Oracle® Composite Application Monitor and Modeler

User's Guide

Release 10.2.0.5

E14546-02

April 2009

ORACLE®

Oracle Composite Application Monitor and Modeler User's Guide, Release 10.2.0.5

E14546-02

Contributing Author: Jacqueline Gosselin

Contributor: Glen Hawkins, Arivalagan Kaliyaperumal, James Kao, Karthik Somasundaram

Contents

3 Exploring the Monitor Workspace

Preface

This guide provides detailed information and procedures for using Oracle Composite Application Monitor and Modeler (henceforth referred to as CAMM). CAMM is a production services monitoring and performance reporting product that can dramatically improve your ability to track the performance and efficiency of complex server-side applications deployed on popular J2EE application server platforms. CAMM reduces the amount of time, effort, and errors associated with the typical manual process associated with setting up and maintaining an Model-driven Application Management (MDAM) system.

Audience

The *Oracle Composite Application Monitor and Modeler User's Guide* is written for those responsible for setting up and maintaining performance monitoring system for complex applications running on the Oracle Application Server, Oracle WebLogic Server, and IBM WebSphere Server platforms. This could be administrators, operation support specialists, architects, or other key members of various development and application operations teams. Additionally, end users who utilize CAMM to browse operational dashboards and performance reports are encouraged to use this manual.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at `http://www.fcc.gov/cgb/consumerfacts/trs.html`, and a list of phone numbers is available at `http://www.fcc.gov/cgb/dro/trsphonebk.html`.

Related Documents

For more information, see the following documents in the *Oracle Composite Application Monitor and Modeler* Release 10.2.0.5 documentation set:

- *Oracle Composite Application Monitor and Modeler Release Notes*
- *Oracle Composite Application Monitor and Modeler Installation and Configuration Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction to CAMM

Leaders in today's IT organizations are under tremendous pressure to deliver mission-critical business applications while dealing with constantly evolving business requirements. To overcome these challenges, many have turned to J2EE and service-oriented architecture (SOA) to help them attain higher levels of performance and agility. As IT organizations deploy more J2EE and SOA applications into QA and production, they start to discover that conventional methods of managing application performance, such as JMX data collection and byte-code instrumentation, are no longer adequate.

CAMM is the only intelligent ASM platform available that can effectively overcome the management challenges of today's complex, distributed J2EE and SOA applications. This chapter further expands on the new concept of an intelligent ASM platform.

This chapter provides the following information:

1.1 Overview

Unlike conventional APM toolkits, Oracle Composite Application Monitor and Modeler (henceforth referred to as CAMM) analyzes J2EE and SOA applications to capture the complex relationships among various application building blocks in its AppSchema model - the core of the Oracle intelligent platform.

Using the insights stored in AppSchema, CAMM is able to deliver an ASM environment that self-customizes out-of-the-box, evolves with change, minimizes expert involvement, and delivers a holistic, service-oriented view across heterogeneous environments. Adopting an intelligent platform such as CAMM enables enterprise to more efficiently manage distributed applications, attain management agility, and lower total cost of ownership.

See the following sections for additional information.

1.1.1 Terminology

The following terminology is used throughout this manual.

Table 1–1 Terminology

Term	Definition
ASM	Application Service Management
APM	Application Performance Management
DIY	Do-It-Yourself
ISV	Independent Software Vendor
Request Trace	A single thread executing a request. Shows as a bar in the hierarchy view.
Request Trace Events	The nodes in the left pane
Request Trace Event View	Transaction event view
SLA	Service Level Agreement
SLO	Service Level Objective
SOA	Service-Oriented Architecture
Transaction Analysis	Transaction tracing
Transaction Hierarchy View	The Transaction Hierarchy View displays a complete transaction hierarchy in a Gantt chart representation showing the execution of each transaction on an absolute timeline.
UI	User Interface
WSRP	Web Services Remote Portlets
WSDL	Web Services Description Language

1.1.2 Managing Complex J2EE and SOA Applications

Today's J2EE and SOA applications enable enterprises to deliver mission-critical business functions to key constituencies - most often their customers, partners, and employees. These composite applications are assembled from many different J2EE components and exposed services distributed across a heterogeneous environment. Unlike conventional monolithic applications of yesteryear, the complexity of today's J2EE and SOA applications has grown exponentially for the following reasons:

- Highly distributed execution

  Interconnected application components executing in different runtime environments significantly increase execution complexity.

- Significant code generation

  Code generation associated with modern application servers and application development frameworks significantly increases architectural complexity.

- Rapid application deployments and changes

  Incremental application deployments and changes at a rapid pace significantly increase operational complexity.

Regrettably, conventional application performance management (APM) toolkits cannot effectively overcome the escalating challenges of J2EE and SOA complexity because they share the following flaws:

- Focus on resource-centric measurements and views

Conventional APM toolkits associate measurements and views to the individual agents. This approach makes managing applications with highly distributed components and runtimes extremely difficult.

- Require deep J2EE, SOA, and application expertise

  Configuring conventional APM toolkits to manage today's J2EE and SOA applications requires teams of experts with deep J2EE, SOA, and application knowledge. Based on their knowledge, these experts perform various Do-It-Yourself (DIY) manual tasks. Heavy reliance on experts strains IT resources and increases dependency risks.

- Depend on repetitive DIY manual processes

  Setting up an effective ASM environment with conventional APM toolkits requires teams of experts to perform DIY manual tasks such as metric selection, metric grouping, threshold setting, alert action configuration, and so on. As new application deployments and updates occur, teams of experts must religiously follow a number of repetitive DIY manual processes to maintain the effectiveness of these APM environments. This manual and expensive approach breaks down and spirals out of control with rising complexity and rapid rate of change.

Given these flaws, enterprises using conventional byte code instrumentation APM toolkits for J2EE must commit significant amount of IT resources to set up and maintain effective APM environments for their distributed J2EE and SOA applications. Clearly, throwing more IT resources to address the complexity problem is not the answer. To be effective at managing today's complex, distributed J2EE and SOA applications across a heterogeneous environment, enterprises must adopt an intelligent ASM platform with the following characteristics:

- Provides holistic, service-oriented views across heterogeneous environments

  An intelligent ASM platform must provide high-level service-oriented metrics that map to low-level technology-centric metrics. These measurements must be organized in a service-oriented fashion to deliver a unified, holistic view of the numerous interconnected application components deployed across heterogeneous environments.

- Requires minimal J2EE, SOA, and application expertise

  An intelligent ASM platform must have the ability to capture complex relationships among various interconnected components of today's J2EE and SOA applications. This ability can help minimize reliance on J2EE, SOA, and application experts for setting up and maintaining effective APM environments.

- Eliminates repetitive DIY manual processes

  An intelligent ASM platform must eliminate repetitive DIY manual processes by delivering the ability to self-customize out-of-the-box and evolve with change. Elimination of these repetitive DIY manual processes is the only way to deal with rising complexity and rapid rate of change with ease.

1.1.3 Delivering a Service-Oriented View Across Environments

Today's mission-critical business functions are powered by J2EE and SOA applications that comprise numerous interconnected components deployed across highly distributed environments. To manage these applications effectively, enterprises must first gain an understanding of the complex relationships among the business functions, associated interconnected components, and the underlying runtime environments. To enable clear and accurate understanding, IT organizations need holistic, service-oriented views that span across heterogeneous environments.

Furthermore, appropriate rendering of these views enables users at different levels of the organization to collaborate with each other and do their respective jobs more efficiently.

Unfortunately, conventional APM toolkits are incapable of providing holistic, service-oriented views due to limitations associated with their management approaches. Let's examine some of these approaches in more detail:

- Server-centric management

  This is a typical approach used by enterprise system management frameworks to gain visibility into the J2EE tier. This resource-centric approach collects availability and performance measurements from various J2EE containers across the enterprise and organizes them into a single view. While adequate for monitoring the health of various servers, this approach does not provide deep enough visibility for application level management.

- JVM-centric application management

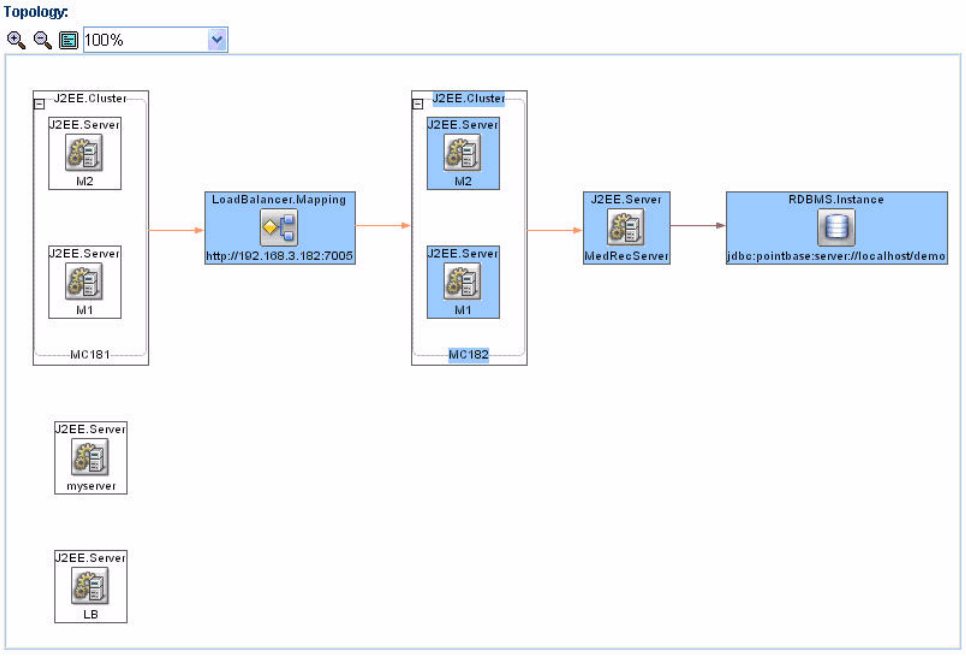
  Commonly used by conventional APM toolkits for J2EE, this resource-centric approach collects low-level technology-oriented measurements from components running in a single JVM. While these toolkits offer ways for users to arbitrarily group measurements from multiple JVMs into logical units, these groupings are imprecise representations of distributed applications. While this approach has been the most common method for monitoring J2EE applications, it increasingly falls short as J2EE applications become more complex, distributed, and service-oriented.

- Transaction tracing

  The transaction-centric approach follows the path of a single transaction across multiple resources and collects low-level technology-oriented measurements along the way. While this approach provides sufficient visibility for distributed applications, it incurs significant overhead per trace and is thus not traditionally employed for production environments. Consequently, conventional APM toolkits employ techniques like sampling rate limitation, sampling window reduction, and overflow protection to control overhead. These visibility-limiting techniques and the need to identify target transactions beforehand make this approach less desirable for managing J2EE and SOA applications continuously.

  Oracle developed the only intelligent ASM platform capable of delivering a holistic, service-oriented view across heterogeneous environments for J2EE and SOA applications. CAMM uses the AppSchema modeling technology to capture complex relationships among distributed applications, software components and runtime infrastructure. The semantic mappings stored in the AppSchema model enables CAMM to accurately measure performance of its managed entities across heterogeneous environments in the appropriate context. Moreover, features like AppSchema Visualization and Navigation significantly improve the overall usability. AppSchema Visualization displays these complex relationships in an organized fashion via different visualization techniques.

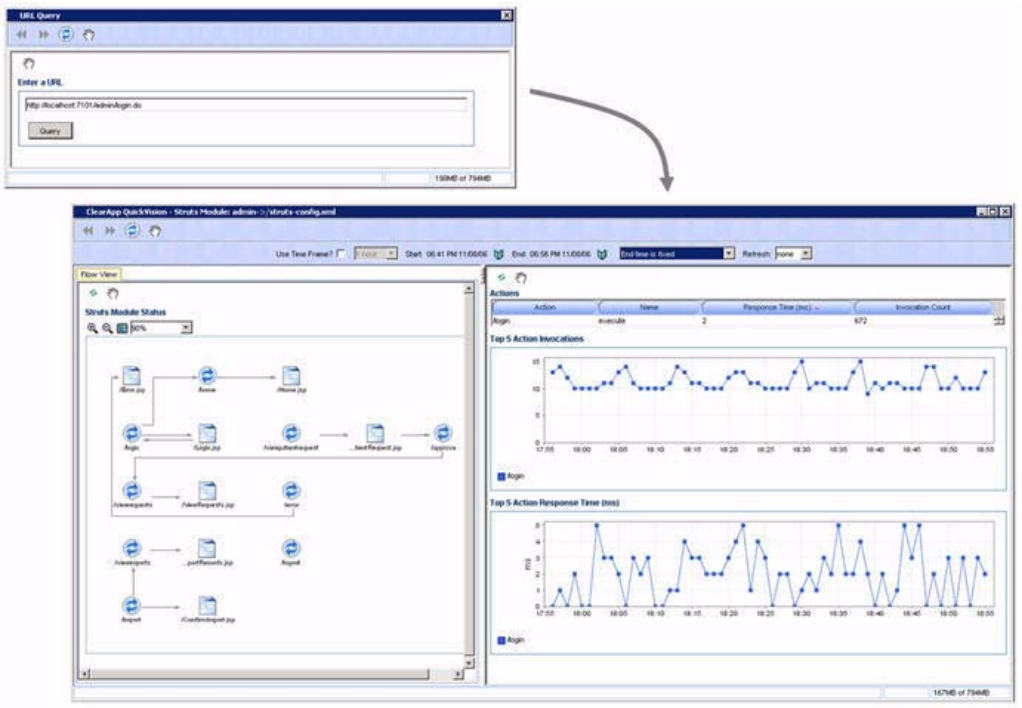
Figure 1–1 Topology View in CAMM



AppSchema Navigation provides efficient ways for you to access relevant information using techniques like hierarchical traversal, architecture model navigation, string queries, drill down, drill out and more.

Use the URL to search for the most appropriate representation of the AppSchema mode.

Figure 1–2 AppSchema View in CAMM



1.1.4 Avoiding Involvement from J2EE, SOA, and Application Experts

Today's enterprises IT organizations are under constant pressure from corporate leadership to create solutions that enable companies to obtain competitive advantages or maintain parity. To churn out applications that meet these fast changing requirements, enterprise developers and architects have turned to J2EE, SOA, and other application development frameworks to maximize efficiency and flexibility. Over time, these experts with specialized knowledge on the way these frameworks are used in their respective IT organizations become instrumental in the software development lifecycle process.

In recent years, demand for J2EE and SOA applications has increased steadily. As a result, IT organizations are now experiencing expertise shortages as existing specialized resources are stretched to their limits. Consequently, IT organizations are seeking new ways to address expertise shortage, minimize reliance on specialized resources, and give experts more bandwidth to focus on value-added activities.

Unfortunately, conventional APM toolkits only make this problem worse. Setting up and maintaining an effective APM environment with conventional APM toolkits requires deep J2EE, SOA, and application knowledge. With these toolkits, experts are needed to determine the architecture of these distributed applications, figure out the configuration of the runtime environments, and select optimal locations to insert performance measurements. These knowledge-intensive tasks require IT organizations to dedicate even more specialized resources, thus further worsening the expertise shortage problems.

It is very difficult to monitor applications created by third-party ISVs and off-shore development teams with these conventional tools due to lack of in-house knowledge.

To overcome these challenges and manage J2EE and SOA performance effectively, IT organizations must adopt an intelligent platform like CAMM that requires minimal expertise to set up and maintain. Unlike conventional APM toolkits, CAMM does not rely on human expertise to set up and maintain customized APM environments. Instead, CAMM uses a unique model-driven approach that leverages the information stored in its AppSchema model to keep the involvement of experts to the minimum. CAMM's unique ability to self-customize out-of-the-box and evolve with change makes it the perfect solution for managing not only custom enterprise applications, but also applications developed by external parties.

1.1.5 Eliminating Repetitive Do-It-Yourself (DIY) Manual Processes

For years, developers and architects relied on repetitive DIY manual processes to measure application performance. Since the advent of Java byte-code injection techniques in the late 1990s, IT organizations have gradually abandoned the completely manual source-code instrumentation techniques in favor of partially automated byte-code instrumentation techniques. Conventional APM toolkits have capitalized on this trend by offering features that would insert byte-code instrumentation automatically. Regrettably, these conventional APM toolkits did little to reduce the repetitive DIY processes required to set up and maintain effective APM environments.

With conventional APM toolkits, IT organizations must go through the following activities repetitively in order to set up and maintain effective APM environments:

- Understand application structure and runtime configuration
- Manually select relevant performance measurements for each application
- Apply context by creating arbitrary metric groups manually

- Update the APM environment when changes occur

The demand on today's IT organizations to efficiently churn out enterprise applications has stretched existing IT resources to their limits. To make matters worse, IT organizations are deploying more applications into production faster and making application changes more frequently. These trends combined with expertise shortages make it more difficult for IT organizations to keep their APM environments up-to-date. As a result, IT organizations look for ways to minimize wasteful activities - such as repetitive DIY manual processes associated with conventional APM toolkits.

CAMM can help IT organizations overcome this challenge. Based on a unique model-driven approach, CAMM is the only intelligent ASM platform that eliminates repetitive DIY manual processes. To achieve this level of self-customization and continuous change adoption, CAMM uses its AppsSchema modeling technology to perform the critical task of analyzing application structure and infrastructure configuration. After capturing these insights in the AppSchema model, CAMM leverages this information to establish a fully customized ASM environment. To keep this environment up-to-date, CAMM continuously updates the AppSchema model as new applications are deployed and changes are applied. CAMM's unique ability to self-customize out-of-the-box and evolve with change enables fast time-to-value, low total-cost-of-ownership (TCO), and maximal return-on-investment (ROI).

1.1.6 CAMM Solution

Today's IT organizations leverage J2EE, SOA, and other application development frameworks to efficiently churn out powerful enterprise applications to meet fast changing business requirements. To ensure these mission-critical applications and services are available and performing at the highest level, enterprises must invest in proper application performance management (APM) solutions. Unfortunately, conventional APM toolkits and their repetitive Do-It-Yourself (DIY) manual processes, once suitable for managing monolithic applications, are no longer effectively at managing these fast changing, highly distributed J2EE and SOA applications running in heterogeneous runtime environments.

A far superior approach for managing J2EE and SOA applications is to use an ASM platform intelligent enough that it eliminates repetitive DIY manual processes and reduces the involvement of expert resources. Furthermore, this platform must be able to deliver a holistic, service-oriented view across heterogeneous execution environments by leveraging a metadata based model to capture the complex relationships among various application building blocks. Finally, enterprises require a solution that is sufficiently agile to handle frequent application and infrastructure changes. In short, today's IT organizations need an intelligent APM platform for J2EE and SOA.

Oracle provides the industry's first intelligent ASM platform for J2EE and SOA. Unlike conventional APM toolkits, CAMM analyzes these applications and captures complex relationships among various application building blocks in its AppSchema model - the brain of this intelligent ASM platform.

Using the insights stored in the AppSchema model, CAMM is able to deliver an ASM solution that self-customizes out-of-the-box, evolves with change, minimizes expert involvement, and delivers a holistic, service-oriented view across heterogeneous environments. Adopting an intelligent platform such as Oracle will enable enterprise to more efficiently manage distributed applications, attain management agility, and lower total cost of ownership.

1.2 Architecture

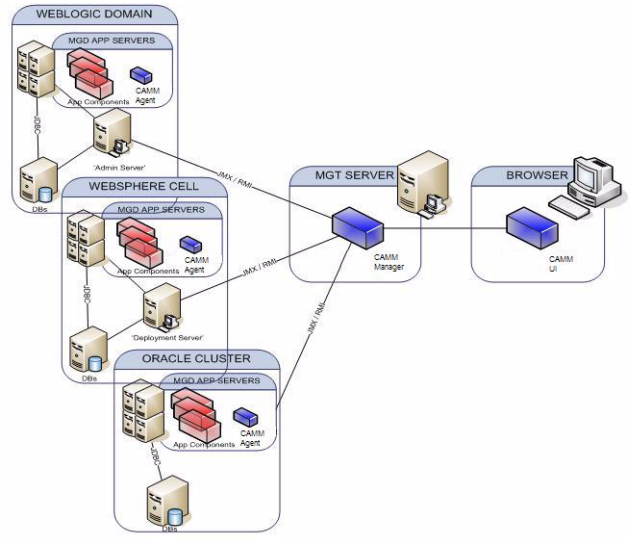
CAMM employs a multi-tier, fully distributed, configurable architecture to provide the scalability and flexibility to meet the changing needs enterprise deployments. CAMM can operate in two main modes: *Service Mode* and *Standalone Application Mode*.

In Service mode, CAMM operates as a service on the machine and automatically begins running when the machine first boots, and remains on perpetually. In this mode CAMM is typically installed on its own machine and dedicated to monitor a group of managed application servers.

To allow remote access to CAMM through a browser, a web container is installed. This web container provisions the CAMM UI applets to the browser and maintains communication with these applets.

Figure 1–3 shows CAMM deployed in Service Mode.

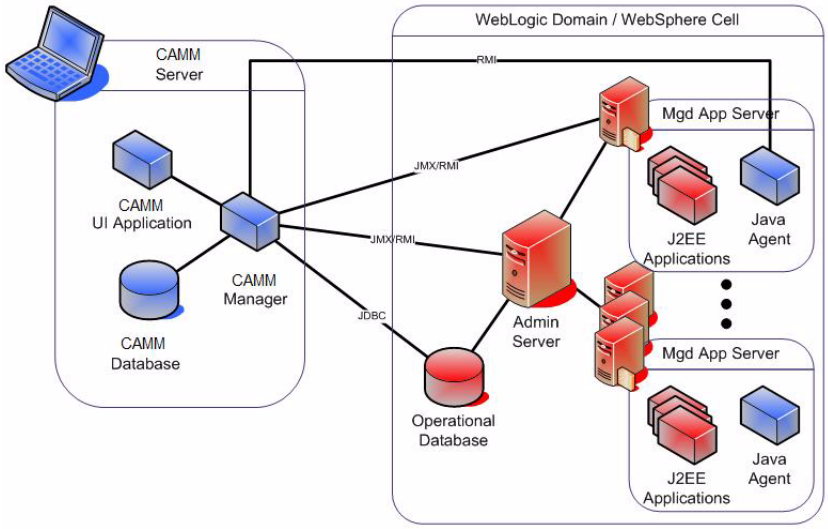
Figure 1–3 Service Mode CAMM Topology



In the Application Mode, CAMM runs as an application. When you start the application, CAMM starts, and when the application is closed, CAMM discontinues operation. Application mode is valuable to run as an occasional debugging tool, perhaps on a laptop.

Figure 1–4 shows CAMM deployed in Application Mode.

Figure 1–4 Application Mode CAMM Topology



The following core components are deployed to form the CAMM ASM system in all modes.

1.2.1 CAMM Java Agents

CAMM Java Agents are the data collectors of the CAMM ASM system. CAMM Java Agents are deployed to all managed application servers to perform a series of tasks including collecting performance managements, tracking contextual relationships, and summarizing data in real-time while introducing as little overhead as possible. At the expiration of the predefined aggregation interval, these agents forward the summarized data to CAMM for additional analysis. For various J2EE platforms such as Oracle SOA Suite, Oracle WebLogic, and IBM WebSphere, CAMM leverages their deployment infrastructures to quickly deploy the CAMM Java Agents to all application servers.

1.2.2 CAMM Manager

CAMM Manager is the core analytical engine of the CAMM ASM system. In real-time, CAMM Manager performs complex mathematical modeling and statistical calculations with summarized data from all CAMM Java Agents. CAMM Manager can be configured with a backup to provide higher level of availability.

CAMM Manager can also be configured without the UI component, also known as headless configuration.

1.2.3 CAMM Database

CAMM stores its analyzed data and application models in a CAMM Database - an operational data repository. This database is always external to the CAMM installation.

1.2.4 CAMM User Interface

The CAMM User Interface (CAMM UI) is the primary user interface for CAMM users. Users can use CAMM UI to view operational dashboards, set Service Level Objectives

(SLOs), define actions, create custom views, analyze monitoring data, and more. The CAMM UI is fully configurable.

2

Exploring the User Interface

This chapter explores the CAMM User Interface. Topics include:

2.1 Starting CAMM UI

Oracle Composite Application Monitor and Modeler (henceforth referred to as CAMM) can operate under two modes: service mode and standalone application

mode. Under standalone application mode, start the CAMM UI by executing the `[CAMM Installation]\bin\standalone.bat` file.

Under the service mode, you can access CAMM UI either through a web browser after starting the manager process either by running "[CAMMInstallation]/bin/acsera.sh" or starting the "Oracle CAMM" Windows service.

2.1.1 Web Browser Access

To access the CAMM UI using a web browser, type the following URL in the address box:

```
https://[CAMM Machine Name or IP]:[port number]/qvadmin [new]
```

Note that 5560 is the default port number for the CAMM web container.

2.2 General CAMM UI Elements

CAMM UI consists of the following core components:

- General Menu and Toolbar

  The General Toolbar provides general purpose functionality such as quick navigation, view refresh, status, version, and shutdown. You can browse through previously viewed data easily by clicking the back and forward buttons. Displayed data can be force refreshed by using the refresh button. You also get status on CAMM or issue a shutdown command from the Manager item on the General Toolbar.

- Double-Click Indicator

  The dotted-grid icon next to a table indicates that the double-click operation is available for that data table. Typically, double-clicking displays next level details.

- Navigation Pane (upper left)

  There are two types of workspaces in the CAMM navigation pane: *Monitor* and *Configure*. In the Monitor workspace, you can navigate the managed environment and monitored applications in a tree. Use the Monitor workspace to traverse the CAMM tree model and identify abnormal activities. Use the Configure workspace to create, modify, and review various configuration settings for CAMM.

- Main Display Window (upper right)

  As you navigate through the CAMM tree model and configuration categories, detailed performance information and configuration settings are displayed in the Main Display Window. You can refresh the Main Display Window at anytime by clicking the Refresh icon.

- Custom Views (lower left)

  You can create custom views in CAMM by simply dragging and dropping display components. All custom views already created are listed in the Custom Views panel.

2.3 Navigating CAMM

When CAMM starts up, the Operational Dashboard appears in the Main Display Window.

This dashboard displays the health of the system. By using the dashboard, you can investigate abnormal behaviors further by using the following navigation techniques.

- Drill Down in Operational Dashboard
- Drill Down in Monitor Workspace

2.4 Drill Down in Operational Dashboard

The Operational Dashboard displays the health indicators for various key entities in the managed environment. CAMM uses traditional traffic light colors to represent the health of these various key entities.

- Normal (Green): Within an acceptable range
- Warning (Yellow): Approaching configured threshold
- Violation (Red): Violated configured threshold

For each component, CAMM uses the following health indicators to provide a comprehensive view. These health indicators are:

- Performance

  The performance health indicator depicts the relative responsiveness of the monitored entity to the configured threshold.

- Availability

  The availability health indicator informs you to what extent a particular entity is available to service requests.

- Errors

  The errors health indicator informs you if the number of errors and exceptions encountered by this entity are approaching or violating the configured threshold.

- Load

  The load health indicator depicts how many operations have been performed and requests have been served by a particular entity.

CAMM is aware of clusters. As such, these indicators display overall health of a particular entity across the entire cluster. You can drill down for more details by clicking the plus (+) icon.

2.4.1 Status of Multiple Health Indicators

The status of multiple health indicators are actually a composite of several metrics. For example, the performance health indicator is comprised of metrics such as Average Execution Time and Number of SLO Violations. Using this detailed information, you can quickly determine the reason behind an abnormal status.

Health Indicator Drill Down Details are currently available for high level business functions only. Currently the following business functions are supported:

- WebLogic Portal - Portal Applications
- Oracle SOA Suite - BPEL Processes
- WebLogic Integration - Processes
- WebSphere Portal - Portal Applications

Further drill-down is possible. By double-clicking the name of the entity, a new viewer relevant to that entity appears. For example, the entities can be business processes running on the Oracle WebLogic Integration platform. By double-clicking on one of these process names, CAMM opens up the Process Flow Viewer.

2.4.2 Process Flow Viewer

The Process Flow Viewer and other viewers provide even more application specific information to enable additional analysis.

The Process Flow Viewer is composed of two panels: the flow of selected processes on the left and the Main Display Window on the right. The Main Display Window shows information corresponding to the item selected in the navigation pane. The left panel shows the flow of the selected process. This is the same view a developer would see in Oracle WebLogic Workshop. This unique feature gives you the ability to communicate with others and solve performance problems faster.

2.4.3 Node Groups

Some process nodes are groups of nodes. By clicking the plus sign (+) and minus sign (-) icons, you can expand and collapse the node group. When a node group is collapsed, the information displayed in the Main Display Window is aggregated automatically for the entire node group.

You can drill down on other entries in the Operational Dashboard by double-clicking the entry. For example, when you select and double-click a row in the Operational Dashboard, a new pop-up window appears and provides relevant lower level statistics. You can select an entry in the Operational Dashboard and bring up detailed performance metrics by double-clicking on the operational dashboard.

2.5 Drill Down in Monitor Workspace

A monitor workspace consists of the Navigation Pane and the Main Display Window. It is used to navigate the monitored components and display relevant metrics.

2.5.1 In Oracle WebLogic

The Monitor Workspace is organized hierarchically using a tree view. Starting from the root of the tree, you can quickly drill down to the information you need. You can expand the tree to see more details by clicking the plus sign (+) icon.

When you select a lower level tree node, the Main Display Window displays performance data and renders graphs relevant to the selected item. For instance, when you select the *csr* desktop node in the *css* portal tree, the Main Display Window shows the Portal desktop performance table, Desktop hits and Desktop response time graphs.

You can double-click a portlet in the portal desktop layout view to drill down on each level and view the portal desktop structure.

2.5.2 In Oracle SOA Suite

The Oracle SOA Suite workspace is similar to WebLogic, but with a focus on BPEL processes, ESB, and Web Services as opposed to portals and WLI processes.

Like Oracle WebLogic, the Monitor Workspace in the left-hand pane is organized hierarchically using a tree view. Starting from the root of the tree, you can quickly drill down to the information you need. You can expand the tree to see more details by clicking the plus (+) icon.

When you select a lower level tree node, the Main Display Window displays performance data and renders graphs relevant to the selected item. For instance, when you select the *Domains* desktop node in the *BPEL Processes* tree the Main Display Window shows the BPEL process performance table, BPEL hits and BPEL response time graphs.

You can double-click on a BPEL Process in the BPEL Process Status view to drill down on each level and view the functional BPEL Process structure.

2.5.3 In WebSphere

The WebSphere workspace is similar to the Oracle SOA Suite and Oracle WebLogic with a few differences in the node details.

The Monitor Workspace is organized hierarchically using a tree view. Starting from the root of the tree, you can quickly drill down to the information you need. You can expand the tree to see more details by clicking the plus (+) icon.

When you select a lower level tree node, the Main Display Window displays performance data and renders graphs relevant to the selected item. For instance, when you select the *Virtual Portals* desktop node in the *WebSphere* portal tree, the Main Display Window shows the Portal desktop performance table, Desktop hits and Desktop response time graphs.

You can double-click on a portlet in the portal desktop layout view to drill down on each level and view the portal desktop structure.

2.6 Configuring Service Level Objectives (SLOs)

In Oracle CAMM, thresholds configured for various measurements are called Service Level Objectives (SLOs). Configuring SLOs is a key activity for establishing and maintaining an effective performance monitoring system. It is easy to configure SLOs in CAMM. By right-clicking on any element, you can configure or view SLOs. This rule applies to the CAMM UI and all other viewers.

2.6.1 Create New SLO

When you select Configure Service Level Objectives, CAMM displays the Service Level Objective Configuration window. This window allows you to apply existing SLOs or create new ones. When you click **Create New SLO**, CAMM guides you through the process of setting up a new SLO.

The steps for SLO creation are as follow:

1. Select a SLO file. CAMM can store SLO configurations in different files to improve configuration portability.
2. Define the SLO Entity Type. CAMM automatically selects the appropriate entity type for you based on the selected monitor element. For example, if you want to set a SLO on a Portal Desktop element, CAMM automatically sets the Entity Type for you.
3. Other information is filled in by default. Normally, there is no need to modify the SLO Entity values.
4. When you are done setting the SLO Entity Type values, click **Create New SLO** to go to the second step of the SLO creation process, Defining the SLO Parameters.

2.6.2 Defining SLO Parameters

Follow these steps to define the SLO parameters:

1. Type in the name of the SLO parameter.
2. Select the performance metric.
3. Define the monitor window size, which determines how long the condition must persist before generating an alert.
4. Set threshold values for the SLO.
5. Select what actions to take when a trigger is fired. A list of preconfigured actions is available in the view pane.
6. Add new actions by going to the Action Configuration node in the Configure Workspace.
7. Click **Save** to set the SLO for this monitored element.
8. You can delete unwanted SLOs for any element from this window.

Note that setting SLOs in CAMM affects the operational dashboard. Typically, a fresh installation of CAMM has no SLOs defined. As a result, most of the traffic light indicators in the operational dashboard use the gray color - an indicator of no status.

2.6.2.1 Propagating Threshold Violation Events

CAMM is designed to propagate threshold violation events up the hierarchy. Therefore, when a SLO is set on a lower level metric, the higher level health indicator light becomes activated. Additionally, the health indicator light for the application server that hosts this component also becomes active. Oracle calls this *containment approach* to SLO event propagation. When a lower level SLO is violated, the violation event propagates all the way up the hierarchy and changes the status of all containers for this event.

For example, say we defined a performance threshold on the average response time metric of the CaseManagement portlet. We would expect the css portal and cgServer health indicator lights to become active because the CaseManagement portlet is part of the css portal, css portal is part of the cssdemo application, and cssdemo application is deployed on cgServer.

2.6.2.2 Types of SLOs

In addition to the containment concept, CAMM categorizes SLOs into the following types:

- Performance
- Availability
- Error
- Load

In the example, the average response time metric is correctly categorized into the performance type.

If you set a SLO on a metric in the load category such as Portal Desktop Visit Count, you will see the activation of load health indicators for all containers of the desktop. In our example, we set a SLO on the Portal Desktop Visit Count of the csr desktop. This activates the load health indictors for css portal, cssdemo application, and cgServer instance.

2.6.2.3 SLO Events Viewer

Right-click on any tree node and select **View Service Level Objective Events** to open a new window. You can see all the SLO violation events triggered for the selected entity. CAMM automatically applies a filter to show only relevant events.

Once new SLOs are added, CAMM updates the relevant graphs to visually display these new thresholds. Table 2–1, " SLO Line Types" explains the different line types.

Table 2–1 SLO Line Types

Line Description	Description
Solid Red Line	A violation threshold that triggers on high.
Solid Yellow Line	A cautionary threshold that triggers on high.
Dashed Red Line	A violation threshold that triggers on low.
Dashed Yellow Line	A cautionary threshold that triggers on low.

2.7 Configuring SLO Blackouts

Use this option to specify blackout time frames to prevent having a specified number of SLOs from being evaluated. You can prevent having unwanted alerts being fired during planned or unplanned down time.

1. Right-click any tree node and select **Configure SLO Blackout** to open a new window.
2. You can view any existing SLO blackout events in the new window.
3. Use this window to create, delete, or view the details of existing events.

Deleting SLO Blackout

1. Select an existing event on the list.
2. Click **Delete SLO Blackout**.
3. Confirm that want to delete the entry and click **Yes**.

Viewing SLO Blackout Summary List

1. Click **SLO Blackout Summary List**.
2. View the details on the existing SLO Blackout events.
3. Click **Show SLO Blackout List** to return to the previous window.

Creating SLO Blackout

1. Click **Create SLO Blackout** to view the detail window
2. Refer to Table 2–2, " SLO Blackout Configuration" to fill in the columns as needed.

Table 2–2 SLO Blackout Configuration

Column/Metric	Description
Blackout Name	Type in the name.
Description	Type in the description of the SLO you are creating.
Blackout By SLO File	Use to blackout at the file level. The SLO files display in a list where you can select them or cancel out of the window. This option restricts the blackout to the SLO file name.

Table 2–2 (Cont.) SLO Blackout Configuration

Column/Metric	Description
Blackout By Individual SLOs	Use to blackout at the SLO level. The SLOs display in a list where you can select them or cancel out of the window. This option restricts the blackout to the SLO name.
Blackout By Entity	Use to blackout at the entity type level. The entity types display in a drop-down menu where you can select the entity. This option restricts the blackout to the entity type selected.
year, month date, hour, minute, duration	Use the guidelines to the right of these columns to enter the appropriate information.
recurring	Select how often you would like to run this blackout event from the drop-down menu.

2.8 Time Frame

In CAMM, you can specify the length of the time the window information is to be displayed. To specify the length of this time window, select the appropriate length in the Time Frame drop-down box. The following Time Frame values are available:

- 1 hour
- 2 hours
- 4 hours
- 8 hours
- 12 hours
- 24 hours

Note: The CAMM default data collection interval is 60 seconds. As you adjust the data collection interval, CAMM automatically adjusts the display time frames. To learn more about how to configure the data collection interval, refer to the *Oracle Composite Application Monitor and Modeler Installation and Configuration Guide*.

CAMM automatically adjusts information displayed to fit the specified time window. You can drill down to see detailed performance information for a specific range of time.

For example, visualize the drill down process with two screen shots of the same graph with different Time Frames of the average response time for Portal campaign. The first graph has a Time Frame of 24 hours. The second graph has a Time Frame of 1 hour. By increasing the granularity of the Time Frame, you are performing a drill down operation.

In this example, an IT Operations staff noticed abnormally high response time with Portal campaign subsystem. The person decided to investigate further to evaluate the extent of the problem. By changing the Time Frame from 24 hours to 1 hour, this user is able to see that between 14:17 and 14:18, the Portal campaign response time jumped from an average of 1000 milliseconds to 5000 milliseconds. While the problem did not persist, it may warrant additional investigation.

2.9 Display Interval

Display Interval indicates the start and end time for the data displayed in the Main Display Window. Display Intervals change as you change the following settings:

- Time Frame
- Interval Context
- Turning Off Time Frame Limitation

2.9.1 Time Frame

When you select a new Time Frame, the Display Interval automatically changes to fit the selected Time Frame. For example, if you were to change the Time Frame from 1 hour to 2 hours, the Start value of the Display Interval changes.

2.9.2 Interval Context

Display Interval can also be changed by setting the Interval Context. The settings for the Interval Context are:

- Interval Context Set To End Time Is Current System Time

  The default Interval Context for CAMM is to use current system time as End value for the Display Interval. In this default setting, you have a sliding Display Interval and can see the latest performance information in the Main Display Window.

- Interval Context Set To End Time Is Fixed

  You can also change the Interval Context setting to use a fixed time as the End value for the Display Interval. By selecting the fixed Interval Context, you can create a fixed time window to display performance data. The fixed time window is particularly useful for performing analytical tasks.

- Date/Time Selector

  When you select to fix the End time for the Interval Context, the CAMM UI enables a pair of Date/Time Selectors to allow you to set Start or End values for the Display Interval. Click the green icon next to the Start and End times to open up the Date/Time Selector.

  The Date/Time Selector allows you to set a specific Display Interval to fit your needs. Additionally, the Date/Time Selector enables CAMM to compare current performance trends with historical data.

Note: Changing the start and end time do conceptually different things. Users are advised to always change their time frame by modifying the end time first, and then the start time. Changing the end time moves the window in time, whereas changing the start time increases/decreases the size of the window.

2.9.3 Turning Off Time Frame Limitation

To support the display of data for more than twenty four hours, CAMM allows you to specify your own time frame for data display. To enable this, set **Interval Context** to **End time is fixed** and make sure the **Use time frame?** check box is unchecked. Turning off time frame limitation allows CAMM to display eight days worth of data.

For example, when you specify the time frame to be eight days by adjusting the start and end times through the Date/Time Selector, CAMM then adjusts its view to display eight days worth of data in a single graph. This feature allows you to perform trending analysis over time.

2.10 Refresh Rate

CAMM does not automatically refresh the information in the Main Display Window because its default Refresh Rate is set to None. However, you can specify how frequently information in the Main Display Window should be refreshed by selecting the appropriate value in the Refresh Rate drop-down box.

CAMM is capable of automatically refreshing the information in the Main Display Window at the following rates:

- 1 minute
- 2 minutes
- 5 minutes

2.11 Queries

This feature provides a quick and easy way for you to locate performance measurements. Based on the query string used, CAMM determines the most appropriate view to display. The following section describes the types of queries supported by CAMM.

2.11.1 URL Query

The URL of an application can be used to find performance measurements in CAMM. A common use case for the URL query is to copy the URL of a slow performing J2EE application from the browser and copy it in the CAMM URL query dialog box.

URL query can be used to look up performance measurements and model visualizations for the following application types:

- BPEL Processes (.bpel)
- Portals (.portal)
- Processes (.jpd)
- Java Page Flows (.jpf)
- Struts Actions (.do)
- Web Services
- Java Server Pages (.jsp)
- Servlets

2.12 Graphs and Data Items

CAMM displays performance information in various formats. Most commonly used display formats in CAMM are tables and graphs.

- As a general rule, you can gain more information about a data item by simply pointing the mouse over the interested item.

- Minimum and maximum response time measurements are stored in their embedded database in addition to average response time measurements. The min and max metrics, if present, are displayed visually in the UI.
- For tables, you can perform a table sort by clicking column headings. Columns can also be rearranged with a simple drag and drop action. The red sort directional arrow appears next to the column currently being sorted.
- You can define the zoom in area using a click and drag operation. To zoom in, click and drag the mouse to the right. To zoom out, click and drag the mouse to the left.

  Tip: For graphs with extreme outliers, graph details are lost due to automatic graph scaling. To work around this problem, you can use the graph zoom in feature to review these details.

- You can toggle on/off vertical ruler in any graph by holding down the **Ctrl** key. The vertical ruler helps visually line up nodes in a graph for easier analysis.

2.13 Right-Click Operations on Tables and Graphs

There are several simple operations you can perform on various tables and graphs in CAMM. The following is a list of the right-click operations associated with tables and graphs:

Table 2–3 Right-click on Tables and Graphs

Right-Click Operation	Description
Copy cell value	The right-click operation is available for tables only. This operation copies the cell value to enable common copy/paste operation (table only).
Export as CSV...	This right-click operation is available for both tables and graphs. This operation saves all the values in the table or graph as a comma separated value (CSV) file. The CSV file can later be imported into other applications such as Microsoft Excel.
Count number of rows	This right-click operation is available for tables only. This operation returns a count for the number of rows in the selected table (table only).
Fit to View	Make the table or graph file the entire pane.
Restore View	Restore the layout to the default state
Show/Hide Table Columns	Remove or un-remove columns from a table (table only)
Enable/Disable Series	Remove or un-remove a line from a graph (graphs only)

For example, you can use the Count number of rows right-click operation to get a total row count for any table.

2.14 Comparative View

CAMM provides a number of analytical tools to enable performance analysis. One of these tools is the Comparative View. To access Comparative View, right-click the CAMM Main Display Window and select **Create Comparative View**.

After the Comparative View window appears, you can use the Date / Time Selector to specify start and end times for each of the two windows in the Comparative View. You can use this tool to compare performance statistics of two different time frames.

Tip: You can use comparative views to determine if current performance of a specific application or component differs greatly from historical performance or baseline performance captured previously.

Comparative views are useful to evaluate current performance characteristics against historical performance characteristics.

2.15 Save as PDF

To improve collaboration among those who work on application performance issues, CAMM provides the ability to save any view as a PDF file. To save a specific view as a PDF file, right-click on the CAMM Main Display Window and select **Save this view as a PDF file**.

2.16 Easy Scroller

Easy Scroller is a feature to help you navigate different views in CAMM. To bring up Easy Scroller, right-click on a view in the Main Display pane and select the **Easy Scroller** option. Drag the box within Easy Scroller to navigate.

2.17 Zoom In and Zoom Out Toolbar

For some of the views, CAMM provides the zooming ability. This capability enables you to zoom into diagrams for more fine-grain details and zoom out for more coarse-grain structure.

On the Zoom In/Zoom Out Toolbar, the icon zooms in on the view by 10%, the icon zooms out on the view by 10%, and the icon returns the view back to normal size (100%). You can use the drop-down box to quickly zoom in or zoom out of the view.

2.18 Custom Metrics

While CAMM intelligently selects relevant performance metrics based on its AppSchema model, you can further customize the monitoring environment by configuring additional custom metrics. In addition, you can use custom metrics in problem diagnostic situations where additional visibility is needed to pinpoint problem root cause.

To configure a new custom metric, right-click and select **Configure Custom Metric** to begin. CAMM walks you through the configuration process.

Custom Metric Configuration window includes the following fields, see Table 2–4:

Table 2–4 Custom Metrics Configuration Window

Field	Description
Name	This text field is for defining the display name for the custom metric.

Table 2–4 (Cont.) Custom Metrics Configuration Window

Field	Description
Resource Name	This drop-down menu is for defining the resource where the custom metric will be collected.
Class Name	This text field is for defining the fully qualified class name (package + class) associated with the custom metric.
Method Name	This optional text field is for defining the method name associated with the custom metric. **Usage:** 1. Type in * - CAMM will hook all methods. 2. Provide comma separated list of methods with no wildcards - CAMM will create method entities and only hooks these methods in the agent. 3. Provide comma separated list of methods with wildcard prefixes or suffixes - CAMM will instruct the agent to hook the methods specified along with the wildcards. 4. Provide 1) or 2) preceded by "!" to create an excluded list - CAMM will instruct the agent to hook all methods in the class not defined in the exclude list. Method field examples: 1. methodA,methodB,methodC 2. ejb*,*context,methodA 3. !ejb*,*context,methodA

After you define the custom metrics, restart the application server instances associated with these customizations. The new custom metrics will be listed under the Custom Metrics node in the CAMM navigation tree.

Newly configured custom metric reports class level performance data, for example invocation count and response time.

2.19 Promote To Dashboard

You can add metrics to the Operational Dashboard by using the Promote to Dashboard feature. The following steps describe how to add new metrics to the Operational Dashboard.

To add new metrics to the operational dashboard:

1. Right-click on a node in the navigation tree and select **Promote to dashboard** to activate the Dashboard Configuration window.
2. Click **Create Dashboard Promotion**.
3. Select the metric to promote to Operational Dashboard from the drop-down menu.
4. Click **Create**.
5. Type in a name in the **Name** box.
6. Click **Save**.
7. Close the Dashboard Configuration window.
8. Click **Yes** to confirm you want to close the window.

   Your action is now a new entry on the Operational Dashboard.

2.20 Functional View

Functional View is a type of AppSchema Visualization - a visual way for CAMM to represent the information stored in its AppSchema model. This view is designed to help you understand how business functions are assembled with various functional building blocks. Table 2–5 provides a list of functional views currently available in CAMM.

Table 2–5 Functional View

Entity Type	Function View	Description
Process	Process Workflow View	This functional view depicts the workflow associated with the selected WLI and Oracle BPEL business process. It shows all the process nodes and the relationships among them.
Pageflow	Pageflow Functional View	This functional view depicts the logical flow associated with a JPS or Struts pageflow. It shows all the pages in a pageflow and the relationships among them.
OSB Proxy Service	Proxy Service Functional View	This functional view depicts the pipeline and stage flow associated with an OSB Proxy Service.

Depending on the type of entity selected, CAMM displays different functional views. Right-click and select Display Functional View to bring up the relevant Functional View associated with the selected entity.

2.21 Topology View

Topology View is another type of AppSchema Visualization - a visual way for CAMM to represent the information stored in its AppSchema model. This view is designed to help you understand how application environments are assembled with various applications, application server instances, and shared resources. This information helps you map composite applications and their building blocks to application server instances and share resources.

The highest level topology view graphically depicts domains, external resources, and shared database resources. The applications used in the following examples are CSS and MedRec demos.

For example, you can have a topology with two CAMM managed resources, CSS Domain and MedRec Domain, two external resources, and a shared database resource. The lines connecting various entities in the Topology Views depict calls made from one entity to another. You can get more information about a specific call by pointing the mouse over a specific line.

It is possible to hide different types of lines in the Topology View. To the line types, right-click the **Topology View** and highlight the **Edge types** option to reveal a list of different edge (arrow) types associated with the current Topology View.

2.21.1 Edge Types and Colors

The kinds of edge types are:

- Deployment Edge Relationship between components defined in the deployment descriptors
- Method Call Dynamic call created during runtime execution

The colors displayed in the topology view are explained in Table 2–6.

Table 2–6 Edge Types Color Codes

Color	Description
Red	Deployment edges. Represents a reference in deployment descriptor (resource-ref, ejb-ref, ejb-local-ref). In struts modules, it will show forwards.
Purple Edge	Used in execution view to show that the method calls did not happen in the selected time frame. For example, probe point was created by agent, but no metrics were present for the selected time frame.
Orange Edge	Used in execution view, orange edge shows an active method call in the selected time frame. For example, there are metrics for that edge in the selected time frame.

You can perform the following actions on edge types:

- To hide all lines of a particular type in the Topology View, un-check a specific edge type. Checking a specific edge type makes these lines appear.
- To hide all lines not connected with a specific entity, select a monitored entity in the Topology View, right-click and select **Hide other edges**.
- To hide all arrows not connected to the managed entity, Highlight An Entity and select **Hide other edges**.
- To display the topology specific to the selected managed resource, select a specific managed resource and double-click.
- Drilling down on a specific managed resource reveals the relationship among the application server instance and the shared resources it uses. It is possible to reach this same view by right-clicking on the entity and selecting Display Deployment Topology option.
- Drilling down on specific application server instance reveals calling relationship among J2EE applications and shared resources. This information is useful to understand how applications are distributed across the infrastructure.

CAMM can also show the Topology View in a tiered fashion, enabling you to better visualize dependency relationships among various servers and share resources. To access tiered Topology View, select an application entity in the Topology View, right-click and select **Display Tier Dependencies**.

CAMM brings up the Tier Dependency Topology View on the left pane and displays the associated External Calls in the Main Display Window. Table 2–7provides a list of metrics in the External Calls table and their descriptions.

Table 2–7 List of Metrics in the External Calls

Column/Metric	Description
Caller Class	Name of the local class making the external call.
Caller Method	Method name in the local class making the external call.
Target URL	Target URL associated with the external call.
Class	Name of the target class associated with the external call.
Method	Name of the target method associated with the external call.
Invocation Count	Total number of invocations for a specific external call.
Response Time (ms)	Average response time in milliseconds for a specific external call.

Tip: External Calls contains information you can use to determine the types of calls made among various servers, applications, and shared resources. Use this information to diagnose problems associated with cross-JVM calls. The target URL provides clues as to the type of external call made.

Note: Topology View is available for nodes under Applications and Resources nodes. The Topology tab displays the Topology View associated with a specific application.

2.22 Architecture View

Architecture View is another type of AppSchema Visualization; a visual way for CAMM to represent the information stored in its AppSchema model. This view is designed to help you understand the structure and behaviors of J2EE and SOA applications at the module and component level. Some Architecture Views also include built-in delay analysis to help identify potential bottlenecks in a given call path.

The Architecture View in CAMM is capable of showing application structure and component relationships at two levels: module and component levels. At each level, CAMM can show both active and potential call paths. Table 2–8 describes various types of Architecture Views.

Drill down on a specific application to launch into the Architecture View. This action demonstrates the logical progression of drilling from high level resource-centric topology view, down through application-centric topology view, to module-centric architecture view. Using this logical drill down, you can understand the structure of your application runtime environments and diagnose problems.

Table 2–8 Various Types of Architecture View

Tab Name	Description
Module Level Execution	This is the default Architecture View at the module level. The Module Level Execution view shows the active calling relationships among various J2EE modules (EAR, WAR, JAR, and so on). Shared resources are also included.
Module Level	The Module Level view shows the potential calling relationships among various J2EE modules. Shared resources are also included. It should also be noted that any object that is not connected within the static view will not be included at this level and if there are no static connections at all between objects, every potential object relationship will be displayed.
Component Level Execution	This is the default Architecture View at the component level. The Component Level Execution view shows the active calling relationships among different J2EE components (EJB, servlet, JSP, and so on). Shared resources are also included.
Component Level	The Component Level view shows the potential calling relationships among various J2EE components. Shared resources are also included. Similar to the module level, any object that is not connected within the static view will not be included at this level and if there are no static connections at all between objects, every potential object relationship will be displayed.

These various types of architecture views are color coded in order to provide additional information. Table 2–9 lists color codes and their meanings.

Table 2–9 Architecture View Color Codes

Background Color	Description
Orange	The orange background color represents entry points into the application or module. The orange color also represents that these entities belong to the same application or module currently selected (in context).
Green	The green background color represents entry points into the application or module. The green color also represents that these entities belong to other applications or modules (out of context). The green color is also used to represent share resources.
White	The white background color represents that these entities belong to the same application and module currently selected (in context).
Blue	The blue background color represents that these entities belong to other applications and modules (out of context). The blue color also represents shared resources.

CAMM graphically depicts active calling relationships among various J2EE modules and shared resources.

2.22.1 Accessing the Architecture View

There are several ways to access the Architecture View. One way is through the Deployments node associated with a specific application under the Application Node. Application specific Architecture View can be accessed using the Deployments node on the Oracle Tree.

The Architecture View can also be accessed as part of a diagnostic drill down from the Operations Dashboard. Double-click on the problematic row on the Operations Dashboard to see a pop-up window with detailed performance data. Further double-clicking takes you to the appropriate Architecture View.

The last way to access the Architecture View is by right-clicking a managed entity and selecting the Architecture View. Right-click and select Architecture View to start the drill down process.

2.22.1.1 Arrows in Architecture Views

The arrows connecting various entities in the Architecture Views depict calls made from one entity to another. You can get more information about a specific call by pointing the mouse over a specific arrow. Mousing over arrows shows the details of a specific call in Architecture View

It is possible to hide different types of arrows in the Architecture View. To do this, right-click on the Architecture View and highlight the Edge types option to reveal a list of different edge (arrow) types associated with current Architecture View. Unchecking a specific edge type hides all lines of that type in the Architecture View. Checking a specific edge type makes these lines appear.

See Table 2–6 for the color descriptions of edge types. To hide all lines not connected with a specific entity, select a monitored entity in the Architecture View, right-click and select **Hide other edges**. Highlight an entity and select **Hide other edges** to hide all arrows not connected to the managed entity.

2.22.1.2 Architecture View Summary

The Architecture View Summary provides the delay analysis associated with the active call path displayed. The table and pie chart displayed in the right pane guides you to leading delay contributors in the displayed call path. Selecting a specific component in

the call path brings up component specific information. You will see the following tabs:

- Summary tab first which includes high-level delay data for both inbound and outbound calls.
- The Instrumentation tab shows detailed method level performance data associated with the selected component. Click the Instrumentation tab to see detailed performance measurements and information at the method level.
- The Errors/Exceptions tab shows the errors metrics associated with the selected portal or BPEL process.
- The SQL Statement tab shows SQL statements and their performance data associated with the selected component.
- The Transactions tab shows the transaction events associated with the selected portal and children below.

2.23 Metric Types

Table 2–10 describes various types of metrics provided by CAMM.

Table 2–10 Metric Types

Examples	Metric Type	Metric Description
Active Sessions Completions Pending Requests Running Instances Max Capacity Messages High	Snapshot Count	A count of the monitored entity at a point in time. CAMM plots these snapshot counts in trend graphs.
Requests Serviced Total Sessions [Processes] Aborted [Processes] Terminated [Method] Invocation Count Bytes Received	Aggregated Count	A count of the monitored entity incrementally aggregated from the beginning of display time window. CAMM shows these aggregated counts in summary tables.
Response Time Elapse Time Connection Delay	Average Timing	Calculated every sampling period (default 60 seconds), the average timing is calculated by dividing the total amount of time needed to complete the monitored business unit of work by the number of completed business units of work. CAMM uses this data in the following two ways: 1. Plot the average timings in trend graphs. 2. Calculate average timing of this business unit of work for the display time window and display in a summary table.
Min/Max	Minimum and Maximum Response Time Measurement	Minimum and maximum response time measurements found per collection sampling intervals. These are stored in their embedded database in addition to average response time measurements. The default is 60 seconds.

3

Exploring the Monitor Workspace

When Oracle Composite Application Monitor and Modeler (henceforth referred to as CAMM) is pointed to a Oracle WebLogic domain, IBM WebSphere cell, or an Oracle SOA Suite cluster, it automatically discovers information about this particular domain including all deployed applications, configuration, resources, and others. CAMM displays this information in the Monitor Workspace under Oracle Tree.

Each node represents a construct in the platforms monitored by CAMM. Each construct is described in this chapter.

This chapter includes the following topics:

3.1 Oracle WebLogic Portals

The Portals node under Oracle Tree contains information about all deployed WebLogic Portal applications in the managed domains. The Portals node is organized hierarchically using the same framework developers use to build these Portal applications. The minimum and maximum response time measurements are stored in

the embedded database in addition to the average response time measurements. These metrics, if present, display visually in the window on the right pane.

For WebLogic Portal, this hierarchy contains the following (Table 3–1):

Table 3–1 WebLogic Portal Hierarchy

Component	Description
Portals	The Portal is the logical containment unit for a Portal application. A typical Portal can contain a few desktops, several books, tens of pages, and hundreds of portlets.
Desktops	The desktop is the top-level container for the portal components included in that specific view of the portal. Portal administrators can create new desktops beyond what portal developers create in WebLogic Workshop.
Books	The top-level book contains all sub-books, pages, and portlets. The top-level book defines the initial menu navigation style used for the desktop. For each sub-book you add to a desktop you can select a different navigation style.
Pages	Pages and sub-books are the navigable containers used for organizing portlets.
Portlets	Portlets are the containers that surface Web content and applications in your desktops.

When you click the Portals node under Oracle Tree, CAMM displays summary information on active portal applications. This summary includes the following (Table 3–2):

Table 3–2 Tree Summary

Metric	Description
Portal web application activity	A summary of user sessions for a specific portal application
Portal completions	Total number of requests fulfilled by a specific portal application
Portal response time (ms)	Average response time for a specific portal application
Portal entitlement response time	Average response time of WebLogic Portal entitlement subsystem for a specific portal application
Portal campaign response time	Average response time of WebLogic Portal campaign subsystem for a specific portal application

For Portal web application activity and Portal performance, CAMM displays information in both table and graph formats. For the other metrics, CAMM shows the information in graph format. When you click the plus (+) icon next to the Portals node, CAMM expands the tree to show all managed portal applications currently deployed on the WebLogic domain.

You can also see information specific to a particular portal application. By selecting a specific portal application, all information displayed in the Main Display Window changes to only show data relevant to this new context. For example, when a user selects a particular portal application under the Portals node, the Main Display Window only shows information specific to that portal application.

At the Portal level, you can navigate to different levels of the portal application by using different tabs. Use the tabs available to quickly access lower level components. Table 3–3 provides a list of the tabs available for portal level nodes and their descriptions.

Table 3–3 Portal Level Tab Description

Tab	Description
Summary	Performance summary specific to the selected portal.
Desktops	Performance summary for all the desktops associated with the selected portal.
Headers	Performance summary for all the headers associated with the selected portal.
Books	Performance summary for all the books associated with the selected portal.
Pages	Performance summary for all the pages associated with the selected portal.
Portlets	Performance summary for all the portlets associated with the selected portal.
Footers	Performance summary for all the footers associated with the selected portal.
WSRP Topology	View WSRP consumer-producer relationships and WSRP deployment topology.
Analysis	Two performance analytics - Multi-Point Regression Analysis performed at the portal level and Entity Performance Ranking performed at the portlet level.
Events	SLO violation events associated with the selected portal.
Errors/Exceptions	Errors metrics associated with the selected portal.
Transactions	Transaction events associated with the selected portal and children below.

3.1.1 Desktops

Expand a particular portal application further to reach the Desktops node. By selecting the Desktops node, CAMM provides a list of currently active desktops associated with that portal application.

This Desktop Summary includes the following metrics:

Table 3–4 Desktop Summary Metrics

Metrics	Description
Desktop arrivals	Total number of requests for a specific desktop
Desktop completions	Total number of requests fulfilled by a specific desktop
Desktop response time (ms)	Average response time for a specific desktop.

Note: Portal desktops are end-user facing entities. Metrics such as Desktop hits and response time represents request arrival rate and application performance respectively. Violations in thresholds set on these metrics would indicate unacceptable end-user experience.

CAMM displays these metrics in both table and graph formats.

For example, when you have two active desktops, you can drill down further to a specific desktop by expanding the Desktops node. Again, clicking on the plus (+) icon expands the tree view for you.

When you select a node in the expanded tree to get more information specific for that desktop, CAMM changes information in the Main Display Window to reflect the new context.

CAMM not only shows the performance metrics associated with a specific node, but it also displays other relevant settings for that node. For example, there can be

pre-configured Service Level Objectives (SLOs). These SLOs are displayed in the graphs as red lines.

Expand the desktop node to see Header, Footer, and Books. You can see detailed information for these components by clicking on the appropriate nodes.

3.1.1.1 Display Portal Desktop - Desktop Structure Viewer

One of the unique capabilities of CAMM is its automatic discovery and modeling of deployed applications. The Desktop Structure Viewer provides visibility into how a portal desktop is organized. To activate the Desktop Structure Viewer, right-click on a specific desktop. Select the Display Portal Desktop menu option to access the Desktop Structure Viewer.

After the Desktop Structure Viewer appears, you can navigate through the portal desktop structure by clicking on the appropriate book, page, or portlet. The ability to see portal desktop structure using the same perspective as portal end-users is a unique value especially for the IT support staff.

With the Desktop Structure Viewer, the IT support staff can speak the same language with end-users while at the same time looking at performance oriented information for a specific component. The IT support staff can also use the Desktop Structure Viewer to isolate a particular performance problem. By drilling down from the top-level desktop to individual portlets, the IT support staff can get more insight into which components are having performance problems.

The Desktop Structure Viewer consists of two main panes. The pane on the left is the Desktop Structure pane. This pane allows you to graphically navigate the portal desktop. The pane on the right is called the Main Display Window. The Main Display window displays performance information in the context of the selected component in the Desktop Structure pane. As you navigate through the portal desktop and click different components, the Main Display Window provides information relevant for that selected context.

The Main Display Window shows relevant performance metrics for different portal desktop components - desktop, books, pages, and portlets.

Since CAMM understands the WebLogic Portal framework and knows that a pageflow can be associated with a portlet, it is designed to allow easy access to the Pageflow Viewer from the Desktop Structure Viewer.

To activate the Pageflow Viewer, double-click the interesting portlet. In turn you can double-click the portlet in the Desktop Structure pane to open the appropriate pageflow in the Pageflow Viewer.

3.1.2 Portlet Drill Down

You can drill down on a portlet in the portal desktop view to activate the Display Architecture View.

1. Select a portlet under a node.
2. Double-click on a name to see the Portal Desktop Status page.
3. In the Portal Desktop Status window, right-click on a service box to select Display Architecture View.
4. See Section 5.4, "Drill Down - Bottleneck Analysis" on how to use the architecture view.

3.1.3 Pageflow Viewer

The Pageflow Viewer has two panes. The pane on the right is the Main Display Window. The Main Display Window shows information corresponding to the item selected in the left pane. The left pane shows either the Flow View or the Component View. You can choose to see either the Flow View or the Component View by selecting the appropriate tab.

The Main Display Window changes to show information relevant to the selected item in either the Flow View or the Component View.

Another way to open the Pageflow Viewer is through the operational dashboard by double-clicking the interested pageflow. The Flow View of the pageflow displays in a newly created Pageflow Viewer.

3.1.4 Books

Expand a particular portal desktop further to see the Books node. By selecting the Books node, CAMM provides a list of currently active books associated with the specific desktop.

This Books Summary includes the following metrics (Table 3–5):

Table 3–5 Book Summary Metrics

Metrics	Description
Book completions	Total number of requests fulfilled by a specific book
Book response time (ms)	Average response time for a specific book

CAMM displays these metrics in both table and graph formats. For example, you can have two active books for the portal desktop. These active books are listed in the table and plotted in the graphs.

You can drill down further to a specific book by expanding its node. Click the plus (+) icon to expand the tree view. Expand the Books node to see a list of specific books configured.

When you select a particular active book, the Main Display Window shows the relevant information in that context.

3.1.5 Pages

Expand a particular book to see the Pages node. By selecting the Pages node, CAMM provides a list of currently active pages associated with the specific book.

This Pages Summary includes the following metrics (Table 3–6):

Table 3–6 Pages Summary Metrics

Metrics	Description
Page completions	Total number of requests fulfilled by a specific page
Page response time (ms)	Average response time for a specific page

CAMM displays these metrics in both table and graph formats. For example, you can have one active page for a book. The active page is listed in the table and plotted in the graphs.

You can drill down further to a specific page by expanding the Pages node. Click the plus (+) icon to expand the tree view. This reveals the next level of components - Portlets.

3.1.6 Portlets

Expand a particular page to see the Portlets node. Select a Portlets node to view a list of currently active portlets associated with the specific page.

This Portlets Summary includes the following metrics (Table 3–7):

Table 3–7 Portlet Metrics

Metrics	Description
Portlet completions	Total number of requests fulfilled by a specific portlet.
Portlet response time (ms)	Average response time for a specific portlet.

CAMM displays these metrics in both table and graph formats. For example, you can have four active portlets for a particular page. These active portlets are listed in the table and plotted in the graphs.

Drill down further to a specific page by expanding the Portlets node. Click the (plus) + icon to expand the tree view. This provides additional information about the page.

3.2 WebSphere Portals

The Portals node under Oracle Tree contains information about all deployed WebSphere Portal applications in the managed cells. The Portals node is organized hierarchically using the same framework developers use to build these Portal applications. The minimum and maximum response time measurements are stored in the embedded database in addition to the average response time measurements. These metrics, if present, display visually in the window on the right pane.

For WebSphere Portal, this hierarchy contains the following (Table 3–8):

Table 3–8 WebSphere Portal Hierarchy

Component	Description
Portals	The Portal is the logical containment unit for a Portal application. A typical Portal can contain a few desktops, several of books, tens of pages, and hundreds of portlets.
WebSphere	The WebSphere is the top-level container for the portal components included in that specific view of the portal. Portal administrators can create new desktops beyond what portal developers create in WebLogic Workshop.
Virtual Portals	The top-level book contains all sub-books, pages, and portlets. The top-level book defines the initial menu navigation style used for the desktop. For each sub-book you add to a desktop you can select a different navigation style.
Content Root	Pages and sub-books are the navigable containers used for organizing portlets.
Header	Portlets are the containers that surface Web content and applications in your desktops.
Pages	Pages are containers within virtual portals, books, and sub-books. Pages often contain labels and portlets.
Labels	Labels are markers defining content within page containers.

When you click the Portals node under Oracle Tree, CAMM displays summary information on active portal applications. This summary includes the following (Table 3–9):

Table 3–9 WebSphere Tree Summary

Metrics	Description
Portal web application activity	A summary of user sessions for a specific portal application.
Portal completions	Total number of requests fulfilled by a specific portal application.
Portal response time (ms)	Average response time for a specific portal application.

For Portal web application activity and Portal performance, CAMM displays information in both table and graph formats. For the other metrics, CAMM shows the information in graph format.

When you click the plus (+) icon next to the Portals node, CAMM expands the tree to show all managed portal applications currently deployed on the WebLogic domain.

You can also see information specific to a particular portal application. By selecting a specific portal application, all information displayed in the Main Display Window changes to only show data relevant to this new context. For example, when a user selects the *WebSphere* portal application under the Portals node, the Main Display Window only shows information specific to WebSphere portal application.

At the Portal level, you can navigate to different levels of the portal application by using different tabs. Use the tabs available to quickly access lower level components.

The following is a list of the tabs available for portal level nodes and their descriptions (Table 3–10).

Table 3–10 Portal Level Tab Descriptions

Tab	Description
Summary	Performance summary specific to the selected portal
Analysis	Two performance analytics - Multi-Point Regression Analysis performed at the portal level and Entity Performance Ranking performed at the portlet level
Events	SLO violation events associated with the selected portal
WSRP Topology	View WSRP consumer-producer relationships and WSRP deployment topology
Errors/Exceptions	Errors metrics associated with the selected portal
Instrumentation	Includes performance data by different types of instrumentation probe points. There are different tabs available: Class, Method, Errors/Exceptions and Transactions. Each tab includes basic information such as Probe Point Name, Invocation Count, and Response Time. This detailed performance data can help you identify low-level bottlenecks. Refer Section 3.5.3, "Instrumentation" for more details.

3.2.1 Virtual Portals

To reach the Virtual Portals node, further expand a particular portal application. By selecting this node, CAMM provides a list of currently active portals associated with that portal application. See to view the Summary for the WebSphere portal application.

This Summary includes the following metrics (Table 3–11):

Table 3–11 Virtual Portals Summary Metrics

Metrics	Description
Virtual Portal Completions	Total number of requests fulfilled by a specific portal.
Virtual Portal Response Time (ms)	Average response time for a specific portal.

CAMM displays these metrics in both table and graph formats. For example, you can have one active portal for the WebSphere portal application. The Content Root is listed in the table and Plotted in the graphs.

You can drill down further to specific portlets by expanding the Content Root node. Again, clicking on the plus (+) icon expands the tree view for you. CAMM changes information in the Main Display Window to reflect new context - portlet.

CAMM not only shows the performance metrics associated with a specific node, but it also displays other relevant settings for that node.

3.2.1.1 Display Virtual Portal - Structure Viewer

One of unique capabilities of CAMM is its automatic discovery and modeling of deployed applications. The Structure Viewer provides visibility into how a portal desktop is organized. To activate the Virtual Portal Viewer, right-click on a specific portal. Select the Display Virtual Portal menu option to access the viewer.

After the Structure Viewer appears, you can navigate through the portal structure by clicking the appropriate header. The ability to see the portal structure using the same perspective as portal end-users is a unique value especially for the IT support staff.

With the Structure Viewer, the IT support staff can speak the same language with end-users while at the same time looking at performance oriented information for a specific component. The IT support staff can also use the Structure Viewer to isolate a particular performance problem. By drilling down from the top-level desktop to individual portlets, the IT support staff can get more insight into which components are having performance problems.

The Structure Viewer consists of two main panes. The pane on the left is the Structure Viewer pane. This pane allows you to graphically navigate the portal desktop. The pane on the right is called the Main Display Window. The Main Display window displays performance information in the context of the selected component in the Desktop Structure pane. As you navigate through the portal desktop and click different components, the Main Display Window provides information relevant for that selected context.

The Main Display Window shows relevant performance metrics for different portal components.

3.2.2 Pages

Expand a particular portal to see the Pages node. By selecting the Pages node, CAMM provides a list of currently active pages associated with the specific book.

This Pages Summary includes the following metrics (Table 3–12):

Table 3–12 Pages Summary Metrics

Metrics	Description
Page Completions	Total number of requests fulfilled by a specific page
Page Response Time (ms)	Average response time for a specific page

CAMM displays these metrics in both table and graph formats.

3.2.3 Portlets

Expand a particular page to see the Portlets node. Select a Portlets node to view a list of currently active portlets associated with the specific page.

This Portlets Summary includes the following metrics (Table 3–13):

Table 3–13 Portlet Metrics

Metrics	Description
Portlet Completions	Total number of requests fulfilled by a specific portlet
Portlet Response Time (ms)	Average response time for a specific portlet

CAMM displays these metrics in both table and graph formats. For example, you can have four active portlets for the Content Root page. These active portlets are listed in the table and plotted in the graphs.

Drill down further to a specific page by expanding the Portlets node. Click the plus (+) icon to expand the tree view.

3.3 Oracle BPEL Processes

The BPEL Processes node in the navigation tree contains information about all deployed Oracle BPEL processes within the managed domain. CAMM organizes information for various process nodes into domains.

In the right-hand pane, you can view the minimum and maximum response time measurements stored in the embedded database in addition to the average response time, arrivals, errors, and completions measurements. These metrics, if present, display visually in the window on the right pane.

When you select the root of the BPEL Processes tree, CAMM displays the BPEL Processes Summary in the Main Display Window.

The BPEL Process Summary includes the following (Table 3–14):

Table 3–14 BPEL Process Summary Metrics

Metrics	Description
Domain	Name of the OC4J domain container
Process	Name of the BPEL process
Arrivals	Total number of currently running instances for a specific BPEL process
Response Time (ms)	Average response time in milliseconds for a specific BPEL process
Completions	Total number of fulfilled requests for a specific BPEL process. A Completed status represents a BPEL process instance that has finished normally.
Errors	Total number of aborted instances of a specific BPEL process
Min Response Time (ms)	Minimum average response time in milliseconds for a specific BPEL process
Max Response Time (ms)	Maximum average response time in milliseconds for a specific BPEL process

CAMM presents these metrics in a table format in the Main Display Window when you select the BPEL Processes node. Graphical representations of two metrics, Arrivals and Completions, are displayed below the table.

When you click the plus (+) icon next to the domains sub-node under the main BPEL Processes node, CAMM expands the tree to show all managed BPEL domains currently deployed on that particular Oracle SOA Suite instance.

You can see information specific to a particular process. By selecting a specific process, all information displayed in the Main Display Window changes to only show data relevant to this new context.

To see the BPEL process work flow associated with a BPEL process, select the node, right-click and select the Display Functional View option. CAMM displays the appropriate functional work flow diagram and associated performance data in a new pop-up window.

See Table 3–15 for BPEL Functional View summary.

Table 3–15 BPEL Functional View Summary

Column/Metric	Description
Activity	Name of a specific activity in the BPEL process
Type	Control Type for a specific node
Arrivals	Number of requests that have arrived for a specific node
Response Time (ms)	Average response time for a specific node
Completions	Number of completed requests for a specific node
Errors	Number of aborted instances for a specific node
Response Time Min (ms)	Minimum response time for a specific node
Response Time Max (ms)	Maximum response time for a specific node

By looking at this summary table, you can determine which BPEL process node is running slowly and whether there are errors.

In addition to the summary, the following views are available for a node:

- Delay Analysis view
- Metadata view
- Partner Links view
- Partner Link Type Role view
- Partner Link Bindings view
- Modeled Entities view
- Topology view

You can get to these views by selecting the appropriate tab.

3.3.1 Delay Analysis View

Delay Analysis gives you a bird's eye view of a specific BPEL process. You can see what nodes in the BPEL process are taking up a majority of the average elapsed time. The red bar indicates the slowest BPEL process group or BPEL process node. The blue represents the time spent for the particular nodes.

3.3.2 Metadata View

The Metadata view displays the tables containing specific metadata associated with the selected active BPEL process being displayed in the left-hand pane. Information provided in this view includes caller and called class metadata information as well as general summarized metadata in relation to the BPEL process and the associated web services. Table 3–16 explains the metadata.

Table 3–16 Metadata View Summary

Column/Metric	Description
SummaryTable -Process	Name of the BPEL process node
SummaryTable -Web Service	Name of the web service being called from the BPEL process
SummaryTable -Version	Version of the web service being called from the BPEL process
SummaryTable -Location	Location of the web service being called from the BPEL process
Caller Table - Caller Class	Class name for the caller class that is calling the BPEL process
Caller Table - Caller Method	Class method for the caller class that is calling the BPEL process
Caller Table -Target Host	Target host that the caller class targeted to instantiate the BPEL process
Caller Table -Target Port	Target port that the caller class targeted to instantiate the BPEL process
Caller Table -Target URL	Target URL that the call class targeted to instantiate the BPEL process
Caller Table - Invocation Count	Number of invocations of the BPEL process instantiated by the caller class
Caller Table - Response Time	Average response time of the BPEL process instantiated by the caller class
Called Clients Table - Called Class	Class name of the class that was called by the BPEL process
Called Clients Table - Target URL	Target URL of the class that was called by the BPEL process
Called Clients Table - Invocation Count	Number of invocations made from the BPEL Process to the called class.
Called Clients Table - Response Time	Response time of the called class

3.3.3 Partner Links View

The partner links view provides detailed information on the various roles related to how and why the partner link service is being utilized. The information provided includes both the caller and callee roles, as well as the partner link type. See Table 3–17.

Table 3–17 Partner Links View Summary

Column/Metric	Description
Partner Link	Name of the partner link
My Role	Role in regards to the BPEL process calling the partner link service
Partner Role	Role of the partner link service
Partner Link Type	Partner link category (type) of the service being called

3.3.4 Partner Link Type Role View

See Table 3–18 describes the columns in the Partner Link Type Role view.

Table 3–18 Partner Link Type Role View Summary

Column/Metric	Description
Name	Name of the partner link
Link Type Name	Category (type) of the partner link
Port Type	Partner link service URL

3.3.5 Partner Link Bindings View

The Partner Link Bindings view provides insight into the actual roles and types of the partner link instances which represent web services that have been bound by the BPEL process. See Table 3–19.

Table 3–19 Partner Link Bindings View Summary

Column/Metric	Description
Partner Link Role	Defines the web service role that the BPEL process will communicate with
Partner Link Type	Defines the web service type that the BPEL process will communicate with
WebService PortType	Name of the web service
WebService Port Namespace ID	URL of the webservice instance

3.3.6 Modeled Entities View

The modeled entities view consist of a list and count of the general entities as catalogued during the discovery phase of the resource configuration. The tables contain both a total entity count as well as a breakdown of the entity count by entity type. See Table 3–20.

Table 3–20 Modeled Entities Summary

Column/Metric	Description
Total Entities Modeled Table - Total	Total entities (static label)
Total Entities Modeled Table - Count	Total number of entities catalogued during the discovery phase of the BPEL process
Modeled Entities Table - Entity Type	Entity type being catalogued as part of the discovery phase of the BPEL process
Modeled Entities Table - Count	Total number of entities catalogued during the discovery phase of the BPEL process for a particular entity type

3.3.7 Topology View

The Topology View utilizes the modeled entities that were captured during the discovery process to provide a bird's eye view of all of the various high-level relationships between BPEL processes, web services, and business services. You can toggle between static and dynamic relationship views using the tabs at the top of the Topology pane.

3.3.8 Node Hierarchy

Expanding a particular BPEL process further, the first item you see is the Node Hierarchy node. By selecting the Node Hierarchy node, CAMM provides a list of nodes associated with the specific process.

When you click the plus (+) icon next to a specific Node Hierarchy node, CAMM expands the tree to show BPEL process nodes in the Node Hierarchy. Click an individual BPEL process node to see the load and performance of the selected node in the Main Display Window.

The BPEL process node information also includes the name of the method invoked. This information is displayed as part of the summary table at the top of the main view window.

3.4 Oracle ESB

The Oracle ESB node under Oracle Tree contains information about all of the deployed Oracle ESB servers running in the managed domain. CAMM organizes the information for various Oracle ESB nodes into various categories.

When you select the root of the ESB tree, CAMM displays the ESB Summary in the Main Display Window.

The ESB Summary includes the following (Table 3–21):

Table 3–21 ESB Summary Metrics

Metric	Description
ESB System	Name of ESB System
ESB Service	Name of the ESB Service identifier
Arrivals	Total number of ESB service instance arrivals
Completions	Total number of ESB service instance completions
Response Time	Total number of completed instances for a specific BPEL process. A Completed status represents a BPEL process instance that has finished normally.

CAMM presents these metrics in a table format in the Main Display Window when you select the ESB node. When you click the plus (+) icon next to the ESB Systems sub-node under the main ESB node, CAMM expands the tree to show all managed ESB Systems currently deployed on that particular Oracle SOA Suite instance.

You can see information specific to a particular ESB System. By selecting a specific ESB System, all information displayed in the Main Display Window changes to only show data and the topology relevant to this new context.

By looking at the summary table, you can find out which ESB node is running slowly and whether there are errors.

Besides the summary, the following views are available for the Node Hierarchy node:

- Service Details view
- Service Parent Details view
- Service Definition view
- Service Operations view
- Operation Routing Rules view
- Topology view

You can get to these views by selecting the appropriate tab.

3.4.1 Service Details View

The Service Details view provides specific information related to the details of the bound service process instances. Instance IDs and other descriptive details are included as part of this view. See Table 3–22.

Table 3–22 Service Details View Summary

Column/Metric	Description
Service Name	Name of the ESB service
GUID	GUID of the ESB service
Qname	Canonical qualified name for the bound ESB service
Description	Description of the ESB service

3.4.2 Service Parent Details View

The Parent Service Details view provides specific information related to the details of the parent of the bound service process instances. Instance IDs, roles, and other descriptive details are included as part of this view. See Table 3–23.

Table 3–23 Service Parent Details View Summary

Column/Metric	Description
Service Name	Name of the parent ESB service
ParentGUID	GUID of the parent ESB service
ParentQname	Canonical qualified name for the parent of the bound ESB service
ParentType	Parent type of the parent ESB service
MyRole	Role of the caller of the parent ESB service instance
ParentRole	Role of the callee of the parent ESB service instance

3.4.3 Service Definition View

The Service Definition view contains information regarding the bound ESB service including the Business Service (ESB) WSDL and Port Type as well as the associated URLs. See Table 3–24.

Table 3–24 Service Definition View Summary

Column/Metric	Description
Service Name	Name of the ESB service
BusinessServiceWSDL	URL of the Business Service WSDL
BusinessServicePortType	Port type of the Business Service
ConcreteServiceWSDL	URL of the Concrete Service WSFL
ConcreteServiceURI	URI for the concrete service

3.4.4 Service Operations View

The Service Operations views provides details regarding the various method operations being executed. All information is provided in regards to the metadata associated with a specific business service instance. See Table 3–25.

Table 3–25 Service Operations View Summary

Column/Metric	Description
Service Name	Name of the ESB service
Name	Service operation name being executed
GUID	GUID of the ESB service
Qname	Canonical qualified name for the bound ESB service
Element	Associated element within the ESB Service
SchemaLocation	Schema location for the associated ESB service
Type	Type of ESB service operation

3.4.5 Operation Routing Rules View

The Operation Routing Rules view provides various details regarding the operation routing rules for Business Service operations. This includes the specific instance business service names being utilized for operations. See Table 3–26.

Table 3–26 Operation Routing Rules View Summary

Column/Metric	Description
Service Name	Name of the ESB service
Name	Instance name ID of the ESB service instance
GUID	GUID of the ESB service instance

3.5 Processes

The Processes node under Oracle Tree contains information about all deployed WebLogic business processes in the managed domain. CAMM organizes information for various process nodes into the following major categories:

- Node Hierarchy
- Persistent Containers
- Instrumentation

The minimum and maximum response time measurements are stored in the embedded database in addition to the average response time measurements. These metrics, if present, display visually in the window on the right pane.

When you select the root of the Processes tree, CAMM displays the Processes Summary in the Main Display Window. See Table 3–27.

Table 3–27 Process Summary Metrics

Metrics	Description
Process	Name of process
Running	Total number of currently running instances for a specific process
Suspended	Total number of suspended instances for a specific process. A Suspended request from a user is a common cause for a process instance to go into a Suspended state.
Frozen	Total number of frozen instances for a specific process
Completed	Total number of completed instances for a specific process. A Completed status represents a process instance that has finished normally.

Table 3–27 (Cont.) Process Summary Metrics

Metrics	Description
Aborted	Total number of aborted instances for a specific process
Terminated	Total number of terminated instances for a specific process. An external Terminate request would terminate a process instance.
Average Execution Time (ms)	Average execution completion time for a specific process

Tip: Statistics on the number of process instances with Terminated, Aborted, and Frozen states can indicate abnormal operation of the WebLogic Integration application or container. It is possible to unfreeze Frozen process instances from WLI Console.

CAMM presents these metrics in a table format in the Main Display Window when you select the Processes node. Graphical representations of two metrics, Running Instances and Average Execution Time, are displayed below the table.

When you click the plus (+) icon next to the Processes node, CAMM expands the tree to show all managed processes currently deployed on the WebLogic domain.

You can see information specific to a particular process by selecting a specific process. All information displayed in the Main Display Window changes to only show data relevant to this new context.

To see the process work flow associated with a particular process, select the process node, right-click and select the Display Functional View option. CAMM displays the appropriate functional work flow diagram and associated performance data in a new pop-up window.

3.5.1 Node Hierarchy

When expanding a particular process further, the first item you see is the Node Hierarchy node. By selecting the Node Hierarchy node, CAMM provides a list of nodes associated with the specific process. See Table 3–28.

Table 3–28 Node Hierarchy Summary

Column/Metric	Description
Node	Name of a specific node
ID	Process Node ID for a specific node
Type	Control Type for a specific node
Method	Node Method Name for a specific node
Arrivals	Number of Requests Arrived for a specific node
Active	Number of Active Instances for a specific node
Elapsed Time (ms)	Average Time Elapsed to Complete an Instance for a specific node
Completions	Number of Completed Instances for a specific node
Aborts	Number of Aborted Instances for a specific node
Exceptions	Number of Exception Encountered for a specific nod.

By looking at this summary table, you can determine which process node is running slowly and whether there are aborts or exceptions.

The following additional views are available for the Node Hierarchy node:

- Delay Analysis view
- Events view

You can get to these views by selecting the appropriate tab.

3.5.1.1 Delay Analysis View

Delay Analysis gives you a bird's eye view of a specific process. You can see what nodes in the process are taking up a majority of the average elapsed time. The red bar indicates the slowest process group or process node. The blue represents the time spent for the particular nodes.

3.5.1.2 Events View

The Events view shows a list of SLO violations events relevant to this process in a table format. The Events view table includes the following information (Table 3–29):

Table 3–29 Events View Summary

Column/Metric	Description
Start Time	Start time for the process instance that violated a SLO
Entity Name	Name of the process node that violated a SLO
SLO Name	Name of the violated SLO
Service URI	URI of the process that violated a SLO
Application	Name of the application that violated a SLO
Event Type	Violation type (violation or cautionary)
Entity Type	Violation Metric type
SLO Threshold	Type of threshold (high or low)
SLO Trigger Value	Value that triggered a SLO violation

When you click the plus (+) icon next to a specific Node Hierarchy node, CAMM expands the tree to show process nodes in the Node Hierarchy. Click an individual process node to see the load and performance of the selected node in the Main Display Window.

The process node information also includes the name of the method invoked. This information is displayed as part of the summary table at the top of the main view window.

3.5.2 Persistent Containers

When you expand a particular process further, the Persistent Containers node is included. By selecting the Persistent Containers node, CAMM provides a list of persistence performance statistics relevant to the selected process.

As you select the root of the Persistent Containers tree, a summary of all Persistent Containers relevant to the selected process is presented. For example, a summary can contain the following high level items:

- Container persistence invocations
- Container persistence response time (milliseconds)
- Entity EJB activity

- Entity EJB cache
- Entity EJB transactions
- Entity EJB locking

These items are displayed in both table and graph formats.

The Persistent Containers Summary includes different tables:

- Entity EJB Activity
- Entity EJB Cache
- Entity EJB Transactions
- Entity EJB Locking

3.5.2.1 Entity EJB Activity Table

Entity EJB Activity table (Table 3–30) includes the following information:

Table 3–30 Entity EJB Activity Table

Metrics	Description
EJB	Name of the Entity EJB
In Use	Number of instances for a specific Entity EJB currently being used from the free pool. [Snapshot Count]
Idle	Number of instances for a specific Entity EJB currently in the idle state in the free pool. These bean instances are available for use. [Snapshot Count]
Waits	Number of Threads currently waiting for a specific Entity EJB bean instance from the free pool [Snapshot Count]
Timeouts	Total number of Threads that have timed out waiting for an available bean instance from the free pool [Aggregated Count]

Tip: Pay attention to Waits and Timeouts metrics. Activities in the Waits metric and increasing count in the Timeouts metric are signs that requests waiting to be serviced by the EJB container. Ideally, 0 should be indicated for these metrics.

3.5.2.2 Entity EJB Cache Table

Entity EJB Cache table (Table 3–31) includes the following information:

Table 3–31 Entity EJB Cache Table

Metrics	Description
EJB	Name of the Entity EJB
Hits	Total number of times an attempt to access the Entity EJB instance from the cache succeeded [Aggregated Count]
Accesses	Total number of attempts to access the Entity EJB instance from the cache [Aggregated Count]
Size	Number of beans instances from this EJB Home currently in the EJB cache [Snapshot Count]
Activations	Total number of beans from this EJB Home that have been activated [Aggregated Count]
Passivations	Total number of beans from this EJB Home that have been passivated [Aggregated Count]

Tip: Passivation (serializing EJB state information to disk) and activation (reconstituting EJB state information from disk) are resource intensive operations. Ideally, it is preferable to see low level of activity in these metrics.

3.5.2.3 Entity EJB Transactions Table

Entity EJB Transactions table (Table 3–32) includes the following information:

Table 3–32 Entity EJB Transactions Table

Metrics	Description
EJB	Name of the Entity EJB
Commits	Total number of transactions that have been committed for this EJB. [Aggregated Count]
Rollbacks	Total number of transactions that have been rolled back for this EJB. [Aggregated Count]
Timeouts	Total number of transactions that have timed out for this EJB. [Aggregated Count]

Tip: High number of EJB Transaction Rollbacks may indicate problems with the data used - for some reason the target database is unable to commit the change. High number of EJB Transaction Timeouts may indicate problems accessing the database including network outage, database lock contention, database outage, and more.

3.5.2.4 Entity EJB Locking Table

Entity EJB Locking table (Table 3–33) includes the following information:

Table 3–33 Entity EJB Locking Table

Metrics	Description
EJB	Name of the Entity EJB
Entries	Number of Entity EJB instances currently locked [Snapshot Count]
Lock Accesses	Total number of attempts to obtain a lock on an Entity EJB instance [Aggregated Count]
Current Waiters	Number of Threads that currently waiting for a lock on an Entity EJB instance [Snapshot Count]
Total Waiters	Total number Threads that have waited for a lock on an Entity EJB instance [Aggregated Count]
Timeouts	Total number Threads that have timed out waiting for a lock on an Entity EJB instance [Aggregated Count]

Tip: Pay attention to Current Waiters and Timeouts. These metrics can indicate possible performance problems caused by EJB Locking. Ideally, 0s should be displayed for these metrics.

By looking at the activities related to Persistence Containers, you can determine if EJB persistence calls are causing performance problems.

3.5.3 Instrumentation

When expanding a particular process further, the last item you see is the Instrumentation node. Click the plus (+) icon next to Instrumentation to expand the tree to reveal the following categories of instrumentation:

- Class
- Methods
- Errors/Exceptions
- Transactions

The Class node in the Instrumentation tree provides the following information (Table 3–34):

Table 3–34 Class Node

Column/Metric	Description
Probe Point	Class name in which instrumentation probe point is inserted
Response Time (ms)	Average response time for a specific class
Invocation Count	Number of times a specific class is called

The Method node in the Instrumentation tree provides the following information (Table 3–35):

Table 3–35 Method Node

Column/Metric	Description
Probe Point	Method name in which instrumentation probe point is inserted
Response Time (ms)	Average response time for a specific method
Invocation Count	Number of times a specific method is called

The Errors/Exceptions and Transactions are described in Section 2.22, "Architecture View".

3.6 Web Services

The Web Services node in the navigation tree contains information about all deployed Web Services in the managed domain. By selecting the Web Services node under Oracle Tree, CAMM shows the Web Services Summary in the Main Display Window.

This summary view lists all discovered web services and their associated URL entry points. Below this list, CAMM lists out all active web services and their performance data (invocation count and response time).

When you click the plus (+) icon next to the Web Services node, CAMM expands the tree to show all monitored web services currently deployed on the WebLogic domain.

When you select a specific web service, CAMM displays performance data associated with the selected web service. Click the plus (+) icon next to a specific web service to expand the tree to show all public operations associated with that web service.

The Operations table provides the following information (Table 3–36):

Table 3–36 Operations Table

Column/Metric	Description
Operation	Name of the web service operation
Invocation Count	Number of times the operation is called

Table 3–36 (Cont.) Operations Table

Column/Metric	Description
Response Time (ms)	Average response time for the operation in milliseconds
Delay (ms)	Overall delay contributed by the operation in milliseconds

3.7 Pageflows

The Pageflows node in the navigation tree contains information about all deployed pageflows in the managed domain. By selecting the Pageflows node under Oracle Tree, CAMM shows the Pageflows Summary in the Main Display Window.

3.8 Services

The Services node in the navigation tree contains information about all external entry points into the managed domain. CAMM currently monitors the following different types of services:

- HTTP
- EJBs
- JDBC

Selecting each service type reveals service summary in the Main Display Window.

The minimum and maximum response time measurements are stored in the embedded database in addition to the average response time measurements. These metrics, if present, display visually in the window in the right pane.

CAMM displays entry point activity summary associated with the selected EJB service.

> **Tip:** Setting thresholds at some of these entry points enables CAMM to monitor the performance of key business services. When a violation event occurs, you can begin investigating from the Service node.

3.8.1 HTTP

Expanding the HTTP node under the Services node reveals a list of discovered HTTP based entry points into the managed domain. HTTP service end points include JSPs, struts actions, and servlet mappings. These discovered HTTP entry points are listed by their root context. When you select a specific HTTP entry point, CAMM displays the associated summary in the Main Display Window.

CAMM displays the activity summary associated with the /admin HTTP service.

Expanding the specific HTTP service, CAMM lists out different entry points by file type - typically *.do* for struts action end point and *.jsp* for JSP end point. Click the plus (+) icon next to different types to reveal a list of specific .jsp and .do files. When a specific file is selected, CAMM displays more detailed performance data.

Method level performance data is displayed when you select a specific HTTP service entry point.

Table 3–37 HTTP Performance Summary

Column/Metric	Description
Servlet	Name of the servlet associated with the selected service

Table 3–37 (Cont.) HTTP Performance Summary

Column/Metric	Description
Method	Name of the method invoked by external call
Arrivals	Total number of requests received by this method
Invocation Count	Total number of method invocations
Response Time (ms)	Average method response time in milliseconds

3.8.2 EJBs

To view the performance summary for EJBs invoked from outside the JVM, click the EJBs node.

Table 3–38 EJB Performance Summary

Column/Metric	Description
EJB	Name of the EJB
Invocation Count	Number of times the EJB is called
Response Time (ms)	Average response time for the EJB in milliseconds
Delay (ms)	Overall delay contributed by the EJB in milliseconds

Tip: As a general rule, external calls that terminate in EJBs are RMI calls. Web services calls that ultimately terminate in EJBs use SOAP and enter the application server via HTTP.

3.8.3 JDBCs

To bring up the performance summary for JDBC operations invocated from outside of the JVM, click the JDBCs note.

Table 3–39 JDBC Performance Summary

Column/Metric	Description
SQL Statement	Generalized SQL Statement executed by the JDBC operation
Class	Name of the class used in the JDBC operation
Method	Name of the method used in the JDBC operation
Invocation Count	Number of times the JDBC operation is called
Response Time (ms)	Average response time for the JDBC operation in millisecond
Delay (ms)	Overall delay contributed by the JDBC operation in milliseconds

3.9 WSRP Producers

The Web Services Remote Portlet (WSRP) Producers node in the navigation tree contains information about the WebLogic WSRP consumer - producer relationships in the managed domain. By selecting an entity in the WSRP node, CAMM displays the performance measurements for the associated WSRP consumer or producer.

WebLogic Portal can act as either a WSRP remote producer or as a consumer. When acting as a consumer, WebLogic Portal's remote--or proxy--portlets are WSRP-compliant. These portlets present content that is collected from

WSRP-compliant producers, allowing you to use external sources for portlet content, rather than having to create its content or its structure yourself.

The following types of portlets can be exposed with WSRP inside a WebLogic portal:

- Page flow portlets
- JavaServer Pages (JSP) portlets
- Struts portlets
- Java portlets (JSR168; supported only for complex producers)
- JavaServer Faces (JSF) portlets

The minimum and maximum response time measurements are captured in addition to the average response time measurements. These metrics, if present, display visually in the window in the right pane.

3.9.1 WSRP Summary

To view the WSRP Producers Summary:

1. Select the WSRP Producers node to show the WSRP Producers Summary tab.

   The WSRP Producers summary includes the following table (Table 3–40):

Table 3–40 WSRP Producers Summary

Column	Description
WSRP Producer	Name of the producer portlet
WSDL URL	URL of the WSD.

2. To view the portlet details, click the Consumer Portlets node under the WSRP Producers.

   The following tables are in this view:

   - WSRP Producer Information
   - WSRP Consumer Portlet Performance

     You can double-click the portlet name to drill down to view more detail.
   - WSRP Producer Portlets

     You can double-click the portlet name to drill down to view more detail.

Table 3–41 WSRP Producers Information

Column	Description
TestPortlets	Defined by the user for the Producer, for example description, handle, and more
URL	Lists the details of each item under the TestPortlet column

Table 3–42 WSRP Consumer Portlet Performance

Column	Description
Portal	The Portal is the logical containment unit for a Portal application. A typical Portal can contain a few desktops, several of books, tens of pages, and hundreds of portlets.
Desktop	The desktop is the top-level container for the portal components included in that specific view of the portal. Portal administrators can create new desktops beyond what portal developers create in WebLogic Workshop.
Book	The top-level book contains all sub-books, pages, and portlets. The top-level book defines the initial menu navigation style used for the desktop. For each sub-book, you add to a desktop you can select a different navigation style.
Page	Pages and sub-books are the navigable containers used for organizing portlets.
Portlet	Portlets are the containers that surface Web content and applications in your desktops.
Response Time (ms)	Average response time in milliseconds.
Completions	Number of Completed Instances for a specific node.
Response Time Min (ms)	Minimum response time in milliseconds.
Response Time Max (ms)	Maximum response time in milliseconds.

Table 3–43 WSRP Producer Portlets

Column	Description
Producer Portlet	Name of the producer portlet
Producer	Name of the producer

3. Click a portlet name in the tree view to see the performances associated with the consumer and producer portlets.

3.9.2 WSRP Topology

Use this option to visually explore WSRP consumer - producer relationships and the WSRP deployment topology.

To view the WSRP Topology:

1. Select the WSRP Producers node to show the WSRP Topology tab.
2. Click the WSRP Topology tab to view the details.

3.9.3 Display Portal Desktop

The portal desktop is described in Section 3.1.1.1, "Display Portal Desktop - Desktop Structure Viewer".

Access the Architecture View:

1. To view the portal desktop for a specific portlet, right-click the portlet name under the Consumer Portlet node.
2. Select Display Portal Desktop.

3. You can drill down to view the Architecture View from this view. See the instructions in Section 3.1.2, "Portlet Drill Down".

3.10 Integration

The Integration node under Oracle Tree contains information about the WebLogic Integration resources in the managed domain. By selecting the Integration node under Oracle Tree, CAMM displays the Integration Summary.

The Integration Summary includes the following (Table 3–44):

Table 3–44 Integration Summary

Metric	Description
Process	Name of process
Running	Total number of currently running instances for a specific process
Suspended	Total number of suspended instances for a specific process
Frozen	Total number of frozen instances for a specific process
Completed	Total number of completed instances for a specific process
Aborted	Total number of aborted instances for a specific process
Terminated	Total number of terminated instances for a specific process
Average Execution Time	Average execution completion time for a specific process

Tip: Statistics on the number of process instances with Terminated, Abort, and Frozen states can indicate abnormal operation of WebLogic Integration application or container. It is possible to unfreeze Frozen process instances from WLI Console.

CAMM presents these metrics in a table format in the Main Display Window when you select the Integration node. Graphical representations of these metrics, Running Instances, Completed Instances, and Average Execution Time, are displayed below the table.

Expand the Integration tree by clicking on the plus (+) icon next to Integration node.

The expanded Integration tree allows you to look at various components of WebLogic Integration and help identify performance bottlenecks. This section explains the nodes under the Integration Tree.

3.10.1 Health

In the expanded Integration tree, the first node you see is the Health node. Under the Health node, CAMM lists various subsystems in WebLogic Integration. By expanding the Health node, you can see the following:

- Execute Queues
- Async Dispatchers
- Sync Dispatchers
- JMS Destinations
- Stateless Containers
- Persistent Containers

You can get to the health information specific to each of these subsystems by clicking the appropriate node. Also, you can get to a particular instance of a subsystem.

3.10.1.1 Execute Queues

In the Execute Queues node, CAMM provides operational statistics of each execute queues configured for WebLogic Integration. Select the Execute Queues node in the Monitor Workspace to display the Execute Queues Summary in the Main Display Window.

The Execute Queues Summary provides the following information (Table 3–45):

Table 3–45 Execute Queues Summary

Metric	Description
Execute Queue	Execute Queue ID
Aggregated Execute Queue	Aggregated execute queue statistics per resource
Idle Threads	Current number of idle threads in a specific Execute Queue
Pending Threads	Current number of pending threads in a specific Execute Queue
Requests	Total number of requests serviced for a specific Execute Queue
Total Threads	Total number of threads configured in a specific Execute Queue

> **Tip:** Pay attention to Idle Threads and Pending Threads counts. Rapidly decreasing Idle Threads count combined with rapidly increasing Pending Threads count can indicate a backup in the Execute Queue.

Use the following guidelines to adjust the Execute Queue Thread Count (Table 3–46):

Table 3–46 Guidelines to Adjust the Execute Queue Thread Count

Execute Queue Is Backed Up?	Application Is CPU Bound?	Adjustment Guideline
Yes	No	Increase execute queue thread count.
Yes	Yes	Decrease thread count and explore JVM or application issues that may be causing high CPU utilization.

CAMM presents these metrics in a table format in the Main Display Window when you select the Health node. Graphical representations of these metrics, Idle Treads, Pending Threads, and Requests, are displayed below the table.

Expand the Health tree by clicking on the plus (+) icon next to Health node. You can get the same summary as previously described for a specific execute queue.

3.10.1.2 Async Dispatchers

In the Async Dispatcher node, CAMM provides operational statistics of each of the Async Dispatchers configured in WebLogic Integration. Select the Async Dispatchers node in the Monitor Workspace to show the Async Dispatchers Summary in the Main Display Window.

The Async Dispatcher Summary includes the following information (Table 3–47):

Table 3–47 Async Dispatcher Summary

Metric	Description
EJB	Name of the Message Driven EJB
In Use	Number of instances for a specific Message Driven EJB currently in use
Idle	Number of instances for a specific Message Driven EJB currently in the idle state
Waits	Number of instances for a specific Message Driven EJB currently in the wait state
Timeouts	Number of instances for a specific Message Driven EJB currently in the timeout state
Commits (Transaction)	Total number of commits performed for a specific Message Driven EJB
Rollbacks (Transaction)	Total number of transaction rollbacks performed for a specific Message Driven EJB
Timeouts (Transaction)	Total number of transaction timeouts performed for a specific Message Driven EJB

Tip: Rapidly increasing counts in MDB Waits and Timeouts metrics may indicate a tuning opportunity for the MBD container. Furthermore, increasing numbers in the Transaction Rollbacks and Timeouts metrics may indicate issues interacting with the database. Ideally, these metrics should not increase rapidly.

CAMM presents these metrics in a table format in the Main Display Window when you select the Async Dispatchers node. Graphical representation of one metrics, Message Driven EJB in use, is displayed below the table.

Expand the Async Dispatchers tree by clicking the plus (+) icon next to Async Dispatchers node. You can get the same summary as previously described for a specific async dispatcher.

3.10.1.3 Sync Dispatchers

In the Sync Dispatchers node, CAMM provides operational statistics of each of the Sync Dispatchers used by WebLogic Integration. Select the Sync Dispatchers node in the Monitor Workspace to show the Sync Dispatchers Summary in the Main Display Window.

The Sync Dispatcher Summary includes the following information (Table 3–48):

Table 3–48 Sync Dispatcher Summary

Metric	Description
EJB	Name of the Stateless EJB
In Use	Number of instances for a specific Stateless EJB currently in use
Idle	Number of instances for a specific Stateless EJB currently in the idle state
Waits	Number of instances for a specific Stateless EJB currently in the waits state
Timeouts	Number of instances for a specific Stateless EJB currently in the timeouts state

Tip: Rapidly increasing counts in Stateless EJB Waits and Timeouts metrics may indicate performance issues and a tuning opportunity for the EJB container. Ideally, these metrics should not increase at a rapid pace.

CAMM presents these metrics in a table format in the Main Display Window when you select the Sync Dispatchers node. Graphical representation of one metrics, Stateless EJB in use, is displayed below the table.

Expand the Sync Dispatchers tree by clicking on the plus (+) icon next to Sync Dispatchers node. You can get the same summary as previously described for a specific sync dispatcher.

3.10.1.4 JMS Destinations

In the JMS Destination node, CAMM provides operational statistics of each of the JMS Destinations used by WebLogic Integration. Select the JMS Destinations node in the Monitor Workspace to show the JMS Destinations Summary in the Main Display Window.

JMS Destination Summary includes the following tables: JMS destination message statistics and JMS destination byte statistics. The JMS destination message statistics table includes the following information (Table 3–49).

Table 3–49 JMS Destination Message Statistics

Column/Metric	Description
JMS Destination	Name of the JMS destination
Message Current	Number of JMS messages currently at a specific JMS destination
Message High	Maximum number of JMS messages at a specific JMS destination
Message Pending	Number of JMS messages pending to be delivered to a specific JMS destination
Message Received	Total number of JMS messages at a specific JSM destination

Tip: Pay attention to Message Pending metric. Too many pending messages in a specific JMS destination could result in a performance slowdown. Rapidly increasing count for the Message Pending metric may indicate a performance problem and a JMS destination tuning opportunity.

The JMS destination byte statistics table includes the following information (Table 3–50).

Table 3–50 JMS Destination Byte Statistics

Column/Metric	Description
JMS Destination	Name of the JMS destination
Byte Current	Byte count of JMS messages currently at a specific JMS destination
Byte High	Maximum byte count of JMS messages at a specific JMS destination
Byte Pending	Byte count of JMS messages pending to be delivered to a specific JMS destination
Byte Received	Total Byte count of JMS messages at a specific JMS destination

CAMM presents these metrics in table format in the Main Display Window when you select the JMS Destinations node. Graphical representations of the following metrics, Message pending and Byte pending, are displayed below the table.

Expand the JMS Destinations tree by clicking on the plus (+) icon next to JMS Destinations node. You can get the same summary as described above for a specific JMS destination.

3.10.1.5 Stateless Containers

In the Stateless Containers node, CAMM provides operational statistics of each of the Stateless Containers used by WebLogic Integration. Select the Stateless Containers node in the Monitor Workspace to show the Stateless Containers Summary in the Main Display Window.

The Stateless Containers Summary includes the following information (Table 3–51):

Table 3–51 Stateless Containers Summary

Metric	Description
EJB	Name of the Stateless EJB
Stateless EJB Transactions	Runtime statistics. You can monitor stateless session EJBs using the metrics in this table.
In Use	Number of instances for a specific Stateless EJB currently being used from the free pool [Snapshot Count]
Idle	Number of instances for a specific Stateless EJB currently in the idle state in the free pool. These bean instances are available for use. [Snapshot Count]
Waits	Number of Threads currently waiting for a specific Stateless EJB instance from the free pool [Snapshot Count]
Timeouts	Total number of Threads that have timed out waiting for an available bean instance from the free pool [Aggregated Count]

CAMM presents these metrics in a table format in the Main Display Window when you select the Stateless Containers node. Graphical representation of one metrics, Stateless EJB in use, is displayed below the table.

Expand the Stateless Containers tree by clicking on the plus (+) icon next to Stateless Containers node. You can get the same summary as previously described for a specific stateless container.

3.10.1.6 Persistent Containers

In the Persistent Containers node, CAMM provides operational statistics of each of the Persistent Containers used by WebLogic Integration. Select the Persistent Containers node in the Monitor Workspace to show the Persistent Containers Summary in the Main Display Window.

The Persistent Containers Summary includes the following tables:

- Entity EJB Activity
- Entity EJB Cache
- Entity EJB Transactions
- Entity EJB Locking

Entity EJB Activity table includes the following information (Table 3–52):

Table 3–52 Entity EJB Activity

Metrics	Description
EJB	Name of the Entity EJB
In Use	Number of instances for a specific Entity EJB currently being used from the free pool [Snapshot Count]
Idle	Number of instances for a specific Entity EJB currently in the idle state in the free pool. These bean instances are available for use. [Snapshot Count]
Waits	Number of Threads currently waiting for a specific Entity EJB bean instance from the free pool [Snapshot Count]
Timeouts	Total number of Threads that have timed out waiting for an available bean instance from the free pool [Aggregated Count]

Tip: Pay attention to Waits and Timeouts metrics. Activities in the Waits metric and increasing count in the Timeouts metric are signs that requests waiting to be serviced by the EJB container. Ideally, 0 should be indicated for these metrics.

Entity EJB Cache table includes the following information (Table 3–53):

Table 3–53 Entity EJB Cache

Metrics	Description
EJB	Name of the Entity EJB
Hits	Total number of times an attempt to access the Entity EJB instance from the cache succeeded [Aggregated Count]
Accesses	Total number of attempts to access the Entity EJB instance from the cache [Aggregated Count]
Size	Number of beans instances from this EJB Home currently in the EJB cache [Snapshot Count]
Activations	Total number of beans from this EJB Home that have been activated [Aggregated Count]
Passivations	Total number of beans from this EJB Home that have been passivated [Aggregated Count]

Tip: Passivation (serializing EJB state information to disk) and activation (reconstitute EJB state information from disk) are resource intensive operations. Ideally, Oracle recommends low level of activity in these metrics.

Entity EJB Transactions table includes the following information (Table 3–54):

Table 3–54 Entity EJB Transactions

Metric	Description
EJB	Name of the Entity EJB
Commits	Total number of transactions that have been committed for this EJB [Aggregated Count]
Rollbacks	Total number of transactions that have been rolled back for this EJB [Aggregated Count]
Timeouts	Total number of transactions that have timed out for this EJB [Aggregated Count]

Tip: High number of EJB Transaction Rollbacks may indicate problems with the data used - for some reason the target database is unable to commit the change. High number of EJB Transaction Timeouts may indicate problems accessing the database including network outage, database lock contention, database outage, and more.

Entity EJB Locking table includes the following information (Table 3–55):

Table 3–55 Entity EJB Locking

Metric	Description
EJB	Name of the Entity EJB
Entries	Number of Entity EJB instances currently locked [Snapshot Count]
Lock Accesses	Total number of attempts to obtain a lock on an Entity EJB instance [Aggregated Count]
Current Waiters	Number of Threads that currently waiting for a lock on an Entity EJB instance [Snapshot Count]
Total Waiters	Total number Threads that have waited for a lock on an Entity EJB instance [Aggregated Count]
Timeouts	Total number Threads that have timed out waiting for a lock on an Entity EJB instance [Aggregated Count]

Tip: Pay attention to Current Waiters and Timeouts. These metrics can indicate possible performance problems caused by EJB Locking. Ideally, 0s should be displayed for these metrics.

CAMM presents these metrics in a table format in the Main Display Window when you select the Persistent Containers node. Graphical representations of three metrics, Entity EJB in use, Entity EJB cache access, and Entity EJB lock access, are displayed below the table.

Expand the Persistent Containers tree by clicking on the plus (+) icon next to Persistent Containers node. You can get the same summary as previously described for a specific persistent container.

3.10.2 Performance

In the expanded Integration tree, the second node you see is the Performance node. CAMM provides the Performance Summary for WebLogic Integration in the Main Display Window when the Performance node is selected.

The Performance Summary includes the following tables: Process Node and Events. The Process Node table provides performance information for various process nodes running in WebLogic Integration. It includes the following information (Table 3–56):

Table 3–56 Performance - Process Node Summary

Column/Metric	Description
Node	Name of a specific node
ID	Process Node ID for a specific node
Type	Control Type for a specific node
Method	Node Method Name for a specific node
Arrival	Number of Requests Arrived for a specific node

Table 3–56 (Cont.) Performance - Process Node Summary

Column/Metric	Description
Active	Number of Active Instances for a specific node
Elapsed Time	Average Time Elapsed to Complete an Instance for a specific node
Completions	Number of Completed Instances for a specific node
Aborts	Number of Aborted Instances for a specific nod.
Exceptions	Number of Exception Encountered for a specific node

Tip: You can use Arrivals and Elapsed Time data collected by CAMM to characterize the performance of your installation. Since CAMM measures performance at cluster level, you are capturing the actual performance of your configuration. You can also perform simple capacity planning analysis by plotting Arrivals versus Elapsed Time (arrival rate versus response time). Ask your Oracle consultant for more information.

The Events table provides a list of SLO violations triggered relevant to WebLogic Integration. It includes the following information (Table 3–57):

Table 3–57 Performance - Events Node Summary

Column/Metric	Description
Start Time	Start time for the process instance that violated a SLO
Entity Name	Name of the process node that violated a SLO
SLO Name	Name of the violated SLO
Service URI	URI of the process that violated a SLO
Application	Name of the application that violated a SLO
Event Type	Violation type (violation or cautionary)
Entity Type	Violation Metric type
SLO Threshold	Type of threshold (high or low)
SLO Trigger Value	Value that triggered a SLO violation

3.10.3 Channels

In the expanded Integration tree, the third node you see is the Channels node. CAMM shows the Channels Summary for various channels configured for WebLogic Integration.

The Channels Summary includes the following information (Table 3–58):

Table 3–58 Channels Summary

Column/Metric	Description
Channel	Name of channel
Type	Channel type
Message Count	Total number of messages processed for a specific channel
Dead Message Count	Total number of dead messages for a specific channel

Tip: Increasing count in the Dead Message Count metric may indicate a configuration issue. When the Message Broker is unable to determine the URI to send a message to, the message is sent to the appropriate deadletter channel. Ensure the URI configured for the channel is reachable.

Expand the Channels tree by clicking the plus (+) icon next to Channels node. You can get the same health summary as previously described for a specific channel.

3.10.4 Subscribers

In the expanded Integration tree, the fourth node you see is the Subscribers node. CAMM shows the Subscribers Summary for various subscribers configured for WebLogic Integration.

Expand the Subscribers tree by clicking the plus (+) icon next to Subscribers node. You can get specific information about an individual subscriber.

3.11 Applications

The Applications node in the navigation tree contains information about all deployed applications in the managed domain. By selecting the Applications node, CAMM displays the Applications Summary.

The Applications Summary includes the following information (Table 3–59):

Table 3–59 Applications Summary

Column/Metric	Description
Application	Name of application
Status	Operations status for a specific application
Response Time (ms)	Average response time in milliseconds for a specific application. This is the average of response times of all JSPs and servlets contained in the deployment archive.
Invocation Count	Total number of invocations for a specific application. This is the total invocation count of all JSPs and servlets contained in the deployment archive.

Tip: Application is a packaging unit in J2EE. Each EAR, WAR, and JAR files deployed to the application server is considered an individual application. These metrics track performance and arrival rate of these entities.

CAMM presents these metrics in a table format in the Main Display Window when you select the Applications node. Graphical representations of the following metrics, Response Time, Invocation Count, and Active Sessions, are displayed below the table.

Expand the Applications tree by clicking the plus (+) icon next to Applications node. You can get more information about a specific application.

CAMM displays performance summary for the selected application in the Main Display Window. You can obtain additional performance data by clicking different tabs in the Main Display Window.

The Applications Summary includes the following tabs (Table 3–60):

Table 3–60 Applications Summary Tabs

Tab Name	Description
Summary	Includes performance data at the application level including time-based trend graphs of Application Response Time, Application Invocation Count, and Application Active Sessions. The invocation count and response time for the top 10 slowest servlets, the usual application entry points, are also included.
Response Times	Includes time-based trend graphs of component response times. Graphs include Servlet Response Time, EJB Response Time, and JDBC Response Time.
Invocations	Includes time-based trend graphs of component invocation counts. Graphs include Servlet Invocation Count, EJB Invocation Count, and JDBC Invocation Count.
Errors/Exceptions	Errors metrics associated with the selected portal.
Transactions	Transaction events associated with the selected portal and children below.
Modeled Entities	Includes a catalog of entities modeled by CAMM. Only the modeled entities associated with the selected application are included.
Instrumentation	Includes performance data by different types of instrumentation probe points. There are different tabs available: Class, Method, and SQL. Each tab includes basic information such as Probe Point Name, Invocation Count, and Response Time. This detailed performance data can help you identify low-level bottlenecks.
Topology	Includes the topology view associated with the selected application.

Under each named application node, CAMM displays performance and other relevant information specific to that application. For example, by clicking the children nodes, the relevant data is displayed in the Main Display Window. Application response time and invocations measurements can be reached by clicking the panes in the Main Display Window.

In this section, we will further expand on the following nodes:

- Services
- Dependencies
- Deployments
- Workshop Projects
- Web Applications
- Stateless Beans
- Stateful Beans
- Entity Beans
- Message Driven Beans

Note: The number of children nodes available under each application node depends solely on the complexity of the selected application. Simple J2EE web applications will not have nodes like Workshop Projects, Stateless Beans, Stateful Beans, Entity Beans, and Message Driven Beans.

3.11.1 Services

The Services node includes all the external entry points associated with the selected application. When this node is selected, CAMM displays a summary view in the Main

Display Window. CAMM displays the performance data associated with various entry points associated with the selected application.

> **Tip:** The children nodes under the Services node include entry point specific performance data. To understand the meaning of these metrics, refer to Section 3.11.1, "Services".

3.11.2 Dependencies

The Dependencies node shows a list of internal and external components and share resources that a specific application depends on for its normal operation. When the Dependencies node is selected, CAMM displays all external references made by the application in the Main Display Window. The following is a list of columns and their descriptions (Table 3–61):

Table 3–61 Dependencies Column Descriptions

Column/Metric	Description
Name	Display name of the component or resource used by the application. If this is undefined in the Deployment Descriptor, the reference name for the component is used.
Reference	Reference name of the component or resource used by the application.
Reference Type	Component or resource type.
Referer Component	Name of the component that is part of the application which obtained the reference to external component or resource.
Referer Module	Name of the module that is part of the application which obtained the reference to external component or resource.

CAMM displays all the references associated with components in the selected application.

The Dependencies node can be further expanded by clicking the plus (+) icon. The children nodes of the Dependencies node are organized by type. Here are the list of dependency types and their descriptions (Table 3–62):

Table 3–62 Dependency Types

Dependency Type	Description
Data Sources	All shared data sources used by the application
Entity Beans	All entity beans used by the application
Session Beans	All session beans used by the application
JMS Queues	All JMS queues used by the application for publishing JMS messages
JMS Topics	All JMS topics subscribed by the application
Web Services	All web services used by the application

When a specific node is selected, CAMM displays relevant performance summary. These nodes can also be expanded by clicking the plus (+) icons. The expanded tree includes specific components and share resources used by the application.

The Performance summary view associated with the Data Sources node under Dependencies provides information on both connection pools and SQL statements.

For more information on the metric description, refer to Section 2.23, "Metric Types".

3.11.3 Deployments

The Deployments node shows the architecture of the deployed application. When this node is selected, CAMM shows all the modules deployed as part of this application. The default view in the Main Display Window shows the active module-level call path. Table 3–63 lists the tabs available as part of this summary view and their descriptions.

Table 3–63 Deployment Tabs

Tab Name	Description
Module Level Execution	Shows the active calling relationships among various J2EE modules (EAR, WAR, JAR, and more). Shared resources are also included. This is the default Architecture View at the module level.
Module Level	Shows the potential calling relationships among various J2EE modules. Shared resources are also included.
Instrumentation	Includes detailed performance data at the method level. The table includes caller components, caller method, callee (target) component, callee module, invocation count, and response time.
SQL Statement	Includes all SQL statements executed as part of this application. It also includes performance information such as invocation count and response time.

Active module-level call path is displayed as the default view for the Deployments node of a selected application.

Double-click a specific module to trigger CAMM to display the architecture of the selected module.

Expand the Deployments node by clicking the plus (+) icon to reveal all the deployed modules in this application. Further expanding the nodes at the *module* level reveals components associated with the selected module. Further expanding the nodes at the *component* level reveals methods associated with the selected component.

When you select one of these children nodes (module, component, and method levels), CAMM displays associated tabs for active call path diagram, static call path diagram, instrumentation and SQL statements.

> **Tip:** Use the active call path diagram as a guide to identify entities with performance data. If an entity does not have performance data, CAMM displays *No data available for the selected time frame* in the Main Display Window.

3.11.4 Workshop Projects

The Workshop Projects node includes performance information about modules and components created using the Oracle WebLogic Workshop. These modules and components include WebLogic Integration processes, WebLogic Integration web services, and WebLogic Portal pageflows.

Workshop Project node and its children nodes provide performance data associated with WLI processes, web services, and WLP pageflows.

When you select a specific children node, CAMM displays detailed performance information.

3.11.5 Web Applications

The Web Applications node includes performance information related to the Web Applications modules and components associated with the selected application. Click the Web Applications node to reveal a performance summary in the Main Display Window. Click the plus (+) icon to expand the Web Applications node to reveal various web modules deployed as part of this application.

Click the plus (+) icon to expand on a specific web module and reveal different groupings for web components, for example, Pageflows, Struts Modules and Servlets. Clicking one of these nodes triggers CAMM to display rolled up performance summary for the entire grouping. You can further expand these nodes by clicking the plus (+) icon to reveal more detailed information. Fully expanded Web Applications node contains all web modules organized by type.

Detailed performance information at the individual pageflow, struts action, and servlet levels will be displayed when you click the lowest level nodes.

3.11.6 Stateless Beans

The Stateless Beans node includes activity information related to the stateless EJB components associated with the selected application. Click the Stateless Beans node to reveal an activity summary in the Main Display Window. Click the plus (+) icon to expand the Stateless Beans node to reveal various stateless EJBs deployed as part of this application.

You can further select individual nodes to obtain detailed activity information. Selecting a specific Stateless Bean node triggers CAMM to display detailed activity metrics.

The detailed view contains the following activity metrics (Table 3–64):

Table 3–64 Stateless Beans Detail View

Column/Metric	Description
EJB	Name of the stateless EJB.
In Use	Number of instances for a specific stateless EJB currently being used from the free pool. [Snapshot Count]
Idle	Number of instances for a specific stateless EJB currently in the idle state in the free pool. These bean instances are available for use. [Snapshot Count]
Waits	Number of threads currently waiting for a specific stateless EJB bean instance from the free pool. [Snapshot Count]
Timeouts	Total number of threads that have timed out waiting for an available bean instance from the free pool. [Aggregated Count]

Note: The metrics reported in the Stateless Beans node are reported by the MBean (Management Bean) of the EJB container. These activity metrics can be used for checking the overall health of the EJB container. When the EJB container is restarted, these metrics are reset.

3.11.7 Stateful Beans

The Stateful Beans node includes activity information related to the stateful EJB components associated with the selected application. Click the Stateful Beans node to reveal an activity summary in the Main Display Window. Click the plus (+) icon to

expand the Stateful Beans node to reveal various stateful EJBs deployed as part of this application.

You can further select individual nodes to obtain detailed activity information.

The Stateful EJB Summary includes the following tables:

- Stateful EJB Cache
- Stateful EJB Transactions
- Stateful EJB Locking

3.11.7.1 Stateful EJB Cache

Stateful EJB Cache table includes the following information (Table 3–65):

Table 3–65 Stateful EJB Cache

Metrics	Description
EJB	Name of the Stateful EJB
Hits	Total number of times an attempt to access the Stateful EJB instance from the cache succeeded [Aggregated Count]
Accesses	Total number of attempts to access the Stateful EJB instance from the cache [Aggregated Count]
Size	Number of beans instances from this Stateful Home currently in the EJB cache [Snapshot Count]
Activations	Total number of beans from this Stateful Home that have been activated [Aggregated Count]
Passivations	Total number of beans from this Stateful Home that have been passivated [Aggregated Count]

Tip: Passivation (serializing EJB state information to disk) and activation (reconstitute EJB state information from disk) are resource intensive operations. Ideally, Oracle recommends low level of activity in these metrics.

3.11.7.2 Stateful EJB Transactions

Stateful EJB Transactions table includes the following information (Table 3–66):

Table 3–66 Stateful EJB Transactions

Metrics	Description
EJB	Name of the Stateful EJB
Commits	Total number of transactions that have been committed for this Stateful [Aggregated Count]
Rollbacks	Total number of transactions that have been rolled back for this Stateful [Aggregated Count]
Timeouts	Total number of transactions that have timed out for this EJB [Aggregated Count]

Tip: High number of EJB Transaction Rollbacks may indicate problems with the data used; for some reason the target database is unable to commit the change. High number of EJB Transaction Time-outs may indicate problems accessing the database including network outage, database lock contention, and database outage.

3.11.7.3 Stateful EJB Locking

Stateful EJB Locking table includes the following information (Table 3–67):

Table 3–67 Stateful EJB Locking

Metric	Description
EJB	Name of the Stateful EJB
Entries	Number of Stateful EJB instances currently locked [Snapshot Count]
Lock Accesses	Total number of attempts to obtain a lock on an Stateful EJB instance [Aggregated Count]
Current Waiters	Number of Threads that currently waiting for a lock on an Stateful EJB instance [Snapshot Count]
Total Waiters	Total number Threads that have waited for a lock on an Stateful EJB instance [Aggregated Count]
Timeouts	Total number Threads that have timed out waiting for a lock on an Stateful EJB instance [Aggregated Count]

Tip: Pay attention to Current Waiters and Time-outs. These metrics can indicate possible performance problems caused by EJB Locking. Ideally, 0s should be displayed for these metrics.

CAMM presents these metrics in a table format in the Main Display Window when you select the Stateful Beans node. Graphical representations of two metrics, Stateful EJB cache access, and Stateful EJB lock access, are displayed below the table.

By looking at the activities related to Stateful EJBs, you can determine if there any abnormal activities associated with Stateful EJBs.

Note: The metrics reported in the Stateful Beans node are reported by the MBean (Management Bean) of the EJB container. These activity metrics can be used for checking the overall health of the EJB container. When the EJB container is restarted, these metrics are reset.

3.11.8 Entity Beans

The Entity Beans node includes activity information related to the Entity EJB components associated with the selected application. Click the Entity Beans node to reveal an activity summary in the Main Display Window. Click the plus (+) icon to expand the Entity Beans node to reveal various Entity EJBs deployed as part of this application.

You can further select individual nodes to obtained detailed activity information. Selecting a specific Entity Bean node triggers CAMM to display detailed activity metrics.

The Entity EJB Summary includes the following tables:

- Entity EJB Activity
- Entity EJB Cache
- Entity EJB Transactions
- Entity EJB Locking

3.11.8.1 Entity EJB Activity

Entity EJB Activity table includes the following information (Table 3–68):

Table 3–68 Entity EJB Activity

Metrics	Description
EJB	Name of the Entity EJB.
In Use	Number of instances for a specific Entity EJB currently being used from the free pool. [Snapshot Count]
Idle	Number of instances for a specific Entity EJB currently in the idle state in the free pool. These bean instances are available for use. [Snapshot Count]
Waits	Number of Threads currently waiting for a specific Entity EJB instance from the free pool. [Snapshot Count]
Timeouts	Total number of Threads that have timed out waiting for an available bean instance from the free pool. [Aggregated Count]

Tip: Pay attention to Waits and Timeouts metrics. Activities in the Waits metric and increasing count in the Timeouts metric are signs that requests are waiting to be serviced by the EJB container. Ideally, 0 should be indicated for these metrics.

3.11.8.2 Entity EJB Cache

Entity EJB Cache table includes the following information (Table 3–69):

Table 3–69 Entity EJB Cache

Metrics	Description
EJB	Name of the Entity EJB
Hits	Total number of times an attempt to access the Entity EJB instance from the cache succeeded [Aggregated Count]
Accesses	Total number of attempts to access the Entity EJB instance from the cache [Aggregated Count]
Size	Number of beans instances from this EJB Home currently in the EJB cache [Snapshot Count]
Activations	Total number of beans from this EJB Home that have been activated [Aggregated Count]
Passivations	Total number of beans from this EJB Home that have been passivated [Aggregated Count]

Tip: Passivation (serializing EJB state information to disk) and activation (reconstituting EJB state information from disk) are resource intensive operations. Ideally, Oracle recommends a low level of activity in these metrics.

3.11.8.3 Entity EJB Transactions

Entity EJB Transactions table includes the following information (Table 3–70):

Table 3–70 Entity EJB Transactions

Metric	Description
EJB	Name of the Entity EJB

Table 3–70 (Cont.) Entity EJB Transactions

Metric	Description
Commits	Total number of transactions that have been committed for this EJB [Aggregated Count]
Rollbacks	Total number of transactions that have been rolled back for this EJB [Aggregated Count]
Timeouts	Total number of transactions that have timed out for this EJB [Aggregated Count]

Tip: High numbers of EJB Transaction Rollbacks may indicate problems with the data used; for some reason the target database is unable to commit the change. High numbers of EJB Transaction Timeouts may indicate problems accessing the database including network outage, database lock contention, database outage, and more.

3.11.8.4 Entity EJB Locking

Entity EJB Locking table includes the following information (Table 3–71):

Table 3–71 Entity EJB Locking

Metric	Description
EJB	Name of the Entity EJB
Entries	Number of Entity EJB instances currently locked [Snapshot Count]
Lock Accesses	Total number of attempts to obtain a lock on an Entity EJB instance [Aggregated Count]
Current Waiters	Number of Threads that currently waiting for a lock on an Entity EJB instance [Snapshot Count]
Total Waiters	Total number Threads that have waited for a lock on an Entity EJB instance [Aggregated Count]
Timeouts	Total number Threads that have timed out waiting for a lock on an Entity EJB instance [Aggregated Count]

Tip: Pay attention to Current Waiters and Timeouts. These metrics can indicate possible performance problems caused by EJB Locking. Ideally, 0s should be displayed for these metrics.

When you select the Entity Beans node, CAMM presents these metrics in a table format in the Main Display Window. Graphical representations of the following metrics, Entity EJB in use, Entity EJB cache access, and Entity EJB lock access, are displayed below the table.

Expand the Entity Beans tree by clicking the plus (+) icon next to Entity Beans node. You can get the same summary as previously described for a specific Entity EJB.

By looking at the activities related to Entity EJBs, you can determine if there any abnormal activities associated with Entity EJBs.

Note: The metrics reported in the Entity Beans node are reported by the MBean (Management Bean) of the EJB container. These activity metrics can be used for checking the overall health of the EJB container. When the EJB container is restarted, these metrics are reset.

3.11.9 Message Driven Beans

The Message Driven Beans node includes activity information related to the message driven EJB components associated with the selected application. Click the Message Driven Beans node reveals an activity summary in the Main Display Window. Click the plus (+) icon to expand the Message Driven Beans node to reveal various message driven EJBs deployed as part of this application.

You can further select individual nodes to obtained detailed activity information.

The Message Driven EJB Summary includes the following tables:

- Message Driven EJB Activity
- Message Driven EJB Transactions

3.11.9.1 Message Driven EJB Activity

Message Driven EJB Activity table includes the following information (Table 3–72):

Table 3–72 Message Driven EJB Activity

Metric	Description
EJB	Name of the Message Driven EJB.
In Use	Number of instances for a specific Message Driven EJB currently being used from the free pool. [Snapshot Count]
Idle	Number of instances for a specific Message Driven EJB currently in the idle state in the free pool. These bean instances are available for use. [Snapshot Count]
Waits	Number of Threads currently waiting for a specific Message Driven EJB instance from the free pool. [Snapshot Count]
Timeouts	Total number of Threads that have timed out waiting for an available bean instance from the free pool. [Aggregated Count]

Tip: Pay attention to Waits and Timeouts metrics. Activities in the Waits metric and increasing count in the Timeouts metric are signs that requests are waiting to be serviced by the EJB container. Ideally, 0 should be indicated for these metrics.

3.11.9.2 Message Driven EJB Transactions

Message Driven EJB Transactions table includes the following information (Table 3–73):

Table 3–73 Message Driven EJB Transactions

Metric	Description
EJB	Name of the Message Driven EJB
Commits	Total number of transactions that have been committed for this EJB [Aggregated Count]
Rollbacks	Total number of transactions that have been rolled back for this EJB [Aggregated Count]
Timeouts	Total number of transactions that have timed out for this EJB [Aggregated Count]

Tip: High numbers of EJB Transaction Rollbacks may indicate problems with the data used; for some reason the target database is unable to commit the change. High numbers of EJB Transaction Timeouts may indicate problems accessing the database including network outage, database lock contention, database outage, and more.

CAMM presents these metrics in a table format in the Main Display Window when you select the Message Driven Beans node. Graphical representation of the Message Driven EJB in use metric is displayed below the table.

By looking at the activities related to Message Driven EJBs, you can determine if there are any abnormal activities associated with Message Driven EJBs.

Note: The metrics reported in the Message Driven Beans node are reported by the MBean (Management Bean) of the EJB container. These activity metrics can be used for checking the overall health of the EJB container. When the EJB container is restarted, these metrics are reset.

3.12 Oracle WebLogic Resources

The Resources node under Oracle Tree contains information for the managed domain organized by logical clusters, machines, servers, and more. You can look for low-level technology metrics organized by technology subsystems for a specific WebLogic Server.

The Resources tree includes the following nodes (Table 3–74):

Table 3–74 WebLogic Resources Tree

Example Node	Description
CSS Domain	Name of the WebLogic Domain configured
b-15/192.168.128.15	ID of the physical machine
OS Metrics	Summary view of three OS metrics collected by OS Agent
CPU	CPU usage data
Memory	Memory usage data
Disk	Disk usage data
cgServer	Name of the WebLogic Server configured
Applications	Performance measurements of all deployed applications running on this server
JDBC	Information of all configured JDBC resources for this server
JMS Servers	Information of all JMS destinations configuration for this server
Execute Queues	Information of all Execute Queues configured for this server
JVM	JVM information including Heap Size for this server
JRockit	JRockit information including Heap Size for this server
Modeling Status	Entities modeled by CAMM for this server
CAMM Modules	Status of the CAMM Java Agent Module for this server

Expand these nodes by clicking the plus (+) icon next to the node name to get more information.

If the CAMM OS Agent is deployed on the machine, clicking on the physical machine ID would show OS metrics collected by the OS Agent. These OS metrics include CPU Usage, Disk Usage, and Physical Memory Usage.

3.13 WebSphere Resources

The Resources node under Oracle Tree contains information for the managed domain organized by logical clusters, machines, servers, and more. You can look for low-level technology metrics organized by technology subsystems for a specific WebSphere Server.

The Resources tree includes the following nodes (Table 3–75):

Table 3–75 WebSphere Resources Tree

Example Node	Description
WPSSUn	Resource name, for example, WPSUn
WebSphere Portal	Machine name, for example, WebSphere_Portal
OS Metrics	Summary view of three OS metrics
CPU	CPU usage data
Memory	Memory usage data
Disk	Disk usage data
WebSphere Portal	Server name, for example, WebSphere_Portal
Applications	Performance measurements of all deployed applications running on this server
JDBC	Information of all configured JDBC resources for this server
JMS Servers	Information of all JMS destinations configuration for this server
Thread Pools	Performance information about all threads used by the container to process request
JVM	JVM information including Heap Size for this server
WebServices	Performance measurements about web services deployed in the container
Sessions	Information about active HTTP sessions
Transactions	Information about transactions performance
Cache	Information about cache performance
ORB	Information about ORB performance
Modeling Status	Modeled entities for the container
CAMM Modules	Status of the CAMM Java Agent Module for this server
Applications	Performance information about the applications deployed in the container

Expand these nodes by clicking the plus (+) icon next to the node name to get more information.

Clicking on the physical machine ID would show OS metrics. These OS metrics include CPU Usage, Disk Usage, and Physical Memory Usage.

3.14 Oracle Resources

The Resources node under Oracle Tree contains information for the managed domain organized by logical clusters, machines, servers, and more. You can look for low-level technology metrics organized by technology subsystems for a specific Oracle AS Server.

The Resources tree includes the following nodes (Table 3–76):

Table 3–76 Oracle Resources Tree

Example Node	Description
Managed System Resource Name	Top-level Resource name, for example, oc4j_soa
Oracle AS Server	Machine name which can be navigated to both within or outside a cluster, for example, oc4j_soa@192.168.1.119 which includes both the server name and the host server IP address
OS Metrics	Summary view of the OS metrics
CPU	CPU usage data
Memory	Memory usage data
Disk	Disk usage data
Applications	Performance measurements of all deployed applications running on this server
JDBC	Information of all configured JDBC resources for this server
JMS Servers	Information of all JMS destinations configuration for this server
Thread Pools	Performance information about all threads used by the container to process requests
JVM	JVM information including Heap Size for this server
BPEL Processes	Performance measurements about BPEL Processes deployed in the container
ESB	Performance measurements about ESB services deployed in the container
Modeling Status	Modeled entities for the container
CAMM Modules	Status of the CAMM Java Agent Module for this server
Applications	Performance information about the applications deployed in the container

Clicking the physical machine ID would show OS metrics. These OS metrics include CPU Usage, Disk Usage, and Physical Memory Usage.

3.15 Custom Metrics

The Custom Metrics node under Oracle tree contains all the custom metrics you defined. Currently CAMM supports custom metrics for Java classes. When Custom Metrics node is selected, CAMM displays various summaries. You can select individual entities to get more detailed performance information.

Expanding the Custom Metrics node reveals a list of Java classes with custom metrics configured.

The following is a list of columns in the Custom Class Performance table and their descriptions (Table 3–77):

Table 3–77 Custom Class Performance

Column/Metric	Description
Caller Class	Fully qualified name of the class that is making the inbound call
Caller Method	Method name in the class that is making the inbound call
Class	Fully qualified name of the class that is the destination of the inbound call
Invocation Count	Total number of times the inbound call is made
Response Time (ms)	Average response time of the inbound call in milliseconds

3.16 CAMM Node

The CAMM node in the navigation tree contains information for the CAMM environment for the monitored WebLogic domain, WebSphere cell, or Oracle AS cluster. Select the CAMM node to see the CAMM Java Agent status for the WebLogic domain.

The CAMM Java Agent status includes the following (Table 3–78):

Table 3–78 CAMM Java Agent Status

Column/Metric	Description
Server	Name of the WebLogic server, WebSphere cell, or Oracle AS cluster
Container Status	Operational status of the WebLogic, WebSphere, or Oracle AS server (running or not)
Agent In Sync	Version synchronization between CAMM and CAMM Agent status (true or false)
EJB Installed	CAMM EJB installation status (true or false)
Agent Installed	CAMM Java Agent installation status
Agent Activated	CAMM Java Agent activation status
Agent Status	CAMM Java Agent operational status
Server Type	Identifies server as administration, individual, or clustered server
Admin URI	Location of the domain admin server
Manager RMI Registry Host	Host name of the CAMM RMI registry
Manager RMI Registry Port	Port number of the CAMM RMI registry
EJB Major Version	CAMM EJB major version
EJB Minor Version	CAMM EJB minor version
EJB Build ID	CAMM EJB build number - for version synchronization check
Agent Major Version	CAMM Java Agent major version
Agent Minor Version	CAMM Java Agent minor version
Agent Build ID	CAMM Java Agent build number - for version synchronization check

Click the Modeling Status node under CAMM node to see a table of all modeled entities in the managed domain. This table shows all the managed clusters, servers, and applications in the CAMM environment. Mismatches between the Modeling Status table and your environment are indications of configuration problems.

You can use this information to debug and resolve CAMM configuration issues.

4

Exploring Configuration Tab

The Configuration tab is the upper-left pane with the "Configuration" tab selected.

The configurations explained in this chapter are:

- Resource Configuration
- User Configuration
- Service Level Objectives by Name
- Service Level Objectives by Metrics
- Service Level Objectives by Entity Type
- Action Configuration

4.1 Resource Configuration

Resource configuration is documented in the *Oracle Composite Monitor and Modeler Installation and Configuration Guide*.

4.2 User Configuration

Select the **User Configuration** node in the Configuration tree to manage user roles, create new users, and delete users.

CAMM uses a permissions-based user security model. This model allows administrators to specify data access rights and end-user's ability to configure CAMM. CAMM supports the following types of roles:

- Admin
- Operator
- User

4.2.1 Admin Role

When installing CAMM, the Admin user role is created by default. This role allows administrators of CAMM to configure the application monitoring environment including the following:

- Add and remove CAMM managed resources.
- Specify domain administration server location.
- Configure SLOs.

- Define actions.
- Customize views.
- Create and maintain other user roles.

4.2.2 Operator Role

When installing CAMM, the Operator user role is also created by default. This role allows operators of CAMM to configure the application monitoring environment including the following:

- Configure SLOs.
- Define actions.
- Customize views.
- Create and maintain user roles.

4.2.3 User Role

This user role has read-only access to CAMM. The Configuration tab is not available to users with the user role.

1. To modify the configuration of existing user accounts, double-click the **user** option. You will see a configuration screen.
2. Select a user and double-click to change the configuration of an existing user account.
3. You can force the user to change the password upon next login. Check the **Must Change Password** check box.
4. Click **Save**.

CAMM supports highly complex password authentication policies. The following password word policy properties can be configured in the *Acsera.properties* file.

- Password length check
- Password complexity check
- Password expiration check

For more information, see *Oracle Composite Application Monitor and Modeler Installation and Configuration Guide*.

4.3 Service Level Objectives by Name

Service Level Objectives by Name node in the Configuration tree allows you to manage SLOs by SLO names.

In this window you can:

- Double-click on a specific SLO to open the Service Level Objective Editor.
- View or edit the selected SLO.

4.4 Service Level Objectives by Metrics

Service Level Objectives by Metrics node in the Configuration tree allows you to manage SLOs by performance metrics. The following are some examples of performance metrics:

- BPEL Process Average Response Time
- Portal Book Average Response Time
- Portal Desktop Visit Count
- JVM Heap Free Current
- Process Node Aborts
- Servlet Invocation Total Count
- Servlet Execution Time Average

In the Service Level Objectives by Metrics window, you can:

- Double-click on a specific performance metric to see all SLOs configured for the selected metric.
- Double-click on a specific SLO to open the Service Level Objective Editor to view or edit the selected SLO.

4.5 Service Level Objectives by Entity Type

Service Level Objectives by Entity Type node in the Configuration tree allows you to manage SLOs by modeled entity type.

The following are examples of entity types:

- Oracle BPEL Process
- IBM Virtual Desktop
- Oracle WebLogic Portal Book
- Oracle WebLogic Portal Desktop
- Oracle WebLogic Process Node
- Oracle WebLogic Process Type
- JVM
- Stateless EJB
- Message Driven EJB

In the Service Level Objectives by Entity Type window, you can:

- Double-click on a specific entity type to see all SLOs configured for the selected entity type.
- Double-click on a specific SLO to open the Service Level Objective Editor to view or edit the selected SLO.

4.6 Action Configuration

Action Configuration node in the Configuration tree allows you to manage actions for CAMM. Actions are triggered by a SLO violation event. CAMM supports the following actions:

- Issue a SMNP trap
- Send an e-mail
- Execute a script
- Log to a file

In the Action Configurations window you can:

- Click **Create Action** to create a specific type of action.

In the Action Configuration window, double-click on a specific action to see its configuration information. You can edit and copy edit selected action. CAMM automatically enforces referential integrity during the deletion process.

Note: You can include a set of SLO variables into e-mail, script, and log actions. This feature significantly increases the value of these actions by using real-time performance data. See Table 4–1 for a list of SLO variables.

Table 4–1 List of SLO Variables

SLO Variable	Description	Example Value
$EventType	SLO event type (Violation or Cautionary)	Event.SLO.Cautionary
$EventAttributes.SLOName	Name of the SLO fired	CSR Portal Desktop Response Time Violation
$Event.Attributes.SLOType	Metric where SLO violation was observed	Metrics.J2EE.JVM.HeapFree
$EventAttributes.TriggerValue	Value of metric when SLO threshold was exceeded	35001
$EventAttributes.TriggerThreshold	Threshold type (High or Low)	High
$Entity.InfrastructureID	Name of the platform	WebLogic
$Entity.NodeID	Server node where SLO violation was observed	B93/192.168.3.93
$Entity.DomainID	Domain in which SLO violation was observed	mydomain
$Entity.ResourceID	Cluster in which SLO violation was observed	my_cluster
$Entity.EntityTypeID	Type of the entity in which the SLO violation was observed	J2EE.JVM
$StartTime	Start time of the SLO violation	1112322030000
$EndTime	Stop time of the SLO violation	1112322045000

Tip: Customize your alert using SLO variables. The following is an example of a customized message for a Mail Action:

SLO Event:
SLO Name = $EventAttributes.SLOName;
Event Type = $EventType;
Trigger Domain = $Entity.DomainID;
Trigger Application = $Entity.ApplicationID;
Trigger SLO Type = $EventAttributes.SLOType;
Trigger Value = $EventAttributes.TriggerValue;
Trigger Threshold = $EventAttributes.TriggerThreshold;
Trigger Element = $Entity.ElementID;
Event ID = $EventID;

5

Performance Analytics

As CAMM collects measurements, it processes incoming data and stores resulting information in the CAMM embedded database. The CAMM performance analytics feature then queries against this database to create statistical models and perform mathematical calculations.

This chapter includes the following performance analytics features:

- Entity Performance Ranking
- Performance Characterization
- Memory Leak Detection
- Drill Down - Bottleneck Analysis
- Drill Out - Impact Analysis

5.1 Entity Performance Ranking

For Portal metrics, you can use CAMM Entity Performance Ranking to accelerate bottleneck isolation.

Entity Performance Ranking becomes available when a node contains multiple children entities. For example, if the Labels node has multiple children entities, clicking on the Analysis tab of the node displays the Entity Performance Ranking.

CAMM provides Entity Performance Ranking on Nodes with Children Entities.

5.2 Performance Characterization

Performance Characterization is a set of performance analytics provided by CAMM. Use the Performance Characterization analytics to visualize actual performance of a monitored entity in a given time frame. The following is a list of Performance Characterization analytics provided by CAMM:

A statistically significant amount of data is required before these graphs will appear. You will need at least 1 hour of continuos load for any graphs to appear and at least 5 hours of continuos load for all of the graphs to appear. Analysis requires that load metrics actually be present and zero points do not count for the purposes of this analysis. It may be helpful to increase your time frame manually until you encompass enough data points.

5.2.1 Multi-Point Performance/Load Regression

CAMM automatically fits a multi-point line through the data set of performance and load measurements collected during a specific time frame. This feature allows you to quickly characterize the performance of an entity for a certain load level in real time.

CAMM performs multi-point regression and fits a 3-point line through the data set.

As a performance analytical tool, the Multi-Point Performance/Load Regression is extremely useful. It enables you to characterize the performance of a system or a component for a certain load range. You can use this analysis to capture the performance baseline of a system with a specific infrastructure configuration such as deployed applications, application server configuration, network topology and hardware capacity and load their characteristics. This analysis can also be used for capacity planning. Since the regression illustrates the actual application performance for a specific configuration, this information is useful to determine how to expand computing capacity to best meet estimated demand.

5.2.2 Performance/Load Scattergram

This feature automatically plots the interceptions between arrival rate and response time for a specific monitored entity into a scattergram. This feature gives you a visual interpretation of actual performance (response time) under a range of loads (arrival rate) in real time.

CAMM plots intersections of arrivals and response time metrics in a scattergram.

5.2.3 Time-Based Performance Distribution

This feature automatically plots the performance (response time) of a specific monitored entity during a time frame specified by you. The X axis is response time and Y axis is time on the clock. This plot gives you a visual representation of a monitored entity's performance over time.

Time-Based Performance Distribution can help you identify abnormal performance patterns.

5.2.4 Performance Histogram

This feature automatically creates a histogram with ten bins (performance ranges) and plots the occurrences of response time that falls in each bin. The X axis is the number of occurrences (frequency) and Y axis is the response time ranges. You can use this histogram to see the performance distribution for a specific monitored entity.

Performance Histogram can help you visualize performance distribution.

5.2.5 Time-Based Performance Trend

This feature automatically plots a trend line associated with the performance (response time) of a specific monitored entity during a time frame specified by you. The X axis is response time and Y axis is time on the clock. This trend line gives you a visual representation of a monitored entity's performance over time.

Time-based Performance Trend allows you to quickly identify abnormal performance patterns during a certain time frame.

5.3 Memory Leak Detection

CAMM has a built-in analytic to detect performance memory leak.

To access Memory Leak Detection, select the JVM node under Resources. When the JVM node is selected, CAMM analyzes memory usage patterns during the time frame specified. Two graphs, Memory Leak Rate and Memory Leak Status, display in the Main Display Window.

You can access Memory Leak Detection by clicking on the JVM node.

The Memory Leak Rate graph enables you to perform time-based trend analysis. The graph depicts the JVM heap memory growth rate in KB/minute for a given period of time. The Memory Leak Status graph enables you to quickly identify the presence of persistent memory leak.

> **Tip:** Persistent JVM heap growth can result in a memory leak. CAMM evaluates the memory growth rate continuously to determine leak status. The Memory Leak Status graph has two values (0 and 1). The value 0 means there is no leak. The value 1 means there may be a leak. When there is a severe leak, the value of the Memory Leak Status graph remains at 1.

5.4 Drill Down - Bottleneck Analysis

Finding performance bottlenecks using CAMM is easy. Often, CAMM guides you to the exact location of the bottleneck. There are some basic ways for performing drill down operations in CAMM.

Table 5–1 Drill-Down Methods

Operation	Description
Double-Clicking	Available on many views in CAMM. Double-clicking triggers CAMM to bring up information one logical level lower than what is currently displayed.
Right-Click Menu	Available on most views in CAMM. From most views, selecting the Display Architecture View option initiates a diagnostic session. From the Architecture View, use the drill down option to go one level deeper.

Whenever possible, use the built-in Delay Analysis to isolate performance bottlenecks. Delay Analysis is available under the Process node as well as in the Architecture View. CAMM Delay Analysis shows the delay contribution associated with a specific component in a process or an active call path to the overall delay.

You can also use Entity Performance Ranking to identify abnormally behaving components and performance bottlenecks.

A red light on the Operational Dashboard indicates that the performance threshold for an application has been violated.

Double-click on the application row in the Operational Dashboard to bring up the detailed performance data pop-up window.

Use the detailed performance data to further isolate the performance problem. This pop-up window contains the following information:

- Servlet Performance [Table]
- Servlet Response Time (ms) [Graph]
- Servlet Invocation Count [Graph]

- EJB Performance [Table]
- Top five EJB Response Time (ms) [Graph]
- Top five EJB Invocation Count [Graph]

Double-clicking on a specific row in this pop-up window brings up the Architecture View associated with the selected row.

Double-clicking on the slow performing component brings up the Architecture View with delay analysis.

> **Tip:** Right-click and check the **Hide other arrows** option to show only arrows on the component which you have clicked on. Follow the slowest Inbound calls to the slowest next level component. Repeat this method to find the bottleneck.

Look at the Fan Out table first to see if any outbound calls are performing poorly. If a specific call is performing poorly, follow the call to the destination component and evaluate further.

Evaluate all the Fan Out calls and follow the slowest Fan Out call.

If Fan Out calls are not performing poorly, it is very likely the performance bottleneck exists within the currently selected component. Use the Fan In table to isolate the poorly performing calls. Double-clicking on a specific inbound call in the Fan In table brings up a pop-up window with detailed performance data associated with this inbound call.

Double-click on the slowest Fan In call to get detailed performance data.

CAMM guides you from the Operational Dashboard to the performance bottleneck that exists at the method level.

> **Tip:** Compare current delay contribution breakdown to baseline delay contribution breakdown to identify any abnormal behavior. This can be done using Comparative Views.

5.5 Drill Out - Impact Analysis

After performance bottleneck is identified, you can use the CAMM unique drill out ability to determine the impact of the bottleneck. The impact analysis can be done in the Architecture View.

To start the drill out process:

1. Select the entity that has been identified as the bottleneck in the Architecture View
2. Right-click and select **Drill Out**. This brings the Architecture View one logical level higher.
3. Use a combination of Drill Out and Show in Context navigation techniques to determine the scope of impact.

To determine the impact of the performance bottleneck, refocus the Architecture View and the module that contains the class in context (white background color).

On this window you can:

- Check the **Hide other arrows** option in the right-click menu to show only arrows on the component you have clicked on and improve visual navigation.

- Use the Architecture View or the Fan In table to identify the upstream components that make inbound calls into the class. The impact analysis would follow these inbound calls to their origins to further determine the impact of the performance bottleneck.

For example, the blue background color of a class indicates this entity belongs to a module that is different from current view context.

6 Exporting Data

This chapter includes the following export features:

- Data Export Modes
- CAMM Export Configuration
- Example of Exported Data for WebLogic

6.1 Data Export Modes

There are three different modes to export performance data collected by CAMM to external databases and other persistence formats. These modes give you flexibility to choose the best way to extract performance data from CAMM.

- Export to File
- Export to Database
- Aggregation Export to File

6.1.1 Export to File

In this mode, CAMM exports its raw performance data as several CSV (comma separated value) files.

6.1.2 Export to Database

In this mode, CAMM exports its raw performance data as several ANSI SQL statements. These SQL statements allow you to create tables and insert data.

6.1.3 Aggregation Export to File

In this mode, CAMM exports its aggregated performance data after it's daily aggregation operation as several CSV files.

6.2 CAMM Export Configuration

The following sections describe CAMM export configuration:

- Data Export Modes
- CAMM Export Configuration
- Example of Exported Data for WebLogic

6.2.1 CAMM Periodic Export Configuration

CAMM stores real-time performance metrics in its internal data repository (MySQL database). If you want to store this data in your historical data repository, CAMM provides automatic means for performance data export. You can control the frequency of export runs, the time when the export should run, and the time range of export data within a day.

Example 6–1 CAMM Export Configuration

```
# Setting for integrated export
AggregationManager.IntegratedExport = false
AggregationManager.ExportDataStartHour = 0
AggregationManager.ExportDataEndHour = 0
AggregationManager.ExportDataSetRangeInHour = 4
AggregationManager.ExportDataSetIntervalInHour = 1
AggregationManager.ExportDataSetDelay = 10000
AggregationManager.ExportStartTime = 0
AggregationManager.ExportEndTime = 0
AggregationManager.ExportFilePurgeTime = 10d
```

By default, automatic data export feature is disabled. To enable it, set AggregationManager.IntegratedExport parameter to true. A pair of parameters, AggregationManager.ExportDataStartHour and AggregationManager.ExportDataEndHour, indicates the time range within a day of the export performance data in which you are interested. By default it is 24 hours.

Parameter AggregationManager.ExportDataSetRangeInHour shows how much data is stored in each export file. By default it is 4 hours worth of data. This means that CAMM will create multiple export data files with 4 hours worth of data.

To minimize export query impact on normal CAMM performance data collection functionality, spread out lengthy data exporting queries. This is achieved by setting AggregationManager.ExportDataSetIntervalInHour parameter. By default it is 1 hour. This means that export query thread will be running every hour.

AggregationManager.ExportDataSetDelay defines cool down time for consecutive export queries. By default it is 10 seconds. This means that the next export query will happen not earlier than 10 seconds after the previous one.

AggregationManager.ExportFilePurgeTime indicates how many days the export data file will be available before CAMM deletes it. By default it is 10 days. Current default values are optimal and this section should not be changed except from enabling the automatic export feature.

Automatic export function relies on the definition of the data to be exported by looking into $CAMM_HOME/config/export.xml file and exports performance data based on the rules defined in this file.

The output directory is specified in export.xml and will be overwritten on each export interval.

6.2.2 Manual Execution of Metric Export

The bin directory on the CAMM manager contains scripts called runExportMetric.sh/bat and runExportEvent.sh/bat that export metrics and events (such as alerts) to CSV files, respectively.

Run runExportMetric.sh like:

```
./runExportMetric.sh <path to export.xml configuration> <start time> <end time>
```

For example:

```
C:\oracle\em10g\bin>runExportMetric.bat c:\oracle\em10g\config\export.xml "4/1/09
16:06:00" "4/1/09 16:36:00"
```

The start time and end time are in the machine's local time zone. The output exported csv files' timestamps will be in UTC/GMT.

6.2.3 export.xml File

The export.xml file contains all the directives and filters required to export performance metrics and events. This file is used by the CAMM Integrated Automatic export feature as well as manual export scripts.

Example 6–2 Contents of export.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<export xmlns="http://www.acsera.com/ns/export" verbose="true" exportMetric="true"
exportEvent="true" metricDataGrain="180s" exportFullMetric="true">
<!--
<output type="jdbc" convertTimeFormat="true"
arguments="access,metric,sun.jdbc.odbc.JdbcOdbcDriver,jdbc:odbc:acsera"/>
-->
<output type="file" convertTimeFormat="false"
arguments="/home/acsera/acsera/export,metric"/>
<entityTypes exportAllTypes="false">
<entityType name="BEA.ProcessNode"/>
<entityType name="J2EE.Dispatcher"/>
<entityType name="J2EE.JDBC.ConnectionPool"/>
<entityType name="J2EE.JVM"/>
</entityTypes>
<!--extra filter -->
<!--
<filters>
<filter key="containerID" values="cgServer"/>
</filters>
-->
<!-- don't modify this -->
<columns>
<column header="Timestamp" type="Timestamp"/>
<column header="EntityID" type="EntityID"/>
<column header="Application" type="Entity" key="applicationID" default=""/>
</columns>
</export>
```

The following tables explain the various attributes.

Table 6–1 Attributes on <export>

Attribute	Description
exportMetric	true/false, whether to export performance metrics
exportEvent	true/false, whether to export SLO events
metricDataGrain	60s, 180s, or 1800s, the metric aggregation tier to export. Note that the population of the 180s tier will be delayed by 1.25 hours and the 1800s tier will be delayed by 7 hours compared to the 60s tier.
exportFullMetric	true/false, whether to include sum, count, min, and max metrics or not.

Table 6–2 Attributes on <output>

Attribute	Description
type	file/jdbc, whether to output to csv files or to write data to another database using JDBC. In the case of JDBC output, the necessary tables will be created automatically by the export mechanism.
convertTimeFormat	true/false, whether to convert the metric timestamp to human readable format (in UTC/GMT). If false, the timestamp will be a long integer. Provide the arguments (jdbc) in a comma separated list.
arguments	See Table 6–3 and Table 6–4 for details.

Table 6–3 Value of the "arguments" Attribute of the <output> Element When the "type" Attribute Is "JDBC"

Attribute	Description
First Parameter	Database type, an arbitrary string
Second Parameter	Table prefix name, should always be metric
Third Parameter	Fully qualified JDBC driver class
Fourth Parameter	JDBC URL

Table 6–4 Value of the "arguments" Attribute of the <output> Element When the "type" Is "file"

Argument	Description
First Parameter	CSV file output directory (this directory will be created, existing files will be overwritten)
Second Parameter	Table prefix name, should always be **metric**

Table 6–5 Attribute of the <entityTypes> Element

Attribute	Description
exportAllTypes	true/false, whether to output all entity types or only the ones specified in <entityType> elements.

Table 6–6 Attribute of the <entityType> Element

Attribute	Description
name	Name of the entityType to include in the export if exportAllTypes is set to false

6.3 Example of Exported Data for WebLogic

The tables in this section describe the fields in the various export files.

- Table 6–7, " Export File Name: metricBEA_ChannelInstance.csv"
- Table 6–8, " Export File Name: metricBEA_ProcessType.csv"
- Table 6–9, " Export File Name: metricBEA_TimerEventGenerator.csv"
- Table 6–10, " Export File Name: metricJ2EE_Dispatcher.csv"
- Table 6–11, " Export File Name: metricJ2EE_EJB_Entity.csv"
- Table 6–12, " Export File Name: metricJ2EE_EJB_Stateless.csv"
- Table 6–13, " Export File Name: metricJ2EE_JDBC_ConnectionPool.csv"

- Table 6–14, " Export File Name: metricJ2EE_JMS_Destination.csv"
- Table 6–15, " Export File Name: metricJ2EE_JMS_Service.csv"
- Table 6–16, " Export File Name: metricJ2EE_JVM.csv"
- Table 6–17, " Export File Name: metricJ2EE_Server.csv"
- Table 6–18, " Export File Name: metricJ2EE_Servlet.csv"

Table 6–7 Export File Name: metricBEA_ChannelInstance.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Fully qualified name of the channel
channelID	Fully qualified name of the channel
serviceID	URL of the service / JPD
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
containerID	Name of the Oracle WebLogic Server instance
nodeID	Name of the physical machine
domainID	Name of the Oracle WebLogic domain
displayNameID	Display name
resourceID	Name of the monitored resource as configured by the user
entityTypeID	Type of the monitored entity
Metric.J2EE.ChannelInstance.MessageCount	JMX metric
Metric.J2EE.ChannelInstance.DeadMessageCount	JMX metric

Table 6–8 Export File Name: metricBEA_ProcessType.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Implementation class name
processID	Display name of the process
serviceID	URL of the service / JPD
projectID	Name of Workshop project / web application module
containerID	Name of the Oracle WebLogic Server instance
nodeID	Name of the physical machine
domainID	Name of the Oracle WebLogic domain
entityTypeID	Type of the monitored entity
applicationID	Name of the Application
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
deploymentID	Unique ID used by Oracle WebLogic to track application deployments
resourceID	Name of the monitored resource as configured by the user

Table 6–8 (Cont.) Export File Name: metricBEA_ProcessType.csv

Field	Description
displayNameID	Display name
controlContainerID	Implementation class name of the process
Metric.J2EE.ProcessType.Arrivals	Instrumentation metric -- number of arrivals
Metric.J2EE.ProcessType.Aborts	Instrumentation metric -- number of aborts
Metric.J2EE.ProcessType.ElapsedTime	Instrumentation metric -- average elapsed time
Metric.J2EE.ProcessType.Active	Instrumentation metric -- number of active requests
Metric.J2EE.ProcessType.VisitCount	Instrumentation metric -- number of completed requests
Metric.J2EE.ProcessType.Exceptions	Instrumentation metric -- number of exceptions

Table 6–9 Export File Name: metricBEA_TimerEventGenerator.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Fully qualified name of the channel
channelID	Fully qualified name of the channel
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
channelTxID	Fully qualified name of the channel
domainID	Name of the Oracle WebLogic domain
displayNameID	Display name
resourceID	Name of the monitored resource as configured by the user
entityTypeID	Type of the monitored entity
Metric.J2EE.TimerEventGenerator.MessageCount	JMX metric
Metric.J2EE.TimerEventGenerator.ErrorCount	JMX metric

Table 6–10 Export File Name: metricJ2EE_Dispatcher.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Fully qualified name of the Execute Queue
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
containerID	Name of the Oracle WebLogic Server instance
nodeID	Name of the physical machine
domainID	Name of the Oracle WebLogic domain
executeQueueID	Name of the Execute Queue as configured by user
displayNameID	Display name
resourceID	Name of the monitored resource as configured by the user

Table 6–10 (Cont.) Export File Name: metricJ2EE_Dispatcher.csv

Field	Description
entityTypeID	Type of the monitored entity
Metric.J2EE.Dispatcher.ServicedRequestsTotalCount	JMX metric
Metric.J2EE.Dispatcher.IdleThreads	JMX metric
Metric.J2EE.Dispatcher.PendingRequests	JMX metric

Table 6–11 Export File Name: metricJ2EE_EJB_Entity.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
methodID	Name of the EJB method executed
domainID	Name of the Oracle WebLogic domain
entityTypeID	Type of the monitored entity
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
ejbID	Name of the EJB
webApplicationID	Name of the web module
displayNameID	Display name
controlContainerTypeID	Identifies type of control
elementID	Implementation class name
processID	Display name of the process
serviceID	URL of the service / JPD
projectID	Name of Workshop project / web application module
containerID	Name of the Oracle WebLogic Server instance
nodeID	Name of the physical machine
ejbComponentID	Name of J2EE component that contains this EJB
applicationID	Name of the Application
resourceID	Name of the monitored resource as configured by the user
controlContainerID	Implementation class name of the process
Metric.J2EE.EJB.Entity.Locking.LockManagerAccessCount	JMX metric
Metric.J2EE.EJB.Entity.ResponseTime	Instrumentation metric -- response time
Metric.J2EE.EJB.Entity.Cache.BeansCurrentCount	JMX metric
Metric.J2EE.EJB.Entity.Cache.AccessCount	JMX metric
Metric.J2EE.EJB.Entity.Pool.WaiterCurrentCount	JMX metric
Metric.J2EE.EJB.Entity.Transaction.CommittedTotalCount	JMX metric
Metric.J2EE.EJB.Entity.Locking.WaiterTotalCount	JMX metric
Metric.J2EE.EJB.Entity.Transaction.TimedOutTotalCount	JMX metric

Table 6–11 (Cont.) Export File Name: metricJ2EE_EJB_Entity.csv

Field	Description
Metric.J2EE.EJB.Entity.Cache.HitCount	JMX metric
Metric.J2EE.EJB.Entity.Locking.WaiterCurrentCount	JMX metric
Metric.J2EE.EJB.Entity.Pool.IdleCount	JMX metric
Metric.J2EE.EJB.Entity.Locking.EntriesCurrentCount	JMX metric
Metric.J2EE.EJB.Entity.VisitCount	Instrumentation metric -- invocation count
Metric.J2EE.EJB.Entity.Locking.TimeoutTotalCount	JMX metric
Metric.J2EE.EJB.Entity.Pool.InUseCount	JMX metric
Metric.J2EE.EJB.Entity.Pool.WaiterTotalCount	JMX metric
Metric.J2EE.EJB.Entity.Pool.TimeoutTotalCount	JMX metric
Metric.J2EE.EJB.Entity.Transaction.RolledBackTotalCount	JMX metric
Metric.J2EE.EJB.Entity.Cache.ActivationCount	JMX metric
Metric.J2EE.EJB.Entity.Cache.PassivationCount	JMX metric

Table 6–12 Export File Name: metricJ2EE_EJB_Stateless.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Implementation class name
projectID	Name of Workshop project / web application module
nodeID	Name of the physical machine
containerID	Name of the Oracle WebLogic Server instance
domainID	Name of the Oracle WebLogic domain
ejbComponentID	Name of J2EE component that contains this EJB
entityTypeID	Type of the monitored entity
applicationID	Name of the Application
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
ejbID	Name of the EJB
resourceID	Name of the monitored resource as configured by the user
displayNameID	Display name
Metric.J2EE.EJB.Stateless.Transaction.TimedOutTotalCount	JMX metric
Metric.J2EE.EJB.Stateless.Pool.WaiterTotalCount	JMX metric
Metric.J2EE.EJB.Stateless.Pool.InUseCount	JMX metric
Metric.J2EE.EJB.Stateless.Transaction.CommittedTotalCount	JMX metric

Table 6–12 (Cont.) Export File Name: metricJ2EE_EJB_Stateless.csv

Field	Description
Metric.J2EE.EJB.Stateless.Transaction.RolledBackTotalCount	JMX metric
Metric.J2EE.EJB.Stateless.Pool.IdleCount	JMX metric
Metric.J2EE.EJB.Stateless.Pool.TimeoutTotalCount	JMX metric

Table 6–13 Export File Name: metricJ2EE_JDBC_ConnectionPool.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Name of JDBC connection pool
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
containerID	Name of the Oracle WebLogic Server instance
nodeID	Name of the physical machine
domainID	Name of the Oracle WebLogic domain
displayNameID	Display name
resourceID	Name of the monitored resource as configured by the user
entityTypeID	Type of the monitored entity
Metric.J2EE.JDBC.ConnectionPool.WaitingForConnectionCurrentCount	JMX metric
Metric.J2EE.JDBC.ConnectionPool.WaitingForConnectionHighCount	JMX metric
Metric.J2EE.JDBC.ConnectionPool.ActiveConnectionsHighCount	JMX metric
Metric.J2EE.JDBC.ConnectionPool.ActiveConnectionsCurrentCount	JMX metric
Metric.J2EE.JDBC.ConnectionPool.FailuresToReconnectCount	JMX metric
Metric.J2EE.JDBC.ConnectionPool.WaitSecondsHighCount	JMX metric
Metric.J2EE.JDBC.ConnectionPool.ConnectionDelayTime	JMX metric

Table 6–14 Export File Name: metricJ2EE_JMS_Destination.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Name of JMS destination
nodeID	Name of the physical machine
containerID	Name of the Oracle WebLogic Server instance
jmsServerRuntimeID	Name of the JMS server
domainID	Name of the Oracle WebLogic domain
jmsDistributedQueueMemberID	Name of the JMS distributed queue member
entityTypeID	Type of the monitored entity

Table 6–14 (Cont.) Export File Name: metricJ2EE_JMS_Destination.csv

Field	Description
jmsQueueID	Name of the JMS queue
jmsRuntimeID	Name of the JMS service
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
jmsDistributedQueueID	Name of the JMS distributed queue
resourceID	Name of the monitored resource as configured by the user.
displayNameID	Display name
Metric.J2EE.JMS.Destination.ConsumersCurrentCount	JMX metric
Metric.J2EE.JMS.Destination.BytesCurrentCount	JMX metric
Metric.J2EE.JMS.Destination.MessagesPendingCount	JMX metric
Metric.J2EE.JMS.Destination.BytesThresholdTime	JMX metric
Metric.J2EE.JMS.Destination.MessagesHighCount	JMX metric
Metric.J2EE.JMS.Destination.BytesReceivedCount	JMX metric
Metric.J2EE.JMS.Destination.MessagesReceivedCount	JMX metric
Metric.J2EE.JMS.Destination.BytesHighCount	JMX metric
Metric.J2EE.JMS.Destination.MessagesCurrentCount	JMX metric
Metric.J2EE.JMS.Destination.ConsumersTotalCount	JMX metric
Metric.J2EE.JMS.Destination.ConsumersHighCount	JMX metric
Metric.J2EE.JMS.Destination.BytesPendingCount	JMX metric

Table 6–15 Export File Name: metricJ2EE_JMS_Service.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Name of the JMS service
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
containerID	Name of the Oracle WebLogic Server instance
nodeID	Name of the physical machine
domainID	Name of the Oracle WebLogic domain
displayNameID	Display name
resourceID	Name of the monitored resource as configured by the user
entityTypeID	Type of the monitored entity
jmsRuntimeID	Name of the JMS service
Metric.J2EE.JMS.Service.ConnectionsHighCount	JMX metric
Metric.J2EE.JMS.Service.ConnectionsCurrentCount	JMX metric
Metric.J2EE.JMS.Service.JMSServersCurrentCount	JMX metric

Table 6–15 (Cont.) Export File Name: metricJ2EE_JMS_Service.csv

Field	Description
Metric.J2EE.JMS.Service.JMSServersHighCount	JMX metric
Metric.J2EE.JMS.Service.ConnectionsTotalCount	JMX metric
Metric.J2EE.JMS.Service.JMSServersTotalCount	JMX metric

Table 6–16 Export File Name: metricJ2EE_JVM.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Name of the JVM
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
containerID	Name of the Oracle WebLogic Server instance
nodeID	Name of the physical machine
domainID	Name of the Oracle WebLogic domain
displayNameID	Display name
resourceID	Name of the monitored resource as configured by the user
entityTypeID	Type of the monitored entity
Metric.J2EE.JVM.JRockit.HeapSizeCurrent	JMX metric
Metric.J2EE.JVM.JRockit.HeapFreeCurrent	JMX metric
Metric.J2EE.JVM.JRockit.PhysMemTotal	JMX metric
Metric.J2EE.JVM.JRockit.PhysMemUsed	JMX metric
Metric.J2EE.JVM.JRockit.GarbageCollectionCountTotal	JMX metric
Metric.J2EE.JVM.JRockit.GarbageCollectionTimeTotal	JMX metric
Metric.J2EE.JVM.HeapFreeCurrent	JMX metric
Metric.J2EE.JVM.JRockit.PhysMemFree	JMX metric
Metric.J2EE.JVM.JRockit.NursurySizeTotal	JMX metric
Metric.J2EE.JVM.JRockit.ActiveDaemonThreads	JMX metric
Metric.J2EE.JVM.JRockit.ActiveThreads	JMX metric
Metric.J2EE.JVM.HeapSizeCurrent	JMX metric
Metric.J2EE.JVM.JRockit.HeapUsedCurrent	JMX metric

Table 6–17 Export File Name: metricJ2EE_Server.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Name of the J2EE server instance
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.

Table 6–17 (Cont.) Export File Name: metricJ2EE_Server.csv

Field	Description
containerID	Name of the J2EE server instance
nodeID	Name of the physical machine
domainID	Name of the Oracle WebLogic domain
displayNameID	Display name
resourceID	Name of the monitored resource as configured by the user
entityTypeID	Type of the monitored entity
Metric.J2EE.Server.RestartsTotalCount	JMX metric

Table 6–18 Export File Name: metricJ2EE_Servlet.csv

Field	Description
StartTime	Clock time (long) at data insertion
EntityID	CAMM unique identifier for the monitored entity
elementID	Name of the servlet implementation class
applicationID	Name of the Application
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
containerID	Name of the Oracle WebLogic Server instance
nodeID	Name of the physical machine
domainID	Name of the Oracle WebLogic domain
servletID	Name of the servlet
webApplicationID	Name of the web module
displayNameID	Display name
resourceID	Name of the monitored resource as configured by the user
entityTypeID	Type of the monitored entity
Metric.J2EE.Servlet.InvocationTotalCount	JMX metric
Metric.J2EE.Servlet.ExecutionTimeAverage	JMX metric

7

Creating Custom Views

This chapter explains how to create custom views which allow users to combine arbitrary graphs, tables, and diagrams together. and to create their own navigation tree hierarchies.

7.1 Custom View Creation

You can create custom views in CAMM by dragging and dropping visual components. The following are steps on how to create a custom view.

To create a custom view:

1. Switch the lower-left pane to the **Custom Views** tab.
2. Right-click the **My CustomViews** node in the Custom Views pane.
3. Select **Create New Child** to start the Custom View creation process.
4. Click **OK** and the new custom view is added to the My Custom View tree.
5. Drag and drop view elements to create your custom views.
6. Click the graph name and drag it to **MyView** in the My Custom Views tree.
7. When you click the graph name and start dragging, make sure a plus (+) icon appears next to your mouse cursor.
8. As you move your mouse cursor to MyView, the same plus (+) icon should reappear when MyView is highlighted. This is an indication that the drag and drop operation is successful.
9. Release the mouse and the Drop Options window appears. You can select to add this view element or create a new view.
10. Click **OK**. Your custom view now contains the new view element.

In addition to adding one view element at a time, you can choose to add all elements displayed in the Main Display Window.

To add all elements displayed in the Main Display Window:

1. Click the 'hand' icon in the Main Display Window then perform a drag and drop operation.
2. When you click the 'hand' icon and start dragging, ensure a plus (+) icon shows next to your mouse cursor
3. As you move your mouse cursor to MyView, the same plus (+) icon should reappear when MyView is highlighted. This is an indication that the drag and drop operation is successful.

4. When you release the mouse, the Drop Options window appears. You can select to add these view elements or create a new view.
5. Click **OK**. Your custom view should now contain the new view elements.
6. To edit or delete a custom view, right-click the interested custom view.
7. Select Edit to open the Cell view manager window. In this window you can edit custom view, delete view elements, and reorder view elements.

8

Custom Dashboards

The Custom Dashboards feature is designed to allow you a more customized display of your everyday activities within the CAMM product. By default, the CAMM UI displays the most active components for the major entity categories. Examples include high-level entities such as Portals, BPEL Processes, ESBs, and Web Services as well as architectural infrastructure entities such as servers.

Custom Dashboards allows you to create your own dashboards and then utilize these dashboards as either the default display or as your own customized viewpoints. They are much more powerful than custom views because you have the ability to modify the various entities significantly as opposed to simply being able to drag-and-drop from existing views into a combined perspective.

This chapter includes the following sections:

8.1 Creating a Custom Dashboard

To initiate the creation of a new custom dashboard, click the **Custom Dashboards** tab in the lower-left hand pane of the CAMM UI.

8.1.1 Layout Templates

There are five different out-of-box layout templates to help format custom dashboards. Select the one with the appropriate layout for the custom dashboard you are looking for.

Click the **Custom Dashboards** option in the bottom left pane to view the templates. Each template has a different layout for components.

8.1.2 Configuring Custom Components

To configure custom components, perform the following steps:

1. Double-click a **Customizable Component** area to configure what you would like to include in each component.
2. Select a component from the drop-down list, for example, the Table type.

8.1.2.1 Custom Table

Construct the contents for the table component using the options in the **Edit Table** window.

8.1.2.1.1 Select Entity

1. Click **Entity Selection**.
2. Select the entries in the Entity table. The selections have to be logical and have a unique entry for each entity.
3. Click **Select Entity** to continue with the configuration.

> **Tip:** Entity Type names are displayed in the SLO configuration, so you can use the information displayed on that screen to determine the Entity Type for a given metric.

8.1.2.1.2 Select Aggregation

1. Click **Define Aggregation**.
2. The key selected must aggregate by the same key metric name.
3. Click **OK**.

8.1.2.1.3 Select Columns

1. Click **Column Selection.**
2. Click the column type buttons as needed on this window to add a column and its label. For example, click ***Add Formula*** and type in `$1*$2` . This means column 1 times column 2.
3. Type in the labels for the columns added. The labels display in the component table header for each column.
4. You can click the **Up**, **Down** or **Delete** buttons to edit the columns.

> **Note:** You must add at least one Property or Metric for the Table View to be displayed. The column types display with an abbreviated letter under the **Type** column. The sequential numbers on the left of this window indicate the column numbers used for formulas.

> **Tip:** Metric names are displayed in the SLO configuration, so you can use the information displayed on that screen to determine the correct name for a given metric.

5. Click the **Show** check mark to indicate if that column should display in the component table after you are done configuring.
6. Click **OK** to save the changes and return to the previous window.

8.1.2.1.4 Select Navigation View

You can map the navigation functionality in the new customer dashboard to other views in order to simplify the overall presentation by adding drill-down capabilities. Note that only expert users such as Oracle support services should utilize this feature because it requires a strong knowledge of the mapping capabilities as well as the application itself.

Views can be selected from a drop-down list, but once completed, you must take extra care to ensure that IDs match up appropriately to handle the drill-down event.

Note: Contact Oracle Support Services if you are interested in this advanced navigation view feature.

8.1.2.2 Custom Chart

You can use charts to provide graphical metrics in the custom dashboard. Routinely, these metrics are the key purpose of setting up the dashboard and are contextually linked to the other custom components being displayed. You can configure various graphs to display the metrics as desired.

Double-click again on a different **Customizable Component** area to configure what you would like to include in each component and select **Chart** from the drop-down list.

8.1.2.2.1 Select Chart Type / Dimensions

You may choose the dimensions of the chart as well as the chart type. Metrics in the chart can be represented by either a pie chart or time series.

1. Select **Time Series Chart Type**.
2. Click **OK**.

8.1.2.2.2 Select Entity

1. Click **Entity Selection**.
2. Select the entries in the Entity table. The selections have to be logical and have a unique entry for each entity.
3. Click **Select Entity** to continue with the configuration.

Tip: Entity Type names are displayed in the SLO configuration, so you can use the information displayed on that screen to determine the Entity Type for a given metric.

8.1.2.2.3 Select Metric

Select the metric that should be displayed in the chart.

Tip: Metric names are displayed in the SLO configuration, so you can use the information displayed on that screen to determine the correct name for a given metric.

8.1.2.2.4 Select Aggregation

Select the metric that should be displayed in the chart.

8.1.2.3 Custom Label

Next, configure a label which can be comprised of any type of text or HTML. A label would routinely be used to provide additional context within the scope of the custom dashboard. An example would be to provide a detailed label for the dashboard to make the content instantly recognizable by colleagues utilizing the new dashboard.

1. Double-click again on a different **Customizable Component** area to configure what you would like to include in each component and select **Label** from the drop-down list.
2. Provide text in the edit box representing what should be displayed in the custom dashboard and select the text alignment from the drop-down menu.
3. Click **OK**.

8.1.2.4 Custom Image

Next, configure an Image custom component. Custom images are static images that are uploaded to the CAMM Manager. They are kept in a server-side format. The most common usage of this feature is to provide you with either a static visual context for the overall information being viewed or to provide a more branded look to the overall custom dashboard being utilized. For a branded look, a company logo might be uploaded and displayed as part of the new custom dashboard.

Double-click again on a different **Customizable Component** area to configure what you would like to include in each component and select **Image** from the drop-down list.

8.1.3 Right-Click Menu for Custom Components

Titles can be changed by a simple right-click. By default, the Custom Components in the template are named numerically as Test #1, Test #2, Test #3, and so forth.

8.1.4 Save

When you have completed configuring the components:

1. Type in a view name in the Dashboard Editor window.
2. Click **Save View**.
3. The new template shows in the Custom Dashboards list.

8.1.5 Save View As

You can use an existing template and duplicate it to edit its configurations without having to recreate a template from scratch.

1. To duplicate a template, double-click its name from the Custom Dashboards window.
2. Type in a new name in the View Name field.
3. Click **Save View As**.
4. The new template view shows in the Custom Dashboards list.

8.2 Creating a Tabbed Dashboard

The tabbed dashboard view displays in the right pane directly across from the Oracle Tree node. You can create and customize the tabbed view using the Custom Dashboards.

1. Click **Create Tabbed**.
2. Type in the View Name. This name appears in the Custom Dashboards list window.

3. Click **Add Tab**.
4. Select the view for this tab from the displayed list.
5. Click **Create Tab**.
6. Type in the Tab Name in the empty text box.
7. Continue to add tabs as needed.
8. Click **Save View**.

8.3 Sharing Views

Use this option to assign a role with read, or read and write, permissions for others to view the dashboard. This allows other users to utilize the custom dashboard. Note that any changes or deletions made to an existing shared dashboard will affect all users that are referencing it.

8.4 Utilizing and Viewing Dashboards

Once you create a new dashboard, you can view it by navigating to the navigation tree in the lower left-hand pane of the CAMM UI.

In addition, you can select your custom dashboard as the main dashboard display by selecting **Set Dashboard View** from the lower left-hand navigation bar and accessing the right-click menu. You will be presented with a list of existing custom dashboards to select from. Once selected. the new custom dashboard will be displayed when the CAMM UI starts up or if you selects the top-most node. Note that the default dashboard can be assigned by using the top menu bar in addition to the navigation tree.

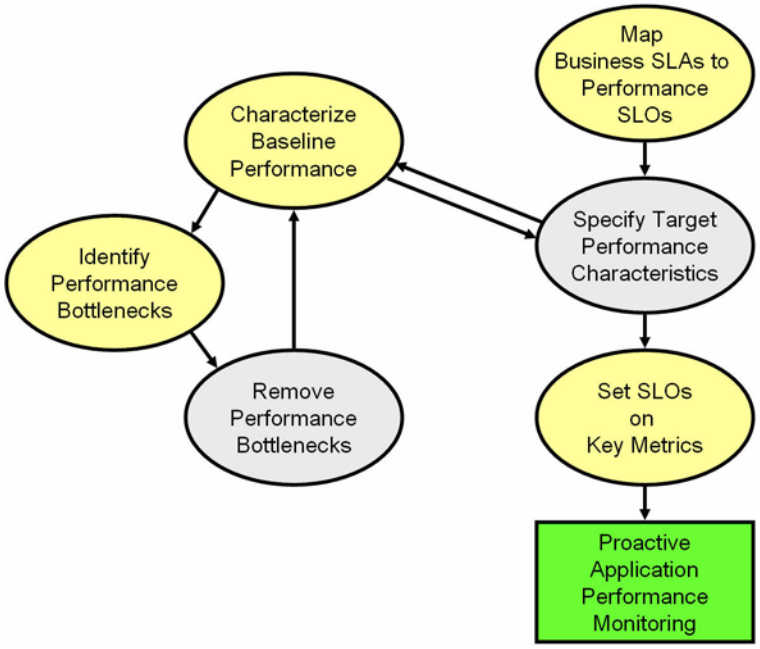
9

Oracle CAMM Methodology

CAMM automatically selects performance metrics and tracks contextual relationships for various applications. The CAMM Methodology focuses on other important activities to allow you to setup and maintain an effective application performance monitoring environment.

These activities include the following:

Figure 9–1 Steps of Oracle CAMM Methodology



The Oracle CAMM Methodology describes a series of steps for CAMM users to establish and maintain a proactive application performance monitoring environment leveraging CAMM's unique capabilities. Figure 9–1 illustrates these steps in a sequential order.

Methodology steps:

1. Map business SLAs to performance SLOs.

   The process of using agreed business SLAs to determine the value of performance SLOs.

2. Specify target performance characteristics.

Specify the ideal application performance characteristics using performance SLOs identified in step 1.

3. Characterize baseline performance.
4. Identify performance bottlenecks.
5. Remove performance bottlenecks.

   Steps 3, 4, and 5 should be grouped together to form a process of incremental performance improvement. Iterations of this process may be required to improve the application performance to meet the performance target as specified in step 2.
6. Set SLOs on key metrics.

Once application performance reaches the targeted goal, you need to set performance SLOs on key metrics to establish a proactive monitoring environment. This environment provides you with warnings when key performance metrics start to report abnormalities. These warnings enable you to proactively solve potential problems before they begin to impact business.

This chapter explains why the Oracle Methodology activities are important and covers the following activities in more detail:

- Map Business SLAs to Performance SLOs
- Characterize Baseline Performance
- Identify Performance Bottlenecks
- Set SLOs on Key Metrics

The detailed sections do not include Specify target performance characteristics (step 2) and Remove performance bottlenecks (step 5) because these activities typically do not involve the use of CAMM. Nevertheless, these activities are still integral parts of the Oracle Methodology.

9.1 CAMM Methodology Activities

The CAMM methodology activities include the following:

- Map Business SLAs to Performance SLOs
- Specify Target Performance Characteristics
- Performance Improvement Process

9.1.1 Map Business SLAs to Performance SLOs

To successfully setup a proactive application performance monitoring environment, the first step is to map a set of business objectives to a set of performance thresholds for you to monitor. These business objectives are often referred to as business service level agreements (SLAs). These business SLAs provide the basic application performance requirements at a high level. As such, mapping these high level SLAs to low level performance thresholds is often a very difficult activity to do well.

Using tools that only measure performance at technology levels (EJB, JSP, servlet, portlet, SQL calls, and so on) to perform this type of activity continues to be very difficult as the correlations between low-level metrics and high-level objectives are often fuzzy at best. Consequently, the mapping activity is considered by many as an art rather than a science.

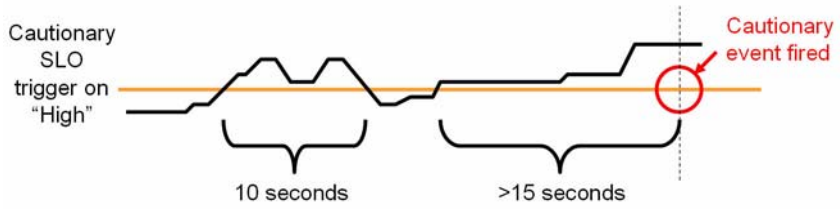
By measuring performance at both technology and functional levels, CAMM makes this mapping activity significantly less complicated. Since functional metrics measure performance for high level constructs such as business processes or portal desktops, mapping business SLAs directly to performance SLOs (Service Level Objectives) is straightforward.

9.1.2 Specify Target Performance Characteristics

Defining the target performance characteristics for the monitored applications is the next step after mapping business SLAs to performance SLOs. Since these SLOs represent absolute minimal performance requirements for these applications, using these *violation* thresholds as target performance characteristics makes little sense. Instead, you need to define what performance range is acceptable for normal operation and when to send out cautionary alerts for abnormal activities.

For some applications, it may be sufficient to just specify a set of *cautionary* performance thresholds. Application performance monitoring tools, such as CAMM, will send out cautionary alerts if these thresholds have been breached. Since these thresholds are cautionary, it may be acceptable to have a few violations before an alert is sent out. By defining the minimal violation duration, you can minimize the number of duplicate alerts generated. Figure 9–2 illustrates this concept.

Figure 9–2 Control Number of Alerts



In Figure 9–2, the minimal violation duration is defined to be 15 seconds. So if the cautionary state does not persist for more than 15 seconds, no cautionary alert would be fired.

For other applications, it is necessary to define both a high and low performance thresholds. Having both thresholds would effectively define a normal range of operation for these applications. With CAMM, both high and low triggers can be set for any SLO.

With a set of clearly defined target performance characteristics, you are able to determine how much performance tuning is needed to achieve the ideal performance range. You will also have a set of cautionary performance thresholds to enable a proactive application performance monitoring environment to be established.

9.1.3 Performance Improvement Process

The activities explained in this section should be grouped together as a single performance improvement process. This process would start with characterizing the baseline performance of our application, move on to identifying performance bottlenecks, and finish with removing performance bottlenecks. We would continue to perform these activities in iterations until the performance of our application meets the target characteristics.

9.1.3.1 Characterize Baseline Performance

Once the specification of the target performance characteristics is completed, the next activity is to capture the performance baseline for our application. The performance baseline will be compared with the set of target performance characteristics to determine if further performance improvement is needed. If so, you will improve application performance iteratively through the next two steps until the performance meets the target characteristics.

9.1.3.2 Identify Performance Bottlenecks

To identify performance bottlenecks, you must first isolate performance abnormalities in your performance baseline. Once you isolate a performance abnormality, you need to determine if this issue is a localized occurrence or a systematic problem. By using monitoring and diagnostic tools available to you, you can perform the analysis needed to identify the cause of the performance bottleneck.

9.1.3.3 Remove Performance Bottlenecks

Once these performance bottlenecks are identified, you need to determine how to remove them. The strategies for bottleneck removal vary by cause. A few examples follow:

- If the cause of the bottleneck is an application defect, the strategy would involve the application development team.
- If the bottleneck is caused by a configuration problem, you would request assistance from system administrators.
- If the bottleneck is in the application server or framework, you would seek help from those vendors.

The following is a list of possible bottleneck removal activities:

- Change application code to fix defects
- Modify environment setting to fix configuration problem
- Install patches to fix software defects
- Replace defective hardware
- Upgrade network infrastructure
- Add computing resources
- Remove resource hogging programs
- Tune back-end connectivity and response time

As you can see from the list, the Remove Performance Bottlenecks activity varies widely by cause. Correctly and quickly finding the appropriate groups to help resolve performance bottlenecks is the key for success for this activity. Once the performance bottleneck removal is completed, you must redo the Characterize Baseline Performance activity to confirm the fix implemented indeed improved performance.

9.1.3.4 Set SLOs on Key Metrics

In addition to setting application specific performance thresholds, it is also important to set performance thresholds on key system metrics and on some selective component metrics. Setting these thresholds will help you establish an early warning system and alert you to smaller issues before they manifest into big production problems.

Setting SLOs on key system metrics involves some basic understanding of how the system behaves under load. If the system becomes unstable or performs poorly when it runs out of free JDBC connections or idle ExecuteQueue threads, these system metrics should be monitored.

To determine which system metrics to monitor, it is critical to figure out the correlations between overall system performance and specific system metrics. You would use this information to decide which of the system metrics to monitor. Once appropriate system metrics are identified, you will then determine the performance range for normal operation and figure out the cautionary as well as violation thresholds.

While it is fairly straightforward to determine which system metrics to monitor and what system metric performance thresholds to set, setting SLOs on key component metrics is significantly more difficult. In theory, you can assume performance degradation at the component level would negatively impact application level performance. However, this assumption may not accurately reflect reality.

To predict application performance by monitoring component level performance metrics, there must be a very strong correlation between the performance of a specific component and that of the application. Sometimes, a drop in performance in one component is compensated by a jump in performance in another. These performance changes in opposite directions at the component level would essentially result in little change at the application level. Therefore, you must be careful not to draw conclusions by monitoring the performance of a few components unless there are strong correlations.

The last task to perform is to associate various actions and responses for various threshold violations. Once these associations are completed, you can begin to use your proactive application performance monitoring environment.

9.2 Map Business SLAs to Performance SLOs

One of the primary reasons companies purchase solutions to establish proactive application performance monitoring is the demand to meet business SLAs. Business SLAs for enterprise applications are a set of service level expectations defined by internal or external customers. In most cases, these business SLAs are defined at such a high level, they are not useful for setting thresholds in application performance monitoring tools.

As a result, the process of mapping business SLAs to performance SLOs is extremely important for companies to meet the service requirements set forth by their customers. Since CAMM monitors performance at both functional and technology levels, it is easy to perform this mapping exercise.

In this section, our example explains how to use CAMM to determine the proper performance threshold values for a set of business SLAs.

In the example, we were given the following high-level business SLAs:

Table 9–1 Example - Guidelines for Business SLAs

Business SLA	SLA Requirement
Fast customer self-service portal.	On average, pages in customer self-service portal should load within 2 seconds. This SLA must be fulfilled 99% of the time.
Customer service representative portal must be as fast as mainframe system.	All pages in customer service rep. portal must load within 6 seconds. This SLA must be fulfilled 99.9% of the time.
Fast to schedule a service call.	On average, scheduling a service call should take less than 30 seconds. This SLA must be fulfilled 99.99% of the time.

Let's map the first business SLA to a performance SLO. This SLA requirement states that the average response time for customer self-service portal (desktop) should be less than 2 seconds. In CAMM, we would set a high-level performance SLO at the desktop. Using the hierarchy in the CAMM UI, you would select the *customer* desktop and right-click to set the SLO.

Because CAMM monitors performance at both functional and technology levels, you can directly translate business SLAs to SLOs on functional metrics. In our example, it is the response time for portal desktop *customer*. For our example, we would proceed to set a violation SLO and a warning SLO.

We can calculate how often violations occur to figure out whether or not our current system is able to meet the SLA requirement 99% of the time. With CAMM, we can see whether there are any obvious violations. If there are any violations or close calls, we should confirm by examining actual data. If we have data for at least 24 hours, we would use the CAMM export function to prepare raw data for this calculation.

For the other two business SLAs, we would set performance SLOs on the appropriate metrics.

9.3 Characterize Baseline Performance

Also known as performance base-lining, characterize baseline performance involves a set of activities to capture the baseline performance of a system under specific level of load. For example, you can measure the baseline performance of a portal application deployed to a WebLogic cluster.

For example, you can have CAMM display the performance data during the first four hours of a load test. The number of active sessions grows at a steady pace for the first ninety minutes. Eventually, the number of active sessions stays at approximately seven hundred as the number of new and expiring sessions reach an equilibrium.

Visualize the portal performance following a typical pattern of slow performance initially and gradually reaching a steady state. The initial slow performance is expected as the application server load components into memory and populates data into its caching mechanism. The performance improves gradually and reaches a steady state after approximately thirty minutes. The performance pattern during this initial thirty-minute period can be characterized as *startup performance during increasing load*. After thirty minutes, performance of the portal application stabilizes.

CAMM's ability to quickly establish an application performance monitoring environment allows you to carry out *characterize baseline performance* painlessly. Because CAMM is able to monitor at cluster level as well as at individual server level, it can characterize performance for the entire cluster or individual servers.

By verifying that a less loaded server has lower resource utilization and faster performance, you can draw the following observations about the performance characteristics of a portal application running on this environment:

- Since Server A's resource utilization is near maximum, we can use the load on that server as the maximum limit for individual servers. We can calculate individual server maximum load limit by using the load metric provided by CAMM.
- We should examine the load balancing algorithm and the configuration of the load balancer.

Comparing load and resource usage of two servers in a cluster confirms resource usage is inversely correlated to the load.

Note: This is a very basic performance characterization of an individual server. Performance of a multi-server cluster cannot be calculated by multiplying performance characteristics of individual servers because of the overhead involved with a clustered configuration. True cluster level performance must be measured with application performance monitoring tools like CAMM.

9.4 Identify Performance Bottlenecks

CAMM can also be used to quickly identify performance bottlenecks in QA, staging, and production settings. You can use the CAMM hierarchical model to identify an application performance bottleneck. Furthermore, you can use CAMM to track down an application performance problem caused by resource starvation.

9.4.1 Performance Bottleneck of a Portal Application

In this example, we will identify the performance bottleneck of a portal application running on WebLogic. Since CAMM organizes performance metrics into a hierarchy, we would start the investigation at the root of the portal hierarchy.

In the example, the *sampleportal* application is barely meeting its SLO. To identify the components that could be the performance bottlenecks, we would expand the portal hierarchical tree to inspect lower level performance metrics.

Since there is only one active portal desktop, continue to look in the Acsera_Desktop_2 tree to the Pages node where you need to identify which page has the worst performance. If it is clear which page is the worst performing, the investigation would continue by traversing down the hierarchy under the slowest page. If there are multiple bad performing pages, investigation would need to continue down multiple paths until a slow performing component is identified.

In this case, the *My Page* is the slowest. Therefore, we will start the investigation by drilling down the hierarchy for My Page.

Drilling down further on the My Page hierarchy, you can compare performance of all portlets that make up the My Page. CAMM clearly identifies the *ThreadedDiscussion* portlet as the worst performing.

In this example, we are able to quickly identify the performance bottleneck in a portal application by using the hierarchical tree provided by CAMM. CAMM's unique ability to organize performance metrics in logical hierarchy allows us to perform performance bottleneck identification quickly and accurately.

9.4.2 Performance Application by Type

In addition to organizing performance metrics by logical hierarchies, CAMM organizes performance information by type. This next example uses multiple performance metrics from different hierarchies in CAMM to determine the source of the performance bottleneck.

In this example, CAMM alerted us that the average response time for *csr* desktop has exceeded both the cautionary and violation SLOs previously defined. Using CAMM to inspect these SLOs violations, we saw significant performance degradation during a ninety minute period. This continuous SLO violation warrants additional investigation to locate the source of this performance slowdown.

By looking at the *Desktop Hits* graph, we can see there was no sudden load increase for the csr desktop during the time period in question. This information allows us to eliminate one potential reason for the performance slowdown.

Next, we would drill down the portal hierarchy to see if any one component is behaving badly.

We do not see any specific portlet behave radically during the time period in question. In fact, when the desktop performance begins to degrade, the performance of all portlets also degrades. Based on this information, we can conclude that no specific portlet in the desktop was the source of this performance slowdown.

9.4.3 System Level Performance

Since the performance problem seems to affect all components in the same way, we should suspect there is some type of performance degradation at the system level. To view system level performance data, we would look under the *Resources* hierarchy. Performance metrics under the Resources hierarchy provides the raw data for us to perform correlation analysis. This type of analysis is needed to determine whether resource starvation is the cause of the performance slowdown.

To perform this type of analysis, we would pull together different performance tables and graphs to create custom views in CAMM. Using custom views, you can analyze the correlation between machine resources, JVM resources, and Application Server managed resources (JDBC connection pool).

Graphs can reveal some interesting patterns.

- OS Agent Abnormalities

  We noticed a sudden drop in CPU utilization and a sudden increase in disk utilization during the time period in question. This pattern indicates a large amount of virtual memory paging activities on this machine. Memory paging to disk is extremely expensive and slows down request processing as indicated by lowered CPU utilization. To understand why page is occurring, we will take a look at performance metrics on the JVM.

- JVM Heap Size

  We noticed that for the initial twelve hours of this example, both total JVM heap size and free JVM heap size grew at a steady pace. The growth of the total JVM heap size stopped at 512 MB - an expected behavior since we configured WebLogic to have a maximum heap size of 512 MB. While we expect the free JVM heap size to stop growing after total heap size reached 512 MB, the free JVM heap size actually starts to drop. Combining this information with some previously obtained information such as no sudden increase in load, we can conclude that there is a high likelihood of a memory leak.

This abnormal consumption of memory caused the total JVM heap to reach its pre-defined maximum. It is also very likely this memory leak caused the increase in virtual memory paging activities and corresponding reduction in CPU utilization. This reduction in CPU utilization impacts the response time for all components running in this machine including JDBC Connections.

- Memory Leak

  We were able to identify a memory leak in the WebLogic JVM gradually caused resource starvation and eventually impacts application performance. In order to further diagnose this problem, a deep-level memory profiling tool is required to understand memory usage of the JVM.

9.5 Set SLOs on Key Metrics

This step in the Oracle Methodology allows you to proactively monitor key system metrics to avoid catastrophic failures such as server hangs (non-responsive), server crashes, cluster hangs, and more. The ability to recognize signs leading up to these catastrophic failures is a must to maintain quality of service for your WebLogic infrastructure.

You can proactively set thresholds and actions for key WebLogic system metrics. Table 9–2 lists the key system metrics for the WebLogic Platform:

Table 9–2 List of Key System Metrics for WebLogic

Key System Metric	Reason to Monitor
ExecuteQueue Idle Thread Count	Running out of ExecuteQueue threads is often a precursor to application server hangs (non-responsive). In some severe cases, when the application server runs out of ExecuteQueue threads, all of its operations would stop working.
ExecuteQueue Pending Request Count	A steady increase in the number of ExecuteQueue pending requests is also a precursor to server hangs. This metric is inversely correlated with the ExecuteQueue Idle Thread Count metric.
Total JVM Heap Size	There are two reasons to monitor this metric: 1. If total JVM heap size grows to predefined maximum, a cautionary event should be fired notifying the administrator. 2. If total JVM heap size suddenly drops to 0, this may be an indication of a JVM crash or a non-operational application server.
Free JVM Heap Size	A steady decrease in the free JVM heap size is an indicator of either a memory leak or misconfigured application server. A JVM running out of heap will experience instability and performance degradation as garbage collector and JVM competes for resources to perform cleanup and object creation respectively.
Open Sessions Count	If open session count drops to 0 and remains at 0 for a period of time, some investigation is warranted. Often this pattern indicates a network or load balancing problem.
Application Invocation Count	If application invocation count drops to 0 and remains at 0 for a period of time, some investigation is warranted. While this pattern often indicates a network or load balancing problem, it could also be a symptom of a hanged server.

Understanding these key WebLogic system metrics, setting the SLO thresholds and assigning appropriate responses are critical to establishing a proactive monitoring. In this example, we will configure SLOs and actions with CAMM.

The first task is to set a cautionary and a violation SLO for ExecuteQueue Idle Thread Count metric so the appropriate person can be alerted when available ExecuteQueue is running low. To configure SLOs, right-click on the Execute Queues metric and select

Configure service level objects. In this example, we will create the following SLOs for ExecuteQueue Idle Threads:

Table 9–3 SLOs for ExecuteQueue Idle Threads

SLO Name	Metric	Threshold Type	Threshold Value	Trigger On
Low ExecuteQueue Idle Threads	Metric.J2EE.Dispatcher.IdleThreads	Cautionary	3	Low
ExecuteQueue Idle Threads Exhaustion	Metric.J2EE.Dispatcher.IdleThreads	Violation	0	Low

When SLO trigger is set to Low, CAMM will fire an alert when current measurement reaches the threshold value AND the previous measurement has a higher value than the threshold.

For example, we would create the following actions for the SLOs previously configured:

Table 9–4 Actions for SLO

SLO Name	Action Name	Action Type
Low ExecuteQueue Idle Threads	Enter Low ExecuteQueue Idle Threads event into server log	Log
ExecuteQueue Idle Threads Exhaustion	Email ExecuteQueue Idle Threads Exhaustion alert	Email
	Send ExecuteQueue Idle Threads Exhaustion SNMP trap to HP Overview	SNMP
	Enter ExecuteQueue Idle Threads Exhaustion event into server log	Log

After configuring these SLOs and actions, we now have a proactive monitoring environment to detect ExecuteQueue resource starvation related problems before a catastrophic event occurs. We would use this approach to establish proactive monitoring for other key WebLogic system metrics.

The following is the Oracle recommendation:

Table 9–5 ExecuteQueue Pending Requests

SLO Name	Metric	Threshold Type	Threshold Value	Trigger On
ExecuteQueue Pending Request Warning	Metric.J2EE.Dispatcher. PendingRequests	Cautionary	5 ~ 10[1]	High
ExecuteQueue Pending Request Violation	Metric.J2EE.Dispatcher. PendingRequests	Violation	10 ~ 20	High

[1] **Threshold values for these SLOs vary by environment.** Figuring out what threshold values to use is an iterative process. Users should gather information about the performance characteristic of their WebLogic environment as the first step. Based on this information, users can set SLOs accordingly. As users continue to improve the performance of their WebLogic environment, they should re-evaluate these threshold values and change them as needed.

When SLO trigger is set to High, CAMM will trigger an alert when current measurement hits the threshold value.

Table 9–6 Total JVM Heap Size

SLO Name	Metric	Threshold Type	Threshold Value	Trigger On
JVM Heap Reached Max	Metric.J2EE.JVM.HeapSizeCurrent	Cautionary	512 MB[1]	High
JVM Heap Reached 0	Metric.J2EE.JVM.HeapSizeCurrent	Violation	0 MB	Low

[1] **Threshold value for this SLO varies by environment.** Users would set this value to the maximum heap size specified in the WebLogic configuration file.

Table 9–7 Free JVM Heap Size

SLO Name	Metric	Threshold Type	Threshold Value	Trigger On
Low JVM Free Heap Warning	Metric.J2EE.JVM.HeapFreeCurrent	Cautionary	72 MB	Low
Low JVM Free Heap Violation	Metric.J2EE.JVM.HeapFreeCurrent	Violation	24 MB	Low

Table 9–8 Open Session Count

SLO Name	Metric	Threshold Type	Threshold Value	Trigger On
No user session in system for 5 minutes	Metric.J2EE.WebApplication. OpenSessionCurrentCount	Cautionary	0[1]	Low

[1] In the example, this SLO would have a measurement window of 5 minutes. By setting the measurement window to 5 minutes, CAMM will fire an alert only if this condition persists for at least 5 minutes.

Table 9–9 Application Invocation Count

SLO Name	Metric	Threshold Type	Threshold Value	Trigger On
No application invocation in system for 5 minutes	Metric.J2EE.Servlet.InvocationTotalCount	Cautionary	0	Low

Setting these SLOs and corresponding actions establishes a proactive monitoring environment for your WebLogic deployment. This proactive monitoring approach allows you to identify problems leading up to catastrophic problems before they impact your system's performance and availability.

9.6 Conclusion

The Oracle CAMM Methodology is a critical aspect of your application performance management strategy. By following this methodology carefully, you will be able to use CAMM to improve your ability to proactively monitor the performance and availability of your deployed applications and WebLogic infrastructure. CAMM's automation reduces time, effort, and errors associated with manual processes. This allows CAMM users to focus on other crucial activities such as the ones listed in the Oracle Methodology.