructure to store composite data (such as instance ID, start date, modification date, and so on), and message delivery status. Cron jobs were created on the database nodes to alternatively crash each one of the two database instances while injecting JMS messages in the system:
  - At HH:00 minutes each hour, soaedg1 is crashed in DBHOST1.
  - At HH:30 minutes each hour, soaedg2 is crashed in DBHOST2.

Tools that Verify the Effect of Failures on the System

The following summary describes the different tools that were used to verify the high availability behavior of the adapter:

1. Statistics from Oracle WLS Administration Console:

   The WLS Admin Console allows verifying the messages present in the queues. To access the messages, click the JMS Module where the destination is deployed and click the destination. Once in the "Settings" screen for the destination, click the Monitoring tab. The Monitoring screen allows you to select the server from which to visualize the messages:

	ID:<785406.1278503101362.0>	Wed Jul 07 04:45:01 PDT 2010	visible	Persistent	176
☐	ID:<785406.1278503100750.0>	Wed Jul 07 04:45:00 PDT 2010	visible	Persistent	176
☐	ID:<785406.1278503100311.0>	Wed Jul 07 04:45:00 PDT 2010	visible	Persistent	176
☐	ID:<785406.1278503100080.0>	Wed Jul 07 04:45:00 PDT 2010	visible	Persistent	176

   Counts of messages in DemoOutQueue: The Administration Console's JMS Monitoring screen shows the number of messages in each server separately. To determine the total number of messages in the Distributed Destination, add the results of all the servers. This can be also obtained with a wlst script (see Appendix A).

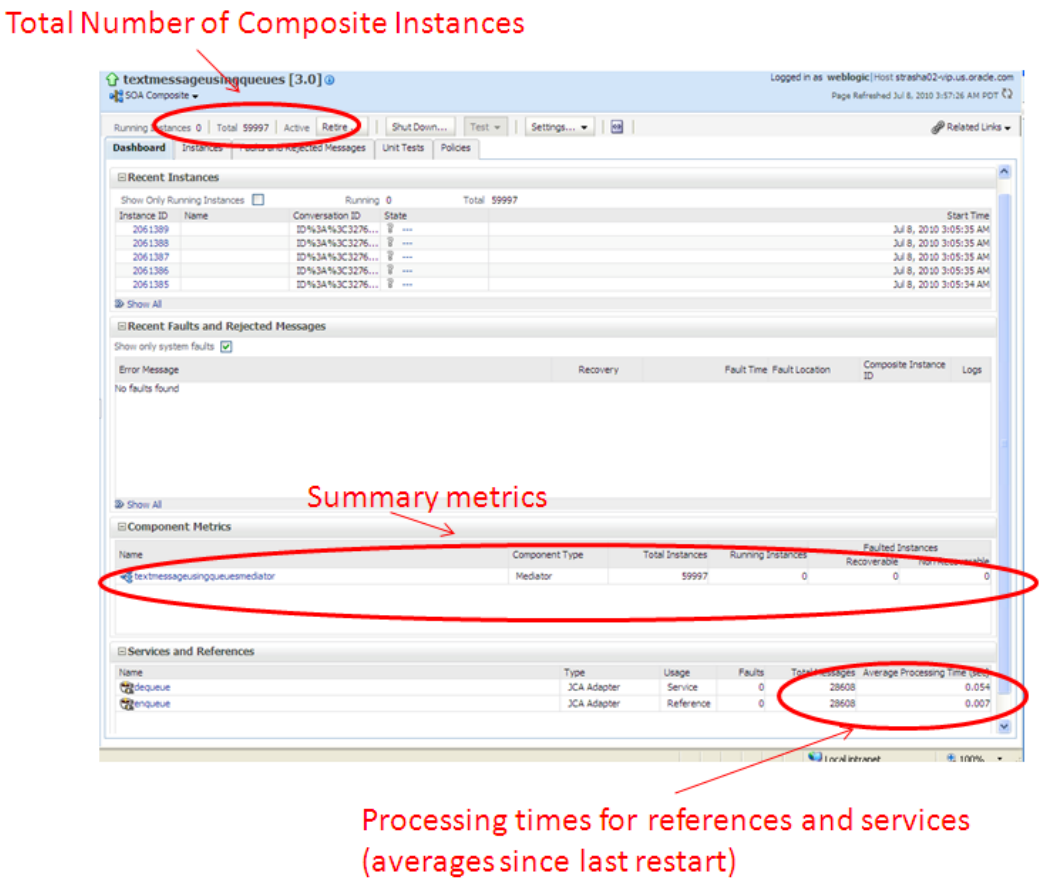
Dumps of Messages in DemoOutQueue: By default, the table where the messages are listed does not show some interesting values. Click **customize table** and add the "JMS Redelivered" column. This messages screen allows you to export all messages to a file. Once all messages are in a file, it is straight forward to determine all the messages that were redelivered:

```
$ cat jmsmessages.xml  | grep JMSRedelivered -c
$ cat jmsmessages.xml  | grep JMSRedelivered | grep false -c
```

This allows you to determine how many messages were redelivered (redelivered because no acknowledgement was received). When a message is redelivered, the JMS Redelivered field changes form `false` to instead display the timestamp of redelivering

2. Statistics from Oracle Enterprise Manager Oracle Fusion Middleware Control

   Oracle Fusion Middleware Control allows you to monitor the Composite Instances. To access the composite information, select **SOA** in the Enterprise Manager navigation tree and select one of the servers in the cluster. Click the composite being tested.

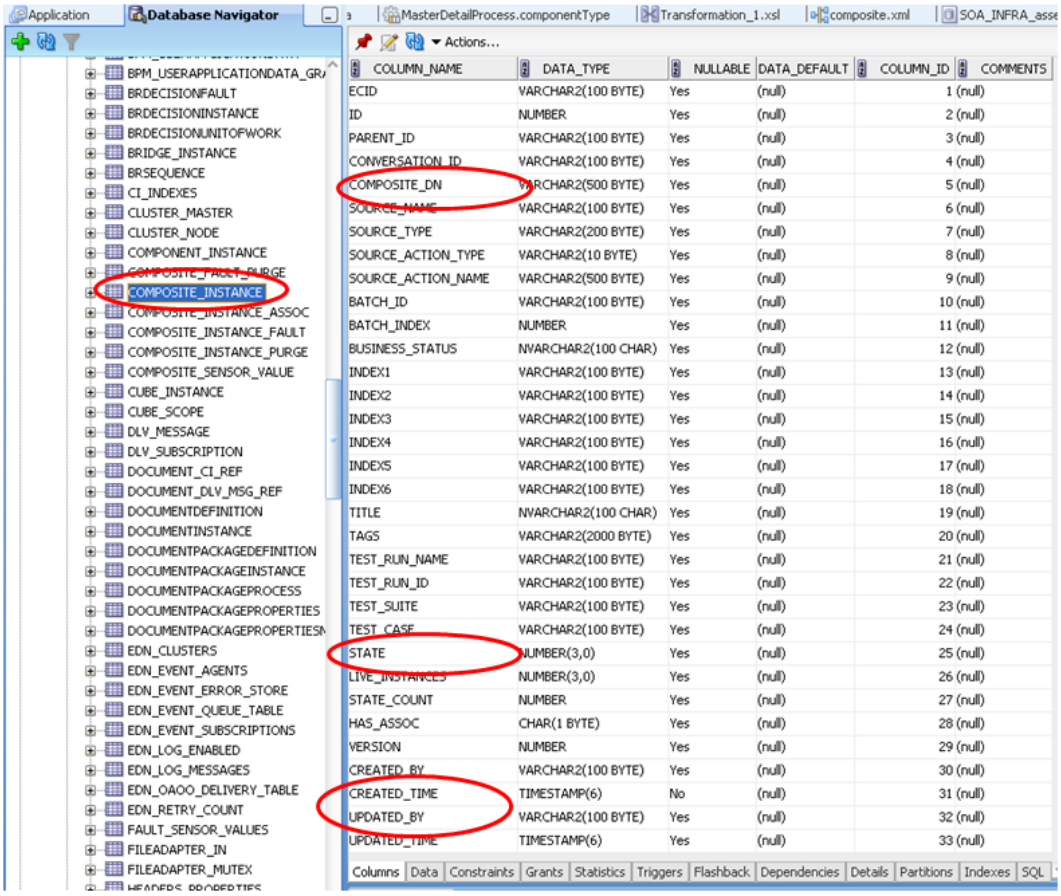


Consider the following information about this screen:

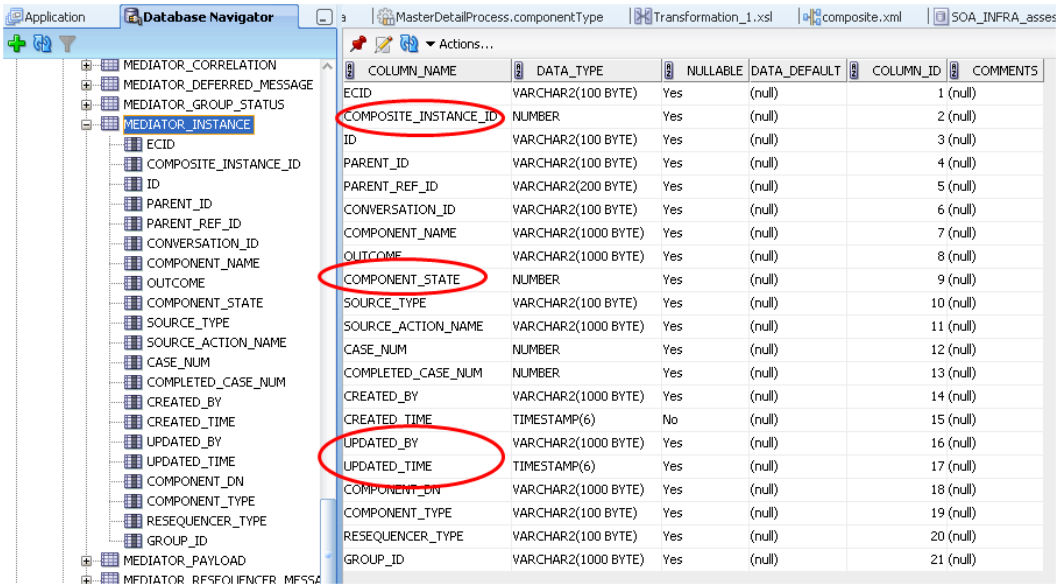
- The number of composite instances does not necessarily match the number of service engine instances. For example, the mediator instantiations may defer from the composite instantiation. The composite instances get created when a message enters a SOA system and each component in the composite creates its own instance in its instance store.
- The average processing times shown for references and services are volatile. For example, they are not persisted and are calculated for instantiations since the last restart. This can help you to determine meaningful averages across restarts.

3. Direct queries to the SOA_INFRA schemas:

   The database schemas and information stored for composites and service engines in the database are likely the most powerful sources of information for analyzing a composite and service engine behavior. The following screenshot shows a view from JDeveloper of the tables used by SOA_INFRA schema. Composite information is store in the COMPOSITE_INSTANCE table in the SOA_INFRA schema.



The following screenshot shows that the Mediator information is stored in the MEDIATOR_INSTANCE table:



The STATE, START, and MODIFY DATEs and the composite_dn/Id obtain some interesting metrics for the composites.

The following SQL queries that can be used as examples for other similar cases:

- The following SQL statement reports the number of composite instances for version 3.0 of the textmessageusingqueues composite:

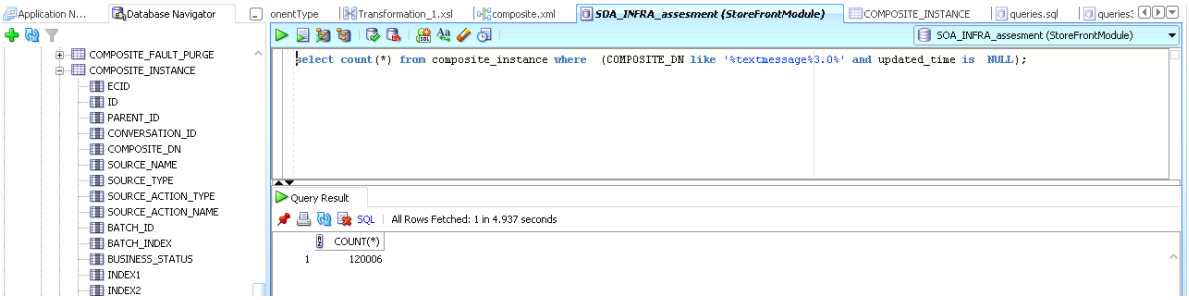
```
SQL>select count(*) from COMPOSITE_INSTANCE where COMPOSITE_DN like '%textmessage%3.0%' ;The following SQL statement reports the number of composite instances, grouping them by their status:

SQL>select count(*),state from COMPOSITE_INSTANCE where COMPOSITE_DN like '%textmessage%3.0%' group by state order by state;
```

The following list describes the different possible states from the composite instance table

```
RUNNING = 0
RECOVERY REQUIRED = 1
COMPLETED SUCCESSFULLY = 2
FAULTED = 3
TERMINATED BY USER = 4
SUSPENDED = 5
STALE = 6;
UNKNOWN= 32;
```

Because the composite instance is not well scoped when the references are not returning results, it is more useful to query the service engine instance. The COMPOSITE_INSTANCE table keeps as NULL the UPDATED_TIME for the composite, as shown in the following screenshot:



- The following SQL statement reports the number of mediator instances and groups them by status:

```
SQL> select count(*),component_state from mediator_instance where COMPONENT_NAME like
'%textmessage%3.0%' group by component_state order by component_state;
```

The following list describes the different values possible for the COMPONENT_STATE field for MEDIATOR instances:

0(000) – No faults in any of the cases till now but some might be running. Running and completed can be verified by the value of CASE_NUM, which is described after the list.

1(001) – At least one case is aborted by user

2(010) – At least one case is faulted (non-recoverable)

3(011) – At least one case is faulted and at least one case is aborted

4(100) – At least one case is in recovery required state

5(101) – At least one case is in recovery required state and at least one case is aborted

6(110) – At least one case is in recovery required state and at least one case is faulted

7(111) – At least one case is in recovery required state, at least one case is aborted and at least one case is faulted

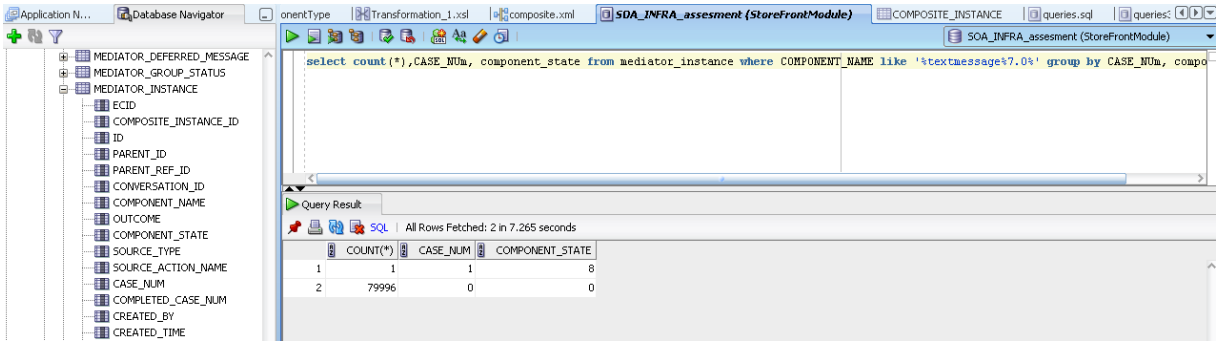
>= 8 and <16 – Running

>=16 - Stale

Once the TOTAL NUMBER of instances is obtained, you can verify the number of COMPLETED instances by querying the CASE_NUM value. This value is initially populated with the total number of cases for a mediator instance. The value is decremented after each case completion. Hence, the case_num should be zero when all instances have completed. For example, the following SQL statement reports the number of instances in different states:

```
SQL> select count(*),CASE_NUM, component_state from mediator_instance where COMPONENT_NAME like
'%textmessage%7.0%' group by CASE_NUm, component_state;
```

For example, the following screenshot shows that 79996 mediator instances have completed and one instance is running:



Finally, query the MEDIATOR_DEFERRED_MESSAGE table to obtain the status of incoming messages:

```
SQL> select count(*), component_dn, status from sh_soainfra.mediator_deferred_message group by
component_dn, status order by component_dn, status;
```

It may be useful to create a separate table to perform timestamp and date calculations for the values in the composite instance without affecting the runtime system. This facilitates analytical work without interfering with runtime operations. This is recommended for test environments. For example, the following SQL statement creates a table and copies the creation time, and modifies the_time and parent composite instance ID for each instance from the existing MEDIATOR_INSTANCE table:

```
SQL>create table textmessageusingqueues_temp(modify_date timestamp,
creation_date timestamp, diff NUMBER(10,6),cid NUMBER );
insert into textmessageusingqueues_temp(modify_date,creation_date,diff, cid)
select updated_time,created_time,CAST(substr((updated_time-
created_time),instr((updated_time-created_time),' ')+8,10) AS
number(10,6)),COMPOSITE_INSTANCE_ID  from mediator_instance where
COMPONENT_NAME like '%textmessage%8.0%' ;
```

Because the component instance is modified (for the last time in the composite's lifecycle) when the instance completes, we can use its value to calculate the time it takes for the component to complete processing. Hence, the following SQL queries report the average time, the maximum time, and the minimum time it takes for the mediator instances to complete:

```
SQL>select max(diff) from textmessageusingqueues_temp;
SQL>select min(diff) from textmessageusingqueues_temp;
```

```
SQL>select avg(diff) from textmessageusingqueues_temp;
```

Notice that these calculations are not possible for the overall composite (including references and services) but only at component level. Hence the total processing tie is the addition of the services, references and component processing times.

With these values it is easy to identify the composite that took the longest to be processed and the impact of failures in the processing time. For example, the following SQL query reports the composite ID that took the longest to be processed. This may be useful to obtain metrics across restarts:

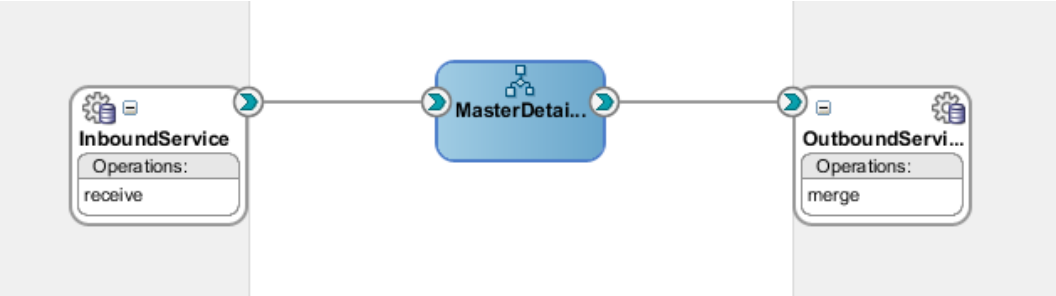
```
SQL>select cid from textmessageusingqueues_temp where diff=(select max(diff)
from textmessageusingqueues_temp);
```

Note: the statistics page for Mediator/BPEL in the Oracle Fusion Middleware Control is volatile; the values shown are only for the instances running since the last start.

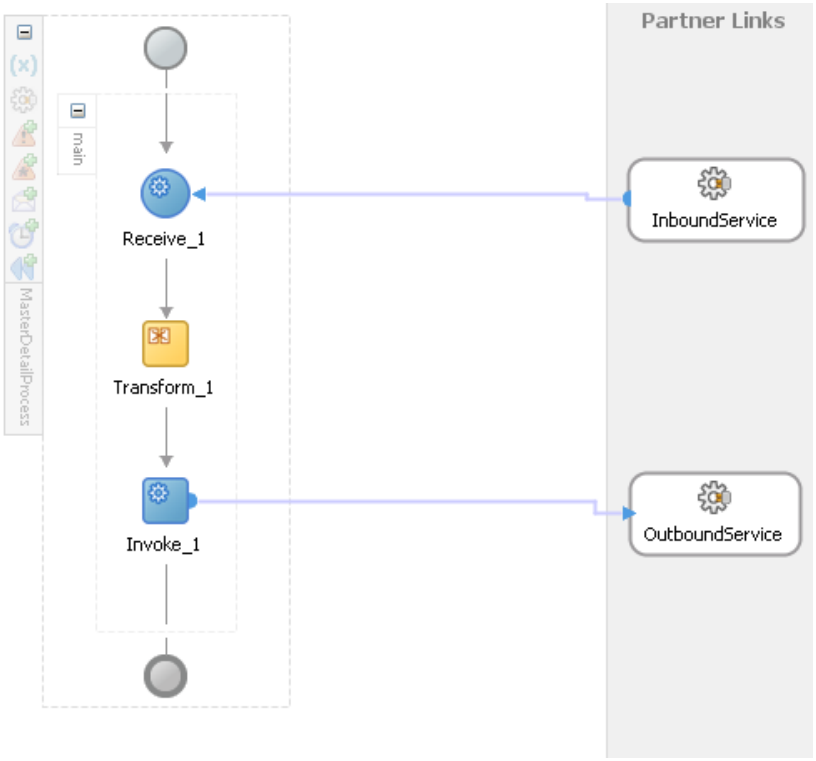
Appendix B: Use Case for Verifying DB Adapter HA Behavior

Description of the Composite Used

For the HA testing, the MasterDetail adapter sample composite was used (available in OTN at http://download.oracle.com/technology/sample_code/products/soa/soasuite/samples.zip). The MasterDetail example shows a simple scenario for replicating data in one set of tables on one database to tables on the same database or on another database. (For the MAA testing described in this white paper, the MasterDetail adapter was replicating on the same database.) The MasterDetail example involves an inbound polling read on the source tables and an outbound write/merge to the destination tables. In this example, there are two sets of department and employee tables, one of which is used for inbound data, and the other set for outbound data.

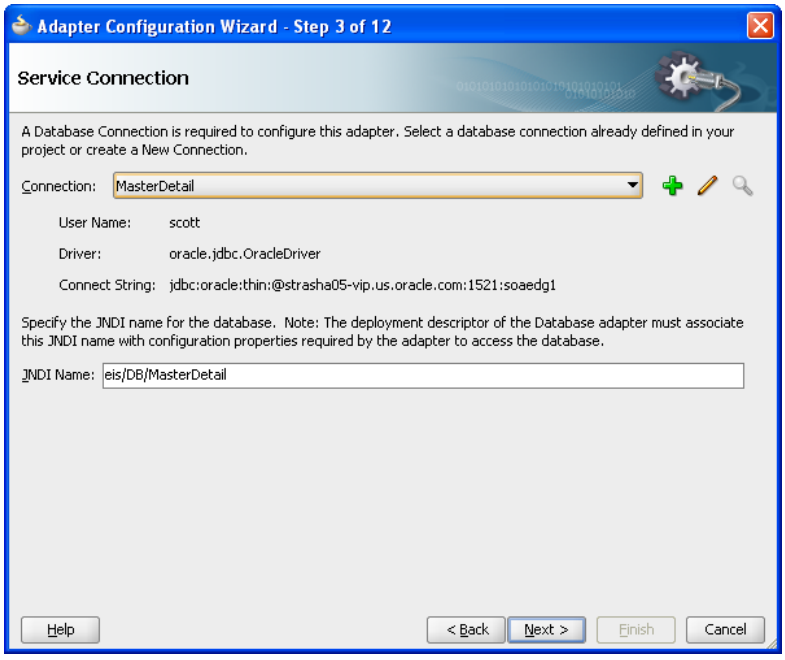


The composite uses a BPEL process with a Receive and an Invoke activity to read from the inbound tables and push to the outbound tables,:



The RECEIVER_EMP and RECEIVER_DEPT tables are used for inbound messaging and the SENDER_EMP and SENDER_DEPT tables are used as outbound tables.

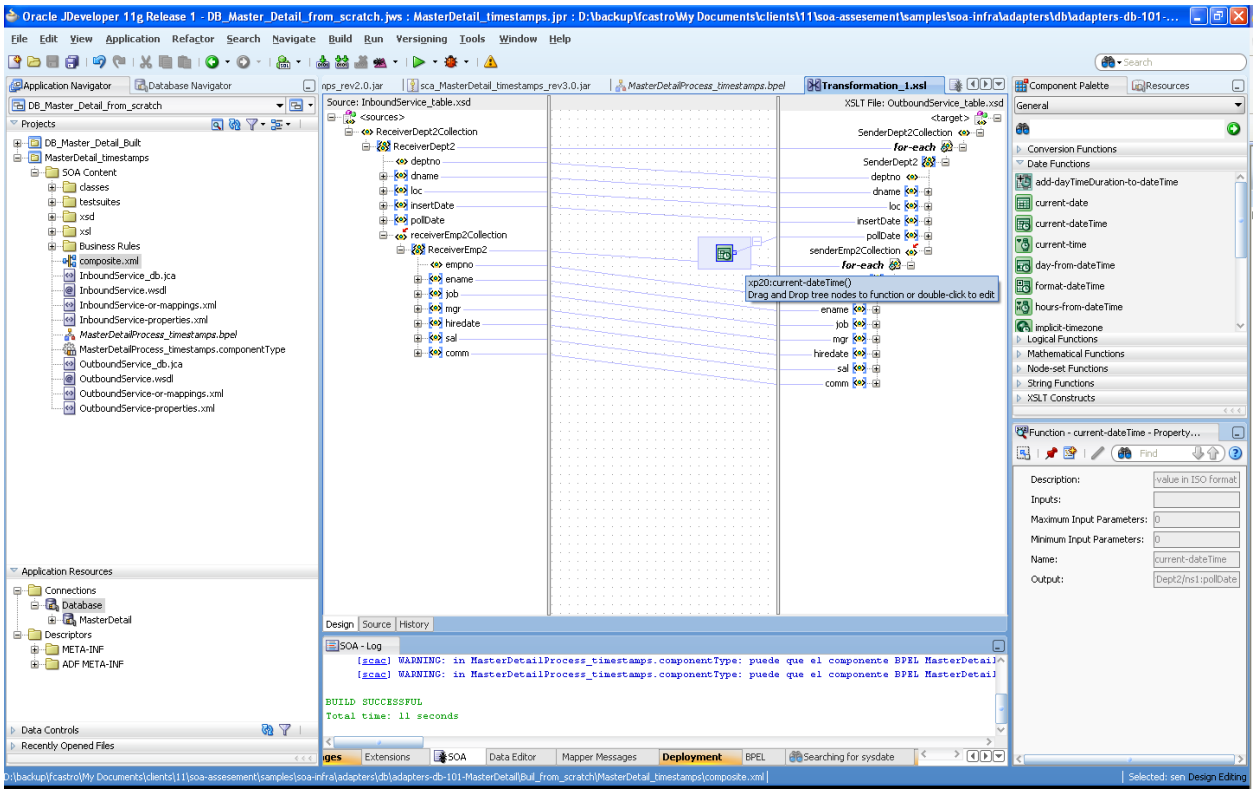
The Database Adapter endpoints were configured using the eis/DB/MasterDetail JNDI name.



The appropriate Database Adapter's data source was configured with the parameters shown in the following table, and deferring from the defaults as recommended in the *Oracle Fusion Middleware High Availability Guide*.

PROPERTY	VALUE
initial-capacity	0
Property command	<property> <name>oracle.net.CONNECT_TIMEOUT</name> <value>10000</value> </property>(*)
connection-creation-retry-frequency-seconds	10
test-frequency-seconds	300
test-connections-on-reserve	true
test-table-name	SQL SELECT 1 FROM DUAL (default)
seconds-to-trust-an-idle-pool-connection	0
global-transactions-protocol	TwoPhaseCommit(**)
keep-xa-conn-till-tx-complete	True (default)
xa-retry-duration-seconds	300
xa-retry-interval-seconds	60

A new transformation rule was added between the receive and invoke activities. The system's current time was added in a row to store the time when the insertion in the outbound tables was performed:



”LogicalDeletePollingStrategy” strategy was used to facilitate correlation of processed rows in inbound tables versus inserted rows in the outbound system. Using `LogicalDelete` may be recommended for troubleshooting and for test purposes (see the “Best Practices for BPEL and Oracle Database Adapter” section). The strategy can be easily reverted to “Delete” or other strategies using the Oracle JDeveloper Adapter Configuration Wizard.

By default the Database Adapter resource adapter is deployed to WLS Servers configured with the eis/DB/SOADemo connection factory. This factory, in turn, uses by default the same data source as the Oracle Fusion Middleware SOA Infrastructure (jdbc/SOADataSource). As a best practice, the SOA infrastructure data sources should be kept separate from the data sources used by applications. To get a different data source in use by the adapter, the Database Adapter resource was redeployed including in its weblogic-ra.xml descriptor the appropriate connection instance data:

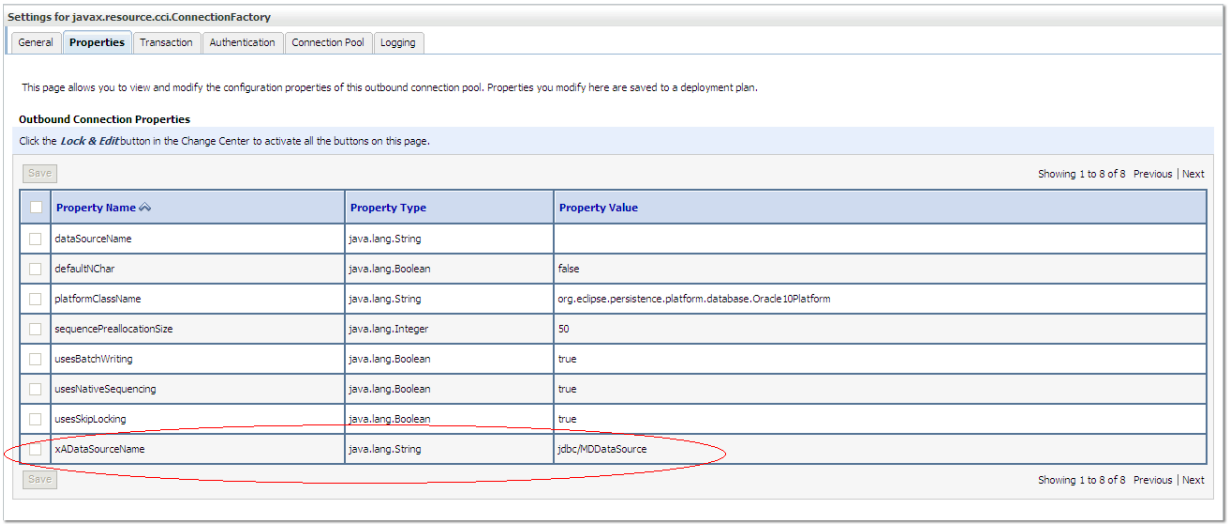
```
<connection-instance>
    <!-- Instances correspond to the individual connection pools for the resource adapter. -->
    <jndi-name>eis/DB/MasterDetail</jndi-name>
    <!-- Instance Level. Specify properties that override any properties specified at the
         group and global levels.
    -->
    <connection-properties>
      <properties>
        <!-- The name of an Application Server connection pool. An XA
             Datasource supports global transactions, jta, and XA.
             Example: jdbc/SOADataSource
        -->
        <property>
            <name>xADataSourceName</name>
            <value>jdbc/MDDataSource</value>
        </property>
        <!-- The name of an Application Server connection pool. A non XA
             Datasource that supports local transactions only.  If both
             xADataSource & dataSource names provided, XADataSource will
             be used for writing, and dataSource for reading. Use both on
             receive/polling where NumberOfThreads > 1.
             Example: loc/BPELSamples
        -->
        <property>
            <name>dataSourceName</name>
            <value></value>
        </property>
        <!-- Optional.  Any database is supported so
             org.eclipse.persistence.platform.database.DatabasePlatform is OK.
             Allows for advanced features and for you to write you own platform
             class for variations from ANSI SQL.
             Other databases:
                 Any         - org.eclipse.persistence.platform.database.DatabasePlatform
                 Oracle (9+) - org.eclipse.persistence.platform.database.Oracle9Platform
                 Oracle      - org.eclipse.persistence.platform.database.OraclePlatform
                 DB2         - org.eclipse.persistence.platform.database.DB2Platform
                 SQLServer   - org.eclipse.persistence.platform.database.SQLServerPlatform
                 Sybase      - org.eclipse.persistence.platform.database.SybasePlatform
                 Oracle Lite - org.eclipse.persistence.platform.database.Oracle9Platform
        -->
        <property>
            <name>platformClassName</name>
            <value>org.eclipse.persistence.platform.database.Oracle10Platform</value>
        </property>
      </properties>
    </connection-properties>
</connection-instance>
```

A multi data source with name jdbc/MDDataSource was created pointing to the Oracle RAC database used by the system. The multi data source included two data sources, configured to support XA Interface for Oracle RAC Database Service Connections.

To make this change effective, perform the following steps:

1. Modify the weblogic-ra.xml file as shown in the example above.
2. Update the Database Adapter.rar archive at `OHOME/soa/connectors/` with the weblogic-ra.xml file.
3. Redeploy the Database Adapter resource adapter (update it) using the Oracle WebLogic Administration Console. Because the no-stage options is used for the Database Adapter, the rar file was updated in the two Oracle Homes used by the system.

As a result, the new JNDI name appeared listed in the connection factories' properties for the adapter and the XADatsourceName was visible in the properties for the connection:



Note: To verify that the new data source is listed for the adapter:

1. Log on to the Oracle WebLogic Administration Console.
2. On the left navigation tree, click **Deployments**->**Database Adapter**->**Configuration**" and expand the javax.resource.cci.ConnectionFactory.
3. Click **eis/DB/MasterDetail**.

The jdbc/MDDataSource multi data source and subjacent data sources were created using the standard WLS Administration Console wizard and targeted to the SOA Cluster. The data sources were configured for Oracle RAC Database Service Connections and with XA support.

Description of the Client Stressing the system:

Scripts were triggered in parallel to simulate realistic row insertions in the inbound tables. To reflect a scenario where rows are inserted at constant rates but not in big batches, iterations of successive `COMMITs` are created and run multiple times. The following simplified version of the script was used during MAA testing:

```
#!/bin/bash
export ORACLE_SID=soaedg1
export ORACLE_HOME=/u01/app/db/11gR2
export PATH=$ORACLE_HOME/bin:$PATH;
loops=$1
deptno=90
echo "Adding emp rows..."
for (( c=1; c<=$loops; c++ ))
do
   echo "Inserting $c ..."
   sqlplus -S scott/tiger << EOF
   INSERT INTO RECEIVER_DEPT VALUES($deptno,'ACCOUNTING','NEW YORK');
```

```
   INSERT INTO RECEIVER_EMP VALUES ($c,'SMITH','CLERK',7902,NULL,800,NULL,$deptno);
   commit;
   exit;
   EOF
  deptno=$((deptno+1))
  echo "deptno: $deptno"
  sleep 1
done
```

Common Mistakes Deploying Database Adapter Composite

The following common mistakes can occur when deploying Database Adapter composites that should be verified when processing is not occurring as expected:

- Not creating an adapter instance entry that matches the location attribute in your db.jca file (or not creating one at all.). (See section 3.A above)
- Setting the location attribute in the db.jca file to the name of the data source directly. The adapter instance uses a level of indirection (eis/DB/...), which points to the data source pool (jdbc/...). It is a common mistake to miss this indirection and give the name jdbc/... directly in the location attribute in the db.jca file
- Not using Multi data source with the appropriate data source an connection pool recommendation for connecting to the inbound and outbound database used by the adapter

Description of the Failures Injected

Failures were injected in the system in the following ways:

- WLS Server Failures: Cron JOBS were created to "ungracefully" stop the servers present in the node every 30 minutes. To guarantee that least one server would be available for consuming messages, the jobs were alternated so that server migration would be triggered alternatively for WLS_SOA1 and WLS_SOA2:
  - At HH:00 minutes during each hour WLS_SOA1 is crashed in SOAHOST1. Because the default behavior is that NM will try to start the server again locally (restart count=1), a one-minute period of time is granted for this restart and the server is stopped again at HH:02. Server migration is triggered then to SOAHOST2.
  - At HH:15 minutes during each hour, WLS_SOA2 is crashed in SOAHOST2. Because the default behavior is that NM will try to start the server again locally (restart count=1), a one-minute period of time is granted for this restart and the server is stopped again at HH:17. Server migration is triggered then to SOAHOST1.
  - At HH:30 minutes during each hour, WLS_SOA1 is crashed in SOAHOST2 . Because the default behavior is that NM will try to start the server again locally (restart count=1), a one-minute period of time is granted for this restart and the server is stopped again at HH:32. Server migration is triggered then to SOAHOST1.
  - At HH:45 minutes during each hour, WLS_SOA2 is crashed in SOAHOST1 . Because the default behavior is that NM will try to start the server again locally (restart count=1) a one minute period of time is granted for this restart and the server is stopped again at HH:476. Server migration is triggered then to SOAHOST2.

With this failure injection cycles, and server migration kicking in, the system is reverted to its original state (WLS_SOA1 running in SOAHOST1 and WLS_SOA2 running in SOAHOST2) in every hour cycle

- File system failures: In both servers, removal of the JMS and TX logs was performed as well as a force unmount of the shared drive hosting these logs. Additional force drop of communications to the NAS device was performed by using iptables.
- Database failures: Cron jobs were created on the database nodes to alternatively "crash" each one of the two Database Instances while injecting rows in the inbound tables.
  - At HH:00 minutes each hour soaedg1 is crashed in DBHOST1
  - At HH:30 minutes each hour soaedg2 is crashed in DBHOST2

Database instances are started after a three-minute delay using Cron jobs.

Tools to Verify the Effect of Failures on the System

The failures where verified in two different ways:

Oracle Enterprise Manager Fusion Middleware Control was used to verify the number of BPEL and composite instances created as well as the faults and rejections. Specifically, for the MasterDetail case, the comparison of the number of rows inserted in the inbound table versus the number of rows produced in the outbound tables provided a good metric of the failure rates. Additionally the number of faults can be determined by querying the COMPOSITE_INSTANCE and CUBE_INSTANCE tables in the SOA-INFRA schema. Some of the following queries were used:

- To report the number of BPEL instances generated for the MasterDetail10.0 composite:

```
SQL> select count(*) from CUBE_INSTANCE where composite_name like '%MasterDetail%' and composite_revision='10.0';
```

- To report the number of COMPOSITE instances generated for the MasterDetail10.0 composite:

```
SQL> select count(*) from COMPOSITE_INSTANCE where COMPOSITE_DN like '%Master_Detail%10.0%' ;
```

- To report all instances created in June 16 that are in STALE status (stratus=6. See JMS Adapter Section 3.d for a description of the different state values)

```
SQL> select count(*) from COMPOSITE_INSTANCE where STATE='34' AND COMPOSITE_DN like '%Master_Detail%10.0%'

SQL> select count(*) from COMPOSITE_INSTANCE where STATE='6' AND COMPOSITE_DN like '%Master_Detail%10.0%' AND CREATED_TIME like '16-JUN%';
```

- To determine the number of composites in different states, issue the following query:

```
SQL> select count(*),state from COMPOSITE_INSTANCE where COMPOSITE_DN like
'%MasterDetail_timestamps%10.0%' group by state order by state;
```

A temporary table can be used to copy results and obtain some metrics for the composites generated. This facilitates analytical work without interfering with runtime operations (obviously, this is primarily recommended for test environments). For example:

```
SQL> create table scott.resultsBufTest(modify_date timestamp, creation_date timestamp, diff
timestamp);
```

This table can then be populated with some relevant information for completion timestamps (and so on) for the composites and do calculations on it without interfering in the runtime tables:

```
SQL> insert into scott.resultsBufTest (modify_date,creation_date,diff) select
modify_date,creation_date,CAST(substr((modify_date-creation_date),instr((modify_date-
creation_date),' ')+8,10) AS number(10,6)) from CUBE_INSTANCE where COMPOSITE_NAME like
'%MasterDetail_timestamps%' and COMPOSITE_REVISION = '10.0';
```

With the above data, the following SQL statement will produce the average time it takes for composites to complete. (**Note**: the statistics page for Mediator/BPEL in the Oracle Enterprise Manager FMW Control is transient/volatile. The values are shown only for instances running since the last start.)

```
SQL>select avg(diff) from scott.resultsBufTest;
```

Database Adapter inbound/outbound correlation may be achieved using PKs and differences between poll/push intervals can be obtained by using timestamps in the insertions (as described in section 3.A above).

```
SQL>select (max(insert_date)-min(poll_date)) from scott.sender_dept2_test4;
```

Finally query the dlv_message table in the soa-infra schema to detect issues with initial message delivery to the service engine. Messages delivered and undelivered are stored in the table. For example:

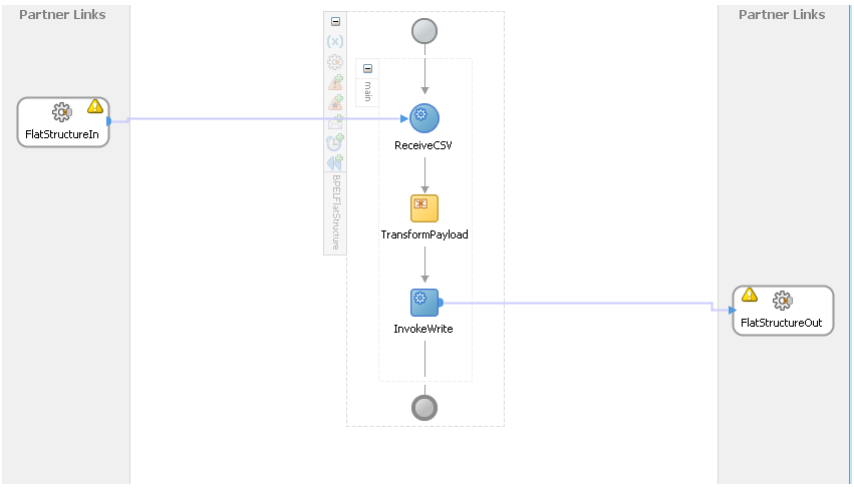
```
SQL>select count(*),state from dlv_message where COMPOSITE_NAME like '%MasterDetail%' and
composite_revision='1.0' group by state;
```

In the query output, the value State=0 indicates that the message has not been delivered to the service engine and that a problem occurred in the inbound operation.

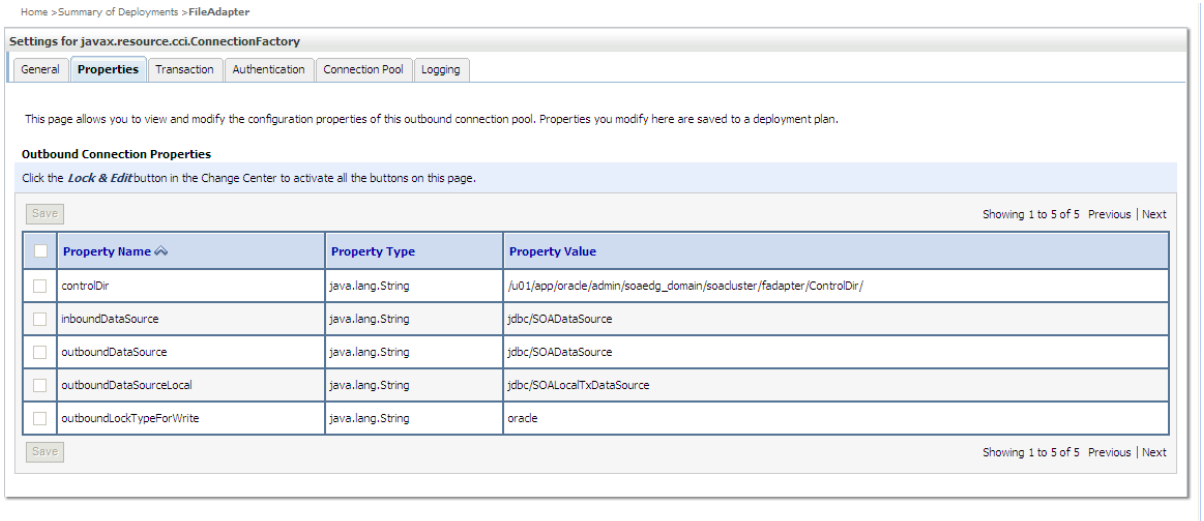
Appendix C Use Case for Verifying File Adapter's HA Behavior

Description of the Composite Used

To verify the expected behaviors and identify best practices for Oracle File Adapter, the "FlatStructure" example was used. This example is available on OTN at http://download.oracle.com/technology/sample_code/products/soa/soasuite/samples.zip). The Flat Structure business process uses the File Adapter to process address book entries from a Comma Separated Values (CSV) file. This is then transformed and written to another file in a Fixed Length Format (concatenating the two fields for "Street" present in the original file). The JDeveloper diagram for the BPEL composite is as follows:



The composite was deployed to the SOA_Cluster directly from JDeveloper. The polling frequency for the inbound adapter was lowered to three seconds. The adapter was configured to use the eis/HAFileAdapter connection factory.



All default configuration parameters were used for the File Adapter connection factories. The jdbc/SOADataSource (created by the configuration wizard during SOA installation) was used for the database mutex. The control directory for the inbound adapter was changed to a location in a shared directory. The appropriate directories (also on the same shared storage) were used for the input and archive directories for the inbound file adapter. The following shows an example "jca" file for the inbound adapter instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<adapter-config name="FlatStructureIn" adapter="File Adapter" xmlns="http://platform.integration.oracle/blocks/adapter/fw/metadata">

  <connection-factory location="eis/HAFileAdapter" UIincludeWildcard="*.txt" adapterRef=""/>
  <endpoint-activation operation="Read">
    <activation-spec className="oracle.tip.adapter.file.inbound.FileActivationSpec">
      <property name="UseHeaders" value="false"/>
      <property name="PhysicalDirectory" value="/u01/app/oracle/admin/soaedg_domain/soacluster/fadapter/Inputdir"/>
      <property name="Recursive" value="true"/>
      <property name="PhysicalArchiveDirectory" value="/u01/app/oracle/admin/soaedg_domain/soacluster/fadapter/ArchiveDir"/>
      <property name="DeleteFile" value="true"/>
      <property name="IncludeFiles" value=".*\.txt"/>
      <property name="PollingFrequency" value="3"/>
      <property name="MinimumAge" value="0"/>
      <property name="OpaqueSchema" value="false"/>
    </activation-spec>
  </endpoint-activation>

</adapter-config>
```

Similarly, a directory on a NAS shared drive was used as the output directory for the outbound file adapter:

```
<?xml version="1.0" encoding="UTF-8"?>
<adapter-config name="FlatStructureOut" adapter="File Adapter" xmlns="http://platform.integration.oracle/blocks/adapter/fw/metadata">

  <connection-factory location="eis/HAFileAdapter" adapterRef=""/>
  <endpoint-interaction operation="Write">
    <interaction-spec className="oracle.tip.adapter.file.outbound.FileInteractionSpec">
      <property name="PhysicalDirectory" value="/u01/app/oracle/admin/soaedg_domain/soacluster/fadapter/OutputDir"/>
      <property name="FileNamingConvention" value="address_%SEQ%.data"/>
      <property name="Append" value="false"/>
      <property name="NumberMessages" value="1"/>
      <property name="OpaqueSchema" value="false"/>
    </interaction-spec>
  </endpoint-interaction>

</adapter-config>
```

Description of the Client Stressing the System

A shell script running on a different node from the ones hosting the SOA cluster was used for injecting files in the input directory. The shell script copied the same payload (one single message per file) with different file names to the input directory (the name of the file was used to store the data when the file got copied). Loops of 9999 files were injected in different tests where the deployed composite was changed with different combinations of parameters (Poling frequency and `MaxRaiseSize`) and payloads (176B and 34kbyte). The following shows a simplified version of the script:

```
#!/bin/bash
test=$2
loops=$1
echo "Start time: " `date` >summary$test.log
echo "Number of files to be copied: "$loops >>summary$test.log
initial_files=`ls  /u01/app/oracle/admin/soaedg_domain/soacluster/fadapter/OutputDir | wc -l`
echo "Number of files initially in output: "$initial_files >> summary$test.log
arch_files=`ls  /u01/app/oracle/admin/soaedg_domain/soacluster/fadapter/ArchiveDir | wc -l`
echo "Number of files initially in archive: "$arch_files >> summary$test.log
for (( c=1; c<=$loops; c++ ))
 do
 echo "Copying file $c ..."
 cp                                                    /orassshare/orcl/assesment/address-csv.txt
/u01/app/oracle/admin/soaedg_domain/soacluster/fadapter/Inputdir/It$c"address""`date +%F_%T`".txt
 sleep 1
done
echo "End time: " `date` >> summary$test.log
cat summary$test.log
```

Description of the Failures Injected

Failures were injected in the system in the following ways:

- WLS Server Failures: Cron JOBS were created to ungracefully stop the servers present in the node every 30 minutes. To guarantee that least one server would be available for consuming messages, the jobs were alternated so that server migration would be triggered alternatively for WLS_SOA1 and WLS_SOA2:
  - At HH:00 minutes during each hour WLS_SOA1 is crashed in SOAHOST1. Because the default behavior is that NM will try to start the server again locally (restart count=1), a one-minute period of time is granted for this restart and the server is stopped again at HH:02. Server migration is triggered then to SOAHOST2.
  - At HH:15 minutes during each hour, WLS_SOA2 is crashed in SOAHOST2. Because the default behavior is that NM will try to start the server again locally (restart count=1), a one-minute period of time is granted for this restart and the server is stopped again at HH:17. Server migration is triggered then to SOAHOST1.
  - At HH:30 minutes during each hour, WLS_SOA1 is crashed in SOAHOST2 . Because the default behavior is that NM will try to start the server again locally (restart count=1), a one-minute period of time is granted for this restart and the server is stopped again at HH:32. Server migration is triggered then to SOAHOST1.
  - At HH:45 minutes during each hour, WLS_SOA2 is crashed in SOAHOST1 . Because the default behavior is that NM will try to start the server again locally (restart count=1), a one-minute period of time is granted for this restart and the server is killed again at HH:476. Server migration is triggered then to SOAHOST2.

With this failure injection cycles and server migration kicking in, the system is reverted to its original state (WLS_SOA1 running in SOAHOST1 and WLS_SOA2 running in SOAHOST2) in every hour cycle

- File System failures: In both server removal of the JMS and TX logs was performed as well as a force unmount of the shared drive hosting these logs. Additional force drop of communications to the NAS device was performed by using iptables.
- Database Failures: The Composite used in the test does not use database resources other than for the soa infrastructure to store composite data (such as instance ID, start date, modification date etc). However, as described above, the database is used as mutex for file locking, so database failures were also verified to characterize the behavior. Cron jobs were created on the database nodes to alternatively "crash" each one of the two database instances while injecting files in the system
  - At HH:00 minutes each hour soaedg1 is crashed in DBHOST1.
  - At HH:30 minutes each hour soaedg2 is crashed in DBHOST2.

Database instances are started after a three-minute delay using Cron jobs.

Tools to Verify the Effect of Failures on the System

The primary objective for the test was to determine the possible scenarios that could cause under and over processing. File counting and comparison of input, output, rejected, and archive folders were used for this purpose. Additionally, the correct processing of files was monitored using both Oracle Enterprise Manager Fusion Middleware Control, and different queries to the database to verify the number of files being processed at any point in time. From the SOA-INFRA database schema, the following SQL statement was used to show the status of files being "found" (0), "ready for processing" (1), and "processed" (2):

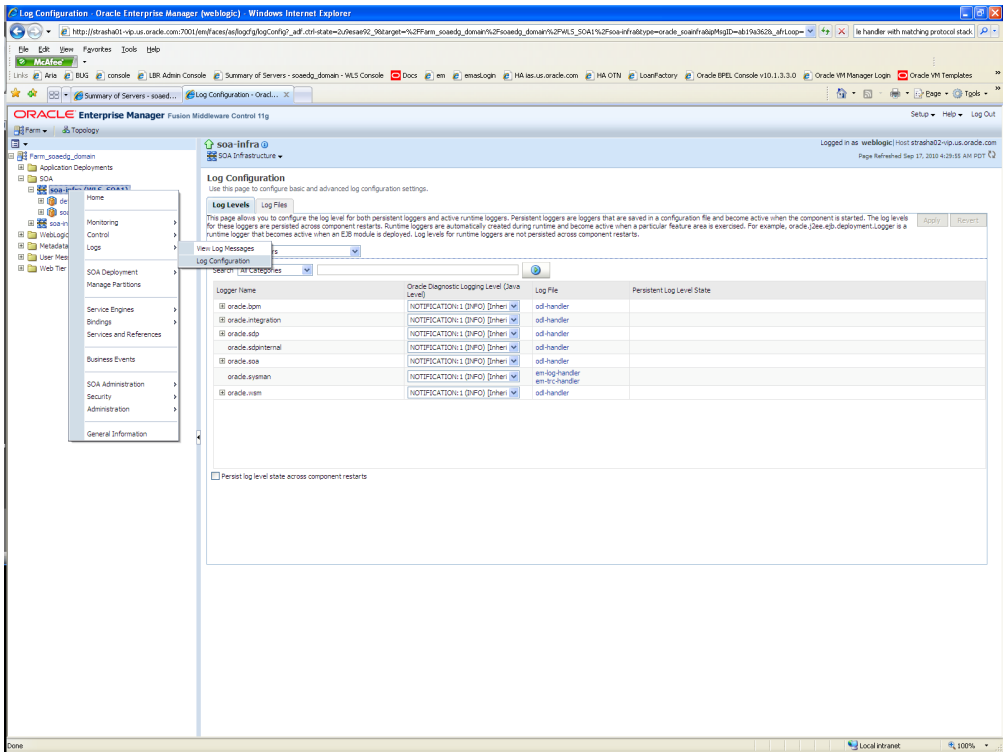
```
SQL> select count(*),file_processed from fileadapter_in group by file_processed order by file_processed;
```

A "relaxed" system should not show many files in state 1 because this would mean that threads are taking very long in processing the files. If the payloads are huge, then this could be reasonable, but not for most cases were payloads do not exceed a few Kbytes. Additionally, after a server crash, entries in status 1 signal that files have been left "dangling". If the server does not come up again (such as in the case of a miss-configuration, a deletion, and so on that would prevent even server migration), the files will be picked up by the other server in the cluster once a grace period of 15-minutes expires.

In a distributed environment, verify that endpoints in both servers are being activated. For JMS and Database adapters, you can do this by monitoring the JMS queues and Connection pools in each server, respectively. For the File Adapter, increase the log level of the oracle.soa.adapter logger and tail the diagnostic and log files for the servers. To do this:

1. With your SOA servers running, log in to Oracle Enterprise Manager Fusion Middleware Control.
2. Expand the SOA item on the left navigation tree and right Click navigate to the `SOA-INFRA` element for the server (soa_infra (WLS_SOA1) or soa-infra(WLS_SOA2)).

3. Select **Logs→Log Configuration.**

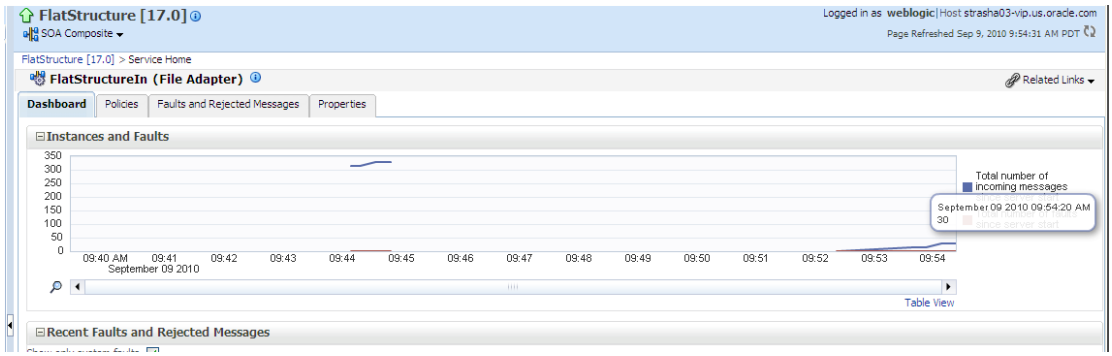


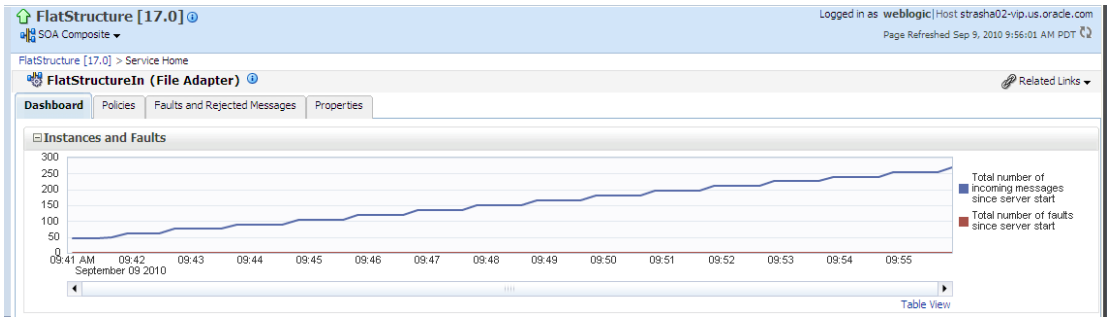
Use the following query to determine which instance has been processed by which server because each server is identified by a separate GUID:

```
SQL> select count(*),file_endpoint_guid from fileadapter_in group by file_endpoint_guid order by
file_endpoint_guid;
```

In the oracle.soa.adapter logger, set Logging Level to "TRACE:32 (FINEST)." With this setting, which should turn off for production system, the detection and consumption of the files should be reported in the log.

The throughput and processing of each endpoint can also be verified by accessing the specific composite dashboard. Once in the dashboard, click the reference/service name at the bottom of the page. The number of messages is shown individually for each server. Access the Service Metrics for the composite.





Notice that these graphics and values are only useful for verifying that both nodes are actively processing files. The information is volatile and isnot preserved across restarts of the Admin Server, nor of the servers themselves.

References

1. *Oracle Fusion Middleware Enterprise Deployment Guide* for Oracle SOA Suite
http://download.oracle.com/docs/cd/E14571_01/core.1111/e12036/toc.htm
2. *Oracle Fusion Middleware High Availability Guide*
http://download.oracle.com/docs/cd/E14571_01/core.1111/e10106/toc.htm
3. *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*
http://download.oracle.com/docs/cd/E14571_01/web.1111/e13738/toc.htm
4. *Oracle Fusion Middleware User's Guide for Technology Adapters*
http://download.oracle.com/docs/cd/E14571_01/integration.1111/e10231/toc.htm

Also, visit the MAA Web site on OTN for a list of Oracle Fusion Middleware high availability documentation and best practice white papers at
http://www.oracle.com/technetwork/database/features/availability/fusion-middleware-maa-155387.html

Oracle FMW 11g R1 SOA High Availability Assessment
January 2011
Author: Fermin Castro
Contributing Authors: Bo Stern, Amandeep Mahajan, Srimant Misra, Stephen Mcritchie, Pradeep Bhat, Viv Schupmann (editor)

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

Oracle is committed to developing practices and products that help protect the environment

0109