Oracle Enterprise Manager
Oracle Fusion Middleware Management

WebLogic Server Management Lab

Session S318968
Oracle Enterprise Manager 11g
WebLogic Server Management Pack Enterprise Edition
Hands-on Lab

Introduction to Enterprise Manager 11g

Oracle Enterprise Manager 11g is the centerpiece of Oracle's integrated IT management strategy, which rejects the notion of management as an after-thought. At Oracle, we design manageability into each product from the start, enabling Oracle Enterprise Manager to then serve as the integrator of manageability across the entire stack encompassing Oracle and non-Oracle technologies. Fueled by this unique vision, Oracle Enterprise Manager 11g has introduced business-driven IT management to help IT deliver greater business value through three highly differentiated capabilities:

- **Business-driven application management**, which combines industry-leading capabilities in real user experience management, business transaction management and business service management to improve application users' productivity while enhancing business transaction availability
- **Integrated application-to-disk management**, which provides deep management across the entire Oracle stack to reduce IT management complexity and eliminate disparate point tools
- **Integrated systems management and support**, which utilizes industry-first technology to bring support services into the IT management console; enabling proactive IT administration, increased application and system availability, and improved customer satisfaction

Oracle WebLogic Server Management Pack Enterprise Edition (EE)

Oracle WLS Management Pack EE provides increased visibility, production assurance, and significant reduction in cost and complexity while managing WLS environments, thereby increasing ROI on WLS investments. It provides application runtime visibility through composite application modeling and monitoring, JVM diagnostics, business transaction management and comprehensive service and infrastructure management including configuration and lifecycle management.

Oracle WLS Management Pack EE delivers comprehensive management capabilities, and makes it easy for the WLS administrators to manage WLS application environments. The pack improves visibility by discovering services as well as underlying Java artifacts, modeling functional and architectural relationships, as well as monitoring usage and performance metrics. Rich out of the box dependency analysis, performance monitoring, and production JVM diagnostics support is provided for Java EE applications and web services.

Further, the pack manages service levels, and provides an error alert management solution. Automated lifecycle management and configuration management capabilities for WebLogic and the underlying infrastructure help to implement Oracle best practices and to analyze changes across services and infrastructure. Java EE infrastructure components are grouped and managed in conjunction with existing components, with topology, service level and alerting functionality. A single centralized console can be used to analyze WLS domain groups, domains, clusters, servers, and the applications themselves throughout the infrastructure. Enterprise Manager provides a superior ownership experience with its unmatched management of Oracle products, as well as third party products.

This lab will demonstrate:

While the WebLogic Server Management Pack Enterprise Edition contains many features, all features of the pack are not included as exercises in this hands-on lab. This hands-on lab includes the exercises listed below. For information on features not covered in this hands-on lab (e.g. Business Transaction Management, configuration
management features from Application Configuration Console and Configuration Change Console, Service Level Management, Coherence Management), please refer to the SOA and Middleware Management page available on the Oracle Technology Network (OTN).

- Navigating the New User Interface for WebLogic Server Management
- Monitoring Multiple WebLogic Domains with WebLogic Domain Groups
- Tracking and Comparing WebLogic Server Configuration Changes
- Debugging a Hung Application Using Root Cause Analysis
- Analyzing JVM Memory Consumption with Heap Comparisons
- Proactively Analyzing Performance Trends and Delay Analysis (OPTIONAL – EXTRA CREDIT)
- Scaling Out Existing WebLogic Clusters (OPTIONAL – EXTRA CREDIT)

Please feel free to seek assistance from the instructor or Oracle Demo staff at any point in time.

Before you begin the exercises in this hands-on lab, please note the following:

- You will be given a virtual machine address to use for this lab. For ease of reference, you may want to write this below:

  Virtual Machine Address: __________________________________________________

- The URL to your Oracle Enterprise Manager Grid Control console is as follows, and should be accessed via the Microsoft Internet Explorer web browser:

  Grid Control URL: https://<Virtual Machine Address>:7799/em
  (e.g. https://ec2-111-222-333-444.compute-1.amazonaws.com:7799/em)

- You may connect to the console with the following Oracle Enterprise Manager administrator credentials:

  Username: sysman
  Password: welcome1
Navigating the New User Interface for WebLogic Server Management

Summary:
Administrators can perform tasks more efficiently through the new and improved interface for managing Oracle Fusion Middleware 11g and Oracle WebLogic Server.

Exercise Walkthrough:
1. After logging in to the Grid Control console, click on the ‘Targets’ tab. Then click the ‘Middleware’ subtab.

   The ‘Middleware’ subtab page identifies all the middleware – both Oracle as well as non-Oracle – that Grid Control centrally manages and monitors. Each WebLogic Domain that is added to Grid Control appears here and you gain a high-level overview of middleware software status and any potential problem areas with that software.

   **Click on the target name ‘/Farm01_medrec_1/medrec_1’ to be taken to the domain’s Home page.**

2. The WebLogic Domain Home page appears. Notice that the user interface has completely changed from prior Grid Control releases. The pages for managing Oracle Fusion Middleware 11g and Oracle WebLogic Server (releases 9.x and 10.x) are ADF-based now - rather than UIX based - and as such provide a simplified and much improved navigation model.

   The interface that appears may look familiar as it is similar to the interface of the Oracle Enterprise Manager Fusion Middleware Control which is bundled and installed with Oracle Fusion Middleware 11g. While the overall look-and-feel of the Grid Control Console and the Fusion Middleware Control Console are similar, the features that each exposes are very different. While Fusion Middleware Control provides real-time monitoring and administration capabilities for Oracle Fusion Middleware, Grid Control provides historical monitoring, alerting and notifications, configuration management, provisioning, service level management, and management support for other non-middleware software (such as databases, packaged applications, hardware, operating system, and third-party software).
On the left-hand side of the page is the target navigation pane displaying a tree of the managed servers in this domain and the applications deployed to them.

**Expand the tree to view all content in the target navigation pane.**

On the right-hand side is the content pane displaying information for the target selected from the left-hand side. The contents pane contains various regions – each with a variety of configuration, availability and performance data concerning the domain. These regions can be dragged and dropped around the page in order to customize the layout of the home page. Notice the ‘Summary’ region has a link to drill down into the WebLogic Server Administration Console, and if this were for instance a SOA Suite or WebCenter Suite domain, there would be a link to drill down to the Oracle Enterprise Manager Fusion Middleware Control Console as well.

Dynamic target menus are available from either the navigation pane or the content pane.

**From the navigation pane, you can right-mouse click on a target to access its menu. From the content pane, you can click on the black down arrow next to the target type icon and label.**

There is a general information icon next to the target type name. When accessed, additional information on the target is displayed; such as the target’s version number, installed location, agent used to monitor the target, and the underlying host machine on which it is installed.

3. **Select the ‘AdminServer’ target from the navigation pane to go to its home page.**
As with any home page, you can gauge the overall health of the managed server from this page. Take a moment to review the various regions’ content on the page. How is the managed server performing? Is there significant load? Are there any alerts?

**Access the target’s menu and peruse its various submenus and available capabilities.** Notice that to startup/shutdown/blackout the server, you must navigate to the target’s menu and drilldown into the ‘Control’ submenu.

4. Oracle Enterprise Manager Grid Control 11g Release 1 introduces two new target types for deployed Java EE applications. These new target types are application deployment and clustered application deployment. While an application deployment represents one application deployed to a single server, a clustered application deployment represents one application deployed across multiple servers that reside in a cluster.

By having application deployments as “first-class” targets, you gain a variety of features in Grid Control. For instance, these new target types have a status metric (up/down); can be started up/shutdown/placed under blackout; support metric thresholds and alerting, and support target privileges.

**Scroll down the home page and select the ‘medrec’ application deployment target from the server’s home page.**

Take a moment to review the various regions’ content on the application deployment home page.

**Access the target’s menu and peruse its various submenus and available capabilities.**
5. Feel free to peruse the domain, or another domain, a bit more to familiarize yourself with the new navigation model and layout.
Monitoring Multiple WebLogic Domains with Oracle WebLogic Domain Groups

Summary:

Administrators can simplify how they manage multiple WebLogic Domains by creating and using WebLogic Domain Groups within Grid Control. Rather than relying on several different WebLogic Server Administration Consoles and rather than navigating to various different domain’s home pages within the Grid Control Console, administrators can gain an overview of availability, performance, configuration, and potential problem areas across all domains from just a few pages – within just 1-3 clicks.

Exercise Walkthrough:
Oracle WebLogic Domain Group targets are not available out-of-box with the Grid Control installation. You must manually create Oracle WebLogic Domain Groups after you add Oracle WebLogic Domains to Grid Control. This exercise will guide you through how to create an Oracle WebLogic Domain Group, and what features are available after you do so.

1. Click on the ‘Targets’ tab. Then navigate to the ‘Groups’ subtab. From the ‘Create’ drop down list, select ‘Oracle WebLogic Domain Group’ and click the ‘Go’ button.

2. The ‘Create WebLogic Domain Group’ page appears prompting you to supply the group’s name as well as to add members to the group.

   First, supply a name for the group, such as ‘Production Domains’, and then click the ‘Add’ button to begin adding WebLogic Domain targets to the group.
3. From the 'Search and Select: Targets' page, select domains to add to the group. Multiple domains can be selected by holding the shift key as you select each domain. When all domains are selected, click the 'Select' button.

4. You are returned to the 'Create WebLogic Domain Group' page with the selected domains appearing in the 'Members' table.

   Click the 'Ok' button to complete creation of the Oracle WebLogic Domain Group.
The ‘Groups’ subtab is updated with the newly created Oracle WebLogic Domain Group.

5. **Click on the newly created Oracle WebLogic Domain Group’s name link to launch the home page for the group.**

The home page can be seen as a “dashboard” view into your domains. The pages available for the Oracle WebLogic Domain Group target type automatically refresh so you can be assured that the most up-to-date information is displayed. The upper region provides an overview of the availability of managed servers across all domains of the group and of potential problem areas via alerts, policy violations and configuration changes. The middle region has several performance metric charts indicating which five managed servers have for example the longest request processing time, most requests, and highest heap usage. The bottom portion of the page provides further details on each managed server.
6. From the ‘Oracle WebLogic Domain Group’ menu, select ‘Performance’.

The ‘Performance Summary’ page appears providing further insight into how the managed servers across the domains are performing. From this summary page, you can modify the time range to limit the amount of performance data displayed on the charts. In addition, as you hover over a performance chart, a tooltip appears providing more detail as well as a vertical line that spans across the charts so that you can easily correlate performance data across all metrics on the page.

7. From the ‘Oracle WebLogic Domain Group’ menu, select ‘Members’.

The ‘Members’ page appears providing additional details for the targets in the domain. From here you can perform process control, view the latest configuration for each target, or compare configurations of targets.
Tracking and Comparing WebLogic Server Configuration Changes

Summary:

Administrators can rely on the WebLogic Server Management Pack Enterprise Edition to perform complete configuration management for WebLogic Servers in their environment.

Grid Control Exercise Walkthrough:
1. On the WebLogic Domain Group’s ‘Members’ page, the last two columns in the table highlight any configuration changes that have occurred to the managed servers in the group over the past 24 hours or week. You can see that some configuration changes have occurred – which could have impacted application performance.

Click on a link in either ‘Configuration Changes’ columns for an Oracle WebLogic Server target type.

Note: If there are no configuration changes identified in the ‘Configuration Changes’ columns, then you still click on the ‘0’ link.

The ‘Configuration History’ page appears for this managed server and identifies all changes that have occurred within the selected time range. You can see that several changes have occurred – changes to web modules, EJB modules, applications, and the server itself.

Note: If there are no configuration changes identified, modify the default value for the ‘Change Discovered after’ property by specifying an earlier date (e.g. 5/20/2010).

So if application performance was fine yesterday but not today, then you can quickly and easily access the history of configuration changes for the WebLogic Servers supporting the application. If there are configuration changes, you can drilldown to further detail in order to understand whether the changes could have negatively impacted performance.
2. Click the web browser’s back button to return to the ‘Members’ page for the WebLogic Domain Group.

Another common use case for configuration management is being able to compare one or more WebLogic Servers. For instance, many times you think that the WebLogic Servers in your stage and production environments are configured identically, but when performance problems occur, you should first validate that the configurations are identical. The Grid Control Console enables you to easily perform such a comparison operation to quickly identify any differences between the managed servers that could have attributed to poor performance. Similarly, a comparison between a live, running WebLogic Server and a saved, ‘gold configuration’ can be done to ensure that a managed server still complies with the gold standard – that no configuration drift has occurred.

To perform a comparison between a running WebLogic Server and a saved gold configuration, select the managed server labeled ‘/Farm01_medrec_1/medrec_1/AdminServer’ in the ‘Members’ table. From the ‘Configuration’ menu above the table, select ‘Compare’.

3. From the ‘Compare Configurations: Second Configuration’ page, click the ‘Saved Configurations’ link to view all saved configurations that you can then choose from in order to perform a compare operation.
4. Select the saved comparison labeled ‘Gold Configuration for Production WebLogic Server’ and click the ‘Compare’ button. Navigate around the comparison results to identify the differences in the configurations.

Notice that you can also compare actual WebLogic Server configuration files (e.g. config.xml). This is helpful should the various properties that Grid Control collects and displays do not include a specific property that you want to analyze and compare.
Debugging a Hung Application Using Root Cause Analysis

Summary:

By using JVM diagnostics within Enterprise Manager Grid Control, administrators can quickly determine the root cause of a hung application in order to get it back up and running as quickly as possible.

Exercise Walkthrough:
1. Click the ‘Middleware’ subtab. Click on the Oracle WebLogic Server target name ‘/secFarm_GCDomain/GCDomain/EMGC_ADMINSERVER’ to be taken to the managed server’s Home page.

   ![Middleware Subtab](image)

   1. The WebLogic Server Home page appears. Looking at the server, we can see that only a single application is deployed on this particular server in addition to the agent deployments (jamagent_ADMINSERVER) which is jadetest. We can also see an alarming amount of alerts have appeared in red in the upper-left hand corner of the WLS Home page.
3. Let’s go ahead and determine which alerts have been kicked off and whether it is connected to the rather high amount of active sessions that are also seen on the home page.

In order to do this, please click on the actual alert number in red.

4. We see a number of alarming alerts. In particular, we can see that there are a large amount of active threads and active sessions indicating a likely application hang scenario.

In order to troubleshoot this further, please click on the active threads alert. If you do not see this specific alert on the Status Report – Critical Alerts page, then from the ‘Previous’/‘Next’ drop down list, select ‘Show All’ and do a find for “active threads” to locate the alert for active threads, and click on the alert’s message link.

We can see that we have had most of the active threads stay locked for a very long time. It is time to actually figure out what the cause of these locked threads is via JVM diagnostics.
5. Click the server breadcrumb link at the top to return back to '/secFarm_GCDomain/GCDomain/EMGC_ADMINSERVER' and from the “WebLogic Server” drop-down menu on the home page, please go to JVM Diagnostics and select “Summary” from the sub-menu to be brought to the JVM diagnostics summary page.

This is where you can also explore JDBC Correlation, compare JVM heap dumps, and many other types of JVM analysis. We will explore JVM heap dump analysis in an upcoming exercise.

6. The JVM Diagnostics Summary shows a shocking amount of overall locks. We can see that the checkout.jsp and additem.jsp are the URLs that are clearly being impacted by these locks as they are the only ones still listed in this real-time summary. Please note that you can expand the timeframe to look at the historical JVM metrics as well.
7. By switching to the Details tab and clicking on checkout.jsp link in the ‘Top Requests’ chart legend, we now have filtered everything by a single URL which may be important if you have multiple requests (particularly if you chose to expand the timeframe). Scrolling down shows that some SQL is being called that is clearly locking the underlying tables in order to attempt an order transaction and we have table lock contention which is blocking at least some if not all of our threads.

You can also click on the “DB” in the top table (or simply choose “DB Wait” from the state drop-down menu) to eliminate the rest of the locks and see how many are actually related to “DB Wait”.

This will highlight the fact that the DB Wait locks might be the fundamental root cause of the other locking threads, but in order to be sure, we need to narrow down which thread is blocking the others.

8. In order to determine whether the database locks are indeed the root cause of all of this thread locking, let’s drill into the actual details of the threads that are in a locked state. From the WebLogic Server drop-down menu, go to JVM Diagnostics, and choose Threads->Real-time Analysis to bring us into a real-time view of the behavior and state of the current threads.
We can see the list of threads and indeed they are all stuck on Cart.java which is part of our Checkout.jsp. This indeed is the root cause of all of our stuck threads. This is obvious simply from the timeframe of thread execution. You can see what is locking each thread. In addition, we can actually choose to trace a thread.

9. Let’s first determine which lock is holding the other threads. Please click on the thread name of one of the stuck threads. As we are particularly interested in the check out calls in our orders, let’s select one with a request value of /checkout.jsp. If we look in the lower left, we can see the details of the Thread Holding the lock.

10. At this point, we could find the thread via a few different methods, but let’s just go ahead and find the OS PID in the list to find the correct thread. This introduces you to the concept of the OS PID which as some of you might know, is completely unique in regards to the JVM environment. This unique identifier ensures that we can always track specific threads.

Once you find the correct thread, click on the “DB Wait” link in the State column which indicates that this is definitely a database lock as we first feared. You can quickly identify the SQL being blocked in the thread via selecting the associated “SQL Hash” link which is a numerical unique identifier.

At this point, we have the exact SQL being locked and we can either contact a DBA or if you are one of those with the proper privileges, you could use the Database Diagnostics and Tuning capabilities to identify the SQL from our application (which is also attempting to lock the table prior to completing the checkout and add item transactions) via the hash and either kill the session blocking your threads or identifying why the table is being locked in the first place as it may be locked on purpose for maintenance reasons or perhaps a general update to the table is being performed externally.

With that, we will conclude this exercise. Please feel free to explore other applications and domains in general. We will look at another mechanism for navigating into JVM Diagnostics in a more general manner in a subsequent exercise which will also explore using saved heap dumps.
Analyzing JVM Memory Consumption with Heap Comparisons

Summary:
By using JVM Diagnostics within Enterprise Manager Grid Control, administrators can easily determine the cause of JVM memory consumption in production, staging, testing, and development environments. They can easily compare those heap dump snapshots between multiple time periods to pinpoint the root cause of the memory consumption/leak so the problem can be quickly resolved.

Exercise Walkthrough:
1. Click on the 'Targets' tab. Then click the 'Middleware' sub-tab. Please go to the bottom of the list of middleware servers to where Related Links are listed. This will allow you to bring up JVM Diagnostics in a more general manner that is not specific to one domain and can provide a broader perspective.

2. You now have visibility into all of the JVM pools being monitored by Enterprise Manager’s JVM Diagnostics. By default, there is no context so there is no data to be displayed. To view data, you can click on various JVM pools to provide you with a view into the behavior of each. Please go ahead and explore to better understand this visibility noting that the drop-down menu can take you into the summary/details and real-time thread metrics for each of these JVM pools as you explored in the previous exercise.

3. Let’s now go ahead and look at how we can take a heap snapshot.

Please choose “/secFarm_GCDomain/GCDomain/EMGC_ADMINSERVER” from the list of JVM pools. From the JVM drop-down menu choose Heap and “Take Snapshot Now” from the sub-menu.

It is easy as that to take a heap snapshot for analysis which is then put into a text file which can be imported into any Enterprise Manager JVM Diagnostics utility. You will see a message telling you where the snapshot was taken.
4. When you are ready to continue, please go to the top of the tree of thread pools and select “All” to get the widest view of all JVM thread pools.

Please note how you can see even the JVM information with the DB locks that we previously explored from this view.

5. Let’s now go ahead and compare to saved heap snapshots.

We can do this by going back to the Pool menu and selecting “Saved Heap Snapshots” from the Heap dropdown.

This is where all of your heap dump snapshots will be saved and made available after they are imported into Enterprise Manager.
6. We see two saved heap snapshots taken from May 24th on the JVM pool that we had examined earlier. We can see that one has a higher memory footprint than the other in regards to percentage of consumed heap memory. Each has a heap size of 1020 and we can see the actual MB that was consumed as well.

<table>
<thead>
<tr>
<th>Available Heaps</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date</strong></td>
</tr>
<tr>
<td>24-MAY-10 08:18</td>
</tr>
<tr>
<td>24-MAY-10 08:11</td>
</tr>
</tbody>
</table>

7. Please click on the one that is marked as 37% as it is the larger of the two.

We can now see the heap usage in regards to which types of objects are consuming the memory.

<table>
<thead>
<tr>
<th>Heap Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object Type</strong></td>
</tr>
<tr>
<td>Array</td>
</tr>
<tr>
<td>Instance</td>
</tr>
<tr>
<td>Constant Pool</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>Constant Pool Cache</td>
</tr>
<tr>
<td>Symbol</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>Other</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>Array</td>
</tr>
<tr>
<td>Instance</td>
</tr>
<tr>
<td>Symbol</td>
</tr>
<tr>
<td>Other</td>
</tr>
<tr>
<td>Compiled IC Holder</td>
</tr>
<tr>
<td>Class</td>
</tr>
</tbody>
</table>

8. Let’s now go ahead and determine the exact objects responsible for the memory consumption.

**First click on “Roots” in the menu bar.**

From here, we can see that Dictionary is consuming the most memory of the objects, but what are the exact classes responsible for the heap consumption?
9. In order to find out, simply click on the Dictionary link to drill into the underlying objects within.

We can see that “jadetest/staticLeak” is referencing the majority of objects.

If you click the small expandable + node to the left of that particular object, the list will be filtered to just the ‘staticLeak’ object. Click the small expandable + node to the left of that object again to get a list of references.

Examining these, it is clear that our memory leak is being caused by references to a large amount of Vectors which in this case was designed on purpose to hold the reference and thus cause a programmatic memory leak.

10. Let’s now do one last thing and compare this heap dump to the one that was taken earlier.
We can do this simply by clicking “Compare Heaps” in the menu bar and selecting the other heap which was consuming 23% of the memory heap at a previous timeframe. Please click on the 23% on that line item to run the comparison.

<table>
<thead>
<tr>
<th>Date</th>
<th>JVM Name</th>
<th>Vendor</th>
<th>Version</th>
<th>Description</th>
<th>Size(MB)</th>
<th>Used(MB)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>24-MAY-10</td>
<td>secFarm_GDomain/GDomain/EMGC_ADMINSERVER</td>
<td>Sun Microsystems Inc.</td>
<td>1.6.0.14</td>
<td>1020</td>
<td>240</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>

11. We can now see a clear comparative summary in this case showing that more objects are clearly being referenced in the larger JVM heap.

By clicking “View Summary”, we can see the size of the jadetest/staticLeak references and the associated referenced objects being held in memory at the time of the heap snapshot.

This can easily be used to compare heap snapshots between timeframes in order to identify memory leaks and performance degradation over time. Please feel free to explore more of the JVM diagnostics if time permits.
<table>
<thead>
<tr>
<th>Type</th>
<th>Signature</th>
<th>Heap 1</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>[Ljava.lang.Objects</td>
<td>25025</td>
<td>39009</td>
</tr>
<tr>
<td>Instance</td>
<td>java.util.Arrays.ElemNet</td>
<td>5000</td>
<td>171</td>
</tr>
<tr>
<td>Instance</td>
<td>java.util.Arrays.ElemNet</td>
<td>4840</td>
<td>113</td>
</tr>
<tr>
<td>Instance</td>
<td>java.lang.StringBuffer</td>
<td>5843</td>
<td>79</td>
</tr>
<tr>
<td>Array</td>
<td>[Ljava.lang.Object[]</td>
<td>2195</td>
<td>1,610</td>
</tr>
<tr>
<td>Array</td>
<td>[C</td>
<td>21040</td>
<td>216,027</td>
</tr>
<tr>
<td>Array</td>
<td>[]</td>
<td>5566</td>
<td>1,541</td>
</tr>
<tr>
<td>class</td>
<td>weblogic.management.provider.Internal</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>class</td>
<td>weblogic.management.provider.Internal</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Instance</td>
<td>com/bea,orb/runtime/DependenciesProxyFactory</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Instance</td>
<td>com/bea,orb/runtime/DependencyBindingContext</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Instance</td>
<td>weblogic.descriptor/AbstractDescriptorManager</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Instance</td>
<td>weblogic.descriptor/AbstractDescriptorManager</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>class</td>
<td>weblogic.descriptor/AbstractDescriptorManager</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Instance</td>
<td>weblogic descriptor/AbstractDescriptorManager</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Instance</td>
<td>com/bea,orb/runtime/DependenciesProxyFactory</td>
<td>114</td>
<td>7</td>
</tr>
<tr>
<td>Instance</td>
<td>com/bea,orb/runtime/DependencyBindingContext</td>
<td>1302</td>
<td>31</td>
</tr>
<tr>
<td>Instance</td>
<td>com/bea,orb/common/ConcurrencyContainer</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Array</td>
<td>[]</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
Proactively Analyzing Performance Trends and Delay Analysis (Optional – Extra Credit)

Summary:

Using Enterprise Manager’s Application Dependency and Performance model-driven approach to performance monitoring empowers both operational and development IT staff to be able to quickly understand an application and perform root cause analysis. Users can drill-down through the various models that are automatically discovered and displayed by Enterprise Manager in order to ensure that they maintain context throughout the analysis of the metrics. Metrics are tied to visible components ensuring that they are understood and easily tied to the associated metadata. The final result is a quick and efficient root cause analysis effort that can get reduce downtime and resource drain on an IT organization for critical Java EE and SOA business applications.

Exercise Walkthrough:

1. Click on the “/Farm01_medrec_1/medrec_1/AdminServer” WebLogic Server from the Middleware subtab. Examine the WebLogic Server home page which illustrates basic performance metrics in both the graphs and tables. We can see the general health and metrics associated with each of the various application deployments. Take a look at some of the overall basic metrics of the server including the heap usage, availability and the overall general performance of the ear files and JSP/Servlet entry points. We will be going deeper into some of those entry points momentarily to look at the overall performance trends and the relationships/dependencies between the components that underlay the JSP/Servlet entry points that you are seeing.

2. Click on the “Physician” link in the Application Deployments table or by expanding the Application Deployment node in the left-hand navigation tree in order to view the overall performance trends and health of the Physician application.
   a. You can now see more specific metrics and basic statistics for the physician (ear) application. You can analyze the various JSP/Serlet metrics as well as other details associated with the response time, load, and EJB calls in the tables and graphs.
b. At this point, we can drill down to Application Dependency and Performance in the context of the ear
file or a specific JSP page. The screenshot below shows where both of these contextual drill-downs
are located with the drop-down menu being for the entire physician application (similar to the
“Dependency” link next to the Physician deployment in the table on the WebLogic Server home page)
or Servlets and JSP table where you can get a more specific JSP page.

![Image of drill-down menu and metrics]

3. You can try out both drill-downs, but for now select the “Application Dependency and Performance” link in
the drop-down menu which will take us to the more detailed metrics for the Physician application
deployment on medrec_1.

So let’s examine what type of performance metrics are now available to us at the component level on some of
these tabs available to us providing details on physician.

a. First, we have the summary tab which provides us with high-level information similar to what we saw
before, but also with max/mins and we now can easily change the time period via the top menu bar
to look at any time period we want whether it be the previous 8 hours or it be a certain time period
the week earlier.
b. Switching to the Response Times, we now can see more specific response times for both the JSP/Servlets in regards to the underlying implemented classes and the EJBs as well being called within the ear file.

c. Let’s take a look at the details on two more of the tabs, the rest can be explored as desired on your own.

This time click the Invocations tab which is clearly for the same objects as identified above but broken out in their own charts.
d. Finally, let’s take a look at the Topology tab which will provide us with a bird’s eye view of the actual topology of our application.

We can see that there is in this case two different physician ear files interoperating via RMI calls and that relationship is shown here as well.

A quick double-click on the physician ear file color-coded in white which is the one within medrec_1 will take us deeper into those relationships as shown below where we can see the relationships and invocations.

4. At this point, we have seen most of the drill-down directly from the context of the physician ear file. Let’s now go to the general Application Dependency and Performance (ADP) page to see how we can take advantage of some of the other model-driven capabilities provided within Enterprise Manager.

Please click the “Application Dependency and Performance” link in the top-right corner of the main window next to where you can change the timeframe you are examining.
a. You can now see the general dashboard which shows all of the various components available in our ADP model. The navigation tree is a model itself and allows you to explore your applications from any entry point or high-level component you like. Please note that we have SOA applications here as well which are running relatively slow and thus most of the right-hand pane will likely be showing alerts for those components initially.

b. Please expand the tree and explore.

Note that the right-hand pane changes to match the context of the node in the navigation tree you have highlighted.

Let’s go to the Services node and expand the underlying HTTP node where we can look at the actual URLs being monitored and treat them as entry points. We can click on first physician application in the tree (please note we have two physician applications in two different WLS domains which are the reason they are separated) and click on the top-level physician node.

Here we will once again see some specific metrics on our physician application on medrec_1.

You can also expand the *.jsp and *.action sub-nodes to see some of the specific URLs.
c. Click on the FacesServlet which is the primary MVC Servlet for all of our actions to see details specifically for that one allowing us to see the overall performance trends.

You can switch time top-most timeframe to 24 hours to get more visibility. Please note though that your training image may not have been up for very long, so you may not see a ton more metrics for the wider time period.

d. Now let’s look at the architecture view again in the general ADP page to see where the delays might be occurring.

Right-click on /physician and choose “Display Architecture View” from the list of selections.

Please note this is also where you can set service level objectives (alerts) at the component level. You can explore that functionality on your own after the exercise as time permits.
e. You can now see the full architecture view with the actual delay analysis next to it and determine where most of the delays are occurring from an architecture standpoint. Please note that several underlying pages are being called via the MVC framework under the FacesServlet. As you can see, most of the delay is in the FacesServlet, so as capacity increases this very well might be where we see delays.

f. Double-clicking on the FacesServlet component brings us a level deeper into all of the EJB and JSP calls being made and the relationships between those calls.

It also shows us a much clearer picture which underlying calls and components are resulting in the delays.

Please try clicking on various objects to understand the overall infrastructure and metrics available for each component and how you can easily see the performance trends or perform root cause analysis using the tool.

Important note: One thing to note is that there are no JDBC calls occurring during the patient search within the actual ear file. You will notice that if you click on the SQL Statements tab for each component, you will see none are made by that specific component on medrec_1. Why is that? The reason is because
of the earlier RMI call. The JDBC call is actually occurring in the other physician app on medrec_2 (a different domain) via an RMI call. In addition, once the JDBC call is initially made, it will persist the EJB and thus will not make a return trip to the database. You can try to explore expanding the timeframe to see some of these dynamic calls (all calls are dynamic in the Architecture View unlike some of the BPEL and Portal views in Application Dependency and Performance), but it may be difficult. Let’s instead explore a different ear file to see that visibility which makes more simple JDBC calls consistently.

5. **Now close the architecture view, and go through the same process with the with the first /medrec HTTP entry point which basically is for medrec_2. Once you bring up the Architecture View, try selecting the SQL Statements tab to see the actual SQL statements.**

You can now see this level of detail identifying the performance trends of the actual SQL statements that underlaying the code.
Scaling Out Existing WebLogic Clusters (Optional – Extra Credit)

Summary:
Administrators can leverage the WebLogic Server Management Pack Enterprise Edition to quickly and efficiently add capacity to a domain in order to improve performance of production applications.

Exercise Walkthrough:
1. Click on the ‘Deployments’ tab.

Then under the ‘Deployment Procedure Manager’ section at the bottom of the page click the ‘Deployment Procedures’ link.

2. A list of deployment procedures appears. A deployment procedure is a hierarchical series of steps to perform a provisioning operation. In other words, the workflow of all tasks that need to be performed for a particular lifecycle management activity is encapsulated in a deployment procedure.

There are two types of activities involved in deployment procedures. The first is design-time activities which can include customizing the default deployment procedure according to administrator’s needs. The second is run-time activities which include using the deployment procedure to actually provision software. Oracle Enterprise Manager 11g Grid Control Release 1 and the WebLogic Server Management Pack Enterprise Edition introduces two new deployment procedures for Oracle Fusion Middleware – Fusion Middleware Provisioning and Fusion Middleware Domain Scale Up. The Fusion Middleware Provisioning deployment procedure can be used to clone an existing, source Fusion Middleware home and WebLogic domain (i.e. cloning from test to production). The Fusion Middleware Domain Scale Up deployment procedure can be used to expand or scale up a domain by increasing the capacity of the WebLogic Server cluster. The focus of this exercise is on the latter deployment procedure – Fusion Middleware Domain Scale Up.

From the list of deployment procedures select the one labeled ‘Fusion Middleware Domain Scale Up’ and click the ‘Schedule Deployment...’ button.
3. The run-time view of this deployment procedure is displayed as a four step wizard. The first step is to select the existing, source WebLogic domain from the domains already being monitored by Grid Control.

On the ‘Select Source’ page, provide the following information, and then click the ‘Next’ button.

- Fusion Middleware Domain Name: clone_domain
- Working Directory: /tmp/fmw_scaleup_source
- Administrator Username: weblogic
- Administrator Password: welcome1
- Administration Server Host Credentials: Select the option for ‘Preferred Credentials’

4. The second step is to configure the domain either by adding a new managed server or by cloning an existing managed server in the domain.

In the tree, select the existing managed server labeled ‘managed_server_1’ and click the ‘Clone Server’ button.
5. Select the newly added managed server within the tree (labeled ‘Managed_Server1’) and on the right hand side of the console, specify the following information, and then click the ‘Next’ button.

- Managed Server Instance Host: ec2as11g.amazonaws.com
- Managed Server Name: managed_server_2
- Listen Address: ec2as11g.amazonaws.com
- Listen Port: 7070
- Configure Machines section: Select to ‘Do not associate with machine’
- Working Directory: /tmp/fmw_scaleup_destination
- Managed Server Credentials section: Select the ‘Preferred Credentials’ option

Note: You need to be sure to specify a different ‘Working Directory’ on this page as what was specified on the ‘Select Source’ page. Since we do not have another machine to which to clone the managed server, we are cloning the managed server to the same machine as the source so the two working directories must be different.

6. The third step is to specify when to perform the actual provisioning operation.

   For the purpose of this exercise, choose to start the operation immediately. Click the ‘Next’ button.
7. The final step is to review the settings already provided.

Scan the page to ensure inputs look accurate, and click the ‘Submit’ button.

8. At this point, a job has been scheduled to scale up the domain. From the ‘Procedure Completion Status’ page you can track progress of this operation.

From the ‘View Data’ drop down list, choose to refresh the content on the page every 30 seconds.
9. As the page refreshes, notice how the status of the run changes from ‘Scheduled’ to ‘Running’. Continue to track progress until the status reaches ‘Succeeded’. This could take approximately five minutes.

10. Once the status has changed to ‘Succeeded’, navigate back to the domain that was to be scaled up and validate that the newly cloned managed server was discovered and up and running. In addition, to ensure that applications are now being served by the newly cloned managed server, access the Calendar application from the listen port of the newly cloned managed server via the url http://<Virtual Machine Address>:7070/Calendar.
This concludes the Oracle Enterprise Manager 11g WebLogic Server Management Pack Enterprise Edition Hands-on Lab. This hands-on lab runs twice during Oracle Open World: Tuesday 2:00-3:00 PM and Thursday 10:30-11:30 AM.

Additional information on WebLogic Server Management can be found...
- at the Demo Booth for Oracle WebLogic Server Management and Java Diagnostics (demo ID 3462) located in Moscone South
- at the following Technical Sessions:

<table>
<thead>
<tr>
<th>Session ID</th>
<th>Title of Technical Session</th>
<th>Date &amp; Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>S317067</td>
<td>WebLogic Server Management for Oracle DBAs</td>
<td>Thursday 9:00 AM</td>
<td>Marriott Hotel, Salon 9</td>
</tr>
<tr>
<td>S317066</td>
<td>Deep Java Diagnostics and Performance Tuning: Expert Tips and Techniques</td>
<td>Thursday 1:30 pm</td>
<td>Marriott Hotel, Salon 9</td>
</tr>
</tbody>
</table>

For additional information, visit:
Oracle Enterprise Manager: SOA, WebLogic Server, and Middleware Management

Oracle Application Management
http://www.oracle.com/technology/products/oem/prod_focus/app_mgmt.html

Oracle Enterprise Manager
http://www.oracle.com/technetwork/oem/grid-control/overview/index.html