Oracle® Transportation Management

Application Scalability Guide

Release 6.3

Part No. E38434-01

November 2012

ORACLE

Oracle Transportation Management Application Scalability Guide, Release 6.3

Part No. E38434-01

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Contents

Send Us Your Comments

Oracle Transportation Management Application Scalability Guide, Release 6.3

Part No. E38434-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: otm-doc_us@oracle.com

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, contact Support at https://support.oracle.com or find the Support phone number for your region at http://www.oracle.com/support/contact.html.

Preface

Intended Audience

The Application Scalability Guide is intended for clients, system administrators, and consultants.

> **Note**: This version of the document assumes you are on the latest RU of 6.0, 6.1, or 6.2, and have all of the recommended Scalability patch fixes.
>
> **Note**: All diagnostic servlets referenced in this document are provided as-is.

Related Documents

- Oracle Transportation Management Administration Guide
- Oracle Transportation Management Installation Guide
- Oracle Transportation Management Technical Architecture Guide

Definition of Related Scalability Terms and Concepts

- **Machine**: A computer hosting one or more instances of Oracle Transportation Management.
- **Web Server**: A pairing of an Apache service and Tomcat servlet container running Oracle Transportation Management. The web tier is used to serve content to browsers and other web-based clients. Though production web servers typical run on a dedicated machine, multiple servers can reside on a single physical machine.
- **Application Server**: A J2EE container instance (WebLogic) running Oracle Transportation Management. The application tier is used to handle business requests from the web tier. Though production machines are typically running on a dedicated machine, multiple machines can reside on a single physical machine.

  Note: Prior to 5.5 CU5, this was referred to as an Application Machine.
- **Cluster**: A logical grouping of application servers dedicated to a set of work.

  Note: Prior to 5.5 CU5, this was referred to as an Application Server.
- **Web Cluster**: A logical grouping of web servers dedicated to a set of work.
- **Failover**: The transfer of work from one server to a backup when the server fails to respond.
- **Topology**: A specification of all web servers and application servers, along with the current distribution of work among them.
- **High Availability**: Use of automatic failover to maximize the availability of Oracle Transportation Management to end users.
- **Scalability**: Use of multiple web servers, application servers, and clusters to increase Oracle Transportation Management throughput by adding machine resources.
- **Web Scalability:** Use of web clusters to increase Oracle Transportation Management availability and throughput. Assigning specific, intensive user interface work to web clusters can reduce resource load and instability on the primary web servers.
- **Light Scalability**: Minimization of messaging and locking needed to coordinate clustered application servers.

Definition of Related Java Terms and Concepts

- **JNDI (Java Naming Directory Interface)**: A Java API for a directory service, allowing clients to query a specific server for services and resources.

- **JMS (Java Message Service)**: A Java message-oriented middleware API for sending messages between two or more clients.
- **RMI (Remote Message Invocation)**: A Java protocol built over TCP/IP, allowing one Java process to call methods in another.
- **JDBC (Java Database Connectivity)**: Industry standard for database-independent connectivity between Java and a range of databases.

Change History

Date	Document Revision	Summary of Changes
11/2012	-01	Initial release.

1. Introduction

Scalability is the name given to Oracle Transportation Management's proprietary solution of application server clustering.

Why Scalability?

Oracle Transportation Management clients would want to set up Scalability for high availability, failover capability, increasing performance, scaling horizontally as demand increases, or offloading certain business functions to different application servers.

Scalability High Availability

Scalability allows for high availability of Oracle Transportation Management by providing automatic failover to another application server. The failover could be caused by a hardware issue, but more likely will be caused by software unavailability. Oracle Transportation Management software unavailability could occur if the application server running Oracle Transportation Management would crash, Oracle Transportation Management would become deadlocked, or if Oracle Transportation Management is swamped with a large number of business requests. During these conditions, it could be possible that the application server would not be able to respond about being able to still handle addition business requests. If any of these situations would occur, Scalability allows the other application servers to take over and become the primary Oracle Transportation Management instance.

Scalability provides for a hot failover solution, which means the failover application server is ready to handle requests during a failover situation. No manual intervention by an administrator is needed as long as the Scalability configuration is setup correctly. The failover application server is not a cold system, and does not need to be started when something arises for the failover server to take over. While a cold application server running Oracle Transportation Management could be setup for failover, it has nothing to do with Scalability functionality.

Scalability Throughput

A major issue with any software is performance. Scalability can help increase performance by providing the ability of application server horizontal scaling. Scalability could be the cure to performance problems, but could also cause other issues. Adding another application server to Oracle Transportation Management may not fix performance. However, by allowing business processes of Oracle Transportation Management to be spread across different application servers the throughput of Oracle Transportation Management will increase. Adding an additional application server for Oracle Transportation Management can increase performance by up to 80%.

Scalability Caveats – When to Use Scalability, Expected Performance

Serious consideration needs to be taken before jumping into the Scalability world. Scalability allows for unlimited horizontal application server scaling capability. However, in order for Scalability to be helpful to performance, the application server needs to be the bottleneck of Oracle Transportation Management. If the application server memory is exhausted, threads are maxed out, and/or the CPU of the server running the application server is pegged, then Scalability can add additional resources to the application tier and improve overall performance. Scalability will be less useful if the database or the web server is the performance bottleneck. Proper tuning of the Oracle Transportation Management software, the application server and the JVM should be explored first before moving to a Scalability environment. Scalability could add unnecessary processing, and compound the overall performance.

Scalability is only supported on Oracle WebLogic Application Servers. Oracle Transportation Management 6.2 supports Scalability on WebLogic. However, it is our recommendation to use the latest release of Oracle Transportation Management since it has all of the newest Scalability features and the required fixes. Oracle Transportation Management databases can be clustered using Oracle

RAC technology, but that is not discussed in this document. Please refer to the supported platforms for Oracle Transportation Management versions in the Technical Architecture document for exact versions. Please also refer to the Administration Guide for minimum server requirements and specifications.

2. What Is Scalability?

Scalability is the name given to Oracle Transportation Management's proprietary solution of application server clustering. While the term "clustering" does not give a completely accurate description of Scalability, it provides an understanding at a high level. Scalability allows for multiple application servers, web servers, and the Java virtual machines (JVM) contained within them to run in separate processes on the same server or on separate physical servers. These application servers then communicate with each other to provide high availability, so that data integrity will be kept synchronized, processes will only run once, and Oracle Transportation Management will be able to failover to another application server. This concept allows Oracle Transportation Management to scale horizontally across application servers, web servers, JVMs, and physical servers.

Scalability is a custom-written mechanism to handle application server clustering and distributed communications for the Oracle Transportation Management application. Scalability does not use the WebLogic clustering solution. Scalability is a propriety scalability solution only for Oracle Transportation Management.

Architecture

How Do Oracle Transportation Management and Scalability Fit Together?

To understand how Oracle Transportation Management and Scalability fit together, the basic Oracle Transportation Management architecture should be reviewed. The Oracle Transportation Management architecture is a three-tier Java web application, which consists of a web tier, an application tier, and a database tier. The database tier is an Oracle Database server. The web tier for Oracle Transportation Management consists of the Apache Web Server along with the Apache Tomcat JSP/servlet container. The application tier is a J2EE-compliant application server like Oracle WebLogic Application Server. The Apache Web Server communicates with the Apache Tomcat servlet container using the Apache mod_jk connector. The Tomcat servlet container communicates with the application server using JNDI and RMI calls. The application server communicates with the Oracle Database using JDBC. The Apache Tomcat servlet container and the J2EE compliant application server both require a JVM to run.

As with any Java application, the memory available to each JVM heap is constrained. On a 32-bit server, the Java heap is limited to 2 GB. This Java heap limit is platform specific. This limit can cause a performance bottleneck in both the web and application servers as the additional load is added to the system. Scalability leverages memory resources from multiple machines or multiple JVMs on a single machine. This ability is achieved by scaling the application server(s) on the application tier.

What Is the Scalability Architecture?

Figure 1 summarizes resources in a typical Scalability environment. Requests from browsers are sent to a third party load balancer[1], which in turn dispatches the requests to individual Oracle Transportation Management web servers. Each Oracle Transportation Management web server analyzes the request and routes it to a cluster of application servers based on the type of work. Application servers can reside on dedicated machines or can share the resources of a single machine. Data requests are then generated by application servers and forwarded to a simple or RAC data tier.

The Scalability architecture consists of three separate components. These components are routing, data synchronization, and cross-process coordination. The Oracle Transportation Management application server Scalability is achieved by utilizing each of these three different components. All of these components are equally important. They are all discussed in more detail below.

[1] This component is not provided by Oracle Transportation Management.

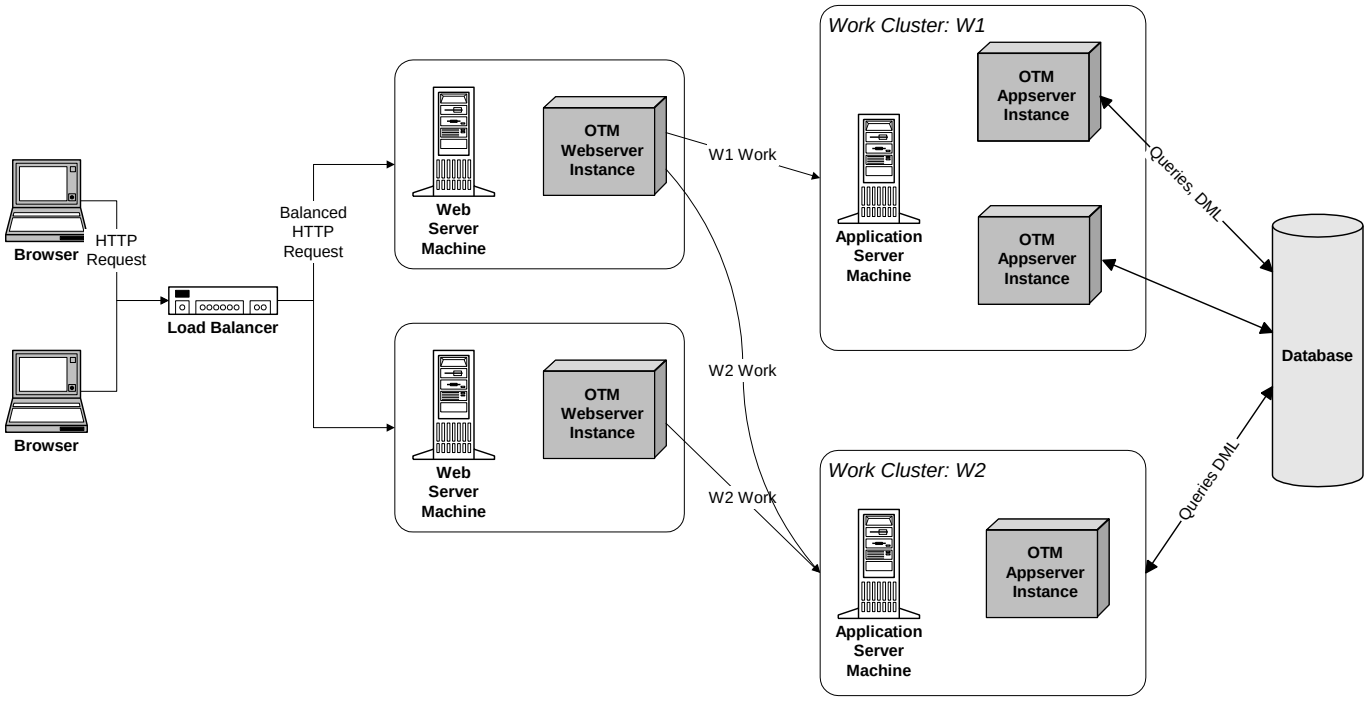


Figure 1: Scalability Environment Resources

What Is Scalability Routing?

Scalability routing is the routing of business process requests to application servers based on request content. Figure 2 summarizes the routing algorithm. It includes two basic steps:

- **Cluster Selection**: Based on request content, the system selects a cluster of application servers to process the request.
- **Server Selection**: Based on the capacities of application servers within a selected cluster, the system selects a specific application server to process the request.

The distribution of work to a cluster can be driven by various performance and business goals. A number of typical scenarios are discussed in the **Common Scenarios** section.

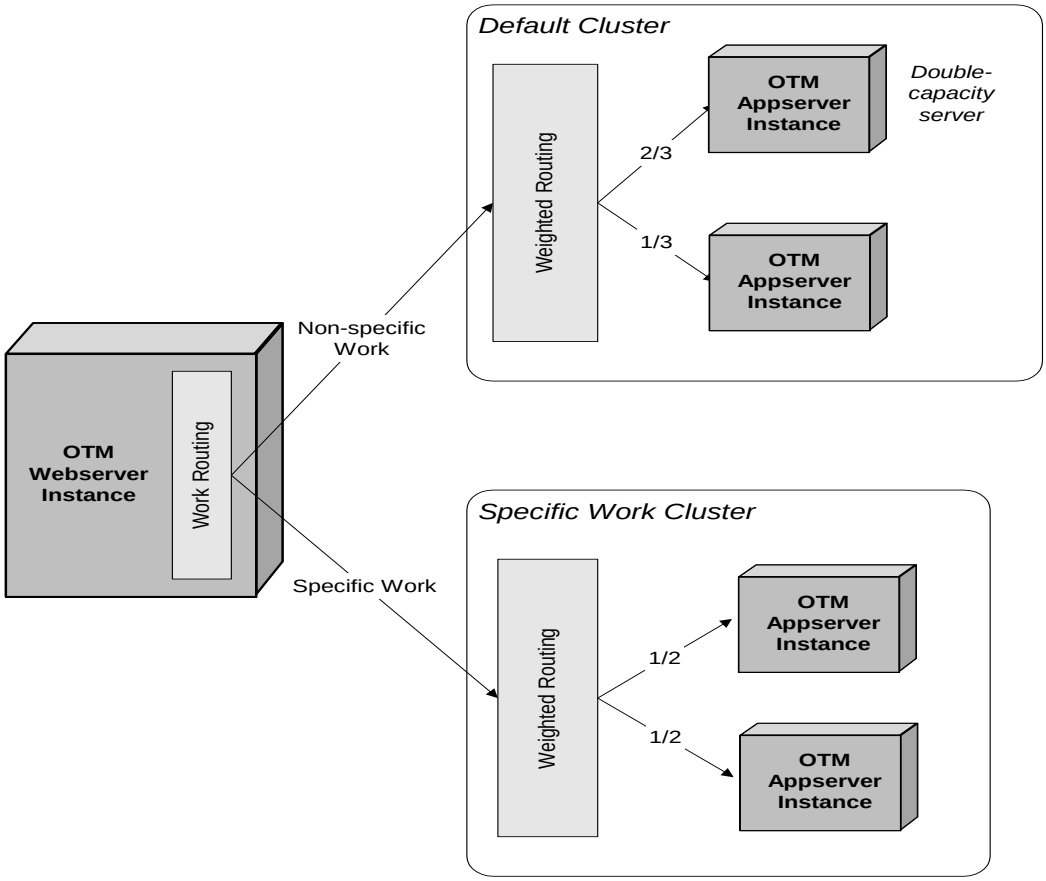


Figure 2: Scalability Routing

Business requests can come from a number of sources. The following sections describe routing subcomponents to handle each of these sources.

Web Server to Application Server Routing

Business requests originating on a web server are routed via JNDI. This routing allows the web server to lookup an application server based on the type of request, and use RMI to invoke the required business API to process the request.

For example, consider an Oracle Transportation Management implementation with two application servers and one web server. One of the application servers is placed in a bulk plan cluster, configured to handle all bulk plans. The other application server is placed in the default cluster, handling any unallocated work. When a user runs the Bulk Plan action from the Order Release screen, the web server logic starts to process the bulk plan business request and determines, by the type of work, which cluster is configured to handle bulk plans. The web server then selects an application server from the cluster and routes the bulk plan to it. If no cluster had been explicitly assigned bulk plan work, the web server would have selected an application server from the default cluster.

Application Server to Application Server Routing

Business requests originating on an application server may also be routed to another application server via JNDI. When a background agent is running on an application server, it issues work requests to the routing component. The router checks to see if the work is handled by another cluster[2]. If so, it selects an application server in that cluster and routes the request via JNDI. RMI is used to invoke the business API on the remote server. This use of the routing component guarantees that if a specific cluster is configured to handle a business function, it will always handle that business function.

For example, an Oracle Transportation Management implementation has two application servers running Oracle Transportation Management. One of the application servers is placed in a bulk plan cluster, configured to handle all bulk plans. The other application server is placed in the default cluster, handling any unallocated work. An agent is configured to run the Bulk Plan action based on some event that occurs. The logic running the agent takes the bulk plan business request and determines, by the type of work, which cluster is configured to handle bulk plans. If the requesting application server is in the cluster, it processes the bulk plan. If not, it selects another application server from the bulk plan cluster and routes the action to it. If no cluster had been explicitly assigned bulk plan work, the requesting application server would have processed the bulk plan.

Weighted Routing

Once a web server or application server selects a cluster to handle a particular request, the routing component must determine which application server within the cluster should process the request. To account for resource differences between application servers, Oracle Transportation Management allows weights to be assigned to each server within a cluster. It then routes out work within the cluster such that the number of requests received by each server is in proportion to its relative weight.[3] Typically, weights are omitted and requests are routed evenly across each server in a cluster.

For example, assume a cluster has two servers. The first server has twice the CPU speed as the second. By assigning a weight of 2 to the first server and 1 to the second, you can direct the routing component to send 2/3 of the messages to the first server and 1/3 to the second.

Failover Routing

When the routing component selects an application server to handle a request, the application server may not be responding. The router attempts to find a failover server to reroute the request as follows:

1. If the application server has been configured with one or more failover clusters, the router selects a failover cluster, in ranked order, and uses weighted routing to select a failover server from the cluster.
2. If the application server has no failover cluster, or no server in a failover cluster is responding, the router selects another server in the original work cluster.
3. If the selected failover server is not responding, the router continues with step 1 for that server.

If no failover server can be found, the router returns an error to the user or the agent.

Routing of Queued Work

Certain work in Oracle Transportation Management is queued up via database tables. This includes scheduled process management, recurring processes and data queue requests. When an item of work

---

[2] i.e. a cluster that does not contain the current application server

[3] The actual routing is done via a weighted randomization rather than round robin.

is queued, the application server uses routing to select a cluster based on the type of work. When the work is pulled out of queue and process, the application server routes the work to an application server within the cluster.

For example, an Oracle Transportation Management implementation has two application servers running Oracle Transportation Management. One of the application servers is placed in a bulk plan cluster, configured to handle all bulk plans. The other application server is placed in the default cluster, handling any unallocated work. Each night a recurring process runs a bulk plan on all new orders received that day. When adding the recurring process, Oracle Transportation Management determines, by type of work, which cluster should handle the request. Only servers in that cluster poll the database to retrieve the bulk plan request, restricting the bulk plan workload to its dedicated cluster.

What Is Scalability JMS Data Synchronization?

Scalability JMS data synchronization is a large component of Scalability which uses messages to keep data synchronized across application server caches. At a very high level, data synchronization works by having application servers communicate with each other by sending messages. These messages use the Java Message Service (JMS) API standard. JMS allows J2EE application servers to write, send, receive, and read messages based on this standard. Basically, two or more J2EE application servers running Oracle Transportation Management send messages to each other and receive messages from each other, so they can communicate and coordinate what processes need to be done. These messages vary based on what business process needs to run, whether certain bean or object caches should be flushed or updated, or whether it is a message for a network topology change. Every Scalability aware application server acts as a JMS server and client.

Cross Machine – Process Coordination

Scalability also consists of a business locking component, which is used to achieve cross application server and JVM data integrity. The business locking component in Scalability is a proprietary locking mechanism. Oracle business locking functionality provides the ability to virtually lock business objects used in Oracle Transportation Management so that data can be kept synchronized and retain its integrity when two processes are trying to modify or use the same objects. This document explains certain key concepts that are needed to fully understand locking mechanisms in Scalability.

Scalability Business Locking – In-Memory Locking vs. Database Locking

An in-memory lock is a local lock on an object achieved in the local JVM through a mutual exclusion mechanism built into the Java language. A non-scalability application server running Oracle Transportation Management uses only a local in-memory Java lock. Scalability uses in-memory locks to lock locally, but also needs to perform locking across application servers and JVMs.

Oracle Transportation Management achieves cross application server and JVM object locking by utilizing Oracle Transportation Management business database locking. Oracle Transportation Management business database locking is the use of the Oracle Database row locking mechanism and the Oracle Transportation Management OBJECT_LOCK table. These two locking mechanisms are extremely different and are used in different situations. The Oracle Database row locking mechanism is a hard lock, while the Oracle Transportation Management OBJECT_LOCK table can be considered a soft lock. These are discussed in more detail in the following sections.

Oracle Transportation Management Hard Locks – BNG Select for Update

The Oracle Transportation Management hard lock relies on and uses a SELECT FOR UPDATE statement to synchronize access to common data across application servers in a Scalability environment. This type of row locking is primarily used in the Oracle Transportation Management business number generator (BNG) to guarantee the same business number is not generated in two separate processes on two application servers.

Oracle Transportation Management Soft Locks – OBJECT_LOCK table

The Oracle Transportation Management soft locking is done in a Scalability environment by storing the lock record in the OBJECT_LOCK database table. After a process retrieves a local Java lock on the business object, it checks the OBJECT_LOCK table for a record to see if another application server machine is holding the lock. When releasing a lock, a Scalability-enabled application server will notify all other application server(s) by using a JMS message to reattempt the object lock.

The most important concept to understand is that most Oracle Transportation Management business object locks only apply to the top-level business object in a transaction such as a shipment or an order.

The number of OBJECT_LOCK records does accumulate over time, and these are now periodically purged by a staged daily cleanup process. See the **Scalability Recommendations** section for more details about these records being removed.

There are several properties that will control object locking. However, they will not be discussed in this document. These properties are very important advanced settings and they should not be modified.

Scalability Topology

The topology of a Scalability implementation is the collection of web servers, application servers, and clusters available to the routing component. It is important to understand the relationship between these resources and how they impact the routing of work.

What Is a Cluster?

An Oracle Transportation Management cluster is a grouping of application servers to handle some set of work. Each application server represents a J2EE Application Server (WebLogic), running the application-tier component of Oracle Transportation Management. Application servers typically reside on disparate server machines, but can be configured to run on a single machine.

Scalability works with clusters so that business processes can be assigned to a group of application servers, and so the business process requests can be routed correctly. In a Scalability topology, one cluster can contain many application servers. For example, an Oracle Transportation Management cluster called OTM_CLUSTER can contain application servers called APP_01 and APP_02. However, a cluster must contain at least one application server in order for the cluster to handle any business processing.

An application server can serve many application server clusters. For example, an Oracle Transportation Management application server machine called APP_02 can be used by the default cluster DEFAULT and a failover cluster, FAILOVER.

What Is a Passive (Failover) Cluster?

A passive (failover) cluster is a grouping of application servers that will take over and handle the business processing requests during a failover situation. Unlike other clustering solutions, Oracle Transportation Management Scalability views all application servers in a failover cluster as hot backups. Each server receives data updates via JMS and is ready to receive requests without delay. The only distinction between an active cluster and a passive cluster is that the passive cluster does not receive any web server, application server, or queued up work requests unless it is acting for a failed application server.

For a failover cluster to work properly, an application server must be configured to have a failover cluster assigned to it. For example, an application server called APP_01 belongs to the DEFAULT cluster. Another application server needs to be mapped to the failover cluster called FAILOVER to automatically failover to servers in that cluster.

Scalability Dynamic Topology

Scalability assumes that all web servers and application servers that could be active in a topology are specified in properties files (on each web and application server instance). To add additional servers requires modification of these properties and cycling of all servers.

Cluster topology, though, can be changed while the system is running. Screens are available to allow for a dynamic configuration of:

- **Clusters**: Clusters can be added or removed.
- **Server Assignment:** Servers can be added to or removed from a cluster.
- **Work Assignment**: Work can be added to or removed from a cluster.

It is important, though, to maintain a DEFAULT cluster with at least one application server. This cluster handles any work not explicitly assigned to another cluster.

Changing the Scalability topology dynamically forces all of the application server machines and web server to update their topology maps by broadcasting a JMS message to all servers

Startup

When the application server(s) start up, Oracle Transportation Management checks the Scalability property to see if it is enabled. If Scalability is enabled, then the Scalability properties are used by the application servers to establish an initial connection. During startup and Oracle Transportation Management Application initialization, all of the application server(s) take each application server and web server topology property and send a message to communicate. The application servers broadcast to all other application servers that they are running. The application server sends a request to a specific servlet to determine its availability. These properties are not used to determine the entire Scalability topology. Once the initial connection is made and an acknowledgment is made, the entire Application Server Scalability network topology is built using the associated Scalability database tables. This map of the topology is built by one of the application servers, and then is broadcasted to the other servers in the topology. The reason this map of the topology is built is so that the business processing requests can be routed to the correct application server. This map enables every application server and every web server to know that other application servers exist. After every application server and web server determines the network topology, messages are sent between each of the application servers to communicate and stay synchronized.

The Scalability topology associated properties are not only used during startup. These properties are used when recovering from a true failover situation. When the default application server has recovered from a failover situation, these properties will be used to communicate to the other application server(s) that it has been restarted, and then it retrieves the topology map from the other application server(s).

Failover Detection

Application server failure is detected by a Java Naming Directory Interface (JNDI) lookup failure. Once a failure occurs, failover routing selects an available application server used to process the business request. The application server entry which was used by the web server before the failure will no longer be used until the application server recovers and becomes usable.

The biggest issue in detecting an application server that has become unavailable is determining whether the application server has actually failed or it is just unresponsive because it is swamped with

business request processing. The JNDI lookup is done for every business request, so detection of the possible failure is instant. There is no amount of set time given to determine whether an application server has failed.

When the failed application server recovers from an application server failure, during startup it broadcasts a message to the other application server(s) and web server(s) that it is available. The recovered application server retrieves the topology map, and it handles business processing. The other application server(s) and web server(s) will now to be able to route business requests to the failed application server.

An application server can become unresponsive even though the server itself did not crash. An application server failure is defined as a web server unable to connect to the application server and make session bean calls. This can happen due to a backlog on the WebLogic execute threads available for RMI requests. There are many different reasons an application server can become unresponsive. This unresponsiveness leads to a false failure, and could prematurely remove the application server from the Scalability topology map. This will cause that particular application server to no longer receive work requests. The Oracle Transportation Management Scalability solution has been greatly improved to determine these false failures and avoid this situation. In particular, a keep alive ping has been added to allow an application server(s) to re-establish communication with a web server. A keep alive ping is available to notify the web server that the application server is still running and available, even if the web server previously detected failure. However, in certain uncontrollable circumstances this false failure is unavoidable. See the **Scalability Failure** section for more information.

Integration in Scalability

Oracle Transportation Management integration has a few slight differences in a Scalability environment. Depending on the Scalability topology, Oracle Transportation Management will indeed handle the load balancing of integrations across application servers. Also, depending on the topology, during an application server failure, any new integration sent to Oracle Transportation Management will automatically be picked up and processed by the other application server(s) that are still running.

However, Oracle Transportation Management integration makes a distinction between integrations which are staged versus integrations that are actively being processed. Once integration transmissions are routed to a particular application server within the cluster, they are not rerouted to another application server in the cluster. The reason for not rerouting these integrations is because Oracle Transportation Management does not know whether an application server has truly failed or it is just simply unresponsive to web requests. If Oracle Transportation Management were to reroute active transmissions to another application server, there is a very good possibility the transmissions would still be properly running on the first application server. This rerouting could then be a possibly of getting data corruption issues. For this reason, if an application server in a Scalability environment goes down it maintains ownership over its active processing transmissions. Only when that application server comes back up does it attempt to reprocess those transmissions.

There are steps that need to be taken when an application server has truly failed in order for those Integrations that were actively being processed to be picked up and processed by the application server(s) still running in the Scalability environment. See the **Scalability Failure** section for more information.

It is also strongly recommended to setup an integration reprocessing process upon startup of failed application servers in any Oracle Transportation Management environment; scalability or non-scalability. By default, all active transmissions owned by the application server are marked as REDO. These transmissions are not processed by any application server unless there is a Redo Transmission Processing recurring process scheduled (or manually triggered). This process enforces a throttle while reprocessing those integrations. To do this you must set up a Redo Transmission Processing recurring process. This particular workflow process finds transmissions with a status of REDO. This process then reprocesses N of these integration transmissions, where N is the throttle count.

Oracle Transportation Management Scalability Relationship with Queues

Oracle Transportation Management Relationship with Oracle Advanced Queuing

Oracle Transportation Management utilizes Oracle Advanced Queuing as a possible way of sending and receiving XML integrations. Oracle Advanced Queuing provides a persistent guaranteed delivery mechanism for integrations. Please see the Oracle Transportation Management Data Guide for more information.

Oracle Transportation Management Scalability can handle Oracle Advanced Queuing within an application cluster. An application cluster can be configured to service particular queues. The queue processing is spread randomly between the application machines within the cluster to which it is assigned. Scalability can also handle Oracle Advanced Queuing processing during an Oracle Transportation Management failover event. There must be a fail over cluster for this to occur.

Oracle Transportation Management Data Queuing

Oracle Transportation Management has a proprietary queuing mechanism for data integration called data queues. Data queues also provide a persistent solution for integrations.

The Scalability solution for Oracle Transportation Management can handle data queues within an application cluster. The application cluster can be configured to service particular sata queues. Data queues can also be handled seamlessly during a failover occurrence. There must be a fail over cluster for this to occur.

Oracle Transportation Management Scalability Failover and Queues

In an Oracle Transportation Management Scalability environment, the system can automatically reroute queued messages from Oracle Advanced Queuing or data queues to the failover cluster. Prior to failover, the servers in the failover cluster do not poll for new messages. Once notified they are servicing a failed server, they automatically begin polling for messages in the Oracle Advanced Queuing and/or data queues. Note that a server in the failover cluster is notified of failure only when a web request to the primary server fails. Oracle Transportation Management does not currently support any keep alive or polling methodologies for the failover server to determine the primary is down. This is a limitation for lights-out operations as they may not send any web requests to the primary server. Future releases may include additional criteria for a failover server to detect the failure of its primary server.

When a failed application server is restarted, it broadcasts a message to let its failover application cluster application servers know that it has restarted and is available. This then allows the failover application server to no longer need to service the queues for the failed application server. The queues are re-initialized again without including the failed application cluster application server.

Now since the failover application server processed the queues, there will be transmissions and transactions which were processed by the failover application server. There may be some which get stuck in a fresh state. If any exist, these will need to be either resent into the queues, or a manual SQL update will need to be done to get these processed by the default application server.

Oracle Transportation Management Web Server Relationship with Web Load Balancing

The Oracle Transportation Management application and the Oracle Transportation Management Scalability solution do not handle the load balancing of requests across web servers; a separate third party load-balancing server is needed for any of the web servers to scale. If a web server that an Oracle Transportation Management user is currently logged into goes down they will not be automatically redirected to another web server. This would be the responsibility of a third party load-balancer to detect the failure and redirect to another web server.

Oracle Transportation Management requires the use of either sticky sessions or session replication by a load balancer. Sticky sessions allow multiple user sessions to be balanced across web servers, but delegate all work for a particular user session to the web server first delegated user work (i.e. typically user login). If the web server servicing a user session goes down and a load balancer redirects the user to another web server, the user's session information is lost and they must log in again. This occurs because of the loss of session between the web servers. The sessions are "sticky" to the web server the user was first logged into, and the sessions are not replicated to other web servers. So, in a non-single sign on environment, the user would receive the login page when they lose their session. But in a single sign on environment, they would get exceptions because the session does not exist. This would only occur if the load balancer relies on the sticky sessions, and is not configured for session replication.

Session replication can be used in place of sticky sessions to maximize scalability and minimize any user impact due to failure. Certain load balancers (or, alternatively, Tomcat) can be configured to replicate user sessions across all known web servers. Though this adds network overhead to every session modification, the balancer can arbitrarily distribute user requests across the web cluster without impacting Oracle Transportation Management user state. If a web server fails, users can be redirected to an alternate web server without losing any session state. Information is not lost and the user is not required to login again. Use of session replication is a customer preference, driven by internal requirements.

The configuration of a load balancer, session replication, or Tomcat scalability is beyond the scope of this document, and is not handled by Oracle Transportation Management at all.

Oracle Transportation Management Relationship with Web Services

Scalability does not handle the load balancing of requests using web services; the application communicating with the Oracle Transportation Management application servers will have to load balance the web service requests itself.

In addition to not load balancing web service requests, depending on the application server used for Oracle Transportation Management; the application server port may be available (pingable) before the Oracle Transportation Management application is ready. This means that the external application may start sending web service requests before Oracle Transportation Management is fully activated. To solve this, the Oracle Transportation Management application opens a new port when it is fully activated and ready to receive web service requests. It is recommended to ping this new port from the external application before sending requests. This port is essentially an offset port from the application server port. The default port is 7101 for WebLogic. This port can be controlled via a property. Please see **Advanced Scalability Properties** for more details on this setting.

Allocation of Work

Scalability routing supports the allocation of specific work to a dedicated cluster. Three types of work can be routed:

- **Domain-Based Requests**: Scalability routing selects a cluster based on the business domain of the current user. Assume, for example, an Oracle Transportation Management implementation supports two major customers, BRAND_A and BRAND_B. Each of these customers is modeled as a separate data domain within Oracle Transportation Management. If a cluster, BRAND_B_CLUSTER, is assigned the domain BRAND_B, the router directs all work on BRAND_B data to the BRAND_B_CLUSTER. Work for the BRAND_A domain is sent to the DEFAULT cluster.

- **Function-Based Requests**: Scalability routing selects a cluster based on the type of work requested. The router supports over 200 types of work including most user and agent actions, screen queries, inbound integrations, process management items and optimization algorithms. Assume, for example, an Oracle Transportation Management implementation needs to offload all recurring and agent bulk plan work to a dedicated cluster. A cluster, BULKPLAN_CLUSTER, is created and assigned the two sources of bulk plan work: BULK PLAN RELEASES TO BUY SHIPMENTS and BUILD SHIPMENT (AGENT). **Table 1: Work Allocation Types** summarizes the types of supported functions and their typical use.
- **OAQ Integration Requests**: Inbound integration requests can be sent to Oracle Transportation Management via Oracle Advanced Queuing. If Scalability is enabled, these queues are not automatically monitored by the application servers. A cluster that can handle integrations must be assigned a particular queue. Servers in this cluster then monitor the queue for new integrations. This provides a mechanism to isolate all Oracle Advanced Queuing integrations to a particular cluster, or to separate integrations by content type (via multiple Oracle queues) and isolate a content type to a cluster.

A cluster can be assigned the combination of a domain and function, e.g., a cluster can be set up to process all bulk plans from domain BRAND_A. When Scalability routing selects a cluster, it uses the following precedence:

1. Find a cluster that is assigned both the current domain and the specified function
2. Find a cluster that is assigned the current domain
3. Find a cluster that is assigned the specified function
4. Use the default cluster

There are many different configurations for work allocation in Scalability.

Application Function or Type	Summary of Work
BUSINESS MONITORS	Queries supporting dynamic business monitors
DIAGNOSTICS	Ad-hoc or periodic diagnostic retrieval
INTEGRATION	Inbound XML integrations via HTTP/HTTPS
OPTIMIZATION	FICO Xpress Optimizer problem solving. In Oracle Transaction Management, many of the planning algorithms require solving of linear and mixed integer programming problems. The system delegates these problems to the FICO Xpress Optimizer package. As a native JNI library, FICO Xpress Optimizer can require large amounts of Java heap and virtual memory outside of the JVM. It can tie up the CPU solving various NP-hard algorithms and, potentially, destabilize the JVM. Scalability can be used to dedicate a cluster to FICO Xpress Optimizer processing, maximizing the resources available to optimization and minimizing the impact on the rest of Oracle Transportation Management functionality.
PLAN NON-PARTITION PLAN PARTITION	Bulk plan partitioning. To scale individual bulk plans, partition bulk plans can be routed to a cluster with multiple servers. The partitions are then balanced across the servers.

Application Function or Type	Summary of Work
PROCESS INTEGRATION	Processing of inbound XML integrations. By routing INTEGRATION to a dedicated cluster but routing PROCESS INTEGRATION to the DEFAULT cluster, the staging of integrations is offloaded. Simply routing INTEGRATION results in additional contention between user and integration activity when processing transactions.
RIQ	Ad-hoc rate queries
UI QUERIES	Queries supporting finders (search) and manager screens
Agent Action	Process triggered by an automation agent action. Each process has a corresponding Application Function with the same name, and a suffix of (AGENT).
Process Management Action	Scheduled or recurring process. Each process management item has a corresponding Application Function of the same name.

Table 1: Work Allocation Types

3. What Is Web Scalability?

Web Scalability is an optional extension of Oracle Transportation Management Scalability architecture to web servers, supporting a cluster of web servers dedicated to specific Oracle Transportation Management work.

In a standard Scalability environment, browser requests are distributed across a set of web servers using a third party load balancer. Once received by a web server, the request is routed to an Oracle Transportation Management application server via Scalability routing. This configuration provides for simple web server scaling, but assumes all requests can be evenly distributed across the web servers.

Certain web functions, though, are resource intensive. Generating ad-hoc reports on the web-tier, for example, can place a heavy load on web server CPU and memory. By dedicating a subset of the web servers to execute resource intensive work, you can increase the availability and stability of the remaining web servers.

Web Scalability is a custom-written mechanism to handle web server clustering for the Oracle Transportation Management application. Web Scalability does not use a Tomcat clustering solution. It is a proprietary approach to reroute work from the default external web servers to a set of internal web servers. It does not replace the need for load balancers or upstream routing to the default web servers.

Web Scalability Topology

What Is a Web Cluster?

An Oracle Transportation Management web cluster is a grouping of web servers to handle some set of work. Each web server represents an Apache/Tomcat Server running the web-tier component of Oracle Transportation Management. Web servers typically reside on disparate server machines, but can be configured to run on a single machine.

Web Scalability works with web clusters so that business processes can be assigned to a group of web servers, and so the business process requests can be routed correctly. In a Web Scalability topology, one cluster can contain many web servers. For example, an Oracle Transportation Management web cluster called OTM_WEB_CLUSTER can contain web servers called WEB_01 and WEB_02. However, a web cluster must contain at least one web server in order for the web cluster to handle any business processing.

A web server can serve many web clusters. For example, an Oracle Transportation Management web server machine called WEB_02 can be used by the default cluster DEFAULT and a reports cluster, REPORTS.

Scalability Dynamic Topology

Scalability assumes that all web servers that could be active in a topology are specified in properties files (on each web and application server instance). To add additional servers requires modification of these properties and cycling of all servers.

Cluster topology, though, can be changed while the system is running. Screens are available to allow for a dynamic configuration of:

- **Clusters**: Clusters can be added or removed.
- **Server Assignment**: Servers can be added to or removed from a cluster.
- **Work Assignment**: Work can be added to or removed from a cluster.

It is important, though, to maintain a DEFAULT cluster with at least one application server. This cluster handles any work not explicitly assigned to another cluster.

Changing the Scalability topology dynamically forces all of the application server machines and web server to update their topology maps by broadcasting a JMS message to all servers

Web Scalability Routing

Web Scalability routing is the routing of business process requests to web servers based on request content. It includes two basic steps:

- **Cluster Selection**: Based on request content, the system selects a cluster of web servers to process the request.
- **Server Selection**: Based on the capacities of web servers within a selected cluster, the system selects a specific web server to process the request.

The distribution of work to a cluster can be driven by various performance and business goals.

Currently, a request rerouted to another web server is piped to that server via a signed, secure servlet. All parameters, forms, and session information are passed from the incoming web server to the routed destination, guaranteeing identical behavior between the servers. The resulting HTTP response is routed back to the browser through the incoming web server. Effectively, this ties up a web server thread in the default web server, but delegates CPU and memory use to the destination.[4]

Allocation of Web Work

Web Scalability routing supports the allocation of specific work to a dedicated web cluster. The following types of work can be routed:

Web Function or Type	Summary of Work
REPORTS	Generation, transformation, and/or distribution of ad-hoc reports. By default, reports are generated, transformed and distributed on the app-tier. By modifying `glog.bipreports.appTier` properties, ad-hoc report overhead can be shifted to the web-tier. Under this scenario, dedicated a web cluster to REPORTS increases the scalability and stability of the default web cluster.
UI Action	Process requested by the user via an Oracle Transportation Management action menu item. Each process has a corresponding Application Function of the same name. By specifying a dedicated application-tier cluster in the web cluster definition, all application-tier calls from the web cluster are routed to a specified application cluster. This effectively routes a UI action to a specification application cluster.

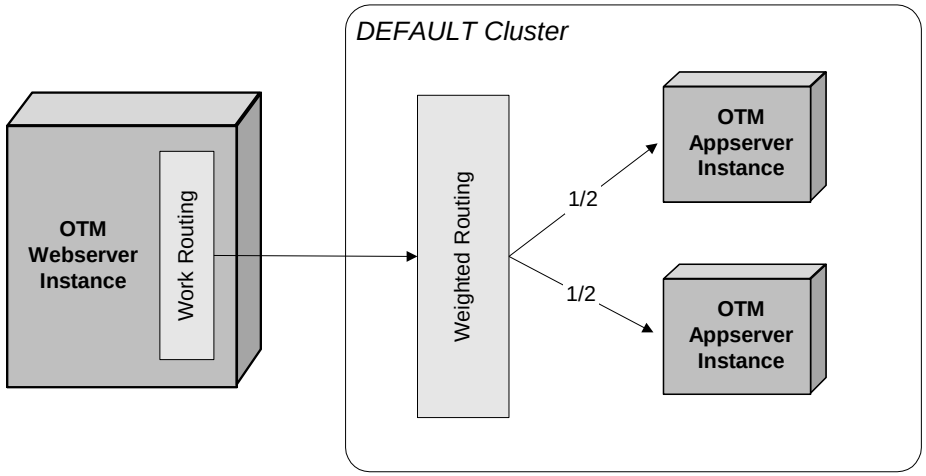
Table 2 – Work Allocation Types

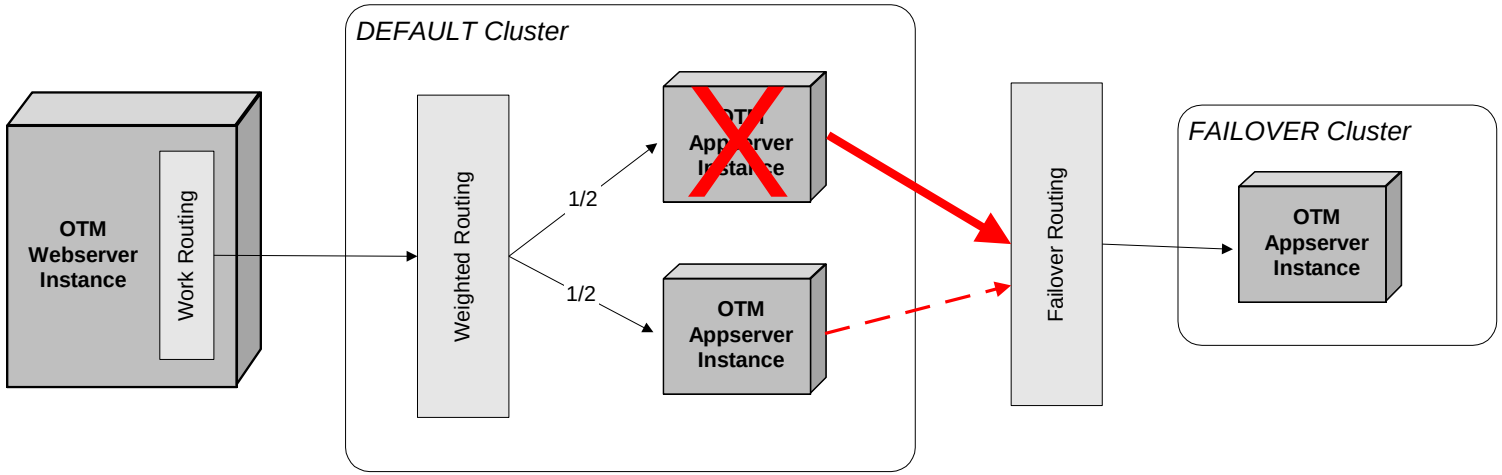
[4] Future releases of Web Scalability may include support for browser redirection to the destination server. This would require the destination server be visible to browsers along with support for session replication.

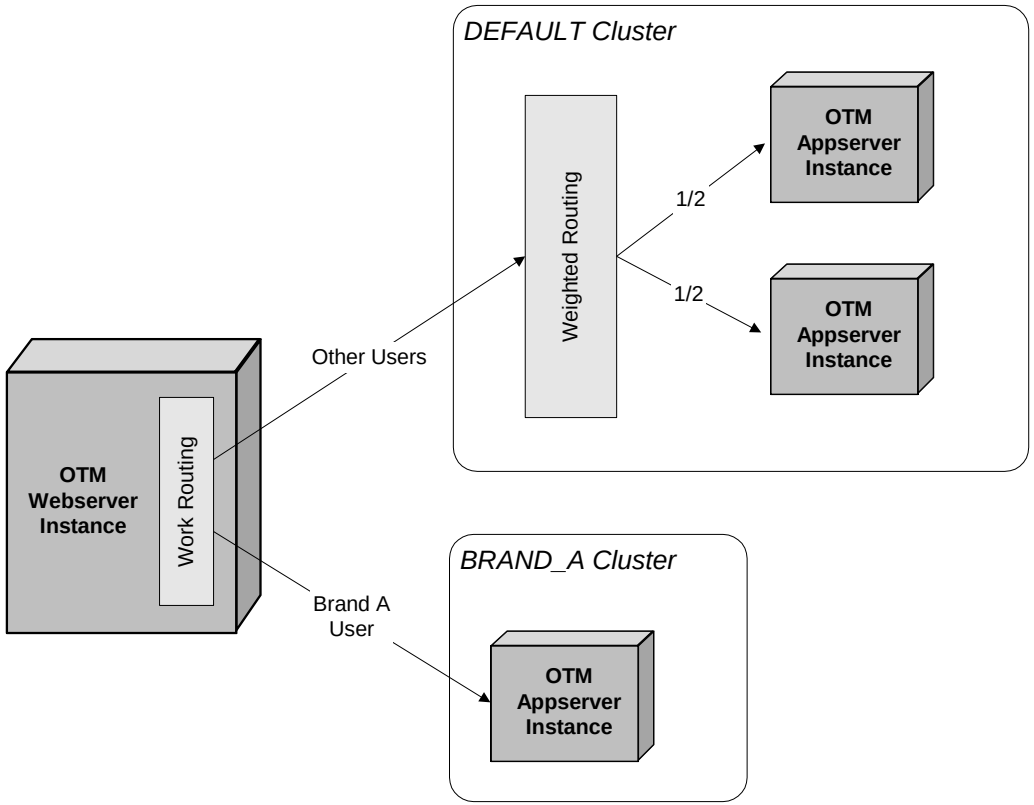
4. Common Scenarios

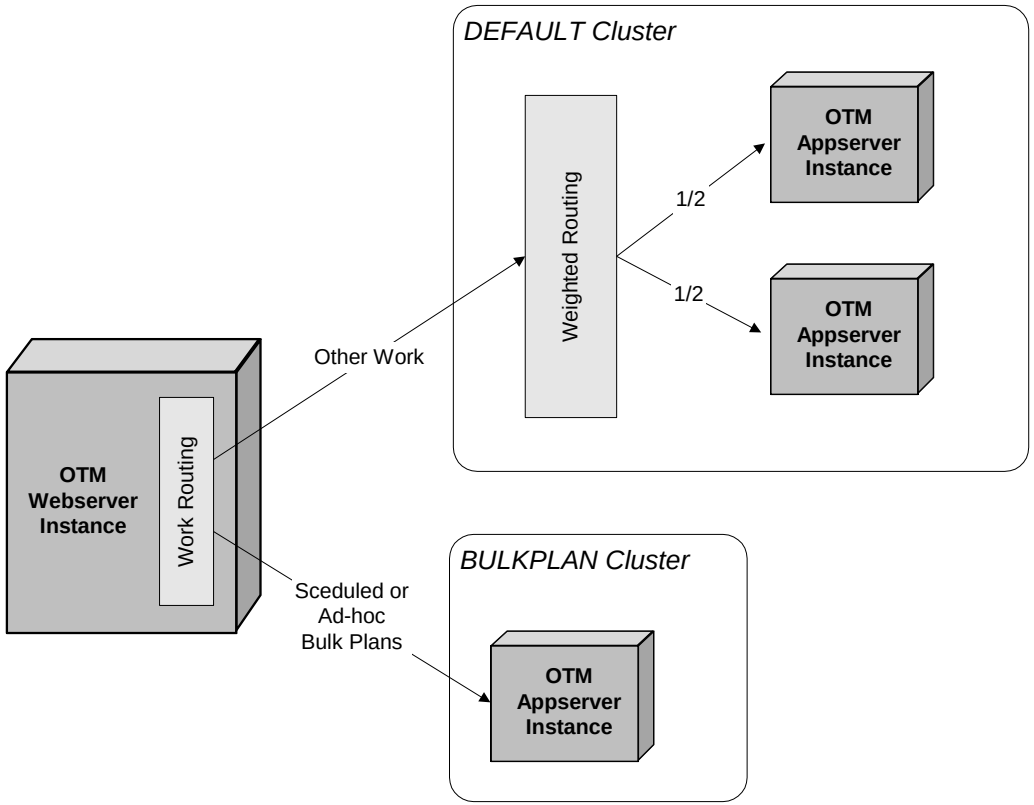
Configuring any Scalability topology starts by carefully planning the application server, web server, and physical machine topology. Every installation starts with a high scalability configuration, which acts as the basis for all other topologies. This section describes the basic topologies used by most Scalability implementations. Using these topologies, more complex scenarios can be configured that combine features from one or more of the standard topologies. For example, an implementation may configure four application servers, one to handle failover, one for customer BRAND_A, one for bulkplans and a default one for all other work.

Table 3: Basic Topologies

Name	Summary
High Scalability	1 default cluster with n active servers (also known as Active-Active) 

Name	Summary
High Availability	1 default cluster with n active servers; 1 failover cluster with 1 passive server (also known as Active-Passive) 

Name	Summary
Domain Separation	1 default cluster with n active servers; m domain clusters, each dedicated to a specified domain and with 1 active server 

Name	Summary
Functional Separation	1 default cluster with n active servers; m functional clusters, each dedicated to a specified set of functions and with 1 active server 

A high scalability configuration provides Oracle Transportation Management the ability to scale by allowing extra processing power. In addition, this configuration provides high availability. During a high scalability configuration, all application servers are handling the workload concurrently. It is automatic that if an application server crashes, one of the other application servers handles both workloads. All of the application servers are processing business process requests concurrently, as well as, sending and receiving messages to communicate.

> **Note**: When adding additional application servers to handle business processes, the additional application servers will improve your performance because they will be able to handle more workload. Using a high scalability configuration, Scalability shares all data between all of the application servers.

A high availability configuration does not help the Oracle Transportation Management application to scale, but it does provide high availability since a failover system is ready to take over whenever there is a problem. The failover application server is a hot backup. In a high availability configuration, the failover application server does not process any client business process requests because it is in failover mode, but does receive all messages to keep bean and data caches synchronized. Scalability has automatic failover and there are no administration tasks to do, as long as, Scalability was setup properly.

Domain separation allows a client to separate Oracle Transportation Management into domain cluster(s). This type of configuration allows certain application server(s) to be dedicated to servicing requests, transmissions, and business processes for a particular domain and its children. For example, two clusters X and Y, could be set up and dedicated for Domain X and for Domain Y. All business process requests for Domain X get routed to Cluster X, and subsequently all processes for Domain Y would be handled by Cluster Y. This configuration allows a client to have high availability for critical domains, and prevents system instability.

Functional separation provides a way to separate business processes onto separate application servers. This configuration allows certain Oracle Transportation Management business processes to be run on a specific application server like bulk planning, UI queries, integration, and rate inquiry queries. When Oracle Transportation Management is set up with this type of configuration and it receives a request for bulk planning, the request will be handled by the specific application server that was set up to handle the request. There are many application functions in addition to the ones listed here. There are almost 280 application functions available that can be used to set up this type of configuration.

Along with these plain types of configurations, there can be many other combinations created from these. There must always be a DEFAULT cluster to handle work not assigned to other clusters. **Figure 3** shows an example of a complex topology handling high scalability, high availability, domain separation, and functional separation.

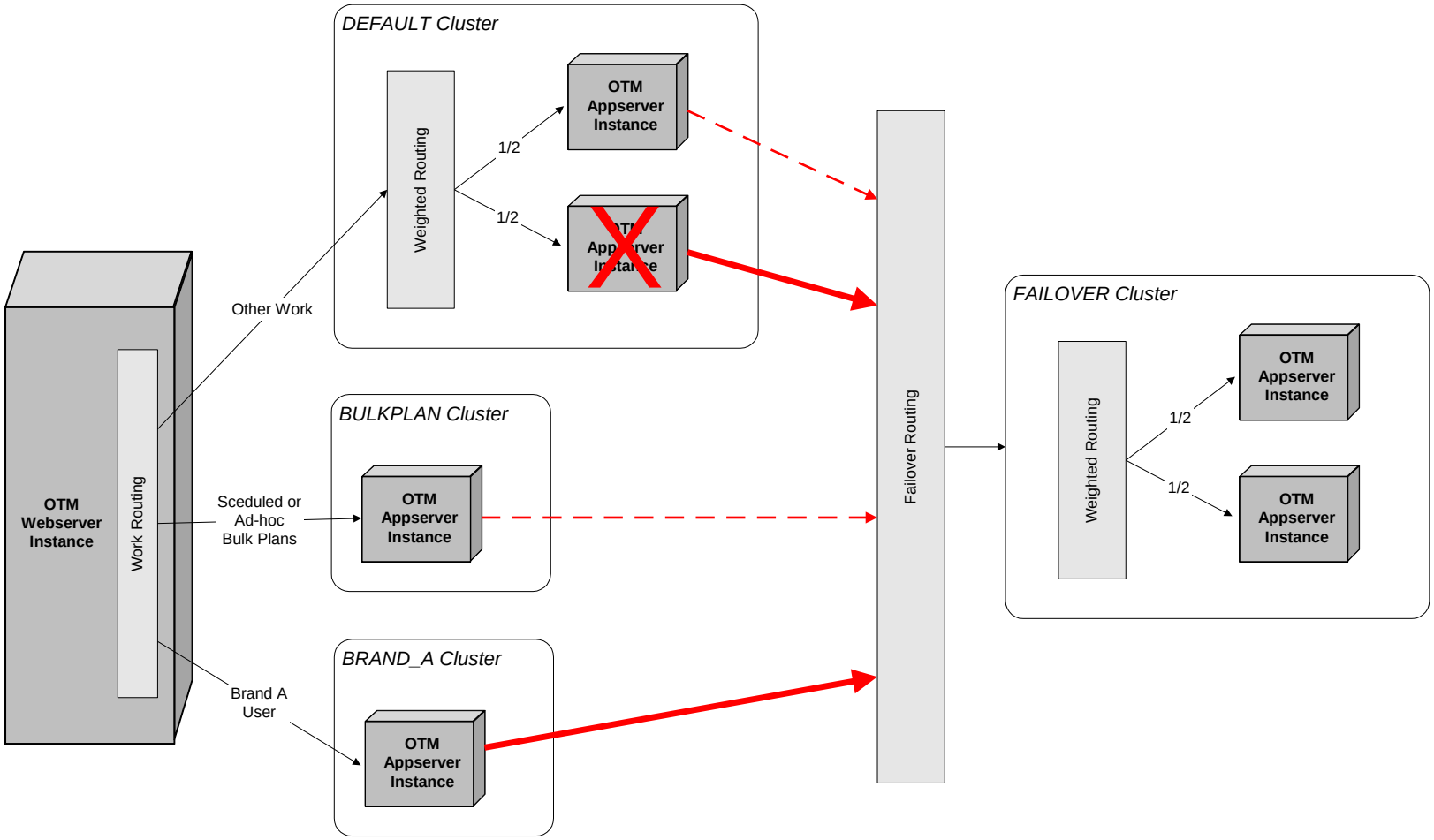


Figure 3: Complex Topology Example

5. Advanced Scenarios

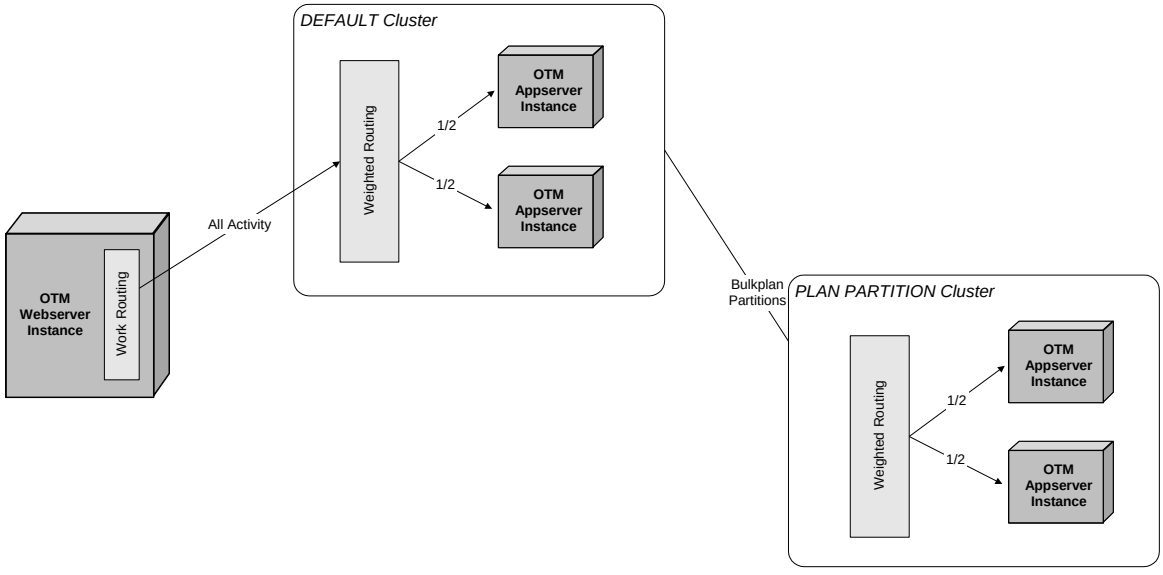
This section describes very advanced topologies not used by most Scalability implementations. Using these topologies, are not recommended for every client. These should only be used when the common scenarios including the combinations are no longer feasible or for some other strong business reason. These should also only be implemented with a high level of scalability knowledge. These advanced scenarios should also be highly tested in a non-production environment.

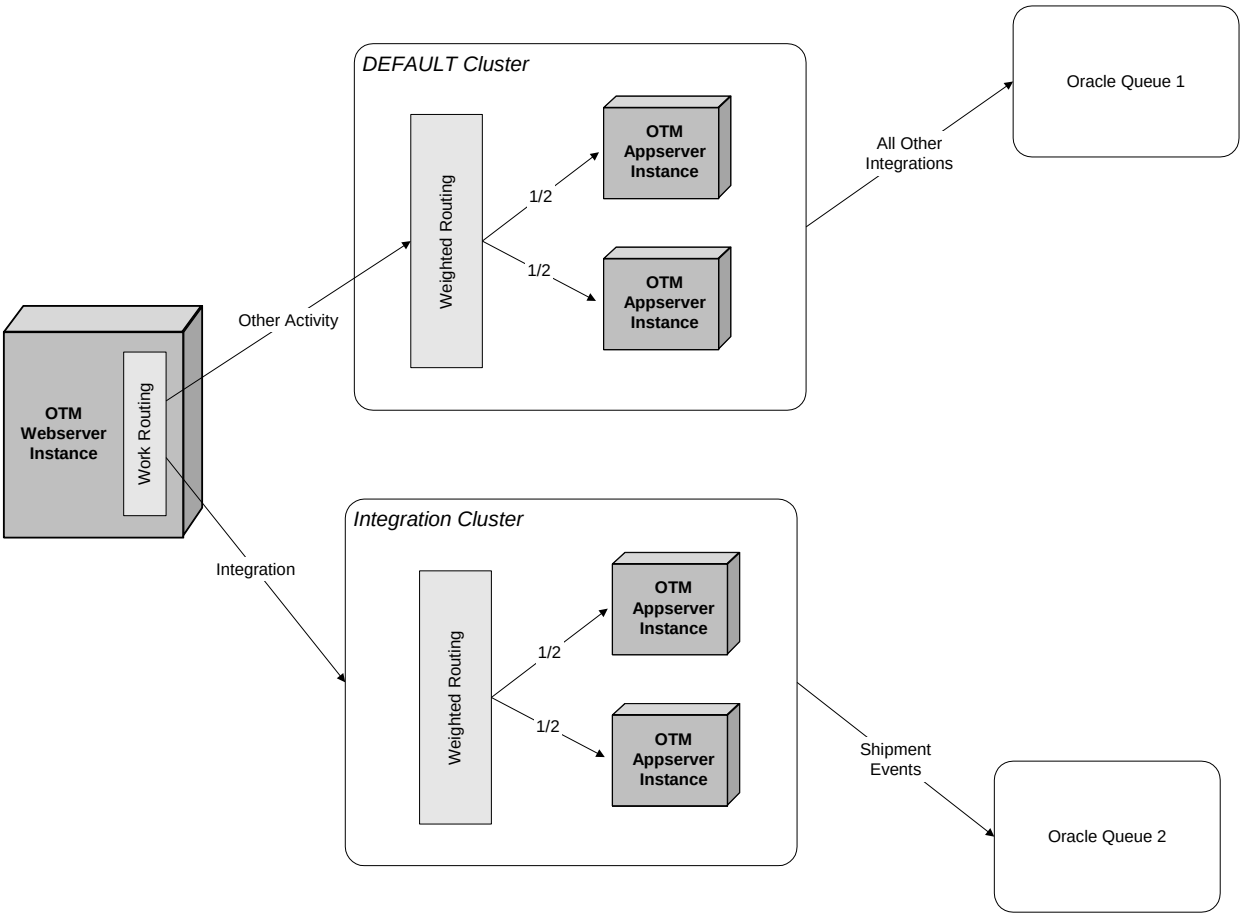
> **Note**: Every Scalability configuration starts with a high scalability configuration, which acts as the basis for all other topologies. In addition, there must always be a default cluster to handle work not assigned to other clusters.

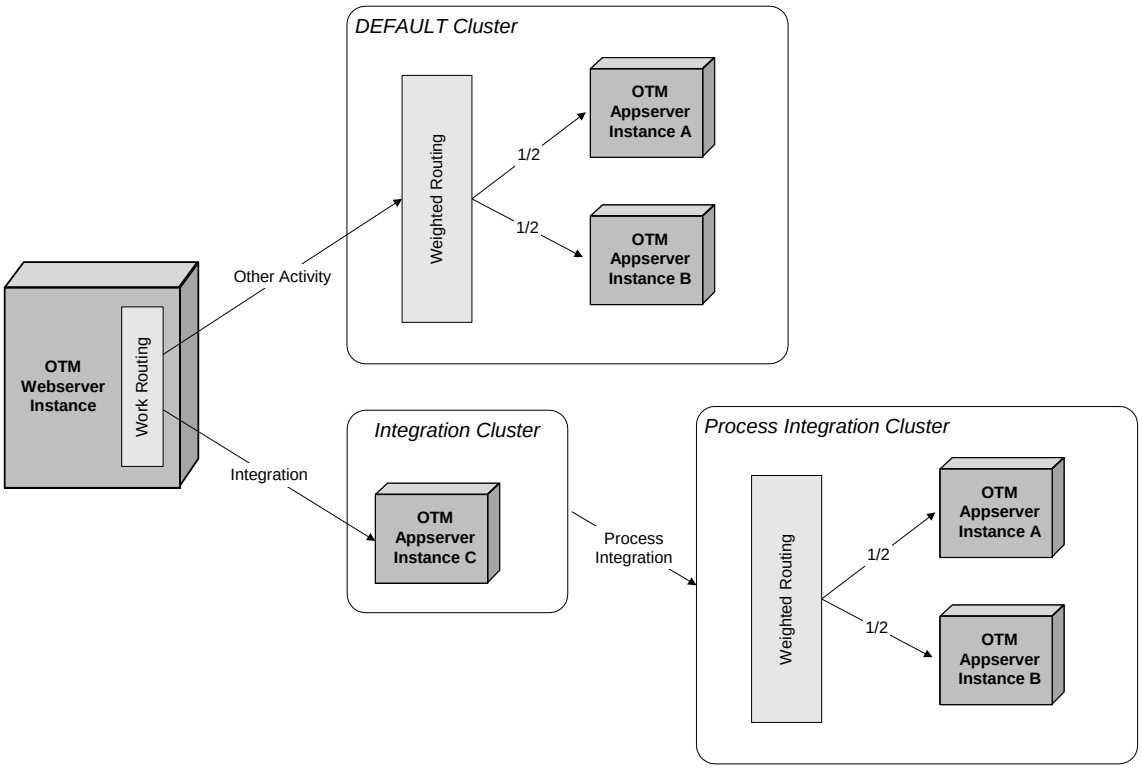
Table 4 shows the examples of these very complex topologies.

> **Note**: These are very advanced scalability scenarios. These **should only be used** after and when the common scalability scenarios are no longer feasible.

Table 4: Advanced Topologies

Name	Summary
Partitioned Bulkplan Scalability	1 default cluster with n active servers; 1 bulkplan partition cluster with n active servers; 

Name	Summary
OAQ Scalability	1 default cluster with n active servers; 1 integration cluster with n active servers; q active Oracle Queues 

Name	Summary
Integration and Staging Scalability	1 default cluster with n active servers ; 1 integration cluster with n active servers ; 1 process integration cluster with exact same active servers as default cluster 

A partitioned bulkplan scalability configuration provides Oracle Transportation Management the ability to increase performance by just scaling only partitioned bulk plans. This configuration also helps to isolate bulk plan partitions to another cluster for performance gains and not have bulk plans affect normal day operations. During a partitioned bulk plan scalability configuration, only the application servers in the DEFAULT cluster are handling all of the workload concurrently. When a bulk plan request is received on the DEFAULT cluster, it will not be processed on this cluster. Instead the bulk plan partition request will be scaled to the PLAN PARTITION cluster in order to be processed.

An Oracle Advanced Queueing Scalability configuration provides Oracle Transportation Management a mechanism to isolate all Oracle Advanced Queuing integrations to a particular cluster, or to separate integrations by content type (via multiple Oracle queues) and isolate a content type to a cluster. Inbound integration requests can be sent to Oracle Transportation Management via Oracle Advanced Queueing. If Scalability is enabled, these queues are not automatically monitored by the application servers. A cluster that can handle integrations must be assigned a particular queue. Servers in this cluster then monitor the queue for new integrations.

Integration and staging scalability configuration provides Oracle Transportation Management the ability to separate integration staging and processing. When routing integrations only to an integration application function cluster, it may create contention between all of the functions running in the default cluster and the workflow resulting from the integration. This configuration allows an alternative topology, which is to route the XML staging (XML processing and persistence) to one cluster and then re-route the integration workflow back to the same machines in the default cluster. This will eliminate the lock contention, and will allow isolation of the just the XML staging.

6. Scalability Limitations

Performance Issues

Shared Resource Overhead

Enabling Oracle Transportation Management Scalability requires coordination of shared resources across all the application servers. This reduces performance since the coordination must be implemented via database objects rather than locally in the app-tier JVM.

In a high availability configuration, with one active server and one failover server, there is an expected performance decrease compared to a single server configuration without Scalability. The typical performance overhead for a single application server is around 20%. While high availability is gained, maximum throughput is slightly reduced.

In a high Scalability configuration, this overhead is still incurred. However, the increased throughput due to increasing the number of active servers provides a scalable and highly available solution.

JMS Overhead

In a Scalability environment, every data change may[5] broadcast the change to every other server. Though the broadcast message is lightweight, the volume of JMS messages can reduce the performance of each application server.

Additional Object Contention

The high Scalability configuration may increase object contention across the servers. For example, order updates coming from upstream systems may be directed to disparate servers. If two of these updates occur on the same order, the servers need to synchronize access to the order. As additional servers increase the potential throughput of integrations, additional contention may occur as well. This may have a dampening effect on the expected performance improvement.

Data Synchronization

Inherently with JMS, there is the possibility of message latency. Oracle Transportation Management does not use persistent JMS because of additional performance concerns. However, an Oracle Transportation Management client can configure the application server so JMS persistence is enabled. There is a possibility of a delay for the JMS messages especially when there is a message for synchronization across application servers.

For example, it is possible that two active application servers could be performing a transaction on the same shipment. The first application server could finish its transaction, broadcast the required messages to the other application server, and by the time the other application server receives the messages it could have already committed the shipment. The shipment modifications done on the second application server could overwrite the modifications done on the first application server. Oracle Transportation Management does provide the ability to automatically detect this latency and fail the second transaction.

The latency of JMS messages could be minimized by increasing the dedicated JMS threads of the application server. This is an application server-specific configuration.

---

[5] A number of optimizations reduce the need for JMS messages based on knowledge of business object triggers within Oracle Transportation Management.

Light Scalability – Minimizing Messaging and Locking Overhead

Oracle Transportation Management 6.1 optimizes performance for dedicated clusters by minimizing JMS messages, database locks, and object locks needed by the cluster. Certain work requests may not require notification of data changes or synchronization of data. Light Scalability analyzes all types of work dedicated to a cluster to determine the set of messages and locks it requires. The default cluster is similarly optimized to only send messages and lock data if required by another cluster.

As an example, assume we have two clusters: a default cluster with one application server; and a UI cluster with two application servers. The UI cluster is assigned the `UI QUERIES` application function. Light Scalability retrieves the required scalability support for all application functions in the UI cluster.[6] Since the UI servers only query data, they do not require JMS notification or lock synchronization. On initialization, neither server registers interest in JMS, other than interest in scalability, security and log file changes. As the default server changes data, it determines no other server is interested in the data and suppresses the generation of JMS update messages and any synchronization locks.

Note that Oracle Transportation Management does not take into account light Scalability for failover servers. Once a server is assigned to a failover cluster, it requires all JMS messages, database locks, and object locks.

[6] from the **LITE_SCA_SUPPORT** XML from the **APP_FUNCTION** table. This schema is defined in **Light Scalability Schema** section but should only be modified under directions from Oracle Support.

7. Initial Configuration

It is extremely important that proper installation of Oracle Transportation Management and configuration of Scalability is done. This section discusses how to set up a high Scalability environment step-by-step. Once this scenario is complete and verified, adjusting the environment for other scenarios may be performed via management screens on a live system. Refer to the **Scalability Templates** section of this document for details.

This section assumes you have *n* application servers and *m* web servers. Each server has Oracle Transportation Management installed for a non-Scalability environment. Software versions on each server are identical.

Installation

Server Identification

For Scalability, each web and application server must be uniquely identified. For each application server identify:

- Its unique URL, accessible by the web servers and other application servers.

  WebLogic: t3://myApp01.domain.com:7001

  where myApp01.domain.com is the DNS entry for the application server
- Its unique display name. This maps to the APP_MACHINE.XID column and cannot exceed 50 characters.
- Its unique target name. This must match the target name supplied during installation (e.g. gc3-myApp01). Note that this target name is automatically filled in by the installer, and is only needed for WebLogic. However if using WebLogic, you should make sure it's unique across all application servers and verify it matches what is in the correct configuration file. For WebLogic this is located in config.xml.

For each web server identify:

- Its unique URL, accessible by the application servers, e.g., http://myWeb01.domain.com:80 where myWeb01.domain.com is the DNS entry for the web server.

  Note: If the common default ports are not used for the application server(s) and or for the web servers please make a note of this when making the following modifications.

- If using Web Scalability, its unique display name. This maps to the WEB_MACHINE.XID column and cannot exceed 50 characters.

Property Modifications

> **Note:** The installer creates a glog.properties file with default values for some of these property entries. These default values should modified as explained below.

On each application server, edit your glog.properties and make the following changes (all entries surrounded by [] should replaced with actual values):

```
# scalability settings
glog.scalability.on=true
glog.log.ID.Scalability.on=true
glog.scalability.thisTarget=[target name]
glog.scalability.thisMachine=[display name]
```



```
glog.scalability.thisMachineURL=[URL]
glog.scalability.defaultServer=DEFAULT
glog.scalability.defaultMachine=[APP_MACHINE_GID of this application server]
glog.scalability.defaultMachineURL=[URL of application server #1]
# application server list
!remove glog.scalability.topologyMachineURL
glog.scalability.topologyMachineURL=[URL of application server #1]
glog.scalability.topologyMachineURL=[URL of application server #2]
    ……
glog.scalability.topologyMachineURL=[URL of application server #n]
# web server list
!remove glog.scalability.topologyWebserverURL
!remove glog.scalability.web.topologyMachineURL
glog.scalability.web.topologyMachineURL=[URL of web server #1]
glog.scalability.web.topologyMachineURL=[URL of web server #2]
    ……
glog.scalability.web.topologyMachineURL=[URL of web server #m]
```

On each web server, edit your glog.properties and make the following changes

```
# scalability settings
glog.scalability.on=true
glog.log.ID.Scalability.on=true
glog.scalability.defaultServer=DEFAULT
glog.scalability.defaultMachine=[APP_MACHINE_GID of application server #1]
glog.scalability.defaultMachineURL=[URL of application server #1]
# application server list
!remove glog.scalability.topologyMachineURL
glog.scalability.topologyMachineURL=[URL of application server #1]
glog.scalability.topologyMachineURL=[URL of application server #2]
    ……
glog.scalability.topologyMachineURL=[URL of application server #n]
```

Special Note for WebLogic

If the WebLogic system password is not default (default password is CHANGEME), you need to include the following property in the glog.properties file on all application servers:

```
weblogic.system.password=[system password]
```

Special Note for Web Scalability

In addition to the above configuration, the following lines in the glog.properties file should be added for Web Scalability support:

```
glog.scalability.web.on=true
glog.scalability.web.defaultServer=[default web cluster]
glog.scalability.web.defaultMachineURL=[URL of web server #1]
glog.scalability.web.thisMachine=[web display name]
glog.scalability.web.thisMachineURL=[URL]
```

Special Note for Scalability if Using the Oracle Transportation Management Proxy Services to Communicate with External Systems Through Proxy Servers

The following line should be added in the glog.properties file for any server where the connection from Oracle Transportation Management server to that server is not required to be routed through a proxy

server. These entries would have to include all of the application servers and web servers within the Scalability topology.

```
glog.integration.http.client.nonProxyHosts=[URL of application server #1]|[URL
of application server #2]|[URL of application server #3]|…
```

Note: The property values are separated by the | character.

Database Modifications

Before running Scalability, you will need to have a DBA increase the max connection pool by 150 per application server that will be used. Please note that even if an application server is in failover mode, it is still initialized at startup and will need to be included in the total connection pool. This is modified in the init.ora under the field *.processes=150.

Database Content Modifications

Execute the following steps to initialize the topology for a high Scalability scenario. These SQL statements should be executed with the application servers down.

Log into your database as "glogowner" and modify the following records:

1. `delete from app_server_machine where app_machine_gid='DEFAULT';`
2. `delete from app_machine_failover where app_machine_gid='DEFAULT';`
3. For each application server,
   ```
   insert into app_machine (app_machine_gid, app_machine_xid, machine_url,
   domain_name)
   values ('[display name]', '[display_name]', '[URL]', 'PUBLIC');
   ```
4. For each application server,
   ```
   insert into app_server_machine (app_server_gid, app_machine_gid,
   load_balance_weight, domain_name)
   values ('DEFAULT', '[display name]', 1, 'PUBLIC');
   ```
5. `update app_machine set machine_url='n/a' where app_machine_gid='ORACLEQUEUE';`

Figure 4 shows the resulting cluster manager for the high Scalability scenario with two application servers: APP-01 and APP-02.

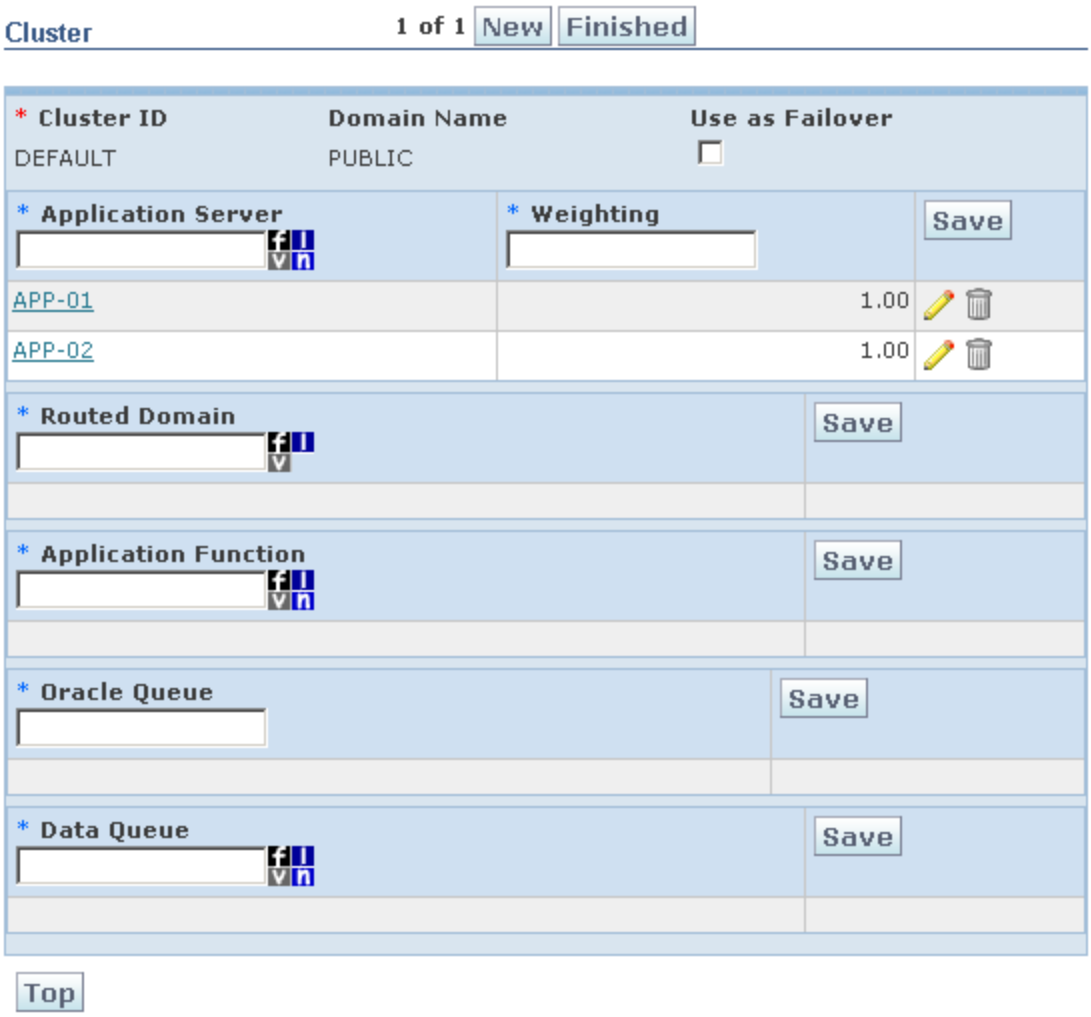


Figure 4: High Scalability Cluster

Special Note for Web Scalability

Execute the following steps to initialize the topology for a Web Scalability scenario. These SQL statements should be executed with the web and application servers down.

Log into your database as "glogowner" and modify the following records:

1. Insert an initial DEFAULT web cluster:
```
insert into web_server (web_server_gid, web_server_xid, domain_name)
values ('DEFAULT', 'DEFAULT', 'PUBLIC');
```
2. For each web server:
```
insert into web_machine (web_machine_gid, web_machine_xid, machine_url,
domain_name)
values ('[web display name]', '[web display_name]', '[URL]', 'PUBLIC');
```
3. For each web server:
```
insert into web_server_machine (web_server_gid, web_machine_gid,
load_balance_weight, domain_name)
values ('DEFAULT', '[web display name]', 1, 'PUBLIC');
```

Figure 5 shows the resulting web cluster manager for the Web Scalability scenario with two web servers: WEB-01 and WEB-02.

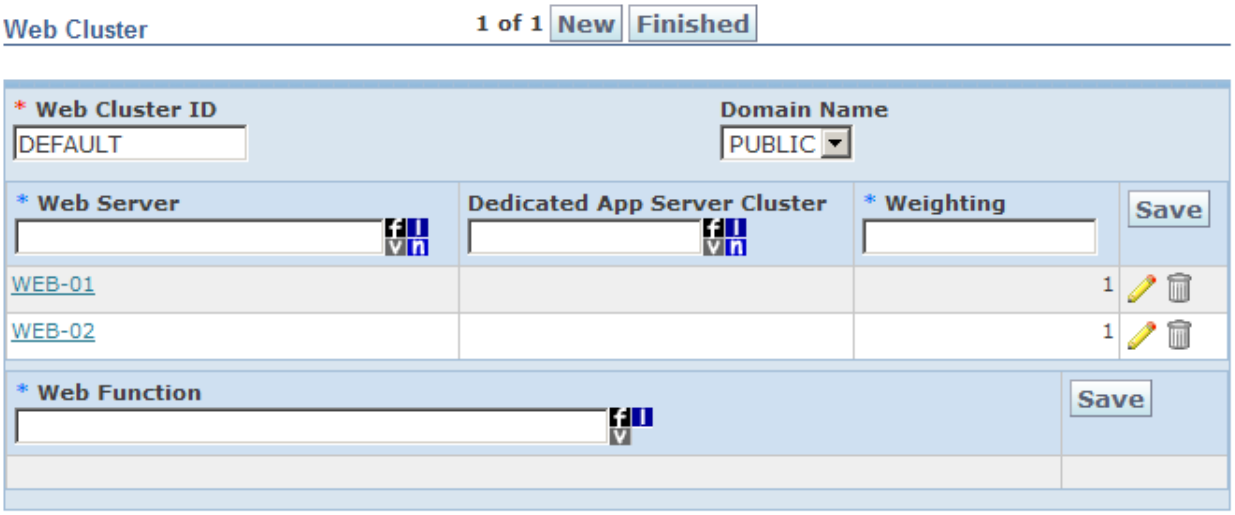


Figure 5: Web Scalability with Two Web Servers

Start

After correctly making the property and database modifications, start the entire scalability topology. Verification should then take place and is the next step.

Verification

Verify Proper Oracle Transportation Management Installation and Scalability Configuration

Scalability needs setup verification to ensure that all application servers, web servers, and physical servers were installed and configured correctly. Although the Oracle Transportation Management login page and the other user interface may work, it does not necessarily mean that Scalability is working correctly. The desired Scalability network topology should be reviewed to understand what type of Scalability setup needs to be accomplished.

All of the Scalability properties should be double-checked to make sure they are correct and that certain properties and database table fields match.

Verify Application Server Communication

The communication between the application servers in the application server cluster(s) needs to be verified.

To verify the communication from the web server(s) to the application server(s), enable the JMS and Scalability logging on the Oracle Transportation Management web servers by setting the following properties. The Webserver Properties Servlet (glog.webserver.properties.WebPropertiesServlet) can be used to set these.

```
glog.log.ID.JMS.on=true
glog.log.ID.Scalability.on=true
glog.log.ID.ScalabilityDetails.on=true
```


To verify the communication between the application servers, enable the LogIDs of JMS, Scalability, and Scalability Details.

Review the logs to verify that there is a connection between the servers, to verify they are communicating, and to verify if the Scalability configuration is working at all. Also, the Scalability logging duplicates the Scalability topology to the log. This logging can be used to verify the topology is right.

The Scalability LogId will display logging about the Scalability initialization and the topology map. For example:

```
Scalability    Initializing server routing. GC3 is scalable
[OTMActivationThread]
Scalability    Topology: sharedDomains=[GLOG, PUBLIC, DBA, SERVPROV, GUEST,
STAGE]
sharedBeans={glog.ejb.notify.db.NotifySubjectServprovPK=[],
glog.ejb.reference.db.StatusTypePK=[],
glog.ejb.appserver.db.AppSharedItemPK=[],
glog.ejb.notify.db.ThresholdProfilePK=[],
glog.ejb.agent.db.AgentEventsInvalidActionPK=[],
glog.ejb.agent.db.AgentEventPK=[], glog.ejb.currency.db.ExchangeRatePK=[],
glog.ejb.workflow.db.WorkflowTopicParamPK=[],
glog.ejb.notify.db.NotifySubjectInvpartyPK=[],
glog.ejb.bngenerator.db.BnRulePK=[], glog.ejb.agent.db.AgentActionPK=[],
glog.ejb.location.db.ExternalSystemPK=[],
glog.ejb.workflow.db.WorkflowParamPK=[],
glog.ejb.notify.db.NotifySubjectContactPK=[],
glog.ejb.bngenerator.db.BnGeneratorPK=[], glog.ejb.notify.db.ThresholdPK=[],
glog.ejb.log.db.LogfilePK=[], glog.ejb.notify.db.AuditControlPK=[],
glog.ejb.bngenerator.db.BnTypePK=[]}
map={(all)={(all)=[BLUE, DARLA, FESS]}}
```

The Scalability Details LogId logs a high level of detail as it shows the individual requests being routed. For example:

```
Debug    ScalabilityDetails    Subscribing for JMS topics on DARLA:t3://otm-
darla-60-wl.us.oracle.com:7001 [OTMActivationThread]
Debug    ScalabilityDetails    Routing GUEST.ADMIN/GUEST/null:
SecuritySessionHome to t3://otm-blue-60-wl.us.oracle.com:7001 [TP-Processor2]
Debug    ScalabilityDetails    Routing GUEST.ADMIN/GUEST/null:
SecuritySessionHome to t3://otm-darla-60-wl.us.oracle.com:7001 [TP-Processor2]
Debug    ScalabilityDetails    Routing GUEST.ADMIN/GUEST/null:
SecuritySessionHome to t3://otm-fess-60-wl.us.oracle.com:7001 [TP-Processor6]
```

The JMS LogId will show logging about the actual JMS messages being sent. For example:

```
JMS    Subscribing to GC3ObjectLockAvailableTopic on DARLA with selector null
[OTMActivationThread]
JMS    Publishing GC3ObjectLockAvailableTopic: message=TRACKING
EVENT:1082898:glog.server.synch.object.ObjectLockKey, properties={sent=2009-02-
24 15:14:57 America/New_York, from=t3://otm-blue-60-wl.us.oracle.com:7001}
```

Verify JMS Is Working

The Message Diagnostic Servlet (`glog.webserver.message.MessageDiagServlet`) can also be used to verify JMS messages are being sent correctly and synchronization is working between application servers. The Message Diagnostic Servlet will show the JMS messages sent and received to and from the J2EE Application Servers. It will not show the individual communication details between the

application server(s) and the web server(s). However, it does show a summary of the communication details between them.

The Message Diagnostic Servlet can also be used to monitor JMS performance. This diagnostic servlet can be used to monitor all JMS messages and provide detailed monitoring of QueryUpdate and CacheRefresh, and BeanUpdate messages. The Message Diagnostic Servlet can also provide statistics on JMS latency and processing.

Message Diagnostic Servlet

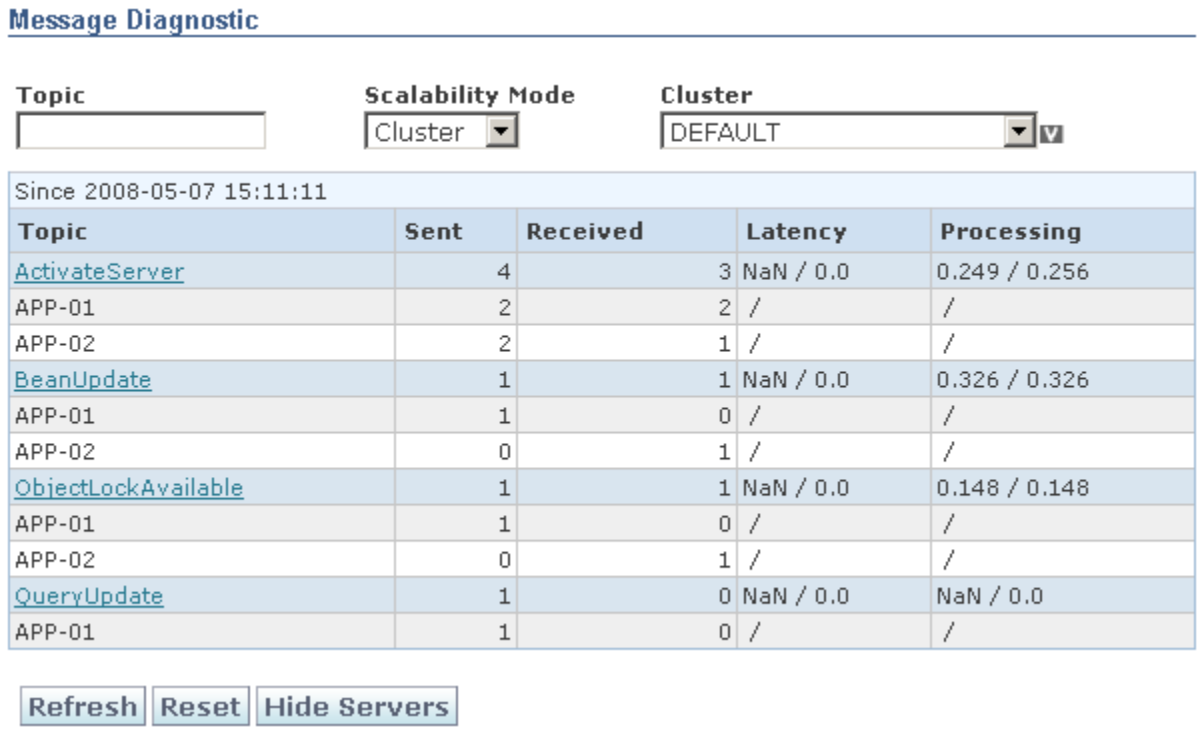


Figure 6: Message Diagnostic Servlet

Verify Routing Works

The Route Tracking Diagnostic Servlet (`glog.webserver.sca.RouteTrackingServlet`) can be used to verify that the Scalability routing is working.

This Route Tracking Diagnostic Servlet provides a web server tracking routing summary, and can also be used to make sure that the weight balancing is correct between the application servers.

Route Tracking Diagnostic Servlet

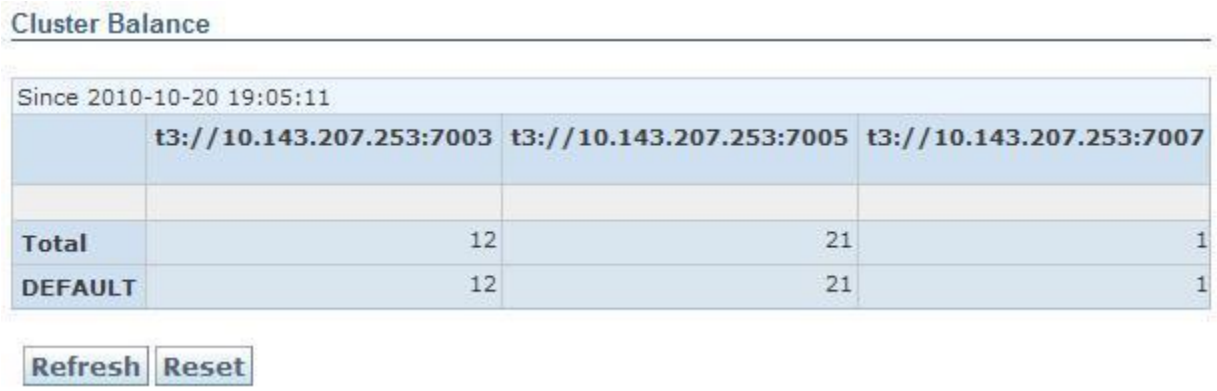


Figure 7: Cluster Balance

Scalability Overview Servlet

In addition to these Scalability monitoring tools, there is another diagnostic servlet, which could help verify the basic Scalability configurations. In order for the tool to work correctly, Scalability must be configured correctly and all application and web servers need to be up and running. The Scalability Overview Servlet (`glog.webserver.appserver.AppServerOverviewServlet`) can be accessed by being logged in as a DBA. This servlet can be found on the DBA menu by following the following menu path **Configuration and Administration > Application Server Management > Scalability Overview**.

> **Note**: The Scalability Overview Servlet was added in Oracle Transportation Management 5.5 CU4.

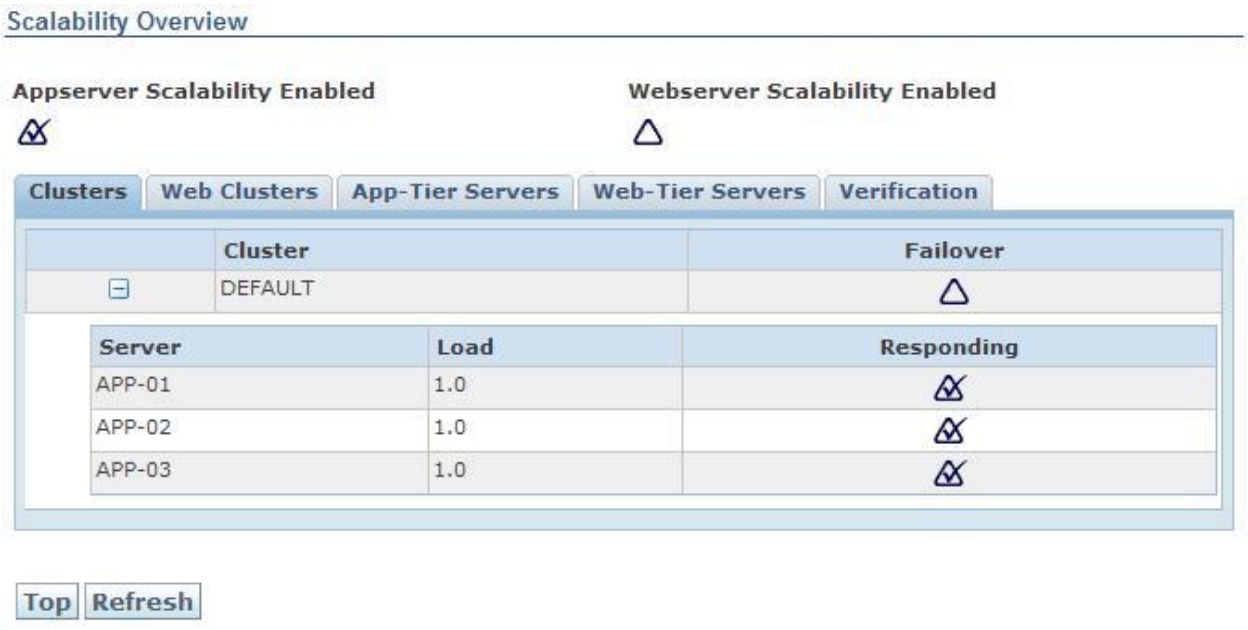


Figure 8: Scalability Overview

See the Scalability Overview online help topic for information about the screen.

Verification FAQs – How to Figure Out What Is Wrong

Scalability Logging

Scalability Exception Logging

Review the glog.web.log and the glog.exception.log for any exceptions that may be occurring.

Exception logging from Scalability may look like:

```
Error    Exception    weblogic.jms.common.JMSException: Connection not found
weblogic.jms.common.JMSException: Connection not found
Javax.naming.NameNotFoundException: jms/TopicConnectionFactory not found
NO MACHINE IN THE DBA/(FUNCTION=NULL) CLUSTER IS RESPONDING
```

Database Records

Upon successful configuration of high Scalability, the following data records should exist:

APP_SERVER				
APP_SERVER_GID	APP_SERVER_XID	DOMAIN_NAME	IS_FAILOVER	TRANSACTION_JMS_FLAG
DEFAULT	DEFAULT	PUBLIC	N	ALL

APP_MACHINE			
APP_MACHINE_GID	APP_MACHINE_XID	MACHINE_URL	DOMAIN_NAME

APP_MACHINE			
[Display name for first server] e.g. APP_01	[Display name for first server] e.g. APP_01	[URL exactly matching glog.scalability.thisMachineURL in glog.properties on the first server]	PUBLIC
[Display name for second server] e.g. APP_02	[Display name for second server] e.g. APP_02	[URL exactly matching glog.scalability.thisMachineURL in glog.properties on the second server]	PUBLIC
…	…	…	…
[Display name for last server] e.g. APP_0[n]	[Display name for last server] e.g. APP_0[n]	[URL exactly matching glog.scalability.thisMachineURL in glog.properties on the last server]	PUBLIC

APP_SERVER_MACHINE			
APP_SERVER_GID	APP_MACHINE_GID	DOMAIN_NAME	LOAD_BALANCE_WEIGHT
DEFAULT	[Display name for first server] e.g. APP_01	PUBLIC	1
DEFAULT	[Display name for second server] e.g. APP_02	PUBLIC	1
DEFAULT	[Display name for last server] e.g. APP_0[n]	PUBLIC	1

8. Scalability Templates

In this section, various Scalability scenarios will be shown by example. As each example assumes a base installation of the high Scalability scenario, changes are limited to data content. Two configuration methods are given for each scenario:

- Administration screens. The DBA.ADMIN user can reconfigure their Scalability scenario by using Application Server Management screens. This is appropriate when you want to modify a live system.
- SQL scripts. Tables controlling Scalability topology can be directly modified by SQL. This requires a restart of all web and application servers and is appropriate for initial installation and deployment of the system.

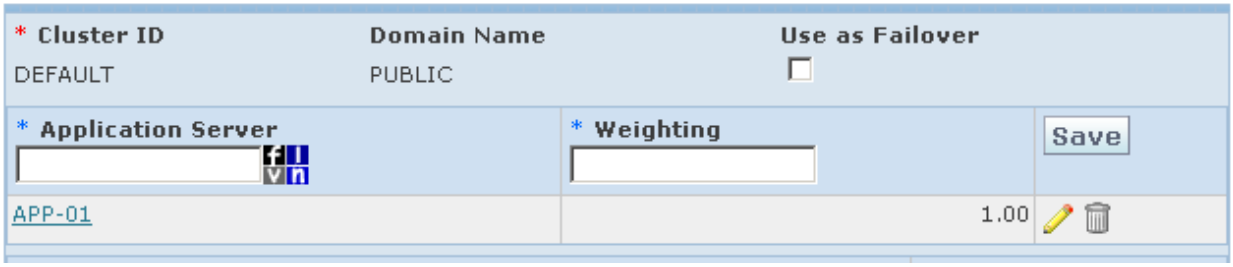
Screen shots should be used as a template, replacing sample machine and cluster names with your own. SQL scripts can similarly be copied and modified to replace canned values with those appropriate for your topology.

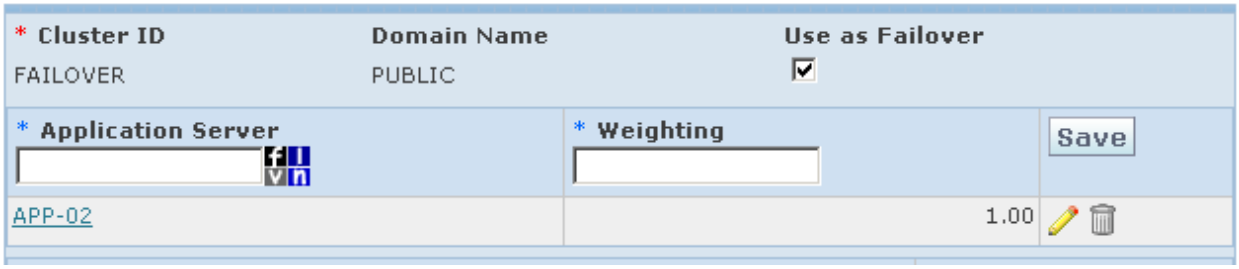
Failover

To dedicate server APP_02 as a failover for server APP_01:

1. Create a new cluster, FAILOVER, to handle a failure in APP-01.
2. Mark the FAILOVER cluster as failover only, by checking the Use as Failover box.
3. Remove the APP-02 server from the DEFAULT cluster
4. Add the APP-02 server to the FAILOVER cluster
5. Assign the FAILOVER cluster to the APP-01 server

Figure 9 shows the resulting Cluster Manager for both the DEFAULT and FAILOVER clusters, as well as the Server Manager for the APP-01 server.





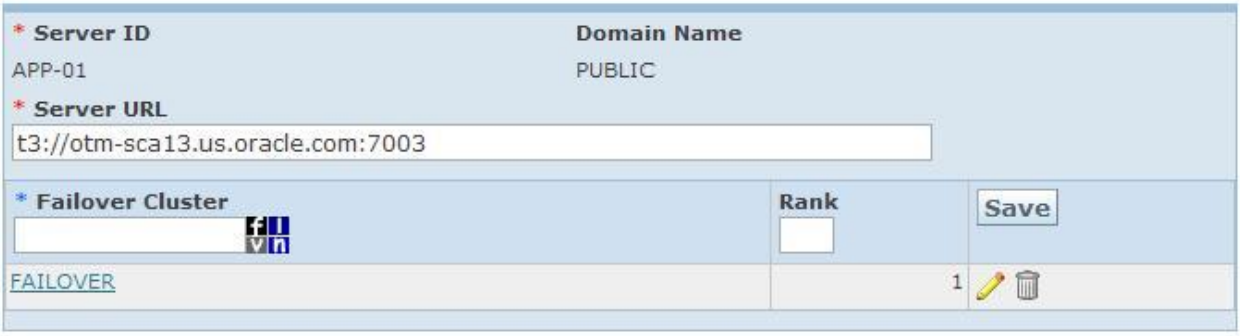


Figure 9: Failover Screen Setup

The following SQL statements perform steps 1-5 from outside the application:

```
insert into app_server (app_server_gid, app_server_xid, is_failover,
domain_name)
    values ('FAILOVER', 'FAILOVER', 'Y', 'PUBLIC');

delete from app_server_machine where app_server_gid='DEFAULT' and
app_machine_gid='APP-02';

insert into app_server_machine (app_server_gid, app_machine_gid,
load_balance_weight, domain_name)
    values ('FAILOVER', 'APP-02', 1, 'PUBLIC');

insert into app_machine_failover (app_machine_gid, failover_app_server_gid,
rank, domain_name)
    values ('APP-01', 'FAILOVER', 1, 'PUBLIC');
```

These SQL statements should be executed when the entire topology is down. Note when running these SQL statements instead of using the User Interface, the new topology will not be reflected until after a complete topology restart.

Database Records

Upon successful configuration of failover, the following data records should exist:

APP_SERVER				
APP_SERVER_GID	APP_SERVER_XID	DOMAIN_NAME	IS_FAILOVER	TRANSACTION_JMS_FLAG
DEFAULT	DEFAULT	PUBLIC	N	ALL
FAILOVER	FAILOVER	PUBLIC	Y	ALL

APP_SERVER_MACHINE			
APP_SERVER_GID	APP_MACHINE_GID	DOMAIN_NAME	LOAD_BALANCE_WEIGHT

APP_SERVER_MACHINE			
DEFAULT	APP-01	PUBLIC	1
FAILOVER	APP-02	PUBLIC	1

APP_MACHINE_FAILOVER			
APP_MACHINE_GID	FAILOVER_APP_SERVER_GID	RANK	DOMAIN_NAME
APP_01	FAILOVER	1	PUBLIC

Domain Separation

To delegate work for a particular domain BRANDA to a dedicated cluster holding server APP_02:

1. Create a new cluster, BRAND A, to handle the domain
2. Remove the APP_02 server from the DEFAULT cluster
3. Add the APP_02 server to the BRAND A cluster
4. Assign the domain BRANDA to the cluster

Figure 10 shows the resulting Cluster Manager for both the DEFAULT and BRAND A clusters.

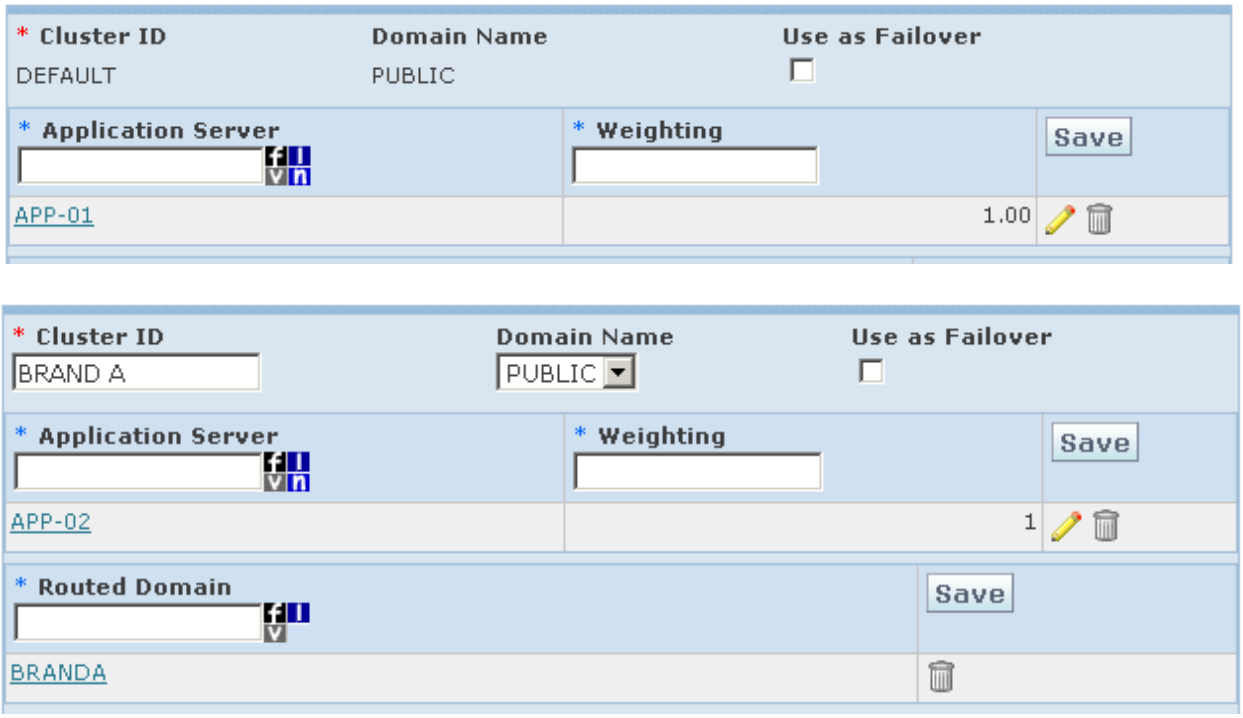


Figure 10: Domain Separation Screen Setup

The following SQL statements perform steps 1-4 from outside the application:

```
insert into app_server (app_server_gid, app_server_xid, is_failover,
domain_name)
    values ('BRAND_A', 'BRAND_A', 'N', 'PUBLIC');

delete from app_server_machine where app_server_gid='DEFAULT' and
app_machine_gid='APP-02';

insert into app_server_machine (app_server_gid, app_machine_gid,
load_balance_weight, domain_name)
    values ('BRAND_A', 'APP-02', 1, 'PUBLIC');

insert into app_server_domain (app_server_gid, routed_domain, include_children,
domain_name)
    values ('BRAND_A', 'BRANDA', 'Y', 'PUBLIC');
```

These SQL statements should be executed when the entire topology is down. Note that when running these SQL statements instead of using the user interface, the new topology will not be reflected until after a complete topology restart.

Database Records

Upon successful configuration of domain separation, the following data records should exist:

APP_SERVER				
APP_SERVER_GID	APP_SERVER_XID	DOMAIN_NAME	IS_FAILOVER	TRANSACTION_JMS_FLAG
DEFAULT	DEFAULT	PUBLIC	N	ALL
BRAND A	BRAND A	PUBLIC	N	ALL

APP_SERVER_MACHINE			
APP_SERVER_GID	APP_MACHINE_GID	DOMAIN_NAME	LOAD_BALANCE_WEIGHT
DEFAULT	APP-01	PUBLIC	1
BRAND A	APP-02	PUBLIC	1

APP_SERVER_DOMAIN			
APP_SERVER_GID	ROUTED_DOMAIN	INCLUDE_CHILDREN	DOMAIN_NAME
BRAND A	BRANDA	Y	PUBLIC

User Interface Query Delegation

To delegate queries for finders, managers and business monitors to a dedicated cluster holding server APP_02:

1. Create a new cluster, UI QUERIES, to handle the queries
2. Remove the APP-02 server from the DEFAULT cluster
3. Add the APP-02 server to the UI QUERIES cluster
4. Assign the function UI QUERIES to the UI QUERIES cluster
5. Assign the function BUSINESS MONITORS to the UI QUERIES cluster

Figure 11 shows the resulting Cluster Manager for both the DEFAULT and UI QUERIES clusters.

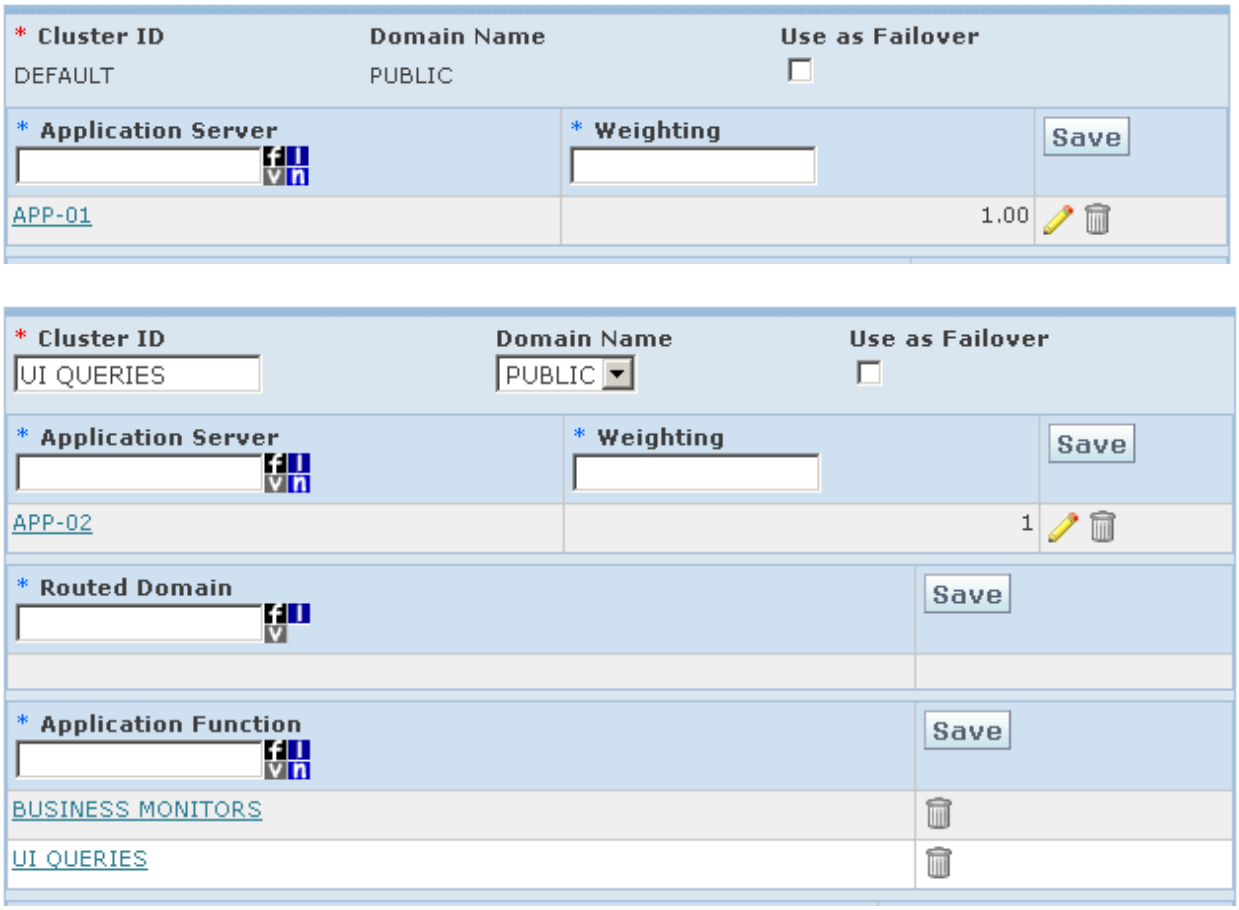


Figure 11: UI Queries Screen Setup

The following SQL statements perform steps 1-5 from outside the application:


```
insert into app_server (app_server_gid, app_server_xid, is_failover,
domain_name)
    values ('UI QUERIES', 'UI QUERIES', 'N', 'PUBLIC');

delete from app_server_machine where app_server_gid='DEFAULT' and
app_machine_gid='APP-02';

insert into app_server_machine (app_server_gid, app_machine_gid,
load_balance_weight, domain_name)
    values ('UI QUERIES', 'APP-02', 1, 'PUBLIC');

insert into app_server_function (app_server_gid, app_function_gid, domain_name)
    values ('UI QUERIES', 'UI QUERIES', 'PUBLIC');
insert into app_server_function (app_server_gid, app_function_gid, domain_name)
    values ('UI QUERIES', 'BUSINESS MONITORS', 'PUBLIC');
```

These SQL statements should be executed when the entire topology is down. Note that when running these SQL statements instead of using the user interface, the new topology will not be reflected until after a complete topology restart.

Database Records

Upon successful configuration of UI query delegation, the following data records should exist:

APP_SERVER				
APP_SERVER_GID	APP_SERVER_XID	DOMAIN_NAME	IS_FAILOVER	TRANSACTION_JMS_FLAG
DEFAULT	DEFAULT	PUBLIC	N	ALL
UI QUERIES	UI QUERIES	PUBLIC	N	ALL

APP_SERVER_MACHINE			
APP_SERVER_GID	APP_MACHINE_GID	DOMAIN_NAME	LOAD_BALANCE_WEIGHT
DEFAULT	APP_01	PUBLIC	1
UI QUERIES	APP_02	PUBLIC	1

APP_SERVER_FUNCTION		
APP_SERVER_GID	APP_FUNCTION_GID	DOMAIN_NAME
UI QUERIES	UI QUERIES	PUBLIC
UI QUERIES	BUSINESS MONITORS	PUBLIC

Bulkplan Delegation

To delegate bulkplans to a dedicated cluster holding server APP_02:

1. Create a new cluster, BULKPLANS, to handle the bulkplans
2. Remove the APP-02 server from the DEFAULT cluster
3. Add the APP-02 server to the BULKPLANS cluster
4. Assign the function BULK PLAN RELEASES TO BUY SHIPMENTS to the BULKPLANS cluster.

Figure 12 shows the resulting Cluster Manager for both the DEFAULT and BULKPLANS clusters.

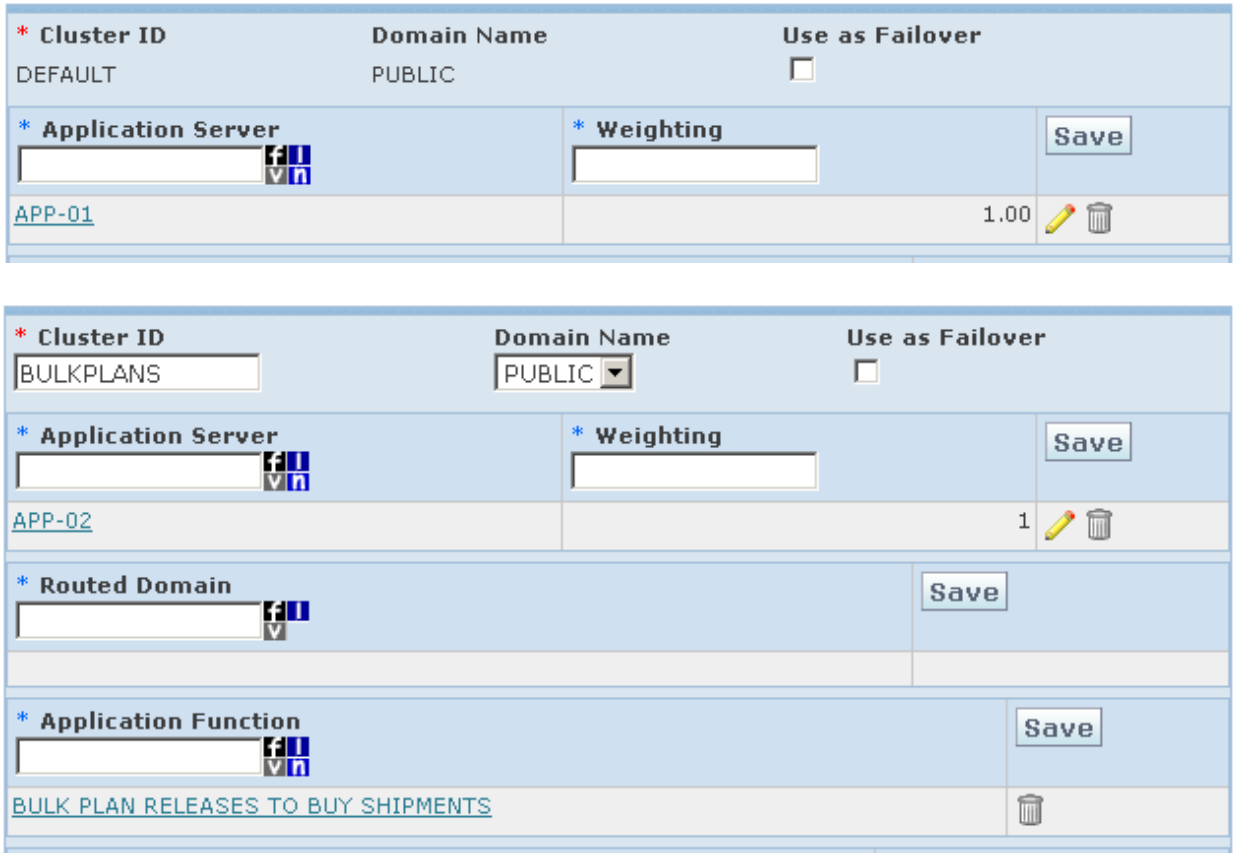


Figure 12: Bulkplan Screen Setup

The following SQL statements perform steps 1-4 from outside the application:


```
insert into app_server (app_server_gid, app_server_xid, is_failover,
domain_name)
    values ('BULKPLANS', 'BULKPLANS', 'N', 'PUBLIC');

delete from app_server_machine where app_server_gid='DEFAULT' and
app_machine_gid='APP-02';

insert into app_server_machine (app_server_gid, app_machine_gid,
load_balance_weight, domain_name)
    values ('BULKPLANS', 'APP-02', 1, 'PUBLIC');

insert into app_server_function (app_server_gid, app_function_gid, domain_name)
    values ('BULKPLANS', 'BULK PLAN RELEASES TO BUY SHIPMENTS', 'PUBLIC');
```

These SQL statements should be executed when the entire topology is down. Note that when running these SQL statements instead of using the user interface, the new topology will not be reflected until after a complete topology restart.

Database Records

Upon successful configuration of bulk plan delegation, the following data records should exist:

APP_SERVER				
APP_SERVER_GID	APP_SERVER_XID	DOMAIN_NAME	IS_FAILOVER	TRANSACTION_JMS_FLAG
DEFAULT	DEFAULT	PUBLIC	N	ALL
BULKPLANS	BULKPLANS	PUBLIC	N	ALL

APP_SERVER_MACHINE			
APP_SERVER_GID	APP_MACHINE_GID	DOMAIN_NAME	LOAD_BALANCE_WEIGHT
DEFAULT	APP_01	PUBLIC	1
BULKPLANS	APP_02	PUBLIC	1

APP_SERVER_FUNCTION		
APP_SERVER_GID	APP_FUNCTION_GID	DOMAIN_NAME
BULKPLANS	BULK PLAN RELEASES TO BUY SHIPMENTS	PUBLIC

9. Scalability Considerations

There are some general considerations which need to be reviewed before moving to Scalability.

The most important general consideration of Scalability, which needs to be enforced at all times is that all versions of the Oracle Transportation Management software, application server(s), and JVM need to match across the entire Scalability topology.

> **Warning**: Scalability does not provide the ability for a live Oracle Transportation Management upgrade. A live Oracle Transportation Management upgrade concept is not supported, and is not recommended.

If running a high availability configuration, servers in the failover cluster may not be sized the same as servers in the default cluster. With multiple servers in the default cluster, failover servers only receive a portion of the overall workload. Even with a two server setup, one default and one failover, the failover server is only active for the time it takes to repair and restart the default server. Reduced throughput due to failover may be acceptable in most circumstances. Sizing of failover servers should be determined by business functionality and expected hardware and software outages on the default server(s).

Active clusters (i.e. a cluster with the Use as Failover check box unchecked) do not support overlapping work. Scalability routing requires that a given work request has a unique cluster handling the work. If multiple clusters handle a particular function or domain, the router selects only one of the clusters – effectively at random. E.g., assume there is a default cluster with no explicit function or domain assignments. Another cluster is defined for failover but has the Use as Failover check box unchecked. The router views both clusters as available for work assignments but they overlap in work (both clusters handle all work). The router may select the failover cluster for a request rather than the default cluster.

Adding an additional application server may not increase performance as expected. The following overhead and bottlenecks should be monitored as an application server is added to the Scalability topology:

- **JMS Processing**: The additional server increases the amount of JMS messages each server receives. This may increase the CPU load on all servers. It may also increase the latency of JMS processing, increasing the chance of cache inconsistencies.
- **Contention**: The additional server increases the number of database record locks and contention on business objects. This may increase the frequency of timeouts across all servers.
- **Database Load**: Oracle Transportation Management is a query-intensive application. Adding an additional server may increase load on the database processes, eventually pegging the Oracle Database. Database resources must also be increased: connections, processes and statements all increase as another application server comes on line.

Be sure to monitor application server CPU usage and thread activity, as well as database performance, for each modification to the Scalability topology.

10. Scalability Recommendations

There are some recommendations which should be implemented when moving to a Scalability environment. These should definitely be done to avoid future potential issues.

Integration Transmission Redo Process

In any Oracle Transportation Management environment, there should be an integration transmission redo process implemented unless the upstream external application will resend integrations. This redo process is very important in a Scalability environment and even in a non-scalability environment.

By setting up this the Redo Transmission Processing, an Oracle Transportation Management client will be able to reprocess integrations that have been stored but are not processed yet. This process is required because these stored integrations will not be processed on a restart. This is even more important in Scalability because once integration gets routed to an application server it does not get re-routed to another application server in the cluster. Therefore, if one of the application servers has a failure, the other application servers in the cluster will not process these integrations. See the **Scalability Failure** section below for more information.

In order to schedule Redo Transmission Processing:

1. Go to Business Process Automation > Process Management > Redo Transmission Processing.
2. Select a Transmission Type of Inbound.
3. Set N as the Redo Number of Inbound Transmissions.
4. Schedule the process to run every T minutes. Typically, you can set N=50, T=5 minutes.

Thus, once you run the update command above, N uncompleted transmissions will be reprocessed on Application Server A within T minutes. N more will be reprocessed at each T minute interval. As long as T is not too frequent, the overhead of scanning for REDO transmissions is minimal: the status column is indexed and histogrammed so querying the status is optimized.

Object Lock Cleanup Process

In a Scalability environment, Oracle Transportation Management you can and eventually will accumulate millions of records in the OBJECT_LOCK table. A workflow process 'Object Lock Cleanup' is available to remove these old and un-owned object lock records. It is strongly recommended to implement this object lock clean up. By cleaning up these locks it will dramatically improve query and DML performance on the OBJECT_LOCK table. There is currently a default staged recurring process available to run this cleanup daily. The default recurring process removes un-owned records, older than 1 day. By default it runs every day at 0500 UMT. This default process can and should be modified to fit individual client timetables and business processes. If you do not currently have this default recurring process on your particular Oracle Transportation Management version, it is strongly advised to create your own recurring process to accomplish the object lock record clean up process.

> **Note:** It is strongly recommended the client does not adjust the day interval to a very high number for the recurring process.

Oracle Advanced Queuing and Data Queues Seamless Failover

When using Oracle Advanced Queuing in an Oracle Transportation Management Scalability environment, the application server listener properties must be set on the application cluster application machines to which the queue is assigned to, as well as the failover application cluster application machines. The failover application cluster does not need to have the Oracle Advanced Queue assigned to it. During a failover event, the failover cluster will process the same queues as long

as the application server listener properties are present. The format of property entry is:
`glog.oaq.integration.{the_topic_queue_name}=1`

In an Oracle Transportation Management Scalability environment where data queues are used, there is no additional configuration needed for seamless failover of Data Queues. During a failover event the data queues which were assigned to the cluster that failed will get processed by the failover application cluster application machines. The failover application cluster does not need the data queue assigned to it.

11. Scalability Failure

Object locks are now cleaned up when restarting an application server. When restarting an application server instance in the cluster the application server deletes any object locks associated with its app_machine_gid. Since this application server is just starting it should not be holding any object locks and these locks may prevent processes from running on another machine. The application server has the object locks, because it was doing processing on those particular objects during the failure. The application server owns these locks until it is restarted.

When an application server restarts it now:

1. Clears any OBJECT_LOCK records it owns
2. Notifies all other Scalability servers that the locks are now available

This allows any waiting servers to secure the locks and continue processing.

Integrations that were not processed are now updated when restarting an application server. The 'FRESH' transmissions that are not processed that were assigned to the failed application server are set to 'REDO' when:

1. That particular failed application server restarts or
2. A user submits a Report Server Failure process management job

There are a few common recommended procedures to follow during a Scalability application server failure. The type of failure is important, as there are different courses of action to follow depending on this type.

> **Note:** These do not cover client specific procedures.

False Failure Procedures

A false failure is when an application server in the Scalability topology becomes unresponsive. This unresponsiveness can be caused for many reasons.

The easiest solution would be to do nothing and let Scalability rebuild the topology. If the application server was only unresponsive, the keep alive ping will let the web servers know it is still alive and running. This will allow the application server to be put back into the topology map, and receive new work. This keep alive ping was recently added, so make sure your particular version of Oracle Transportation Management has this keep a live capability before relying on it. See the **Light Scalability Schema** in the **Reference** section on how to change the default value.

Another solution although not the fastest solution, would be to identify the falsely failed application server, and re-start it. This is not recommended since the keep alive ping will be able to rebuild the topology. The application server would then be put back into the Scalability topology and be routed work. However, the time to restart the application server varies between clients, hardware and several other factors. So this restart time may not be feasible or acceptable.

True Failure Procedures

A true failure is when an application server in the Scalability topology crashed.

The only solution is to reboot the application server that failed.

True Long Term Failure Procedures

A long term failure is when an application server in the Scalability topology will no longer be part of the topology because of hardware failure or some other reason for extended time down.

An Oracle Transportation Management client with a truly failed application server can have an administrator re-route integrations from that failed application server to another application server in the cluster. There is a process control topic called, Report Server Failure. This can be found under the Configuration and Administration > Process Management > Report Server Failure. This topic allows the user to select a Scalability application server and report its failure. This, in turn:

1. Sets any 'FRESH' transmissions owned by the application server to 'REDO'. The Redo Transmission Processing process running on other machines in the cluster will pick up these transmissions.
2. Clears any OBJECT_LOCK locks held by the failed machine. This also includes cleaning up any mediator/object locks.
3. Notifies all Scalability machines that object locks held by the failed machine are now available.

This allows for transmissions running on the failed machine to be processed by the active application server(s) in the cluster. This will also allow for transmissions blocked on the active machines to run to completion.

> **Note:** The topic XML for the "Report Server Failure" can also be sent in via integration. This would allow for an external monitoring system to send this XML automatically if it detected a JVM crash, an application server crash, or hardware failure.

12. Scalability FAQs and Potential Pitfalls

Scalability FAQs

The following are FAQs received from Oracle Transportation Management clients or consultants.

Q: In a Scalability configuration, when I choose to run a Process Request (Recurring Process available through the Process Management menus) and click the Run Now button, which Scalability cluster or server will that run on?
A: Scalability will route to the correct cluster for the process request. However, if the Oracle Transportation Management client has a complex Scalability configuration, they want to check to make sure the message does not come back to the original application server.

Q: I am trying to set up a Scalability cluster and I am little confused about what the Use as Failover check box does. Could you explain it please?
A: The Use as Failover check box specifies whether the cluster is dedicated to failover. By default, all Scalability clusters are active – they can have UI and agent work. To ensure that a failover cluster acts as a hot backup (i.e. receives no requests until the default server fails over), you must check the Use as Failover check box. Note that this does not complete the failover setup. The failover cluster must be assigned to an application server to act as its failover. See Section 9 for more information.

Q: Will there be any difference in a Scalability configuration for custom threads?
A: First, this would depend on the Scalability configuration. When first implementing Scalability in a complex configuration, it is not recommended to have differences in custom thread groups. However, over time the custom threads could be tuned if desired. The memory usage of idle threads is minuscule, and this may not be worth the hassle.

Q: Does special attention need to be taken when setting up the FICO Xpress Optimizer libraries for a Scalability environment?
A: No.

Q: What are all of the Oracle Transportation Management application functions that are available, and how do I know I selected the correct ones?
A: All of the Oracle Transportation Management application functions can be found by using the Application Function search page. The search page is located in the user interface here: **Configuration and Administration > Cluster Management > Application Functions**. All of the Oracle Transportation Management application functions are well-named, and the name explains what the Oracle Transportation Management application function does. See the discussion of **Allocation of Work** in the **What Is Scalability?** section for more information.

Q: If using Oracle Database RAC in a Scalability environment, does RAC require a different configuration?
A: No.

Q: In an Oracle Transportation Management high availability configuration, do all of the web servers take users to only the one application server in the default cluster? And if the application server fails, users will be down until we add the secondary application server to the default cluster? Is this correct?
A: If the Scalability configuration is setup correctly as high availability, then, yes all web servers will be routing (taking users) to only one application server (in the default cluster). If the application server fails, Scalability will automatically send business process requests from the users to the secondary application server (in the failover cluster). There is no need to add the secondary application server to the default cluster. Scalability is a hot-failover solution.

Q: In an Oracle Transportation Management high Scalability configuration, does either web server take users to either application server based on the weight that you configure. Is this correct?
A: Yes. Either web server will route (take users) to either app server based on the weight that is configured.

A few additional points on this though. The above statement of YES is only true if neither web server is configured to only handle lets say just integrations. Scalability is only an application server Scalability solution. We do not load balance on the web server(s). A third party product is required to handle web server load balancing.

> **Note**: The weight configured is used with randomization to determine which application server to route the business process request to. This means that the routing within a cluster cannot be viewed as truly "round-robin".

Q: What happens if a user is connected to the application server and it goes down? Does it transition automatically or will the user have to log of and log back in? In an Oracle Transportation Management high Scalability configuration, if one of the application servers is down for two days, will users be directed to it and error out or will they only be taken to the application server that is up and running?
A: This answer depends on the Scalability configuration. This question needs to be specific because it depends on the configuration and what exactly is the user doing. Depending on the configuration, what exactly the user is doing, and where in the business process the request is, the transition could be automatic or the user may have to log back in and re-run their business process. In a single application server cluster with high Scalability, if one of the application servers is down for two days, users will be directed only to the application server that is up and running.

Q: What is an Oracle Transportation Management no Scalability configuration? In my mind, this means that users will only be taken to the default application server by all the web servers? Is this correct? Is this similar to a high availability configuration?
A: Not sure what is meant by a no Scalability configuration. No Scalability means basic Oracle Transportation Management runs with one application server. This is what the client was doing on previous versions of Oracle Transportation Management before even considering scalability. Users would only be taken to that one application server.
This is not similar to a high availability configuration at all. With high availability, the passive (failover) cluster is a "hot" backup. It is live (hot, up and running) and processing work (updating caches) resulting from JMS broadcasts from the default cluster.

Q: Right now, we are only running on two web servers pointing to only on application server. Do we need Scalability? If we turn Scalability on, we would choose either a high availability or high Scalability configuration. I think going to a high availability active configuration will be the best for us. What would be the best configuration for us?
A: Does the client need Scalability? There are only a few reasons why a client should move to Scalability.

1. The Oracle Transportation Management client's application server is maxed out on memory, or the CPU of the server running the application server is always pegged to 100%.
2. The client wants a failover application server in place for a failover situation.
3. The client wants to offload Oracle Transportation Management domains or Oracle Transportation Management application functions like bulk plans to another application server.

So a single cluster with two active application servers may be the best solution for the client depending on what they are trying to accomplish with Scalability.

Q: Can you provide pros/cons for each of the basic Scalability configurations?
A: A complete list of pros and cons would be considerably lengthy. However, here are some of the biggest pros and cons of each of the basic configurations.

Scenario

In an Oracle Transportation Management high availability configuration, the failover cluster is "hot", and it does process all of the cache refreshes.

Pros: The failover is ready to take over right away when a failover situation occurs. This is an automatic failover.
Cons: The failover system is running and processing the cache refreshes for a failover event that may never occur. It may be better to use a single cluster with two active application servers.

Scenario

In an Oracle Transportation Management high Scalability (single cluster with two active application servers) configuration, both application servers are processing business requests with a randomized weight balancing.

Pros: Both application servers are processing business requests, and this can dramatically increase performance. This configuration can also help with maxed out memory, because now there are two JVMs (processes) which can use the memory available. If one of the application server machines in the cluster fails, then all requests will automatically be sent to the other application server machine.
Cons: There may be more contention in the application servers because both are processing requests. This can get even worse if both are processing on the same shipment or order.
If a failure event occurred on both application server machines in the cluster, then there would be no failure cluster to fail to.

There are many more pros and cons of each of these configurations, as well as, many more Scalability configurations.

Q: How do I add a third application server into the mix if it is determined that we need an extra application server in the future?
A: The answer depends on the type of configuration, and what the new configuration would be. The answer also depends on whether the application server would be added dynamically or not. The existing configuration would need to be known, and what the new configuration would be before giving any specifics. There would need to be additional properties added to all of the existing app/web properties files. Review section 7 for details on adding a new application server to a high availability environment.

Q: Can Scalability be used to do a live upgrade of the Oracle Transportation Management product?
A: No. Scalability cannot be used for a live Oracle Transportation Management upgrade. This will not work, and could cause disastrous results. Any attempt of using Scalability to do a live upgrade is not supported. The use of Scalability for a live upgrade is not recommended, and is very strongly discouraged.

Q: Why aren't my integrations being reprocessed by the other application server after the first application server fails?
A: Please follow the instructions located in the **Integration Transmission Redo Process** section of the **Scalability Recommendations** and the instructions in the **Scalability Failure** section.

Q: We get this exception in the exception log, "Application server is not ready to receive integration requests". What does this mean?
A: The Oracle Transportation Management integration has a special check to make sure the application server(s) do not accept transmissions until Startup is complete. The Scalability solution will not route transmissions to an application server until the correct port is open. Upstream systems should be paying attention both to HTTP 503 errors and transmission failures due to Oracle Transportation Management un-readiness.

Scalability Pitfalls

The following are common Scalability pitfalls and solutions which have happened to Oracle Transportation Management client(s), or that could happen.

- An Oracle Transportation Management client had URLs that did not match between the properties and the Scalability database tables. This can cause an exception similar to NO MACHINE IN THE DBA / (FUNCTION=NULL) CLUSTER IS RESPONDING. The properties settings for `glog.scalability.thisMachineURL`, `glog.scalability.defaultMachineURL`, and `glog.scalability.topologyMachineURL` need to match exactly to the correct MACHINE_URL in the APP_MACHINE database table.
- An Oracle Transportation Management client was missing the `glog.scalability.defaultMachine` property. When servers are too busy to respond, this can cause an inability to direct messages to the current server.
- An Oracle Transportation Management Client's properties files did not have the correct unique targets in them. The `glog.scalability.thisTarget` property needs to match exactly to the name (target) of the application server (not the DNS entry for the application server).
- An Oracle Transportation Management client's properties files did not have the correct settings for the `glog.scalability.thisMachine` and `glog.scalability.defaultServer` properties. The `glog.scalability.thisMachine` property needs to match exactly to the corresponding APP_MACHINE_GID. The `glog.scalability.defaultServer` property needs to match exactly to the default APP_SERVER_GID.
- An Oracle Transportation Management client had two active clusters. Both of these active clusters had no specific work assigned to them. This allows a possible overlap of cluster work that can confuse Scalability routing. This is not necessary in a high Scalability configuration. Each of the application servers in the default cluster will failover to the other servers in the cluster. Remove the second active cluster, or assign specific business functions to avoid any possible conflicts.
- An Oracle Transportation Management client thought they had a failover application server cluster. However, it was not truly a failover. The client forgot to check the Use as Failover check box.
- An Oracle Transportation Management client had more than two or three application servers and they did not change the correct settings for tomcat. They needed to change the settings for tomcat.
- An Oracle Transportation Management client did not have the same application server ports for both of the application servers in their topology. Different application server port numbers for multiple instances are perfectly alright, and are discretionary to the client. However, between the time of installation of both instances and the Scalability setup these differences were forgotten about and lost. The Scalability properties and database entries were then added using the wrong port. This effectively made one of the application servers unreachable within Oracle Transportation Management, even though that application server started and appeared to being running fine. However, no web requests or work requests were ever sent to that application server. The client should either change the properties files and database records to match the correct port, or change the port settings for the specific application server.

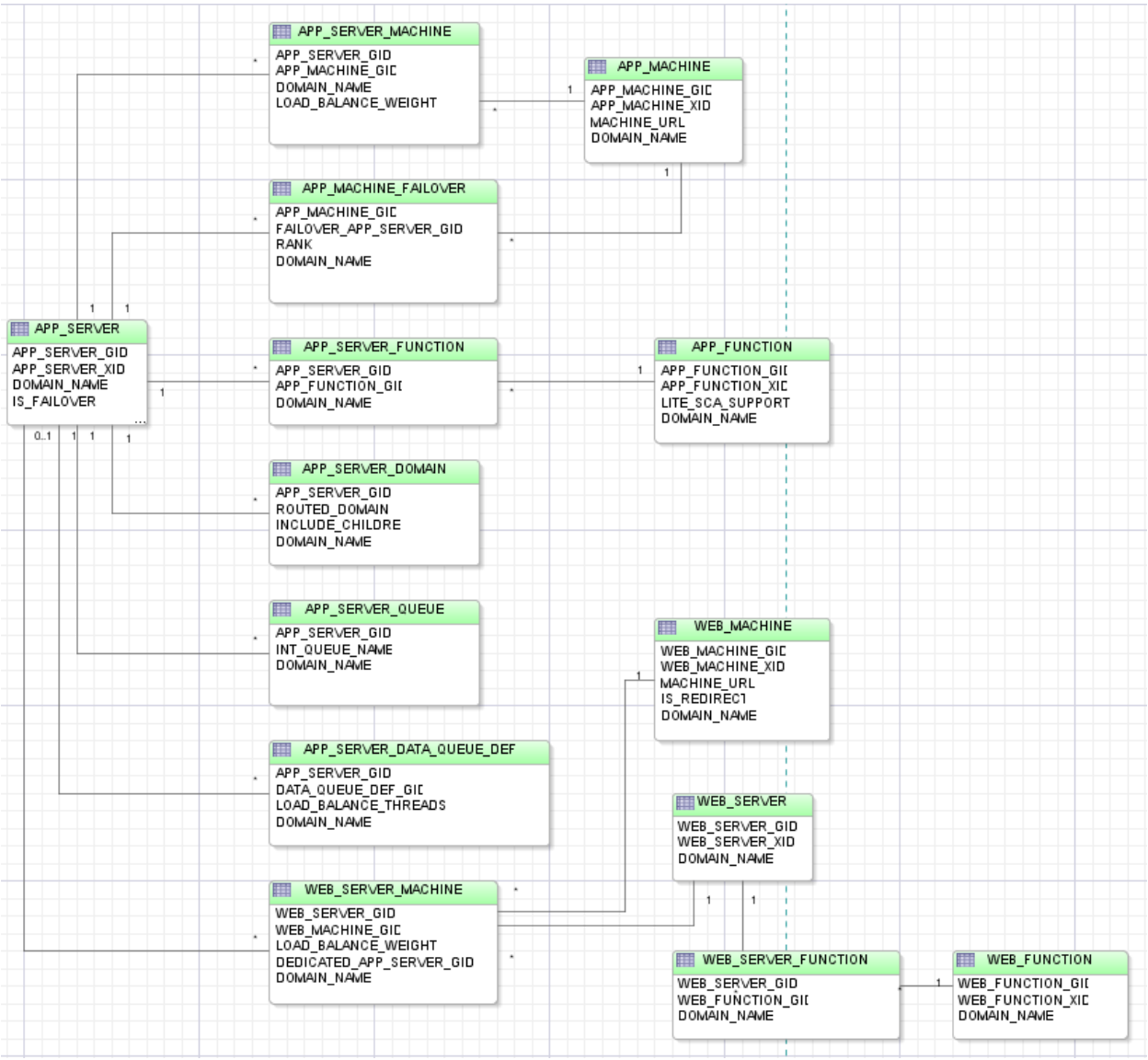
13. Reference

This section documents Scalability database tables and properties. It provides an understanding of what is involved and needed to successfully implement Scalability.

Scalability Database Tables

This section describes the various Scalability related database tables. The **Database Diagram 1** shows a hierarchal database diagram of most of the related Scalability tables.

- The APP_MACHINE table represents the application server.
- The APP_SERVER table represents the cluster.
- The APP_SERVER_MACHINE table provides the mapping of application servers to a cluster.
- The APP_MACHINE_FAILOVER table represents the mapping of an application server to failover clusters.
- The APP_SERVER_DOMAIN table allocates work from a domain to a specified cluster.
- The APP_SERVER_FUNCTION table allocates work from an Application Function to a specified cluster.
- The APP_SERVER_QUEUE table allocates work from an Oracle Queue to a specified cluster.
- The APP_SERVER_DATA_QUEUE_DEF table allocates work from an Oracle Transportation Manager Data Queue to a specified cluster.
- The APP_FUNCTION table holds assignable work types for application-tier scalability.
- The WEB_MACHINE table represents the web server.
- The WEB_SERVER table represents the web cluster.
- The WEB_SERVER_MACHINE table provides the mapping of web servers to a web cluster.
- The WEB_FUNCTION table holds assignable work types for web-tier scalability.



Database Diagram 1: Scalability database tables

Database tables 1-8 provide more details about individual columns.

APP_MACHINE		**Application machine table. Represents the actual application server.**					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
APP_MACHINE_GID	VARCHAR2(101)	X	X				Application machine Global ID.
APP_MACHINE_XID	VARCHAR2(50)		X				Application machine ID.

APP_MACHINE		Application machine table. Represents the actual application server.					
MACHINE_URL	VARCHAR2(512)		X				The URL of the Application Machine.

APP_SERVER_MACHINE		Application Server machine table. The table provides the link between the Application Server Cluster and the Application Server Machine.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
APP_SERVER_GID	VARCHAR2(101)	X	X			APP_SERVER	Application Server cluster Global ID.
APP_MACHINE_GID	VARCHAR2(101)	X	X			APP_MACHINE	Application machine Global ID.
LOAD_BALANCE_WEIGHT	NUMBER(4)		X				The number used for assigning weights to handle request balancing.

APP_SERVER		Application servers table. This table defines the Application Server Cluster.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
APP_SERVER_GID	VARCHAR2(101)	X	X				Application Server cluster Global ID.
APP_SERVER_XID	VARCHAR2(50)		X				Application Server cluster ID.
IS_FAILOVER	VARCHAR2(1)		X	Y/N	'N'		Flag to specify if this Application Server is a failover.

APP_MACHINE_FAILOVER		Application Machine Failover table. This table defines the Application Server Cluster for an Application Server Machine.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
APP_MACHINE_GID	VARCHAR2(101)	X	X			APP_MACHINE	Application machine ID.
FAILOVER_APP_SERVER_GID	VARCHAR2(101)	X	X			APP_SERVER	The Application Server cluster Global ID used for Failover for this Application Machine.
RANK	NUMBER(2)		X				The number used for assigning rank to

APP_SERVER_FUNCTION		Allows the execution of Application Functionality on one Application Server (Cluster) in the whole topology.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
APP_SERVER_GID	VARCHAR2(101)	X	X			APP_SERVER	Application Server (Cluster) ID
APP_FUNCTION_GID	VARCHAR2(101)	X	X			APP_FUNCTION	The Application Function to execute on this Application Server (Cluster).

APP_SERVER_DOMAIN		Allows the execution of all functionality specifically for an Oracle Transportation Management Domain on one Application Server Cluster.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**

APP_SERVER_DOMAIN		Allows the execution of all functionality specifically for an Oracle Transportation Management Domain on one Application Server Cluster.					
APP_SERVER_GID	VARCHAR2(101)	X	X			APP_SERVER	Application Server (Cluster) ID
ROUTED_DOMAIN	VARCHAR2(50)	X	X			DOMAIN	The Domain to execute on this Application Server (Cluster).
INCLUDE_CHILDREN	VARCHAR2(1)		X	Y/N			A flag to indicate whether Child Domains should be included to run in this Application Server (Cluster).

APP_SERVER_QUEUE		Maps OAQ queues to application servers clusters.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
APP_SERVER_GID	VARCHAR2(101)	X	X			APP_SERVER	Application Server Cluster ID
INT_QUEUE_NAME	VARCHAR2(128)	X	X				Oracle Advanced Queue name.

APP_SERVER_DATA_QUEUE_DEF		Maps Oracle Transportation Management data queues to application servers clusters.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
APP_SERVER_GID	VARCHAR2(101)	X	X			APP_SERVER	Application Server Cluster ID
DATA_QUEUE_DEF_GID	VARCHAR2(101)	X	X			DATA_QUEUE_DEF	Oracle Transportation Management Data Queue Definition

WEB_MACHINE		Web machine table. Represents the actual web server.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
WEB_MACHINE_GID	VARCHAR2(101)	X	X				Web machine Global ID.
WEB_MACHINE_XID	VARCHAR2(50)		X				Web machine ID.
MACHINE_URL	VARCHAR2(512)		X				The URL of the Web Machine.
IS_REDIRECT	VARCHAR2(1)		X	Y,N	N		If N, requests should be piped to this machine; otherwise uses browser redirection.[7]

WEB_SERVER_MACHINE		Web Server machine table. The table provides the link between the Web Cluster and the Web Machine.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
WEB_SERVER_GID	VARCHAR2(101)	X	X			WEB_SERVER	Application Server cluster Global ID.
WEB_MACHINE_GID	VARCHAR2(101)	X	X			WEB_MACHINE	Application machine Global ID.
LOAD_BALANCE_WEIGHT	NUMBER(4)		X				The number used for assigning weights to handle request balancing.

[7] Browser redirection is not supported in 6.1.

WEB_SERVER_MACHINE		Web Server machine table. The table provides the link between the Web Cluster and the Web Machine.					
DEDICATED_APP_SERVER_GID	VARCHAR2(101)					APP_SERVER	Forces all application-tier requests from this work to a single app-tier cluster.

WEB_SERVER		Web servers table. This table defines the Web Server Cluster.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
WEB_SERVER_GID	VARCHAR2(101)	X	X				Web Server cluster Global ID.
WEB_SERVER_XID	VARCHAR2(50)		X				Web Server cluster ID.

WEB_SERVER_FUNCTION		Allows the execution of Web Functionality on one Web Server (Cluster) in the whole topology.					
Column Name	**Type**	**Pk**	**Nn**	**Ck**	**Default**	**Ref. Table**	**Col. Comment**
WEB_SERVER_GID	VARCHAR2(101)	X	X			WEB_SERVER	Web Server (Cluster) ID
WEB_FUNCTION_GID	VARCHAR2(101)	X	X			WEB_FUNCTION	The Web Function to execute on this Web Server (Cluster).

Light Scalability Schema

Figure 13 shows the schema is used by application functions to specify their Light Scalability support:

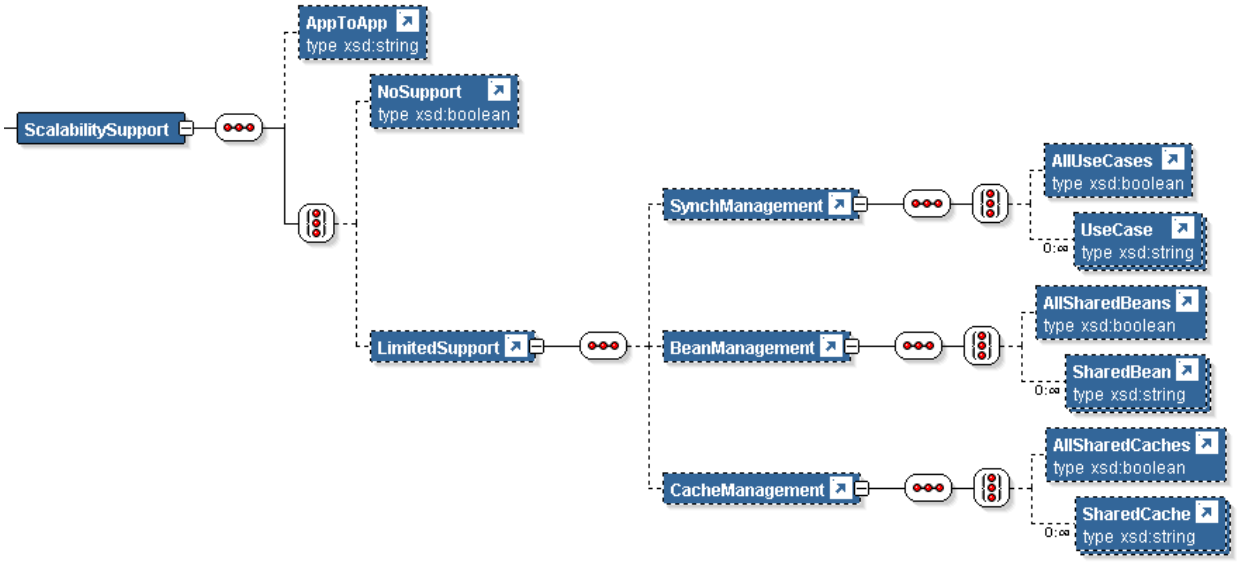


Figure 13: Light Scalability Schema

Synchronization use cases include:

- Audit
- BusinessNumbers
- DataQueue
- Diagnostics
- ObjectLocks
- ProcessControl

Bean and cache messaging support are proprietary to Oracle Transportation Management.

Note that all application servers support the following updates and locks, regardless of the dedicated application function:

- Scalability topology updates
- Log setting updates
- Security updates

Scalability Properties

These Scalability properties need to be set correctly so Scalability will work. There are ten basic properties. Most of the basic properties are needed on both the application server and web server.

Scalability Properties
`glog.scalability.on` The glog.scalability.on property turns scalability on or off. **Usage**: `glog.scalability.on=[true | false]`

Scalability Properties

glog.log.ID.JMS.on

The glog.log.ID.JMS.on property turns the JMS logging on or off.

Usage: glog.log.ID.JMS.on=[true | false]

glog.log.ID.Scalability.on

The glog.log.ID.Scalability.on property turns the Scalability logging on or off.

Usage: glog.log.ID.Scalability.on=[true | false]

glog.scalability.thisTarget

The glog.scalability.thisTarget property is the target name of the WebLogic Application Server that this property will be used on. This name of the application server must be unique.

Usage: glog.scalability.thisTarget=[exampleOTM-appServerName01]

glog.scalability.thisMachine

The glog.scalability.thisMachine property is the unique display name of this application server. This name must match the data in the APP_MACHINE_GID column of database table, APP_MACHINE. The following values are only examples.
Usage: glog.scalability.thisMachine=[DEFAULT | APP-01 | APP-02 | … | APP-0n]

glog.scalability.thisMachineURL

The glog.scalability.thisMachineURL property is the URL of this current application server. This URL needs to match exactly the URL data that is in the MACHINE_URL column of the database table of APP_MACHINE.

WebLogic Usage: glog.scalability.thisMachineURL=t3://appServerName01.domain.com:7001

glog.scalability.defaultServer

The glog.scalability.defaultServer property is the ID of the default cluster. This ID must match the data in the APP_SERVER_GID column of the database table, APP_SERVER.

Usage: glog.scalability.defaultServer =[DEFAULT | FAILOVER | BULKPLANS | …]

glog.scalability.defaultMachine

The glog.scalability.defaultMachine property is the ID of the default Machine. For an application instance, this ID is simply the application machine ID for the current server. For a web server instance, the ID should be set to one of the application servers in the default cluster. It must match the data in the APP_MACHINE_GID column of the database table, APP_MACHINE.

Usage: glog.scalability.defaultMachine =[DEFAULT | SECONDARY | …]

Scalability Properties

glog.scalability.defaultMachineURL

The glog.scalability.defaultMachineURL property is the URL for an application server in the default cluster. This URL needs to match exactly the URL data that is in the MACHINE_URL column of the database table of APP_MACHINE.

WebLogic Usage: glog.scalability.defaultMachineURL
=t3://appServerName01.domain.com:7001

glog.scalability.topologyMachineURL

The following is the list of properties for all of the available application server(s). These are only used by the web server(s) to poll for network topology. Remember to put only one of these properties per line.

WebLogic Usage:

```
!remove glog.scalability.topologyMachineURL
glog.scalability.topologyMachineURL=t3://appServerName01.domain.com:7001
glog.scalability.topologyMachineURL=t3://appServerName02.domain.com:7001
```

glog.scalability.topologyWebserverURL

The following is a list of properties for all of the available web server(s). There are only used by the application server(s) and are not needed in the properties on the web server(s). Remember only one of these properties per line.

```
!remove glog.scalability.topologyWebserverURL
glog.scalability.topologyWebserverURL=http://webServer01.domain.com:80
glog.scalability.topologyWebserverURL=http://webServer02.domain.com:80
```

If Web Scalability is enabled, these properties are deprecated. Please see **glog.scalability.web.topologyMachineURL** for more information.

The following properties are only needed if configuring web clusters under Web Scalability.

Web Scalability Properties

glog.scalability.web.on

The glog.scalability.web.on property turns web scalability on or off.

Usage: glog.scalability.web.on=[true | false]

Web Scalability Properties

glog.scalability.web.thisMachine

The glog.scalability.web.thisMachine property is the unique display name of this web server. This name must match the data in the WEB_MACHINE_GID column of database table, WEB_MACHINE. The following values are only examples.

Usage: glog.scalability.web.thisMachine=[DEFAULT | WEB-01 | WEB-02 | … | WEB-0n]

glog.scalability.web.thisMachineURL

The glog.scalability.web.thisMachineURL property is the URL of this current web server. This URL needs to match exactly the URL data that is in the MACHINE_URL column of the database table of WEB_MACHINE.

Usage: glog.scalability.web.thisMachineURL=http://webServerName01.domain.com:80

glog.scalability.web.defaultServer

The glog.scalability.web.defaultServer property is the ID of the default web cluster. This ID must match the data in the WEB_SERVER_GID column of the database table, WEB_SERVER.

Usage: glog.scalability.web.defaultServer =[DEFAULT | …]

glog.scalability.web.defaultMachineURL

The glog.scalability.web.defaultMachineURL property is the URL for a web server in the default web cluster. This URL needs to match exactly the URL data that is in the MACHINE_URL column of the database table of WEB_MACHINE.

Usage: glog.scalability.web.defaultMachineURL =http://appServerName01.domain.com:80

glog.scalability.web.topologyMachineURL

The following is the list of properties for all of the available web server(s). These replace the glog.scalability.topologyWebserverURL properties used in standard scalability solutions. Remember to put only one of these properties per line.

Usage:

```
!remove glog.scalability.topologyWebserverURL
!remove glog.scalability.web.topologyMachineURL
glog.scalability.web.topologyMachineURL=http://webServer01.domain.com:80
glog.scalability.web.topologyMachineURL=t3://webServer02.domain.com:80
```

The following properties are used to control the collection associated JMS data, which is displayed in the Message Diagnostic Servlet. These are all true by code default, and do not reside in a properties file by default. These are just here for reference purposes.

JMS Data Collection Properties

JMS Data Collection Properties
glog.scalability.topic.trackUpdates glog.scalability.topic.trackUpdates=[true | false] If true, track general message statistics
glog.scalability.topic.trackLatency glog.scalability.topic.trackLatency=[true | false] If true, track message latency
glog.scalability.beanUpdate.trackUpdates glog.scalability.beanUpdate.trackUpdates=[true | false] If true, track statistics specific to BeanUpdateTopic
glog.scalability.cacheRefresh.trackUpdates glog.scalability.cacheRefresh.trackUpdates=[true | false] If true, track statistics specific to CacheRefreshTopic
glog.scalability.queryUpdate.trackUpdates glog.scalability.queryUpdate.trackUpdates=[true | false] If true, track statistics specific to QueryUpdateTopic

Advanced Scalability Properties

These are a set of advanced scalability properties. These should be modified with extreme caution, and with a lot of testing before moving into a production environment.

Advanced Scalability Properties
glog.scalability.activatePortOffset The glog.scalability.activatePortOffset property is port offset from the RMI port of the application server that should be used in any Oracle Transportation Management environment for determining if Oracle Transportation Management is fully initialized and activated. This property should be used particularly for web service integration requests. A third party application should ping the (RMI port + this property offset value) port to determine when the server is ready for integrations. This property defaults to 100. The creation of this ready socket can be suppressed by setting glog.scalability.activatePortOffset to 0. **Usage**: glog.scalability.activatePortOffset=100

Advanced Scalability Properties

`glog.scalability.invokablePortOffset`

A keep-alive ping exists for application servers to account for missed JMS synchronization on startup. The keep alive relies on an invocation port ping on each application server. This port is set up after all other startup classes have activated (including JMS messaging). The port is controlled by this property, which is the number offset from the primary WebLogic port (e.g. 7001). It defaults to 200.

At each keep-alive ping, the application server invokes all other pingable application servers to:

1. Make sure the source server is in the scalability map.

2. Make sure JMS connections exist between the source and destination servers..

Usage: `glog.scalability.invokablePortOffset=200`

`glog.scalability.keepAliveInterval`

The glog.scalability.keepAliveInterval property controls the number of seconds between the keep alive pings from an application server to the web servers. This property is helpful when a web server thinks an application server is down due to unresponsiveness. This ping will re-establish the connection between the application server and web servers. The default is 300 seconds (5 minutes).

`Usage: glog.scalability.keepAliveInterval=300`

glog.scalability.cachedQuery

The glog.scalability.cachedQuery is a set of properties for the user interface queries that are cached for drop lists. This list is used within Scalability to determine which queries need a message when the cache needs to refreshed. It is not recommended for a client to change this property at all.

Usage: `glog.scalability.cachedQuery=`

glog.scalability.restrictToServers

The glog.scalability.restrictToServers property controls which application clusters will be used in the scalability topology. This property will restrict scalability the clusters it will use. This property is not recommended for a client production environment, but could be useful in a testing environment.

Usage: `glog.scalability.restrictToServers=DEFAULT`

`glog.scalability.restrictToMachines`

The glog.scalability.restrictToMachines property controls which application servers will be used in the scalability topology. This property will restrict Scalability the application servers it will use. This property is not recommended for a client production environment, but could be useful in a testing environment.

Usage: `glog.scalability.restrictToMachines=DEFAULT`

Advanced Scalability Properties
`glog.objectLock.delayForScalability` The glog.objectLock.delayForScalability property controls the number of milliseconds a scalability application server should wait before trying to obtaining an object lock. This time delay will give other application servers a greater chance of obtaining the object lock. By default, a scalability application server releasing a lock will naturally favor any lock waiters on the same JVM. This occurs because the object lock release operation notifies the local mutex, updates the database and sends out the JMS notification. It is extremely likely that any local lock waiters will wake up and obtain the lock before any of the other Scalability application servers get a chance. **Usage**: `glog.scalability.delayForScalability=2000`
`glog.scalability.lite.on` This property suppresses light scalability support. All application servers will require JMS updates and object synchronization. **Usage**: `glog.scalability.lite.on=false`

The following properties have been depreciated in Oracle Transportation Management 5.5 and no longer have any effectiveness. These should be ignored.

```
glog.scalability.mediatorTimeout.duration
glog.scalability.mediatorTimeout.pass
```

Scalability Network Topology Monitoring

Scalability provides the ability to monitor application servers in the network topology and raise automated lifetime events. These Scalability network topology changes events can be used to notify interested parties by e-mail, fax, or by the message center. There are only currently two events and they are raised when an application server is not responding or when an application machine is restarted.

- **MACHINE – NOT RESPONDING**: This lifetime event is raised when a web server gets a connect exception while communicating with an application server. This event may be raised more than once when an application server goes down, and as each web server detects the failure. Please note that this event requires at least one application server to still be running to send out any notifications.
- **MACHINE – RESTARTED**: This lifetime event is raised when an application server restarts.

Please note that since these special lifetime events always run in a DBA.ADMIN context, the registration must be in the PUBLIC domain.

Scalability vs. WebLogic Clustering

Since Scalability is a proprietary solution for Oracle Transportation Management, it is different from WebLogic clustering. The main reason for Oracle Transportation Management needing its own custom solution is to have control over the business and domain routing capability. It was also done to try to enable Oracle Transportation Management to be application server independent. By using and implementing WebLogic clustering, Oracle Transportation Management would then be tied to using that specific application server. Although this document will not discuss all of the differences between Scalability and WebLogic clustering it will point out a few important and interesting differences.

WebLogic clustering is a proprietary solution written by BEA for WebLogic to scale an application. WebLogic allows for replication and failover objects such as EJB and RMI objects to be shared between the application servers, but does not allow for Oracle Transportation Management specific objects to be shared without some major code changes. This could cause a large problem in Oracle Transportation Management, since Oracle Transportation Management makes use of specific business object caching. Scalability provides the mechanism to replicate Oracle Transportation Management business object data across application servers. During a WebLogic failover, the application server may retrieve the object from the database, and then start to reprocess the request. During a failover situation in Scalability, the business object and data would already be present, and would start reprocessing the request right away. Scalability does support dynamic additions of application servers like WebLogic. However, before the next time the application server and web server are restarted, the Scalability properties configuration will need to be updated so all application servers will be initially aware of every other application server involved in the application cluster.

