

OracleTM Composite Application Monitor and Modeler (CAMM)TM 7.5 User Guide

ORACLE®

Copyright © 2008 Oracle, Inc.

Trademarks

Oracle AS[®] and Oracle SOA Suite[®] is a registered trademark of Oracle[®] Corporation in the United States, other countries, or both.

WebLogic[®] is a registered trademark of BEA[®] Corporation in the United States, other countries, or both.

WebSphere[®] is a registered trademark of International Business Machines Corporation in the United States, other countries, or both.

Sun[®], Sun Microsystems, Java[™], and Solaris[™] are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Microsoft[®], Windows[®], Windows NT[®], and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Intel[®], Intel logo, Intel Inside[®], Intel Inside logo, Intel Centrino[®], Intel Centrino logo, Celeron[®], Intel Xeon, Intel SpeedStep[®], Itanium[®], and Pentium[®] are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX[®] is a registered trademark of The Open Group in the United States and other countries.

Linux[®] is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Document Revised: October 29, 2008

Table of Contents

Chapter 1: About This Guide	21
Overview	21
Audience	22
Getting Help	22
Using This Guide	22
Typographical Conventions	22
Feedback	24
Chapter 2: Introduction	25
Overview	25
Terminology	26
Managing Complex J2EE™ and SOA Applications	27
Delivering a Service-Oriented View Across Environments	29
Avoiding Involvement from J2EE™, SOA, and Application Experts	31
Eliminating repetitive Do-It-Yourself (DIY) manual processes	32
Solution	33
Architecture	33
CAMM™ Java Agents	35
CAMM™ Manager	35
CAMM™ Database	35
CAMM™ User Interface	35
Chapter 3: Exploring the User Interface	37
Starting CAMM™ UI	38
Web Browser Access	38
Command Line Access	39
General CAMM™ UI Elements	40
Navigating CAMM™	42
Drill Down in Operational Dashboard	42
Drill Down in Monitor Workspace	46
In WebLogic®	46
In Oracle SOA Suite®	47
In WebSphere®	49
Configuring Service Level Objectives (SLOs)	52
Create New SLO	53
Defining SLO Parameters	54
Configure SLO Blackouts	59
Time Frame	62
Display Interval	64
Time Frame	64
Interval Context	65
Turning Off Time Frame Limitation	66
Refresh Rate	66
Queries	67
URL Query	67
Graphs and Data Items	69
Right-Click Operations on Tables and Graphs	72
Comparative View	73
Save as PDF	76
Easy Scroller	78

Zoom In and Zoom Out Toolbar.....	79
Custom Metrics.....	79
Promote To Dashboard.....	82
Functional View.....	85
Topology View.....	87
Edge Types and Colors.....	88
Architecture View.....	93
Metric Types.....	103
Chapter 4: Exploring the Monitor Workspace.....	105
Oracle™ Tree.....	106
WebLogic® Portals.....	106
Desktops.....	111
Display Portal Desktop - Desktop Structure Viewer.....	113
Portlet Drill Down.....	115
Pageflow Viewer.....	117
Books.....	118
Pages.....	120
Portlets.....	122
WebSphere® Portals.....	124
Virtual Portals.....	128
Display Virtual Portal - Structure Viewer.....	130
Pages.....	131
Portlets.....	133
Oracle® BPEL Processes.....	134
Delay Analysis View.....	139
.....	139
Metadata View.....	139
Partner Links View.....	142
Partner Link Type Role View.....	143
.....	143
Partner Link Bindings View.....	144
Modeled Entities View.....	145
Topology View.....	146
Node Hierarchy.....	147
Oracle® ESB.....	148
Service Details View.....	151
Service Parent Details View.....	152
Service Definition View.....	153
Service Operations View.....	155
Operation Routing Rules View.....	156
Processes.....	156
Node Hierarchy.....	160
Delay Analysis View.....	162
Events View.....	162
Persistent Containers.....	165
Entity EJB Activity Table.....	168
Entity EJB Cache Table.....	169
Entity EJB Transactions Table.....	169
Entity EJB Locking Table.....	170
Instrumentation.....	170
WLI Web Services.....	171
Public Operations.....	174
Instrumentation.....	176

Web Services	177
Pageflows.....	179
Services	180
HTTP	181
EJBs	184
JDBC's	185
WSRP Producers.....	185
WSRP Summary	186
WSRP Topology.....	189
Display Portal Desktop.....	190
Integration	192
Health.....	194
Execute Queues.....	194
Async Dispatchers.....	196
Sync Dispatchers	198
JMS Destinations	199
Stateless Containers	202
Persistent Containers	203
Performance	207
Channels	209
Subscribers.....	210
Applications	211
Services	216
Dependencies	216
Deployments	219
Workshop Projects	221
Web Applications.....	223
Stateless Beans	225
Stateful Beans.....	228
Stateful EJB Cache	229
Stateful EJB Transactions.....	230
Stateful EJB Locking.....	231
Entity Beans.....	232
Entity EJB Activity.....	235
Entity EJB Cache	236
Entity EJB Transactions	237
Entity EJB Locking.....	238
Message Driven Beans.....	239
Message Driven EJB Activity.....	240
Message Driven EJB Transactions.....	241
WebLogic® Resources	242
WebSphere® Resources	245
Oracle® Resources	248
Custom Metrics.....	249
Grid Control™	251
Chapter 5: Exploring Configuration Workspace.....	273
Resource Configuration.....	274
User Configuration.....	276
Admin Role	277
Operator Role.....	277
User Role.....	278
Service Level Objectives by Name	279
Service Level Objectives by Metrics	280

Service Level Objectives by Entity Type	281
Action Configuration	282
Chapter 6: Performance Analytics	289
Entity Performance Ranking	289
Performance Characterization	291
Multi-Point Performance/Load Regression	291
Performance/Load Scattergram	292
Time-Based Performance Distribution	293
Performance Histogram	293
Time-based Performance Trend	294
Memory Leak Detection	294
Drill Down - Bottleneck Analysis	296
Drill Out - Impact Analysis	304
Chapter 7: Exporting Data	311
Remote Log Retrieval	311
Lardemo.sh File	311
Lardemo.sh File	313
Data Export Modes	323
Export to File	323
Export to Database	323
Aggregation Export to File	323
Data Export Configuration	323
Example of Exported Data for WebLogic	323
Chapter 8: Creating Custom Views	337
Custom View Creation	337
Instance Specific Custom Views	342
Chapter 9: Custom Dashboards	343
Overview	344
Working with Custom Dashboards	344
Create Custom Dashboard	344
Layout templates	345
Configuring Custom Components	348
Custom Table	348
Custom chart	351
Custom label	354
Custom image	354
Right-Click Menu for Custom Components	355
Save	355
Save View As	355
Create Tabbed Dashboard	355
Share Views	357
Preview dashboards	357
Chapter 10: Oracle CAMM™ Methodology	361
Oracle™ Methodology Activities	363
Map Business SLAs to Performance SLOs	363
Specify Target Performance Characteristics	364
Performance Improvement Process	365
Characterize Baseline Performance	365
Identify Performance Bottlenecks	365
Remove Performance Bottlenecks	366
Set SLOs on Key Metrics	367

Map Business SLAs to performance SLOs.....368
Characterize Baseline Performance370
Identify Performance Bottlenecks376
Set SLOs on Key Metrics383
Conclusion.....387

List of Figures

Figure 2-1: Topology View in CAMM™	30
Figure 2-2: AppSchema™ View in CAMM™	31
Figure 2-3: Service Mode CAMM™ Topology	34
Figure 2-4: Application Mode CAMM™ Topology	34
Figure 3-1: Accessing CAMM™ UI through a web browser	39
Figure 3-2: Accessing CAMM™ UI Using Client.bat	39
Figure 3-3: General Toolbar and Tabs	40
Figure 3-4: Workspace, Main Display Window, Custom Views	41
Figure 3-5: CAMM™ Operational Dashboard	42
Figure 3-6: Health Indicator Drill Down	43
Figure 3-7: Health Indicator Drill Down Details	43
Figure 3-8: Process Flow Viewer	44
Figure 3-9: Operational Dashboard Drill Down	45
Figure 3-10: Operational Dashboard Drill Down, Continued	45
Figure 3-11: Drill Down in WebLogic® Monitor Workspace	46
Figure 3-12: Monitor Workspace Drill Down Details	46
Figure 3-13: Portlet Drill Down	47
Figure 3-14: Drill Down in Oracle SOA Suite® Monitor Workspace	48
Figure 3-15: Oracle SOA Suite® Monitor Workspace Drill Down Details	48
Figure 3-16: BPEL Drill Down	49
Figure 3-17: Drill Down in WebSphere® Monitor Workspace	49
Figure 3-18: Monitor Workspace Drill Down Details	50
Figure 3-19: Portlet Drill Down	51
Figure 3-20: Right-Click to Configure SLOs	52
Figure 3-21: Select a Specific SLO File to Store New SLO Configuration	53
Figure 3-22: Create New SLO: Define SLO Entity Type	53
Figure 3-23: Create New SLO: Define SLO Parameters	54
Figure 3-24: Initial Operational Dashboard with Gray Status Lights	55
Figure 3-25: Setting SLO on Portlet Activates Health Indicators	56
Figure 3-26: SLOs in Four Buckets	57
Figure 3-27: Right-click on Tree Elements to View SLO Events	58
Figure 3-28: SLO Events Viewer	58
Figure 3-29: Various Line Types Represent Different SLOs Graphically	59
Figure 3-30: SLO Blackout	60
Figure 3-31: SLO Blackout List	60
Figure 3-32: SLO Blackout Summary List	61
Figure 3-33: SLO Blackout Configuration	61
Figure 3-34: Time Frame Selection	63
Figure 3-35: Portal Campaign Response Time in Different Time Frames	64
Figure 3-36: Impact of Time Frame Change on Display Interval	64
Figure 3-37: Interval Context set to End Time is Current System Time	65
Figure 3-38: Interval Context set to End Time is Fixed	65

Figure 3-39: Date/Time Selector	65
Figure 3-40: Use Time Frame? Check Box	66
Figure 3-41: Refresh Rate Drop-down Box	66
Figure 3-42: Copy URL	67
Figure 3-43: Queries Menu	67
Figure 3-44: Enter URL Query	68
Figure 3-45: Found the Portal Desktop View Relevant to the URL	68
Figure 3-46: Mouse Over to Display Data Item Details	69
Figure 3-47: Min/Max Metrics	69
Figure 3-48: Sort Data by Column	70
Figure 3-49: Rearranging Table Columns	70
Figure 3-50: Zoom In to See Graph Details	71
Figure 3-51: Zoom in to See Graph Details	71
Figure 3-52: Right-Click Operations to Get a Total Row Count	72
Figure 3-53: Right-Click to Activate Comparative View	73
Figure 3-54: Comparative Views to Evaluate Performance Characteristics	75
Figure 3-55: Right-Click on Any View to Save As a PDF	76
Figure 3-56: File Sharing Improves Collaboration	77
Figure 3-57: Use Easy Scroller to Navigate Complex Views	78
Figure 3-58: Drop-down Box Enables Quick Zoom In / Zoom Out	79
Figure 3-59: Configure Custom Metric	80
Figure 3-60: Define and Modify Custom Metrics	80
Figure 3-61: Custom Metrics Node	82
Figure 3-62: Custom Metric Reports Class Level Performance Data	82
Figure 3-63: Dashboard Configuration Window	83
Figure 3-64: Create Dashboard Promotion	83
Figure 3-65: Define Display Name	83
Figure 3-66: Dashboard Promotion Configuration	84
Figure 3-67: New Entry on Operational Dashboard	84
Figure 3-68: Accessing Functional View	86
Figure 3-69: Process Functional View	86
Figure 3-70: Web Service Functional View	87
Figure 3-71: Highest level Topology View	87
Figure 3-72: Mouse Over Specific Arrow for Tool Tip	88
Figure 3-73: Hide Arrows in Topology View	89
Figure 3-74: Hide Other Edges	90
Figure 3-75: Drill Down to Reveal Relationships	90
Figure 3-76: Drill Down to Display Relationship	91
Figure 3-77: Access Tier Dependency Topology View	91
Figure 3-78: Architecture View	93
Figure 3-79: Active Calling Relationships	95
Figure 3-80: Application Specific Architecture View	96
Figure 3-81: Details of a Specific Call in Architecture View	97
Figure 3-82: Edge Types	98
Figure 3-83: Hide Other Edges	99
Figure 3-84: Access Architecture View - Summary View	100

Figure 3-85: Detail Performance Data	101
Figure 3-86: Instrumentation Tab	102
Figure 3-87: Errors/Exceptions Tab	102
Figure 3-88: SQL Statements Tab	103
Figure 4-1: Main Nodes in the Oracle™ Tree	106
Figure 4-2: Portal Performance Table and Graphs	108
Figure 4-3: Portals Node Expanded	108
Figure 4-4: Display css Portal Specific Metrics	109
Figure 4-5: Portal Level Nodes Tabs	109
Figure 4-6: Desktop Summary for css Portal Application	111
Figure 4-7: csr Desktop Specific Metrics and Thresholds	112
Figure 4-8: Expanded csr Desktop Tree	113
Figure 4-9: Desktop Structure Viewer	113
Figure 4-10: Desktop Structure Viewer	114
Figure 4-11: Pageflow Viewer with Flow View	115
Figure 4-12: Drill Down Portlet to Access Architecture View	116
Figure 4-13: Flow or Component View in Pageflow Viewer	117
Figure 4-14: Pageflow Viewer with Component View	117
Figure 4-15: csr Desktop Expanded to Show Books Node	118
Figure 4-16: Expand Books to View Cases	119
Figure 4-17: Cases Book Node Expanded to Show Pages Node	120
Figure 4-18: Expanding the Manage Case Page Reveals the Portlets Node	122
Figure 4-19: Manage Case Page Node Expanded to Show Portlets Node	123
Figure 4-20: Expanding the Portlets Node Reveals the Specific Portlet Nodes	124
Figure 4-21: WebSphere® Portal Performance Table and Graphs	126
Figure 4-22: WebSphere® Portals Node Expanded	126
Figure 4-23: Display WebSphere Portal Specific Metrics	127
Figure 4-24: WebSphere® Portal Level Nodes Tabs	127
Figure 4-25: Desktop Summary for WebSphere Portal Application	128
Figure 4-26: Portlet Specific Metrics and Thresholds	129
Figure 4-27: Display Virtual Portal Structure Viewer	130
Figure 4-28: Virtual Portal Structure Viewer	131
Figure 4-29: Virtual Portal Node Expanded to Show Pages Node	132
Figure 4-30: Content Root Node Expanded to Show Portlets Node	134
Figure 4-31: BPEL Process Summary	135
Figure 4-32: BPEL Processes Node Expanded	136
Figure 4-33: SOAOrderBooking BPEL Process Summary	137
Figure 4-34: Functional Work Flow View	137
Figure 4-35: Delay Analysis for RequestQuote Process	139
Figure 4-36: Metadata View	140
Figure 4-37: Partner Links View	142
Figure 4-38: Partner Link Type Role View	143
Figure 4-39: Partner Link Bindings View	144
Figure 4-40: Modeled Entities View	145
Figure 4-41: BPEL Topology View	146
Figure 4-42: SOAOrderBooking BPEL Process Node Hierarchy Summary	147

Figure 4-43: Drill Down to See the Entire BPEL Node Hierarchy	147
Figure 4-44: Process Node Specific Performance Information	147
Figure 4-45: ESB Summary	148
Figure 4-46: ESB Systems Node Expanded	150
Figure 4-47: ESB Systems Topology	150
Figure 4-48: OrderBooking ESB Summary	151
Figure 4-49: OrderBooking ESB Service Details View	152
Figure 4-50: OrderBookingProcess Service Parent Details View	153
Figure 4-51: OrderBookingProcess Service Definition View	154
Figure 4-52: OrderBookingProcess Service Operations View	155
Figure 4-53: OrderBookingService Operation Routing Rules View	156
Figure 4-54: Processes Summary	157
Figure 4-55: Processes Node Expanded	158
Figure 4-56: RequestQuote Process Specific Metrics	159
Figure 4-57: Functional Work Flow View	159
Figure 4-58: RequestQuote Process Node Hierarchy Summary	160
Figure 4-59: Delay Analysis for RequestQuote Process	162
Figure 4-60: Drill Down to See the Entire Node Hierarchy	163
Figure 4-61: Process Node Specific Performance Information	164
Figure 4-62: Persistent Containers Summary	165
Figure 4-63: Persistence Containers Summary Information	166
Figure 4-64: Persistence Containers Summary Information Continued	167
Figure 4-65: WLI Web Services Summary	172
Figure 4-66: Web Services Node Expanded	173
Figure 4-67: SchedulingSystemWS Web Service Specific Metrics	173
Figure 4-68: SchedulingSystemWS Public Operations Summary	174
Figure 4-69: Class Panel Displays Instrumentation Data at the Class Level	176
Figure 4-70: Instrumentation Data at the Method Level	176
Figure 4-71: Web Services Summary in CAMM™	177
Figure 4-72: Web Services Node Expanded	178
Figure 4-73: MedRecWebServices Performance Data	178
Figure 4-74: Pageflows Summary	179
Figure 4-75: EJB Service Entry Point Activity Summary	181
Figure 4-76: Admin HTTP Service	181
Figure 4-77: Method Level Performance Data	183
Figure 4-78: WSRP Producers	185
Figure 4-79: WSRP Producers Summary	186
Figure 4-80: WSRP Details	187
Figure 4-81: WSRP Consumer Portlet Performance	189
Figure 4-82: WSRP Producer Portlet Performance	189
Figure 4-83: WSRP Topology	190
Figure 4-84: Display Portal Desktop	191
Figure 4-85: Integration Summary	192
Figure 4-86: Expanded Integration Tree	193
Figure 4-87: Execute Queues Summary	194
Figure 4-88: Async Dispatcher Summary	196

Figure 4-89: Sync Dispatchers Summary	198
Figure 4-90: JMS Destinations Summary	199
Figure 4-91: Stateless Containers Summary	202
Figure 4-92: Persistent Containers Summary	203
Figure 4-93: Performance Summary	207
Figure 4-94: Channels Summary	209
Figure 4-95: Subscribers Summary	210
Figure 4-96: Applications Summary	211
Figure 4-97: Expanded Tree for cssdemo Application and cssdemo Summary .	213
Figure 4-98: Services - Summary View	216
Figure 4-99: Dependencies - Associated Components	217
Figure 4-100: Dependency - Data Sources	218
Figure 4-101: Deployment Node	220
Figure 4-102: Deployment - Module Detail	220
Figure 4-103: Workshop Projects Nodes and It's Children	222
Figure 4-104: Expanded Web Applications Node with Performance Summary	223
Figure 4-105: Fully Expanded Web Applications Node	224
Figure 4-106: Expanded Stateless Beans Node with Activity Summary	225
Figure 4-107: Further Expand Stateless Beans	226
Figure 4-108: Expanded Stateful Beans Node With Activity Summary	228
Figure 4-109: Expanded Entity Beans Node With Activity Summary	232
Figure 4-110: Entity Beans - Detail Activity	234
Figure 4-111: Expanded Message Driven Beans Node With Activity Summary	239
Figure 4-112: Expanded Resources Tree for WebLogic®	242
Figure 4-113: JRockit related graphs under JRockit Node	243
Figure 4-114: OS Agent Metrics	244
Figure 4-115: Expanded Resources Tree for WebSphere®	245
Figure 4-116: Thread Pools Expanded Node	247
Figure 4-117: OS Metrics	247
Figure 4-118: Expanded Resources Tree for Oracle®	248
Figure 4-119: OS Metrics	249
Figure 4-120: Custom Metrics - Expanded	250
Figure 4-121: Custom Metrics Node Performance Summary	250
Figure 5-1: Resource Configuration	274
Figure 5-2: New Resource Creation Form	274
Figure 5-3: Resource Configuration Form for WebLogic®	275
Figure 5-4: Resource Configuration Form for BEA Admin Server	276
Figure 5-5: User Configuration	276
Figure 5-6: Change User Account Configuration	278
Figure 5-7: Service Level Objectives by Name	279
Figure 5-8: Service Level Objective Editor	280
Figure 5-9: Service Level Objectives by Metrics	281
Figure 5-10: Service Level Objectives by Entity Type	282
Figure 5-11: Action Configuration	283
Figure 5-12: SNMP Trap Action Configuration	284

Figure 5-13: Send E-Mail Action Configuration	284
Figure 5-14: Script Action Configuration	285
Figure 5-15: Log Action Configuration	285
Figure 5-16: Enforcing Referential Integrity	285
Figure 6-1: Entity Performance Ranking View	290
Figure 6-2: Entity Performance Ranking with a WebSphere® Portal	291
Figure 6-3: Multi-point Regression	292
Figure 6-4: Performance Scattergram	292
Figure 6-5: Time-Based Performance Distribution	293
Figure 6-6: Performance Histogram	293
Figure 6-7: Time-Based Performance	294
Figure 6-8: Memory Leak Detection	295
Figure 6-9: Drill Down from Operational Dashboard	296
Figure 6-10: Drill Down Details	297
Figure 6-11: Architecture View with Delay Analysis	298
Figure 6-12: Detailed Performance Data on ActionServlet	299
Figure 6-13: Hide Other Arrows	300
Figure 6-14: Fan Out Call to UserEJB	301
Figure 6-15: Bottleneck Detected	303
Figure 6-16: Fan In Call Detail	304
Figure 6-17: Drill Out	305
Figure 6-18: Drill Out UserEJB	305
Figure 6-19: Drill Out to Show in Context	306
Figure 6-20: Detailed Performance Data	307
Figure 6-21: Mouse Over to Obtain More Information	307
Figure 6-22: AdminSessionEJB	308
Figure 6-23: Uncheck Less Relevant Arrows	308
Figure 6-24: Potential Call Path Diagram Without Unnecessary Arrows	309
Figure 8-1: Creating New Custom View	338
Figure 8-2: New Custom View is Added to My Custom Views Tree	338
Figure 8-3: Add a View Element to a Custom View by Drag and Drop	339
Figure 8-4: Drop Options	339
Figure 8-5: New View Element Added to Custom View	340
Figure 8-6: New View Elements Added to Custom View	341
Figure 8-7: Right-Click Menu for Custom Views	341
Figure 8-8: Cell View Manager	342
Figure 9-1: Create Custom Dashboard	344
Figure 9-2: Custom Dashboard Layout Templates	345
Figure 9-3: Custom Dashboard Template 1	346
Figure 9-4: Custom Dashboard Template 2	346
Figure 9-5: Custom Dashboard Template 3	347
Figure 9-6: Custom Dashboard Template 4	347
Figure 9-7: Custom Dashboard Template 5	348
Figure 9-8: Custom Dashboard Select Table Component	348
Figure 9-9: Custom Dashboard Edit Table	348
Figure 9-10: Custom Dashboard Entity Selection	349

Figure 9-11: Custom Dashboard Aggregation Selection	350
Figure 9-12: Custom Dashboard Add Columns	350
Figure 9-13: Navigation View Selection	351
Figure 9-14: Chart Type Selection	352
Figure 9-15: Custom Dashboard Entity Selection	353
Figure 9-16: Chart Metrics Selection	353
Figure 9-17: Chart Metrics Aggregation Selection	354
Figure 9-18: Label Configuration	354
Figure 9-19: Custom Dashboard Edit Table	355
Figure 9-20: Create Tabbed Dashboard	356
Figure 9-21: Tabbed View Editor	356
Figure 9-22: Add Tab	356
Figure 9-23: Share Dashboard	357
Figure 9-24: Preview Dashboard	357
Figure 9-25: Sample Preview Dashboard	358
Figure 9-26: Dashboard Navigation	359
Figure 9-27: Default Dashboard Selection	360
Figure 10-1: Steps of Oracle™ CAMM Methodology	362
Figure 10-2: Control Number of Alerts	364
Figure 10-3: Configure Service Level Objectives	369
Figure 10-4: Cautionary and Violation SLOs	369
Figure 10-5: WebLogic® Cluster Configuration	370
Figure 10-6: CAMM™ Displaying Portal Activities	371
Figure 10-7: Visual Analysis Using CAMM™	371
Figure 10-8: Application Performance on Server 192.168.3.185	372
Figure 10-9: Application Performance on Server 192.168.3.186	373
Figure 10-10: Comparing Load Differences Between Servers in the Cluster ...	373
Figure 10-11: Comparing Response Times Between Servers in a Cluster	374
Figure 10-12: Compare Resource Usages in a Cluster-Server 192.168.3.185 ...	374
Figure 10-13: Compare Resource Usages in a Cluster-Server 192.168.3.186 ...	375
Figure 10-14: Comparing Load and Resource Usage in a Cluster	376
Figure 10-15: Portal Application Not Able to Meet Its Performance SLO	377
Figure 10-16: Drill Down on the Hierarchy to Identify Active Desktop	377
Figure 10-17: Drill Down on Hierarchy to Find the Slowest Page	378
Figure 10-18: Drill Down on Hierarchy to Find the Slowest Portlet	378
Figure 10-19: Visually Identify SLO Violation	379
Figure 10-20: Analysis of Portal Components Revealed Consistent Behavior ..	380
Figure 10-21: Use a Custom View to Correlate Different Metrics	381
Figure 10-22: Metrics from OS Agent Reveals Abnormalities	381
Figure 10-23: JVM Heap Metrics Reveals Abnormal Memory Usage	382
Figure 10-24: Connection Response Time Impacted by Paging Activities	382
Figure 10-25: Mechanics of SLO Trigger Set to Low	384
Figure 10-26: Mechanics of a SLO Trigger Set to High	385

List of Tables

Table 1-1: Conventions	22
Table 3-2: SLO Blackout Configuration	61
Table 3-6: Edge Types Color Codes	88
Table 3-7: List of Metrics in the External Calls	92
Table 3-8: Various types of Architecture View	94
Table 3-9: Architecture View Color Codes	94
Table 3-10: Metric Types	103
Table 4-1: WebLogic® Portal Hierarchy	107
Table 4-2: Tree Summary	107
Table 4-3: Portal Level Tab Descriptions	109
Table 4-4: Desktop Summary Metrics	111
Table 4-5: Book Summary Metrics	118
Table 4-6: Pages Summary Metrics	120
Table 4-7: Portlet Metrics	122
Table 4-8: WebSphere® Portal Hierarchy	125
Table 4-9: WebSphere® Tree Summary	125
Table 4-10: Portal Level Tab Descriptions	127
Table 4-11: Virtual Portals Summary Metrics	129
Table 4-12: Pages Summary Metrics	132
Table 4-13: Portlet Metrics	133
Table 4-14: BPEL Process Summary Metrics	135
Table 4-15: BPEL Functional View Summary	138
Table 4-16: Metadata View Summary	140
Table 4-17: Partner Links View Summary	142
Table 4-18: Partner Link Type Role View Summary	143
Table 4-19: Partner Link Bindings View Summary	144
Table 4-20: Modeled Entities Summary	145
Table 4-21: ESB Summary Metrics	149
Table 4-22: Service Details View Summary	152
Table 4-23: Service Parent Details View Summary	153
Table 4-24: Service Definition View Summary	154
Table 4-25: Service Operations View Summary	155
Table 4-26: Operation Routing Rules View Summary	156
Table 4-27: Process Summary Metrics	158
Table 4-28: Node Hierarchy Summary	161
Table 4-29: Events View Summary	162
Table 4-30: Entity EJB Activity Table	168
Table 4-31: Entity EJB Cache Table	169
Table 4-32: Entity EJB Transactions Table	169
Table 4-33: Entity EJB Locking Table	170
Table 4-34: Class Node	171
Table 4-35: Method Node	171

Table 4-36: WLI Web Services Summary	172
Table 4-37: Public Operations Node Hierarchy Summary	175
Table 4-38: Instrumentation Class Panel Summary	176
Table 4-39: Instrumentation Method Panel Summary	176
Table 4-40: Operations Table	179
Table 4-41: HTTP Performance Summary	184
Table 4-42: EJB Performance Summary	184
Table 4-43: JDBC Performance Summary	185
Table 4-44: WSRP Producers Summary	186
Table 4-45: WSRP Producers Information	187
Table 4-46: WSRP Consumer Portlet Performance	187
Table 4-47: WSRP Producer Portlets	188
Table 4-48: Integration Summary	193
Table 4-49: Execute Queues Summary	195
Table 4-50: Guidelines to Adjust the Execute Queue Thread Count	196
Table 4-51: Async Dispatcher Summary	197
Table 4-52: Sync Dispatcher Summary	199
Table 4-53: JMS Destination Message Statistics	201
Table 4-54: JMS Destination Byte Statistics	201
Table 4-55: Stateless Container Summary	202
Table 4-56: Entity EJB Activity	205
Table 4-57: Entity EJB Cache	205
Table 4-58: Entity EJB Transactions	206
Table 4-59: Entity EJB Locking	206
Table 4-60: Performance - Process Node Summary	208
Table 4-61: Performance - Events Node Summary	208
Table 4-62: Channels Summary	210
Table 4-63: Applications Summary	213
Table 4-64: Applications Summary Tabs	214
Table 4-65: Dependencies Column Descriptions	216
Table 4-66: Dependency Types	218
Table 4-67: Deployment Tabs	219
Table 4-68: Stateless Beans Detail View	227
Table 4-69: Stateful EJB Cache	229
Table 4-70: Stateful EJB Transactions	230
Table 4-71: Stateful EJB Locking	231
Table 4-72: Entity EJB Activity	235
Table 4-73: Entity EJB Cache	236
Table 4-74: Entity EJB Transactions	237
Table 4-75: Entity EJB Locking	238
Table 4-76: Message Driven EJB Activity	240
Table 4-77: Message Driven EJB Transactions	241
Table 4-78: WebLogic® Resources Tree	242
Table 4-79: WebSphere® Resources Tree	245
Table 4-80: WebSphere® Resources Tree	248
Table 4-81: Custom Class Performance	251

Table 4-82: CAMM™ JAgent Status	251
Table 5-1: Resource Creation Form	274
Table 5-2: BEA WebLogic® Based Resource	275
Table 5-3: List of SLO Variables	286
Table 6-1: Drill Down Methods	296
Table 7-1: Commands and their Functions	312
Table 7-2: Export File Name: metricBEA_ChannelInstance.csv	323
Table 7-3: Export File Name: metricBEA_ProcessType.csv	324
Table 7-4: Export File Name: metricBEA_TimerEventGenerator.csv	325
Table 7-5: Export File Name: metricJ2EE_Dispatcher.csv	326
Table 7-6: Export File Name: metricJ2EE_EJB_Entity.csv	327
Table 7-7: Export File Name: metricJ2EE_EJB_Stateless.csv	328
Table 7-8: Export File Name: metricJ2EE_JDBC_ConnectionPool.csv	330
Table 7-9: Export File Name: metricJ2EE_JMS_Destination.csv	331
Table 7-10: Export File Name: metricJ2EE_JMS_Service.csv	332
Table 7-11: Export File Name: metricJ2EE_JVM.csv	333
Table 7-12: Export File Name: metricJ2EE_Server.csv	334
Table 7-13: Export File Name: metricJ2EE_Servlet.csv	335
Table 10-1: Example - Guidelines for Business SLAs	368
Table 10-2: List of Key System Metrics for WebLogic®	383
Table 10-3: SLOs for ExecuteQueue Idle Threads	384
Table 10-4: Actions for SLO	384
Table 10-5: ExecuteQueue Pending Requests	385
Table 10-6: Total JVM Heap Size	386
Table 10-7: Free JVM Heap Size	386
Table 10-8: Open Session Count	386
Table 10-9: Application Invocation Count	387

1

About This Guide

This chapter includes the following topics:

- [Overview](#)
- [Audience](#)
- [Getting Help](#)
- [Using This Guide](#)
- [Typographical Conventions](#)
- [Feedback](#)

Overview

This guide provides detailed information and procedures for using Oracle Enterprise Manager Grid Control - Composite Application Monitor and Modeler (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 CAMM™ *User's Guide* is primarily used by those responsible for setting up and maintaining performance monitoring system for complex applications running on the Oracle Application Server®, BEA 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 can use this manual.

Getting Help

Paid customers are entitled to Oracle™ Standard Technical Support. To reach Oracle™ Technical Support, send an e-mail to support@Oracle.com.

Using This Guide

This guide includes the following chapters:

- [About This Guide](#)
- [Introduction](#)
- [Exploring the User Interface](#)
- [Exploring the Monitor Workspace](#)
- [Exploring Configuration Workspace](#)
- [Performance Analytics](#)
- [Exporting Data](#)
- [Creating Custom Views](#)
- [Custom Dashboards](#)
- [Oracle CAMM™ Methodology](#)

Typographical Conventions

The following table describes the conventions used in this guide.

Table 1-1: Conventions

Conventions	Descriptions
Windows, dialog box names, and menu options	Names of windows and dialog boxes are first letter (of each word) capitalized, with no other special formatting. For example: <ul style="list-style-type: none">• Type in your name in the Logon Window.• The Process Flow Viewer has two panels.• Right-click to select Service Level Objectives Events.

Table 1-1: Conventions

Conventions	Descriptions
User input and directory/file names	<p>User input and directory/file names are in <code>courier</code> font type. For example:</p> <ul style="list-style-type: none"> • Issue the <code>ls</code> command with <code>-al</code> arguments to see the full list. • The <code>acsera_preferences</code> directory starts with a period.
Buttons	<p>A button or a similar element when clicked on a page is in bold. For example:</p> <p>Click Create New SLO.</p>
System outputs and important points	<p>System outputs, such as confirmation messages or alerts, are first letter capitalized and <i>bold italics</i>. For example:</p> <p>The system displays the following message: <i>Do you want to save your changes before moving on?</i></p> <p>Important points are also in <i>bold italics</i>. For example the name of the portlet in this example:</p> <p>CAMM™ clearly identifies the <i>ThreadedDiscussion</i> portlet as the worst performing.</p>
Referenced topics	<p>Referenced topic headings are in blue and underlined. For example:</p> <p>See Typographical Conventions for more information.</p>
Referenced books/documents	<p>Referenced book or document titles are in <i>italics</i>. For example:</p> <p>See <i>CAMM™ Configuration Guide</i> for more information.</p>
Notes and Tips	<p>Notes and Tips are contained within two horizontal lines. For example:</p> <hr/> <p>Note: This is a note.</p> <hr/> <hr/> <p>Tip: This is a tip</p> <hr/>
Blank pages	<p>There will be a blank page with header and footer at the end of the chapters ending on odd page numbers. This format is designed to accommodate printing multiple chapters on duplex paper and keeping the collation even.</p>

Feedback

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

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

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

- E-mail: support@Oracle.com

At Oracle™, we are dedicated to improving the user documentation. We are not able to reply to your comments individually, but we welcome your feedback in order to better improve our services.

2

Introduction

This chapter includes the following topics:

- [Overview](#)
- [Architecture](#)

Overview

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.

While a few enterprises deal with this issue by simply ignoring the need to manage application performance, most choose to implement custom manual processes so that they can effectively use conventional application performance management (APM) toolkits. Unfortunately, as some of these enterprises discovered, to be truly effective with these conventional APM toolkits, the repetitive Do-It-Yourself (DIY) manual processes must be followed religiously. Doing so not only incurs additional resource overhead, but also adds significant time lag for getting applications into QA and production.

In fact, these conventional APM toolkits and their repetitive DIY manual processes act like pairs of rigid handcuffs handicapping these organizations' ability to efficiently deliver applications and meet the requirements of today's highly competitive business climate.

To be effective and efficient, enterprises need to manage these highly distributed J2EE™ and SOA applications in a holistic fashion while using the least possible amount of IT resources and specialized expertise. They need an intelligent Application Service Management (ASM) platform that can deliver the following:

- Provides a holistic, service-oriented view across heterogeneous environments
- Requires minimal involvement from J2EE™, SOA, and application experts
- Eliminates repetitive Do-It-Yourself (DIY) manual processes

Semantic modeling is the only way to inject intelligence into APM platforms transforming it into an automated and much broader ASM solution. This model-driven approach enables the ASM platform to understand the structure of these J2EE™ and SOA applications and the configuration of their distributed runtime environments. Using the insights captured during semantic modeling, these new ASM platforms can intelligently and automatically insert performance measurements, apply service context, track calling relationships, monitor application performance, send real-time alerts, and evolve with application and infrastructure changes.

Oracle™ provides the industry's first and only model-driven intelligent ASM platform for J2EE™ and SOA. Unlike conventional APM toolkits, Oracle Enterprise Manager Grid Control - Composite Application Monitor and Modeler (CAMM™) analyzes J2EE™ and SOA applications to capture the complex relationships among various application building blocks in its AppSchema™ model - the core of Oracle's 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.

Are you ready to unhinge your conventional APM handcuffs?

Terminology

Table 2-1: Terminology

Term	Definition
ASM	Application Service Management
APM	Application Performance Management
DIY	Do-It-Yourself
ISV	Independent Software Vendor.

Table 2-1: Terminology (Continued)

Term	Definition
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.

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 yesteryears, the complexity of today's J2EE™ and SOA applications has grown exponentially for the following three 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 three 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, etc. 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.

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.

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 will enable 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**

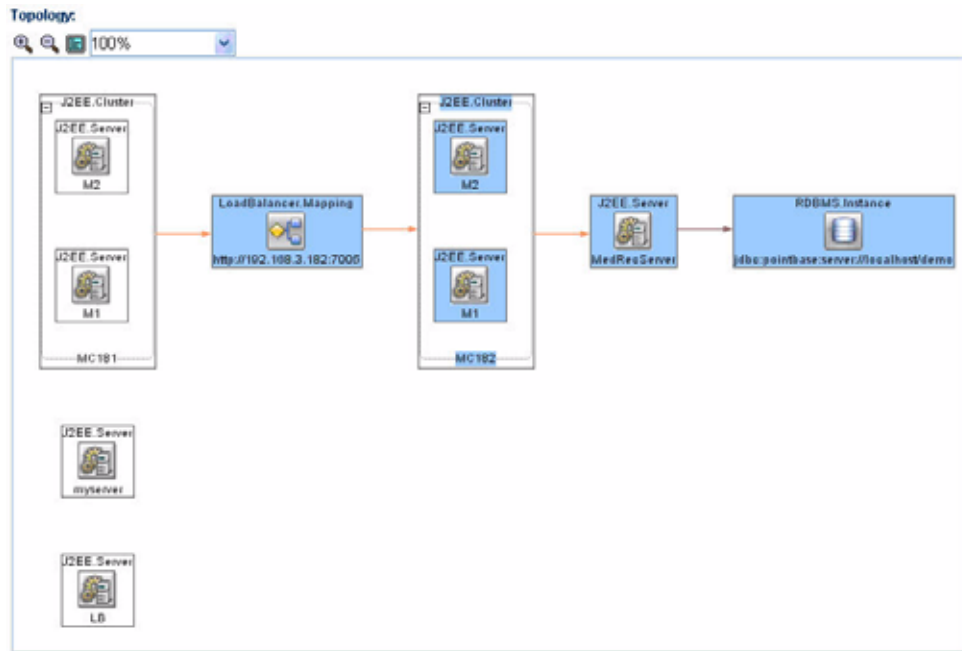
Commonly used by conventional APM toolkits for J2EE™, this resource-centric approach collects low-level technology-oriented measurements from components running in a single JVM. While these toolkits offer ways for users to arbitrarily group measurements from multiple JVMs into logical units, these groupings 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.

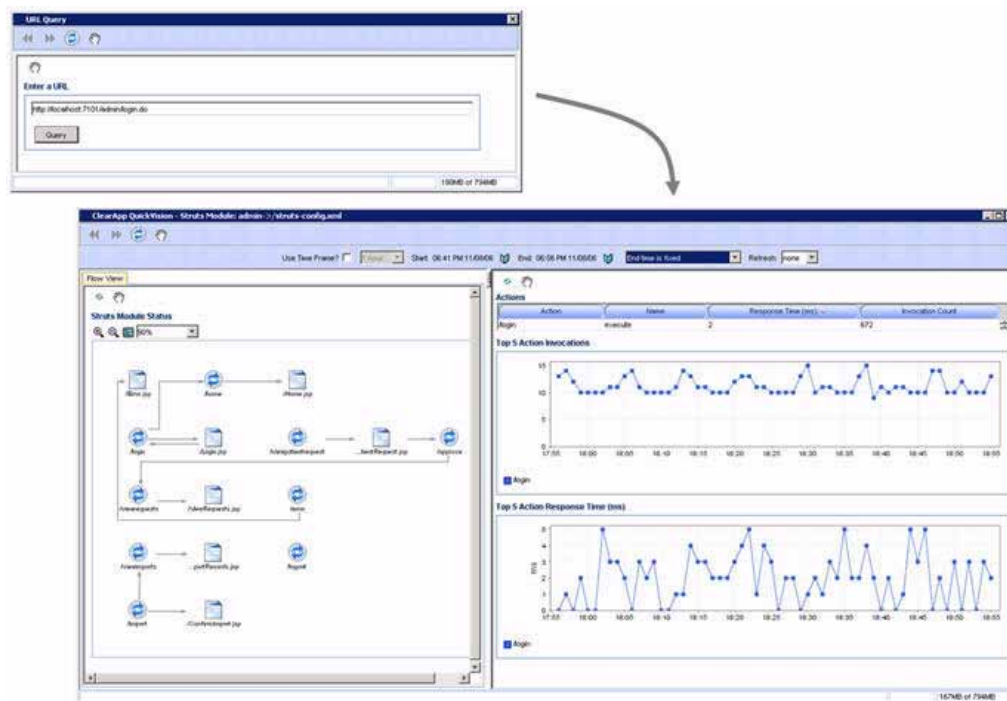
Figure 2-1: Topology View in CAMM™



AppSchema Navigation provides efficient ways for you to access relevant information via 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 2-2: AppSchema™ View in CAMM™



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.

Eliminating repetitive Do-It-Yourself (DIY) manual processes

For years, developers and architects rely 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 matter 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 AppSchema 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).

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.

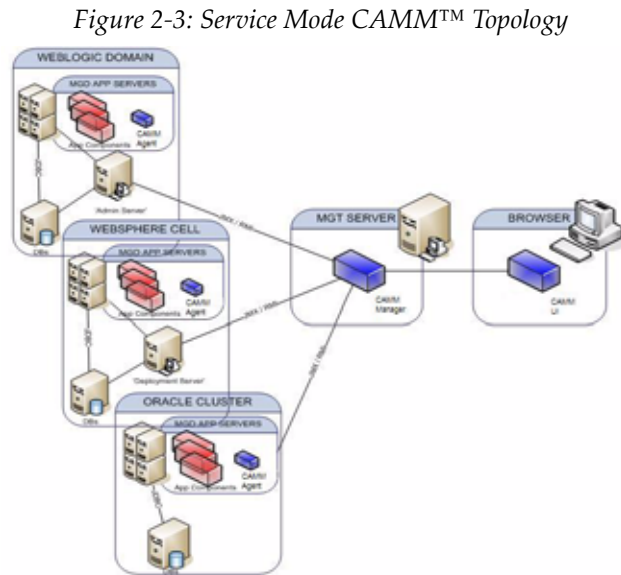
Architecture

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 *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.

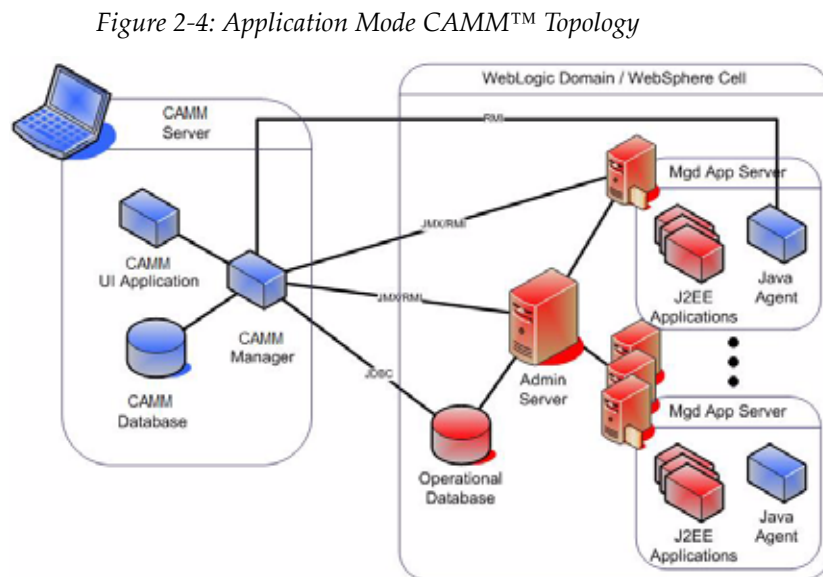
To allow remote access to CAMM™ through a browser, a web container is installed. This web container provisions the CAMM™ UI applets to the browser and maintains communication with these applets.

[Figure 2-3](#) shows CAMM™ deployed in Service Mode.



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 2-4](#) shows CAMM™ deployed in Application Mode.



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

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®, BEA WebLogic® and IBM WebSphere®, CAMM™ leverages their deployment infrastructures to quickly deploy the CAMM™ Java Agents to all application servers.

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.

CAMM™ Database

CAMM™ stores its analyzed data and application models in an CAMM™ Database - an operational data repository. CAMM™ Database is an integral part of CAMM™'s turn-key solution and is installed during the installation process.

Creating CAMM™ Database on an external database system is possible as long as the external database fulfills the performance requirements of CAMM™.

CAMM™ User Interface

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.

3

Exploring the User Interface

This chapter includes the following topics:

- [Starting CAMM™ UI](#)
- [General CAMM™ UI Elements](#)
- [Drill Down in Operational Dashboard](#)
- [Configuring Service Level Objectives \(SLOs\)](#)
- [Configure SLO Blackouts](#)
- [Time Frame](#)
- [Display Interval](#)
- [Refresh Rate](#)
- [Queries](#)
- [Graphs and Data Items](#)
- [Right-Click Operations on Tables and Graphs](#)

- [Comparative View](#)
- [Save as PDF](#)
- [Easy Scroller](#)
- [Zoom In and Zoom Out Toolbar](#)
- [Custom Metrics](#)
- [Promote To Dashboard](#)
- [Functional View](#)
- [Topology View](#)
- [Architecture View](#)
- [Metric Types](#)

Starting CAMM™ UI

Oracle Enterprise Manager - Composite Application Monitor and Modeler (CAMM™) can operate under two modes: application and service modes. Under application mode, CAMM™ UI can be started by executing the [CAMM™ Installation]\bin\standalone.bat.

Under the service mode, CAMM™ UI can be accessed either through a web browser or through command line.

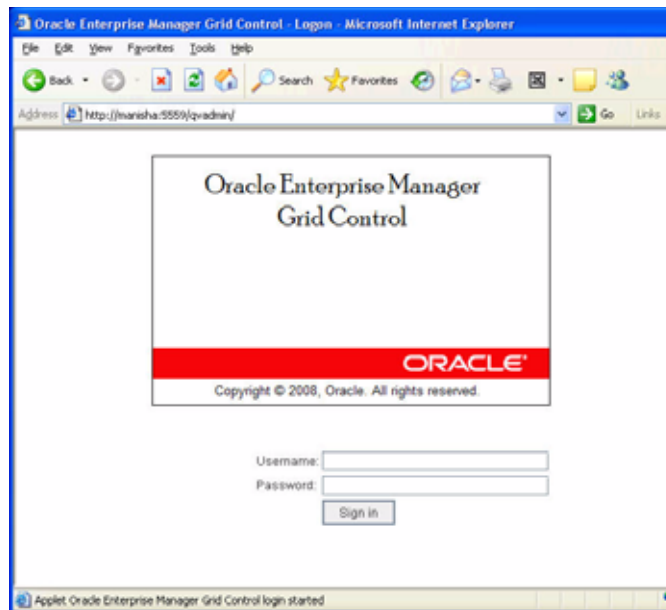
Web Browser Access

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

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

In [Figure 3-1](#) example, the CAMM™ Machine Name is *localhost* and the port number is 5557. It should be noted that 5557 is the default port number for CAMM™'s web container. [Figure 3-1](#) is the login page for CAMM™ UI in a web browser.

Figure 3-1: Accessing CAMM™ UI through a web browser



Command Line Access

To access CAMM™ UI using command line, execute the [CAMM™ Installation]\bin\client.bat file. [Figure 3-2](#) is the first screen you will see after running the client.bat file.

Figure 3-2: Accessing CAMM™ UI Using Client.bat






In both cases, the CAMM™ UI appears after successful user authentication.


General CAMM™ UI Elements

CAMM™ UI consists of the following core components:

- General Toolbar

The General Toolbar provides some 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.

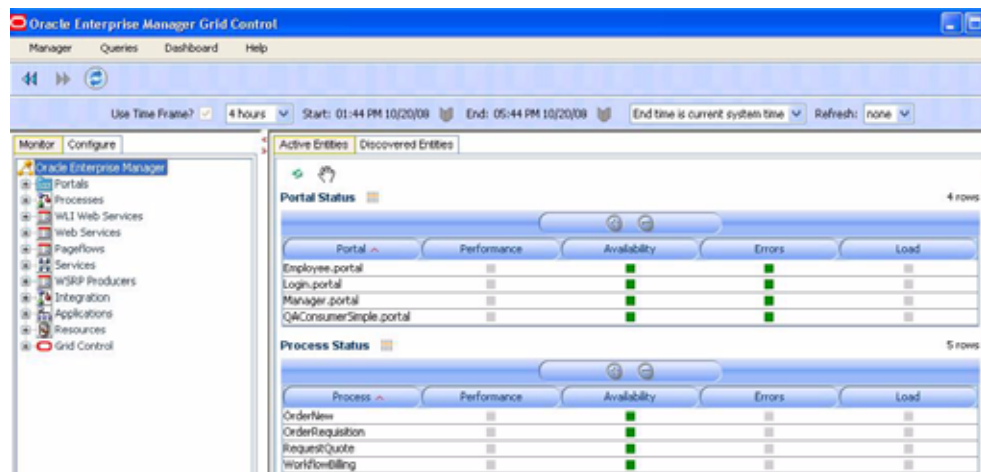
- Double-Click Indicator

The  icon indicates that the double-click operation is available for that visual element. Typically, double-clicking will display next level details.

- Tabs

CAMM™ UI is organized into two workspaces - **Monitor** Workspace and **Configure** Workspace. These workspaces are organized in a tabular fashion. You can access the appropriate workspace by clicking on the corresponding tab.


Figure 3-3: General Toolbar and Tabs



- Navigation Pane

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. Use the Monitor workspace to traverse CAMM™'s tree model and identify abnormal activities. Use the Configure workspace to create, modify, and review various configuration settings for CAMM™.

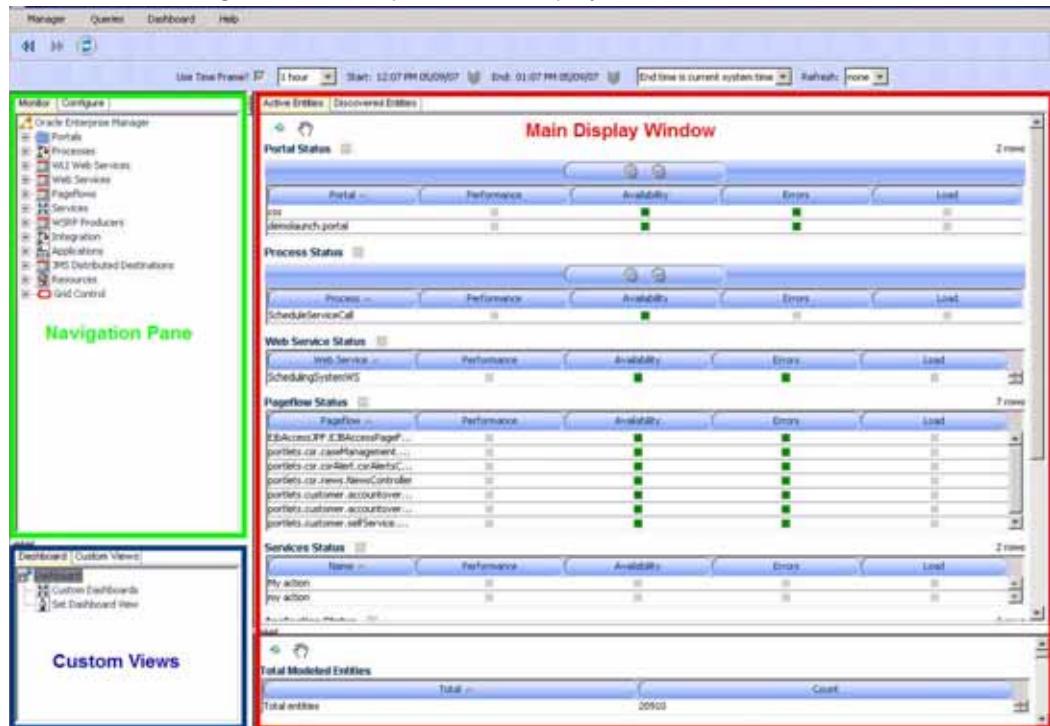
- Main Display Window

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

- Custom Views

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.

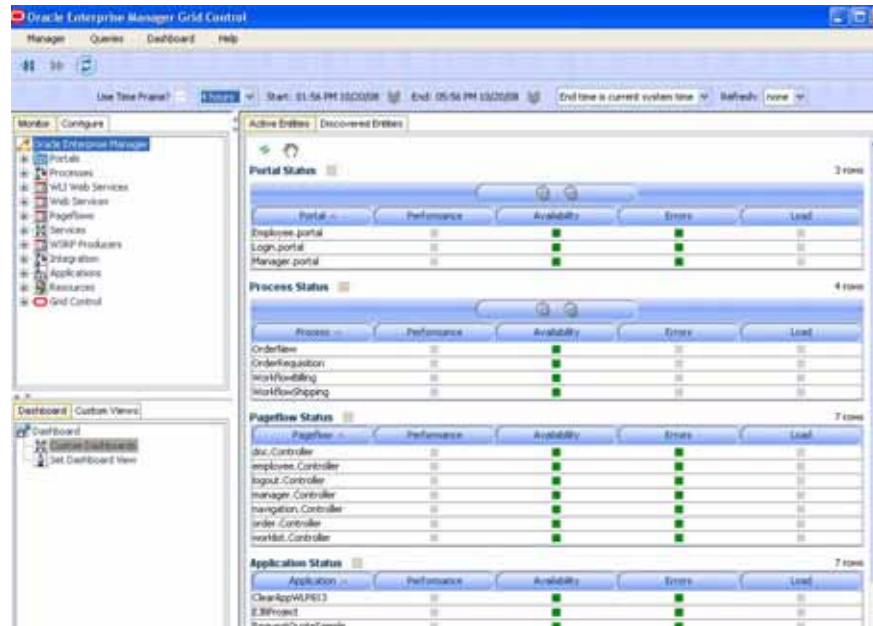
Figure 3-4: Workspace, Main Display Window, Custom Views



Navigating CAMM™

When CAMM™ starts up, the Operational Dashboard is displayed in the Main Display Window.

Figure 3-5: CAMM™ Operational Dashboard



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

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

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 four different types of health indicators to provide a comprehensive view. These health indicators are:

- Performance

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

- **Availability**
The availability health indicator tells us to what extent a particular entity is available to service requests.
- **Errors**
The errors health indicator tells us if the number of errors and exceptions encountered by this entity are approaching or violating the configured threshold.
- **Load**
The load health indicator tells us how many operations have been performed and requests have been served by a particular entity.

CAMM™ is cluster aware. As such, these indicators display overall health of a particular entity across the entire cluster. You can drill down to more details by clicking on the + icon.

Figure 3-6: Health Indicator Drill Down



Figure 3-7: Health Indicator Drill Down Details



As you can see, 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 figure out 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:

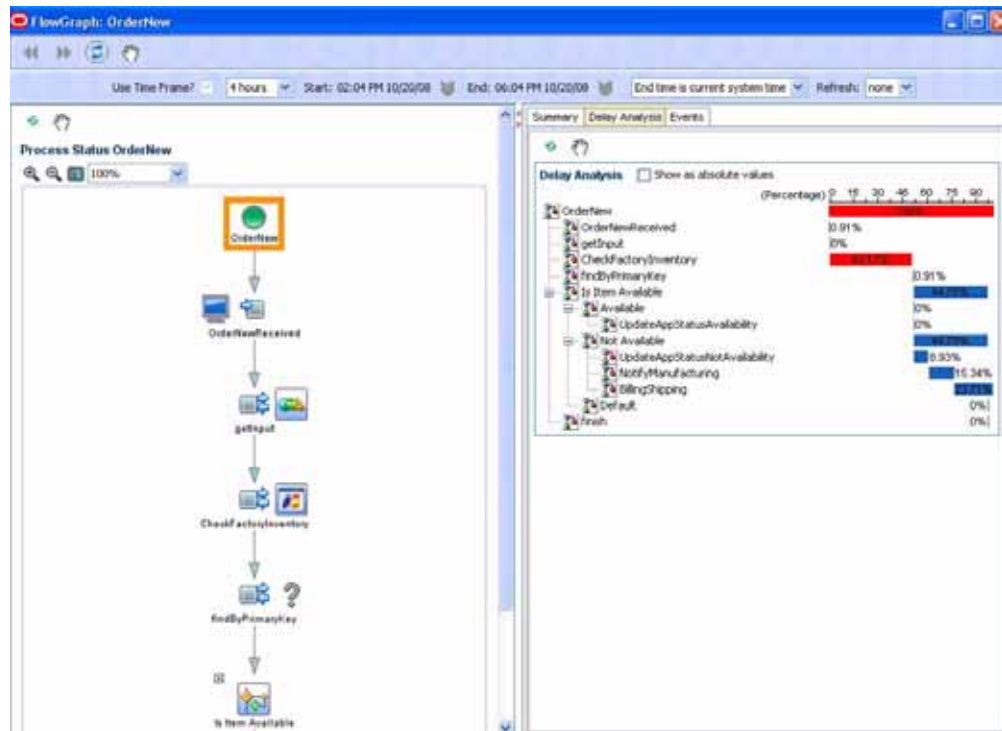
- WebLogic Portal® - Portal Applications

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

Further drill down is possible. By double-clicking on the name of the entity, a new viewer relevant to that entity appears. For example, the entities illustrated in [Figure 3-7](#) are business processes running on the BEA WebLogic Integration® platform. By double-clicking on one of these process names, CAMM™ opens up the Process Flow Viewer.

The Process Flow Viewer and other viewers provide even more application specific information to enable additional analysis. See [Figure 3-8](#).

Figure 3-8: Process Flow Viewer



[Figure 3-8](#) shows the Process Flow Viewer and its two panels. The panel on the right is the Main Display Window. The Main Display Window shows information corresponding to item selected in the navigation pane. The left panel shows the flow of the selected process. In the case of [Figure 3-8](#), this is the same view a developer would see in BEA WebLogic Workshop®. This unique feature gives you the ability to communicate with others and solve performance problems faster.

Notice some process nodes are groups of nodes. By clicking on + and - 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. When you select and double-click on a row in the Operational Dashboard, a new pop-up window appears and provides relevant lower level statistics. The following screen shots depict this type of drill down. You can select an entry in the Operational Dashboard and bring up detailed performance metrics by double-clicking on the operational dashboard.

Figure 3-9: Operational Dashboard Drill Down

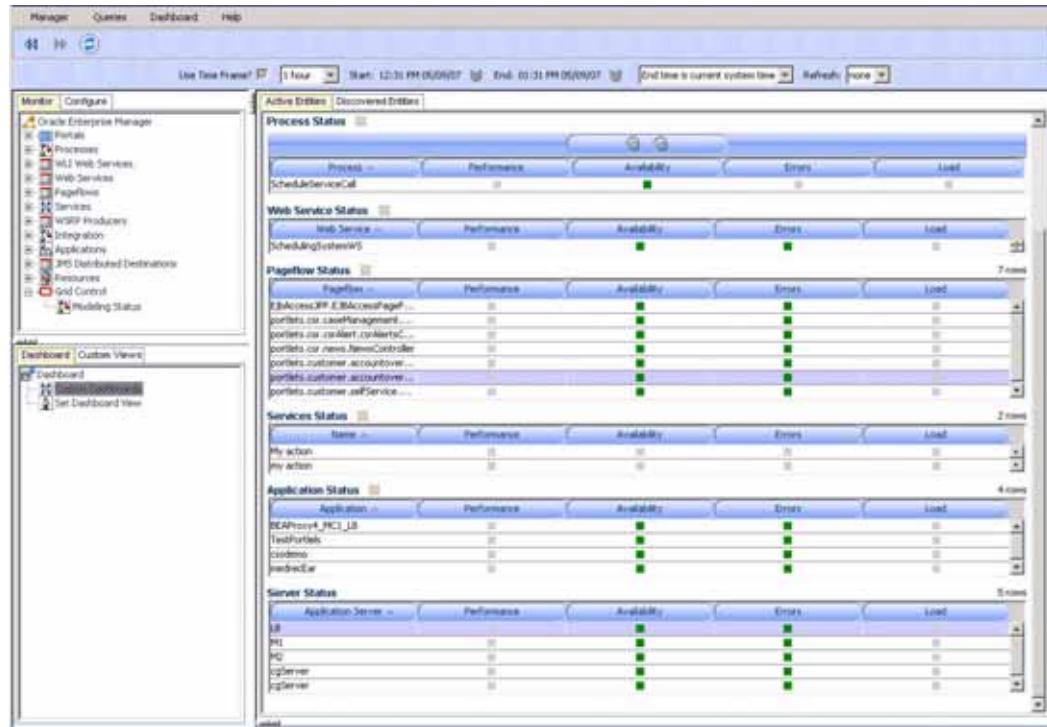
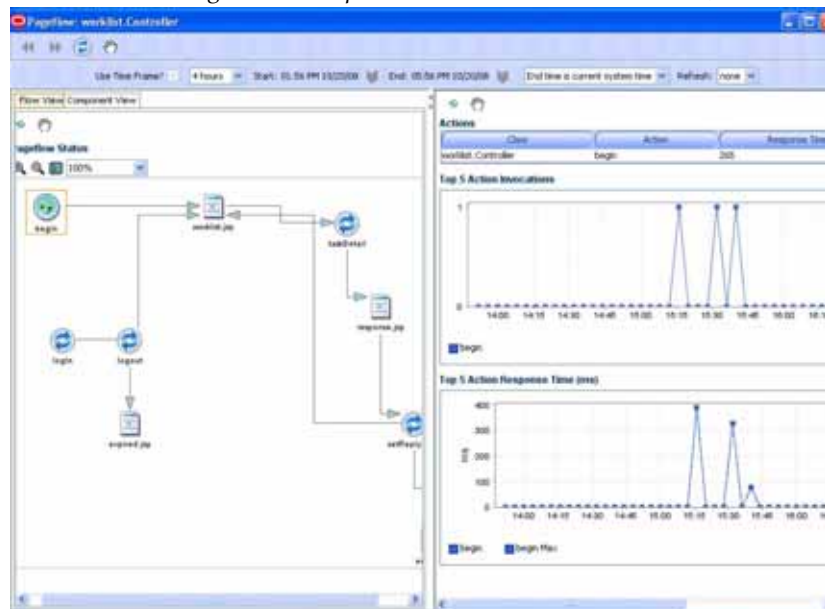


Figure 3-10: Operational Dashboard Drill Down, Continued

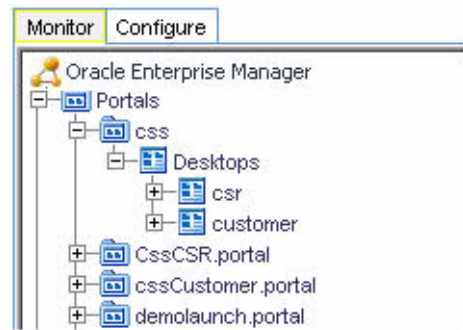


Drill Down in Monitor Workspace

In 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 on the + icon.

Figure 3-11: Drill Down in WebLogic® Monitor Workspace



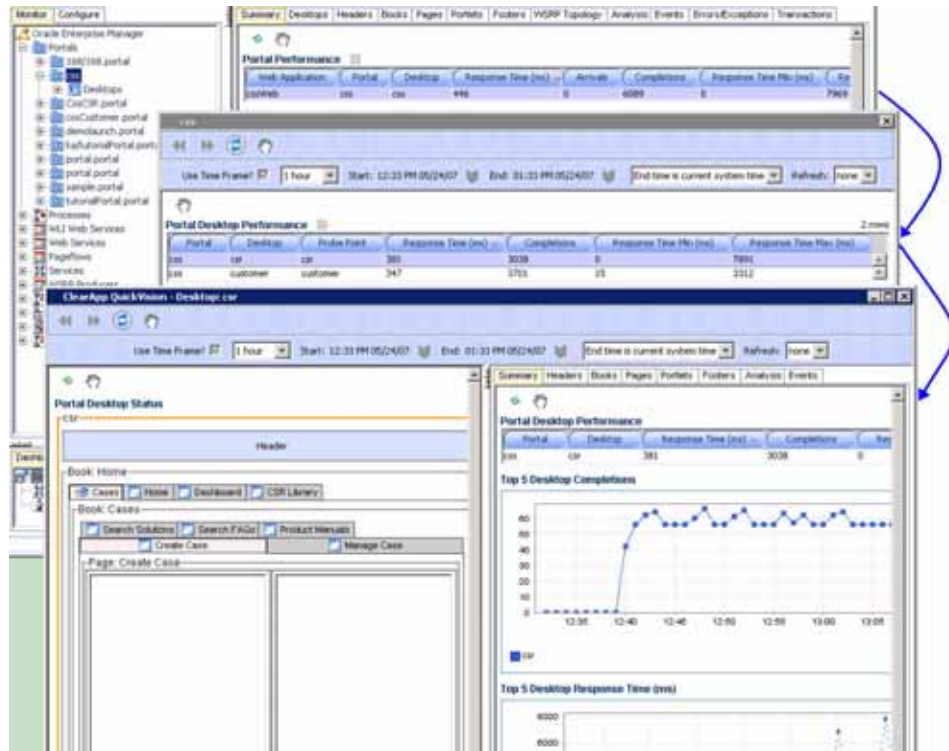
When a lower level tree node gets selected, the Main Display Window displays performance data and render graphs relevant to the selected item. For instance, when you select the *csr* desktop node in the *css* portal tree (see [Figure 3-11](#)), the Main Display Window shows the Portal desktop performance table, Desktop hits and Desktop response time graphs (see [Figure 3-12](#)).

Figure 3-12: Monitor Workspace Drill Down Details



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. See [Figure 3-13](#).

Figure 3-13: Portlet Drill Down

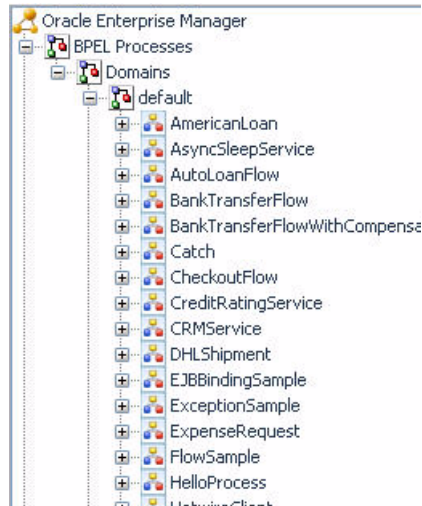


In Oracle SOA Suite®

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

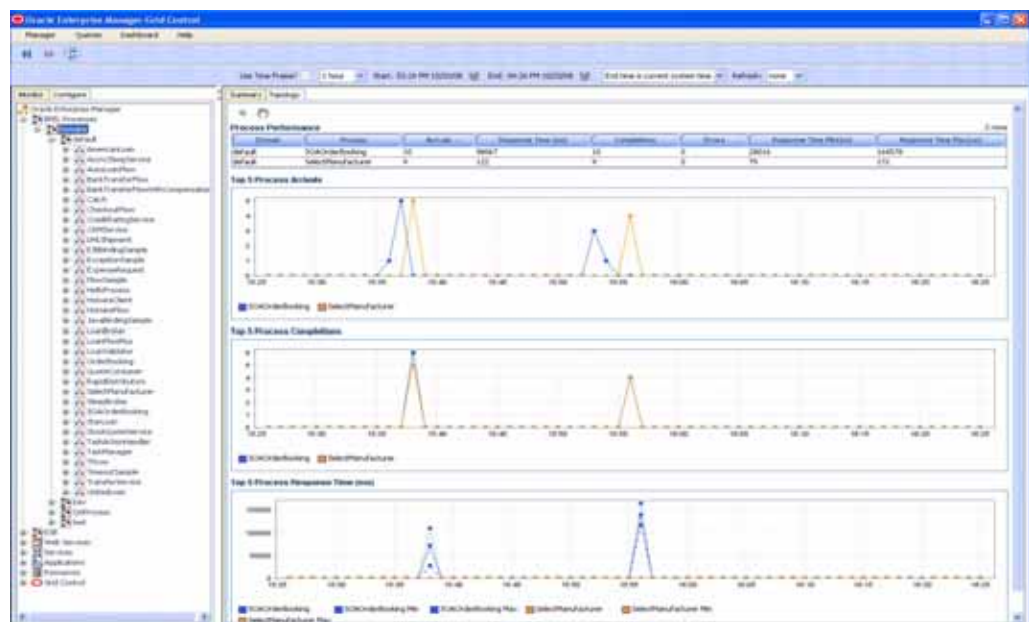
Like BEA, the Monitor Workspace in the left-hand pane is organized hierarchically using a tree view. Starting from the root of the tree, you can quickly drill down to the information you need. You can expand the tree to see more details by clicking on the + icon.

Figure 3-14: Drill Down in Oracle SOA Suite® Monitor Workspace



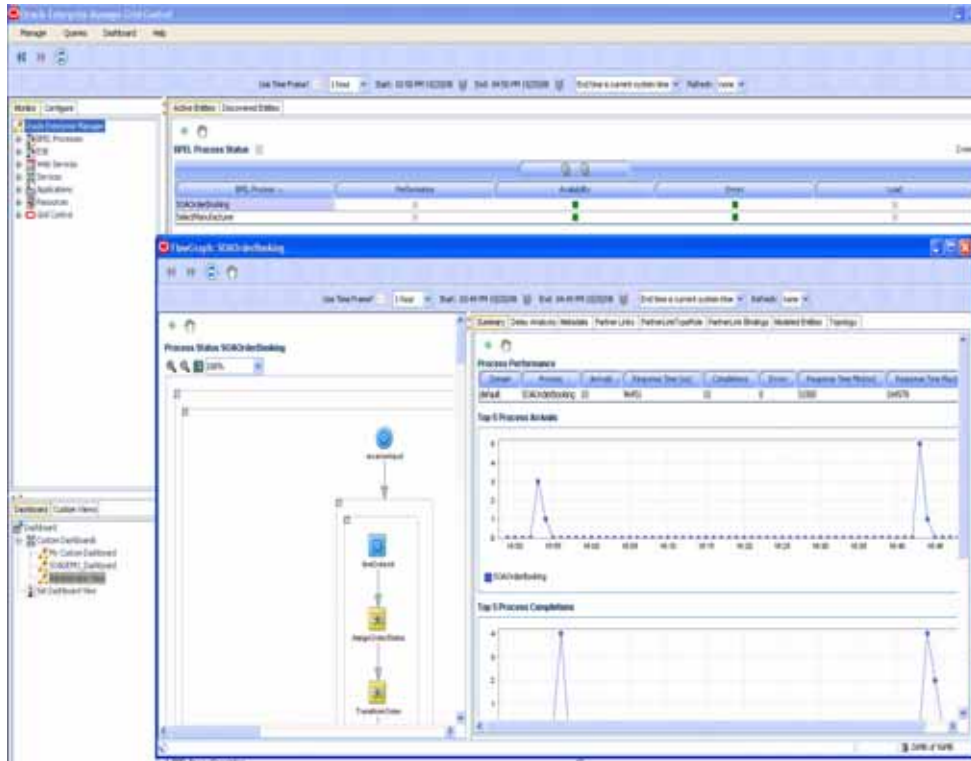
When a lower level tree node gets selected, the Main Display Window displays performance data and render graphs relevant to the selected item. For instance, when you select the *Domains* desktop node in the *BPEL Processes* tree (see [Figure 3-14](#)), the Main Display Window shows the BPEL process performance table, BPEL hits and BPEL response time graphs (see [Figure 3-17](#)).

Figure 3-15: Oracle SOA Suite® Monitor Workspace Drill Down Details



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. See [Figure 3-16](#).

Figure 3-16: BPEL Drill Down

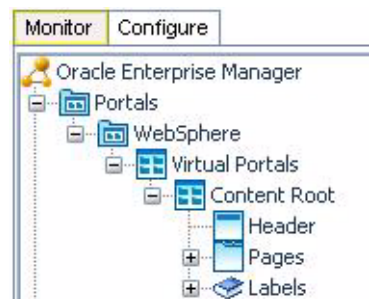


In WebSphere®

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

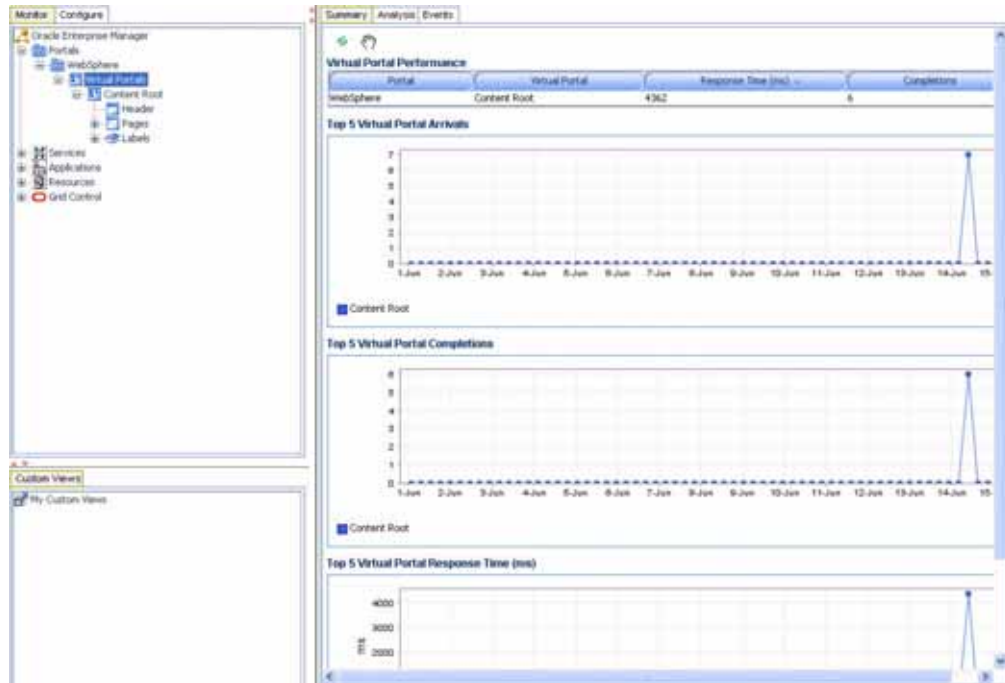
The Monitor Workspace is organized hierarchically using a tree view. Starting from the root of the tree, you can quickly drill down to the information you need. You can expand the tree to see more details by clicking on the + icon.

Figure 3-17: Drill Down in WebSphere® Monitor Workspace



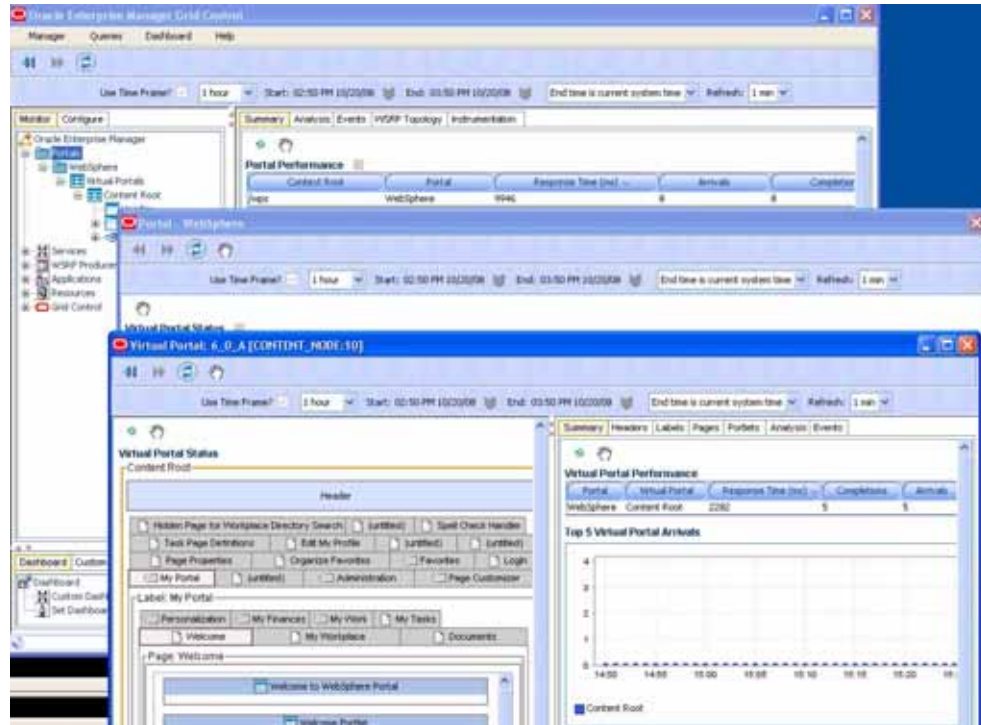
When a lower level tree node gets selected, the Main Display Window displays performance data and render graphs relevant to the selected item. For instance, when you select the *Virtual Portals* desktop node in the *WebSphere* portal tree (see [Figure 3-17](#)), the Main Display Window shows the Portal desktop performance table, Desktop hits and Desktop response time graphs (see [Figure 3-18](#)).

Figure 3-18: Monitor Workspace Drill Down Details



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. See [Figure 3-19](#).

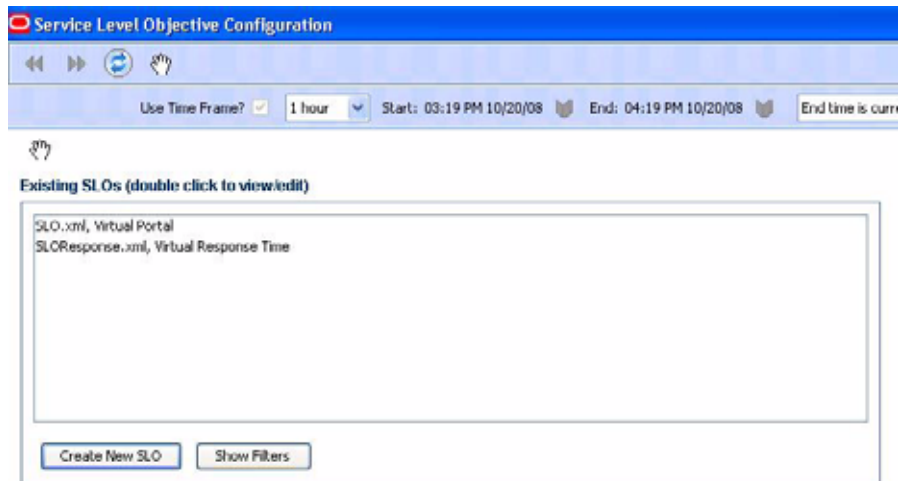
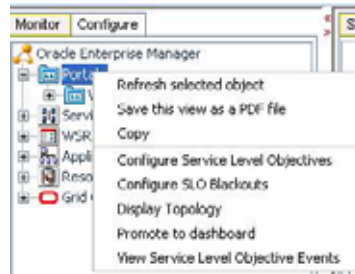
Figure 3-19: Portlet Drill Down



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 extremely easy to configure SLOs in CAMM™. Simply right-click on any element, you will be able to configure or view SLOs. This rule applies to CAMM™ UI and all other viewers.

Figure 3-20: Right-Click to Configure SLOs



Create New SLO

When you select Configure Service Level Objectives, CAMM™ opens up the Service Level Objective Configuration window. This 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. Here are the screen shot examples of the SLO creation process:

Figure 3-21: Select a Specific SLO File to Store New SLO Configuration

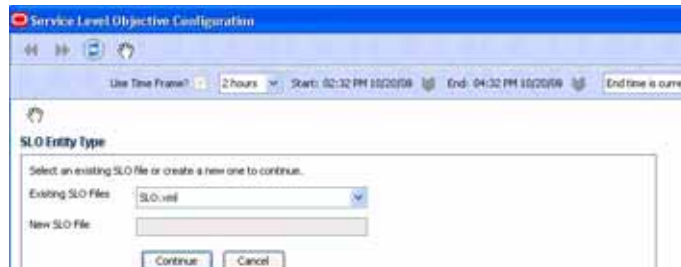
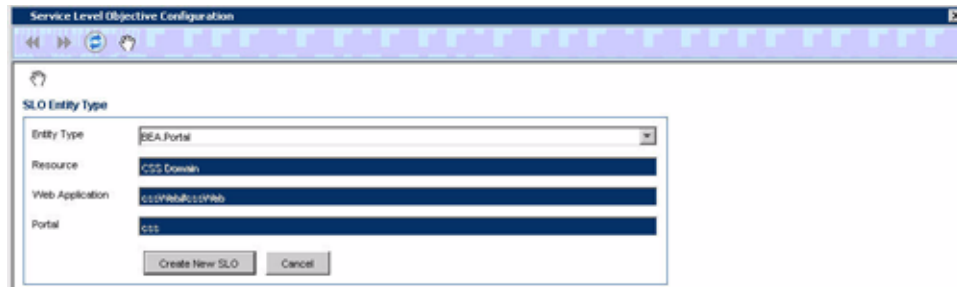


Figure 3-22: Create New SLO: Define SLO Entity Type

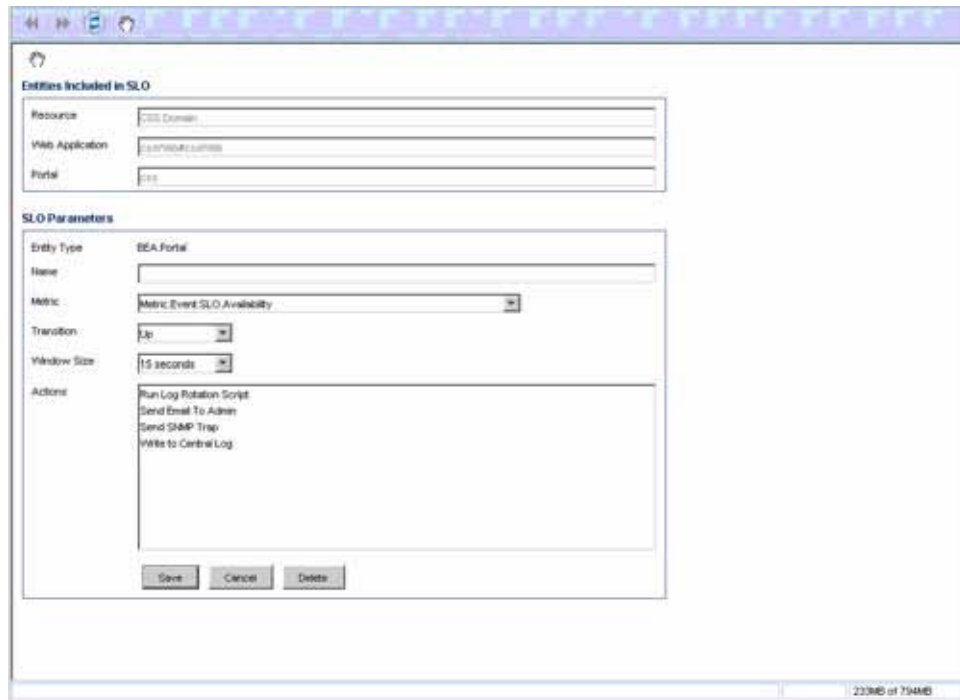


The steps for SLO creation are as follow:

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 the **Create New SLO** button to go to the second stop of the SLO creation process, Defining the SLO Parameters.

Figure 3-23: Create New SLO: Define SLO Parameters



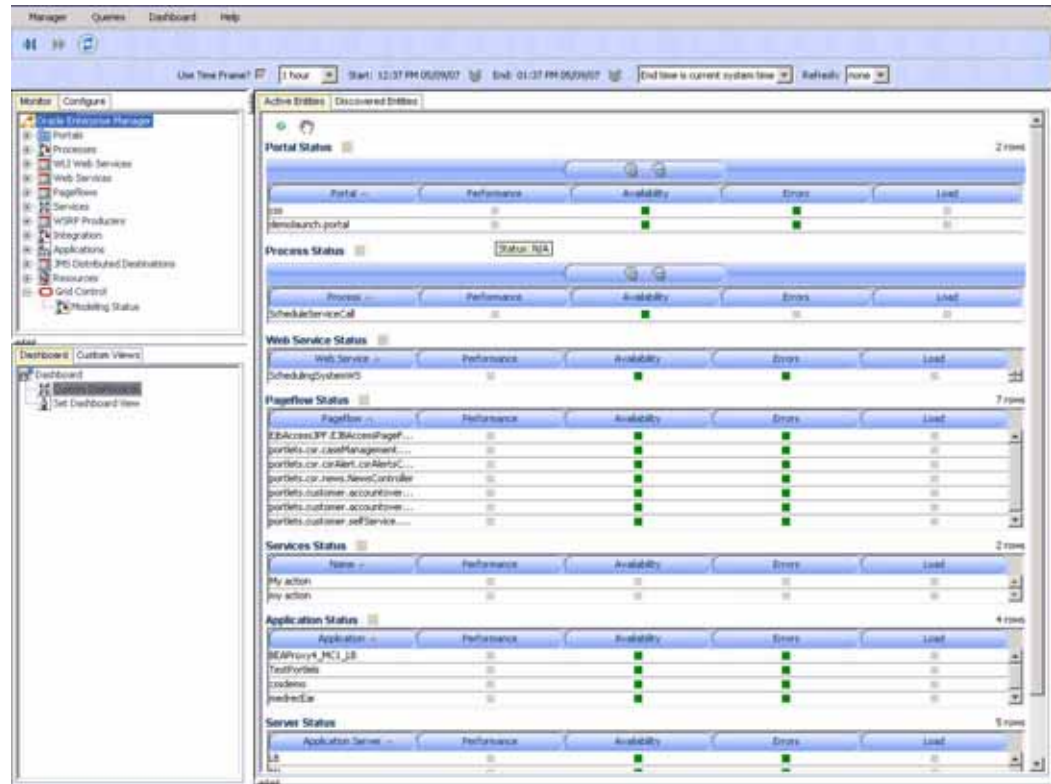
Defining SLO Parameters

Follow these steps to define the SLO parameters:

1. Type in the name.
2. Select the performance metric.
3. Define the monitor window size.
4. Set threshold values for the SLO.
5. Select what actions to take when a trigger is fired. A list of pre-configured 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 may also delete unwanted SLOs for any element from this window.

It's also important to know 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. [Figure 3-24](#) is an example of the operational dashboard when no SLOs are defined.

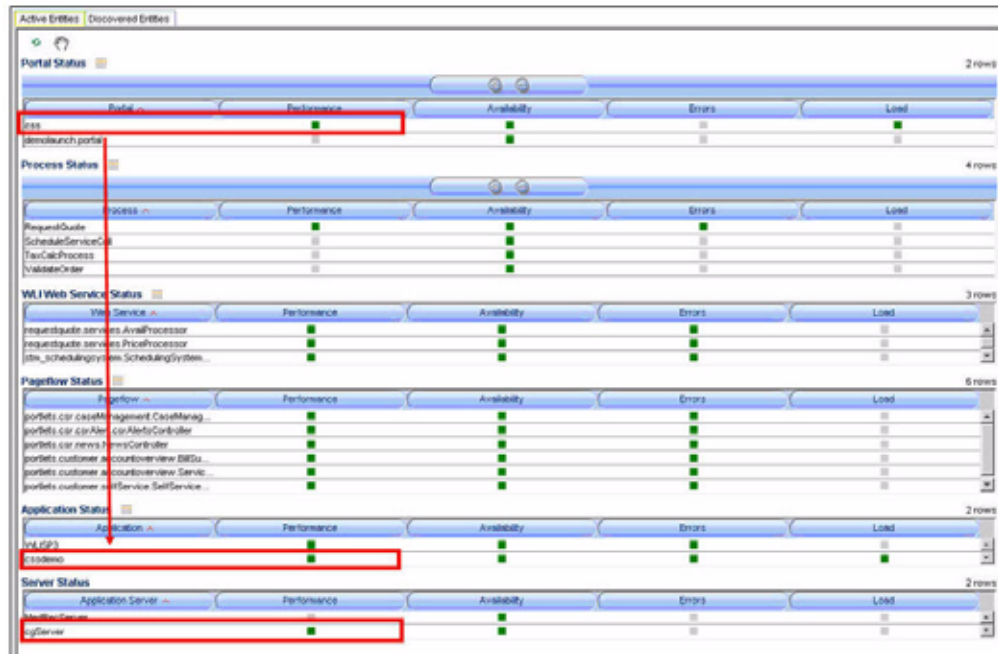
Figure 3-24: Initial Operational Dashboard with Gray Status Lights



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 change status of all containers for this event.

In our first example, 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. [Figure 3-25](#) illustrates the containment approach.

Figure 3-25: Setting SLO on Portlet Activates Health Indicators



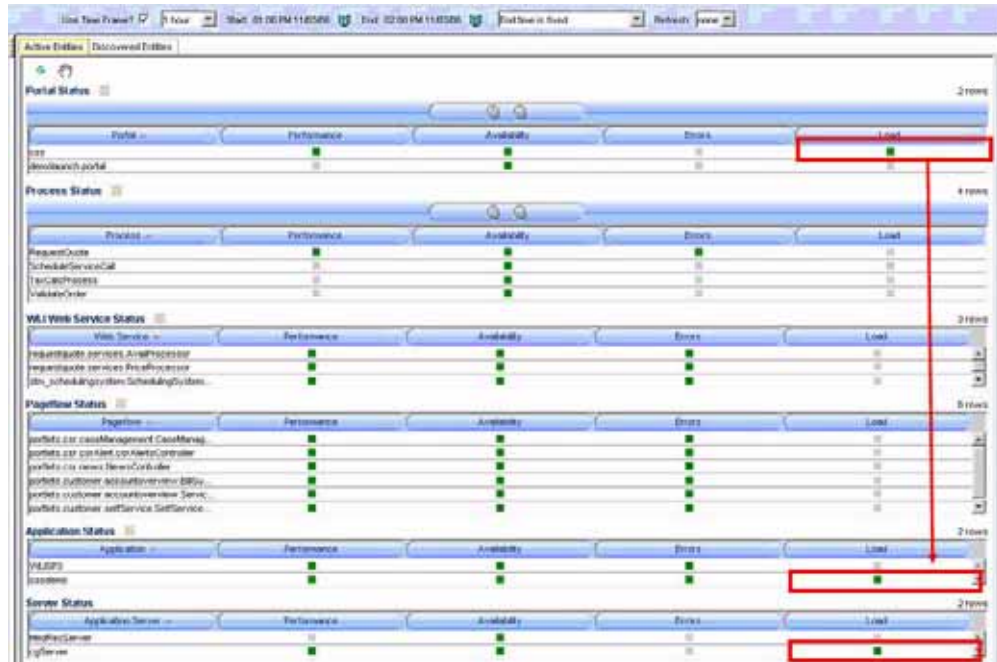
In addition to the containment concept, CAMM™ categorizes SLOs into four buckets:

- performance
- availability
- error
- load

In the [Figure 3-25](#) example, the average response time metric is correctly categorized into the performance bucket.

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 indicators for css portal, cssdemo application, and cgServer instance. [Figure 3-26](#) is the screen shot of this example.

Figure 3-26: SLOs in Four Buckets



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

Figure 3-27: Right-click on Tree Elements to View SLO Events

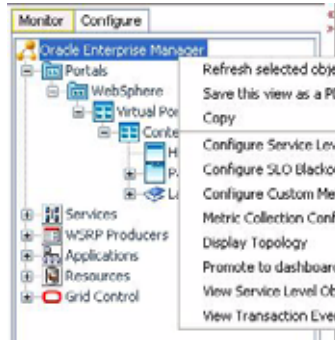


Figure 3-28: SLO Events Viewer

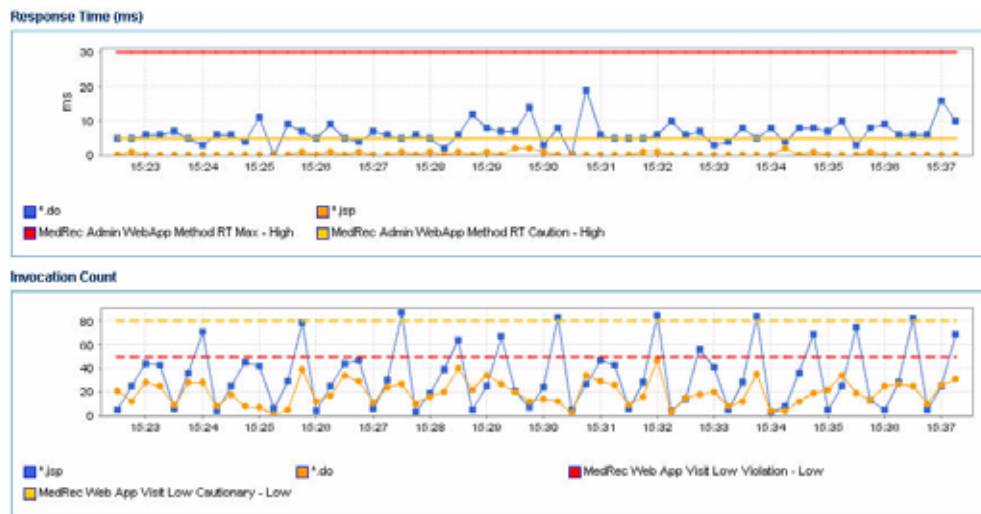


Once new SLOs are added, CAMM™ updates the relevant graphs to visually display these new thresholds. The following screen shots depict this effect.

Table 3-1: 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.

Figure 3-29: Various Line Types Represent Different SLOs Graphically

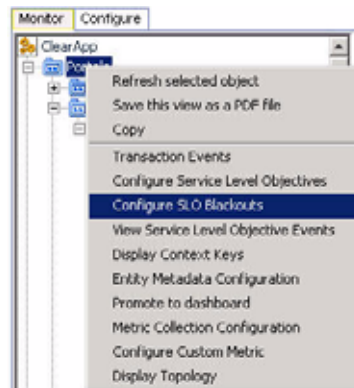


Configure SLO Blackouts

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.

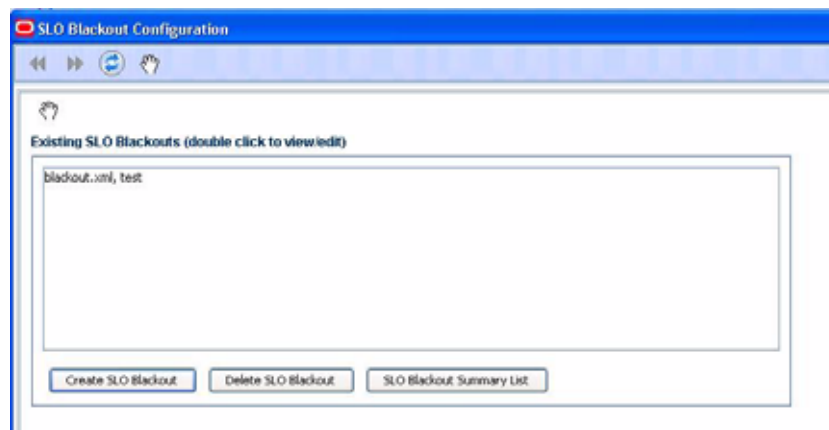
1. Right-click on any tree node and select Configure SLO Blackout to open a new window.

Figure 3-30: SLO Blackout



2. You can view any existing SLO blackout events in the next window. See [Figure 3-31](#).
3. Use this window to create, delete or view the details of existing events.

Figure 3-31: SLO Blackout List



Delete SLO Blackout

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

SLO Blackout Summary List

1. Click **SLO Blackout Summary List**.
2. View the details on the existing SLO Blackout events. See [Figure 3-32](#) for an example.

3. Click **Show SLO Blackout List** to return to the previous window.

Figure 3-32: SLO Blackout Summary List

File Name	Name	Description	Date	Time	Duration	Recurring
blackout.xml	Deeservlet	blackout from 1300 to 1500	1/29/2007	18:30	120	weekly
J2EE Servlet.xml	J2EE Servlet	at SLO file level	5/19/2007	13:30	120	none

2 rows

Show SLO Blackout List

Create SLO Blackout

1. Click **Create SLO Blackout** to view the detail window.

Figure 3-33: SLO Blackout Configuration

SLO Blackout Configuration

Blackout Name:

Description:

Blackout By SLO File: 1 SLO File(s) selected.

Blackout By SLO:

Blackout by Entity:

Blackout Period:

year: (yyyy)

month: (1 - 12)

date: (1 - 31)

hour: (0 - 23)

minute: (0 - 59)

duration: (minutes)

recurring:

2. Use [Table 3-2](#) to fill in the columns as needed.

Table 3-2: SLO Blackout Configuration

Column/ Metric	Description
Blackout Name	Type in the name.
Description	Type in the description of the SLO you are creating.

Table 3-2: SLO Blackout Configuration (Continued)

Column/ Metric	Description
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.
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.

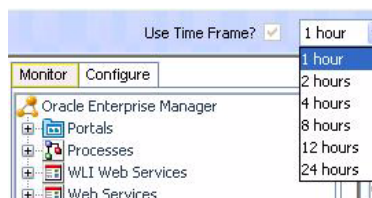
Time Frame

In CAMM™, you can specify the size of the time window information 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

Figure 3-34: Time Frame Selection



Note: CAMM™'s 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, please refer to the *CAMM™ Deployment Guide*.

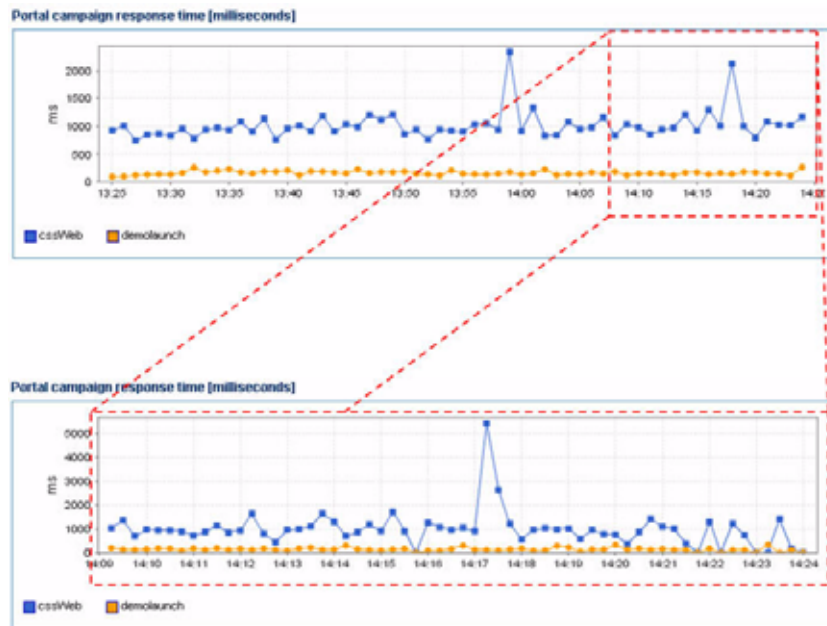
CAMM™ automatically adjusts information displayed to fit the specified time window. You can drill down to see detailed performance information for a specific range of time. The following two figures illustrate how the same data is displayed differently when using different Time Frames.

[Figure 3-35](#) illustrates the drill down process with two screen shots of the same graph with different Time Frames. These graphs visualize the average response time for Portal campaign, a WebLogic® Portal subsystem. The top graph has a Time Frame of one hour. The bottom graph has a Time Frame of fifteen minutes. 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 shortly before 14:00 and 14:20. The person decided to investigate further to evaluate the extent of the problem. By changing the Time Frame from one hour to fifteen minutes, 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.

Note: By default, fifteen minutes is not an option in the drop-down Time Frame menu. You can customize the application to make this option available. The default values start from one hour.

Figure 3-35: Portal Campaign Response Time in Different Time Frames



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

When you select a new Time Frame, the Display Interval automatically changes to fit the selected Time Frame. Here is an example of how the Display Interval would change by changing the Time Frame from 1 hour to 2 hours:

Figure 3-36: Impact of Time Frame Change on Display Interval



Notice when you change the Time Frame, the Start value of the Display Interval changes.

Interval Context

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

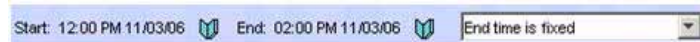
The default Interval Context for CAMM™ is to use current system time as End value for the Display Interval. In this default setting, you have a sliding Display Interval and can see the latest performance information in the Main Display Window.

Figure 3-37: Interval Context set to End Time is Current System Time



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.

Figure 3-38: Interval Context set to End Time is Fixed



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.

Figure 3-39: Date/Time Selector

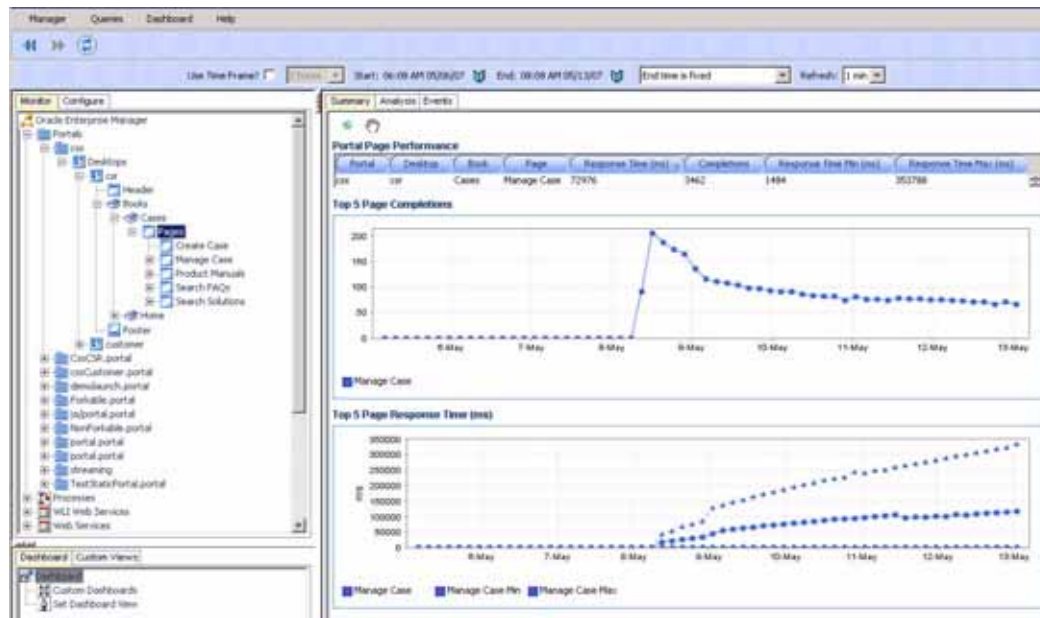


The Date/Time Selector allows you to set a specific Display Interval to fit your needs. Additionally, the Date/Time Selector enables CAMM™ to compare current performance trends with historical data.

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 the **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.

Figure 3-40: Use Time Frame? Check Box

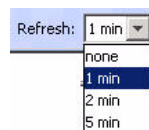


In [Figure 3-40](#) you specified the time frame to be eight days by adjusting the start and end times through the Date/Time Selector. CAMM™ then adjusts its view to display eight days worth of data in a single graph. This feature allows you to perform trending analysis over time.

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.

Figure 3-41: Refresh Rate Drop-down Box



CAMM™ is capable of automatically refreshing the information in the Main Display Window at five different rates:

- 1 minute
- 2 minutes
- 5 minutes

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™.

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 CAMM's™ URL query dialog box. See [Figure 3-44](#).

Figure 3-42: Copy URL

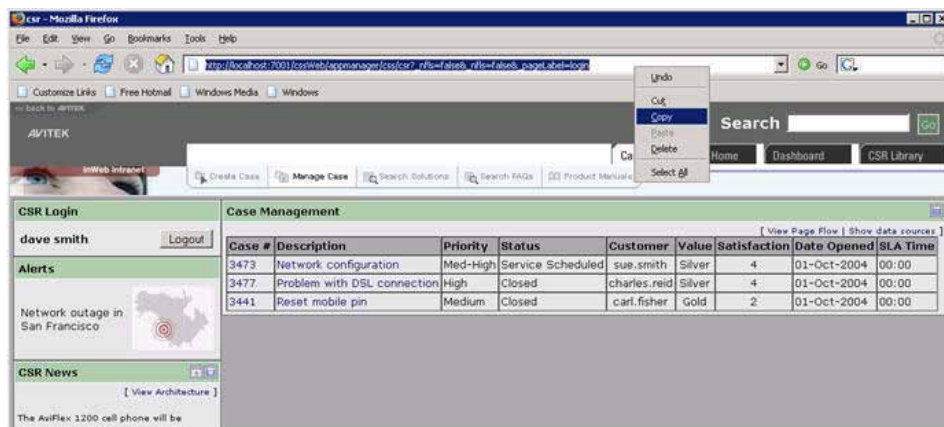


Figure 3-43: Queries Menu



Figure 3-44: Enter URL Query

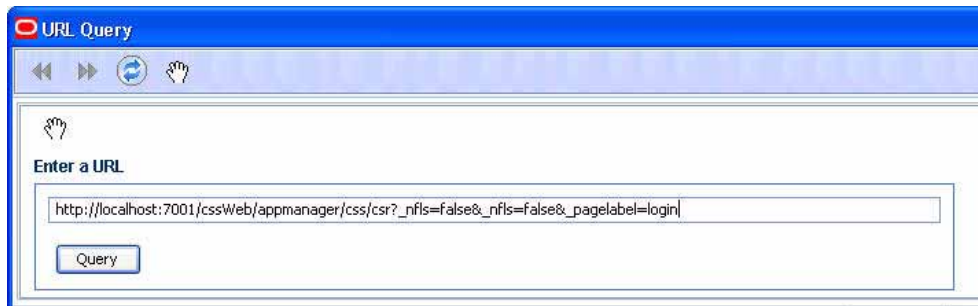
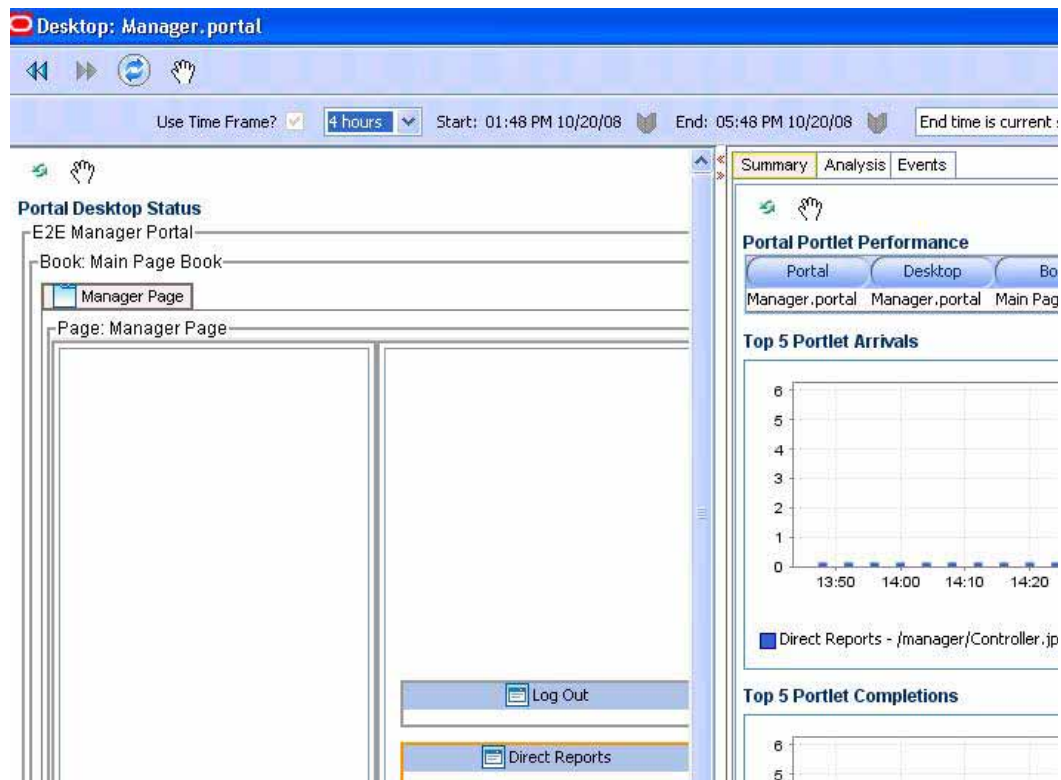


Figure 3-45: Found the Portal Desktop View Relevant to the URL



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

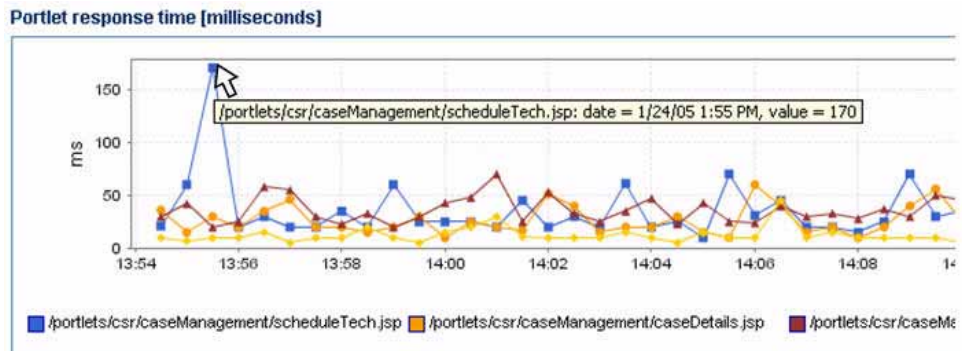
- BPEL Processes (.bpel)
- Portals (.portal)
- Processes (.jpd)
- Java Page Flows (.jpf)
- Struts Actions (.do)
- Web Services (.jws)
- Java Server Pages (.jsp)

- Servlets

Graphs and Data Items

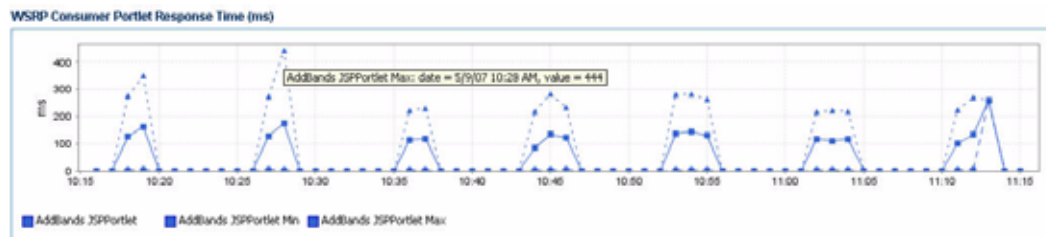
CAMM™ displays performance information in various formats. Most commonly used display formats in CAMM™ are tables and graphs. As a general rule, you can gain more information about a data item by simply pointing the mouse over the interested item. See [Figure 3-46](#).

Figure 3-46: Mouse Over to Display Data Item Details



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. See [Figure 3-47](#).

Figure 3-47: Min/Max Metrics



For tables, you can perform table sort by clicking on column headings. Columns can also be rearranged with simple drag and drop action. [Figure 3-48](#) shows the effect of a mouse click on the column heading Probe point. Notice the red sort directional arrow to the right of the column heading. This red sort directional arrow appears next to the column currently being sorted. [Figure 3-49](#) is a screen shot of a user rearranging columns.

Figure 3-48: Sort Data by Column

Probe point	Hits	Response time	Portal	Desktop
/portlets/csr/caseManagement/caseDetails.jsp	96	12	css	csr
/portlets/csr/caseManagement/scheduleTech.jsp	96	29	css	csr
/portlets/csr/caseManagement/caseList.jsp	96	32	css	csr
/portlets/csr/caseManagement/caseList.jsp	192	38	css	csr

Figure 3-49: Rearranging Table Columns

Hits	Response time	Portal	Probe point	Context
96	12	css	/portlets/csr/caseManagement/portlets/csr/caseManagement/CaseManagementController.jsp	cs
96	29	css	/portlets/csr/caseManagement/portlets/csr/caseManagement/CaseManagementController.jsp	cs
96	32	css	/portlets/csr/caseManagement/portlets/csr/caseManagement/CaseManagementController.jsp	cs
192	38	css	/portlets/csr/caseManagement/portlets/csr/caseManagement/CaseManagementController.jsp	cs

You can define the zoom in area using a click and drag operation. [Figure 3-50](#) is an example of the zoom in feature to see details of a highly complex graph. To zoom in, click and drag the mouse to the right. To zoom out, click and drag the mouse to the left.

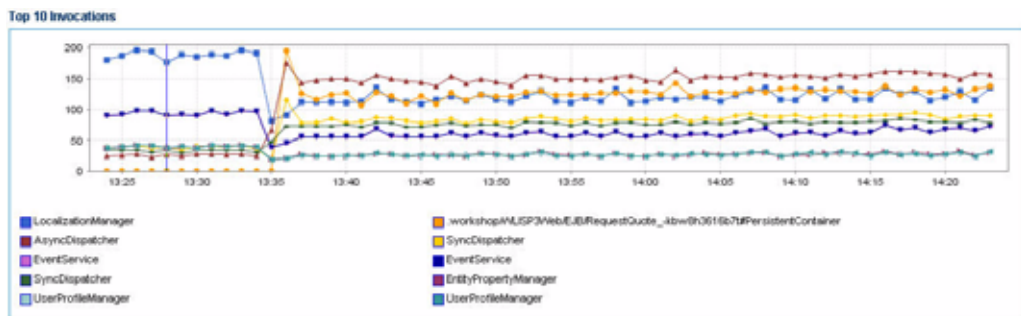
Figure 3-50: Zoom In to See Graph Details



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.

Figure 3-51: Zoom in to See Graph Details



Right-Click Operations on Tables and Graphs

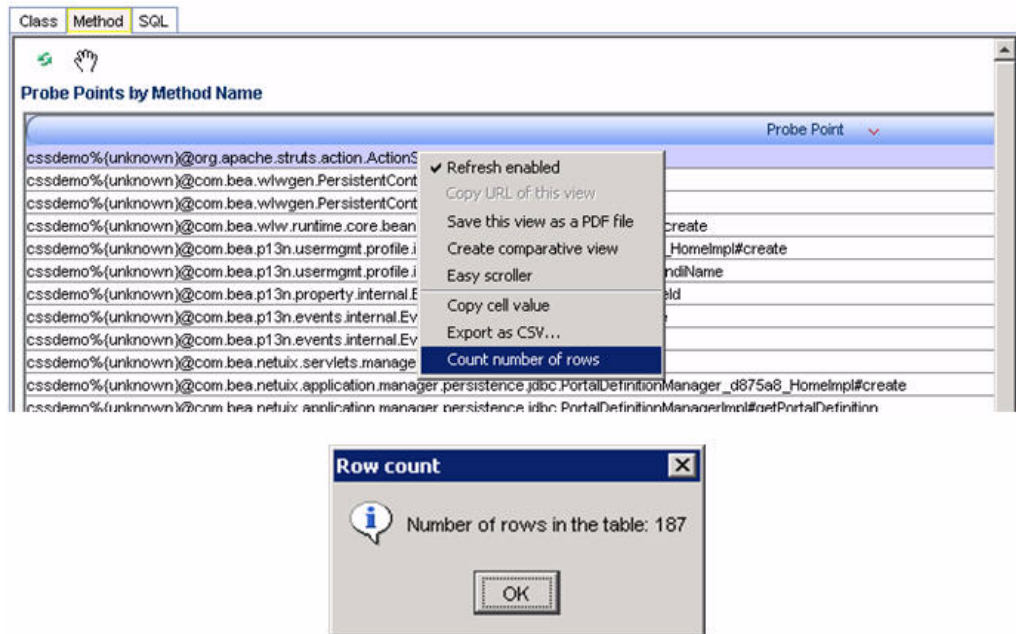
There are several simple operations that can be performed on various tables and graphs in CAMM™. The following is a list of the right-click operations associated with tables and graphs:

Table 3-3: Right-Click on Tables

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.
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.

[Figure 3-52](#) depicts the Count number of rows operation. Use the Count number of rows right-click operations to get a total row count for any table.

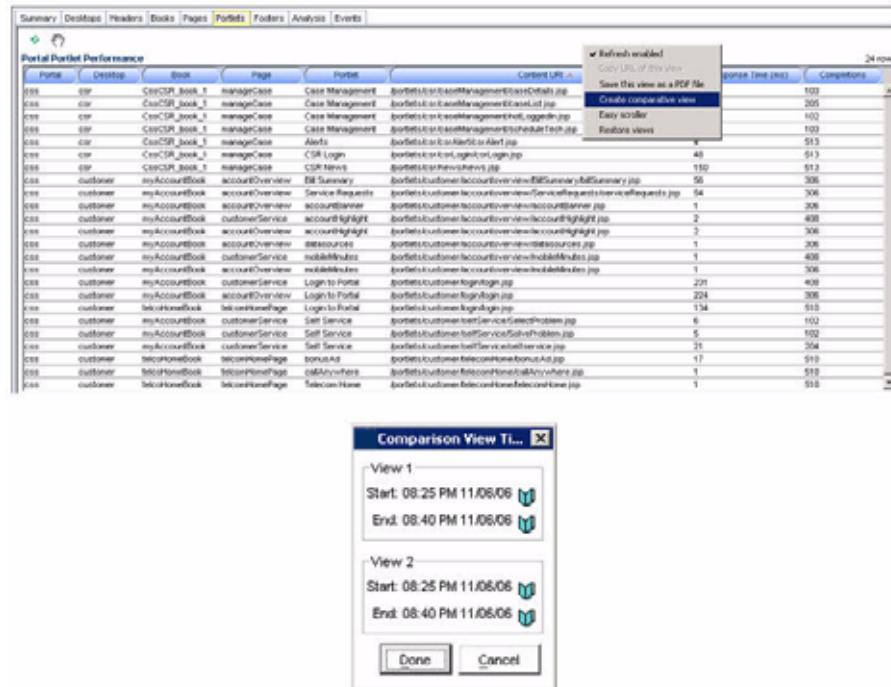
Figure 3-52: Right-Click Operations to Get a Total Row Count

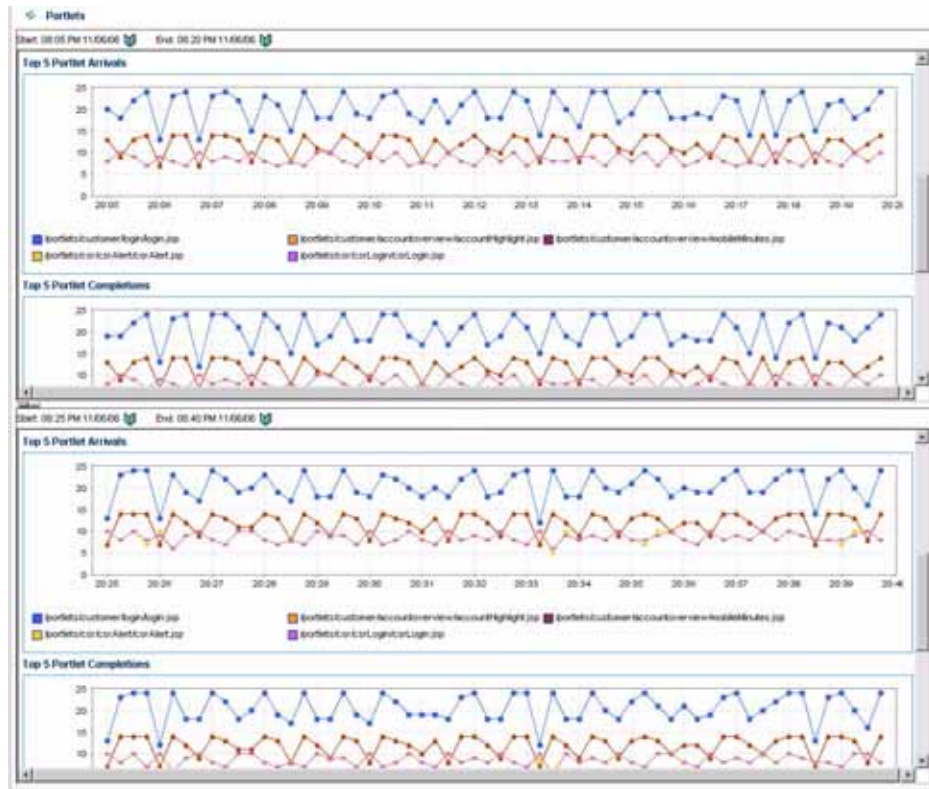


Comparative View

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 on CAMM™'s Main Display Window and select Create Comparative View.

Figure 3-53: Right-Click to Activate Comparative View





After the Comparative View window shows, you can use the Date / Time Selector to specify start and end times for each of the two windows in the Comparative View. This tool can be used 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.

[Figure 3-54](#) is an example of using comparative views to determine if current delay characteristics are significantly different from past delay characteristics. Comparative views are useful to evaluate current performance characteristics against historical performance characteristics.

Figure 3-54: Comparative Views to Evaluate Performance Characteristics



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 CAMM™'s Main Display Window and select Save this view as a PDF file.

Figure 3-55: Right-Click on Any View to Save As a PDF

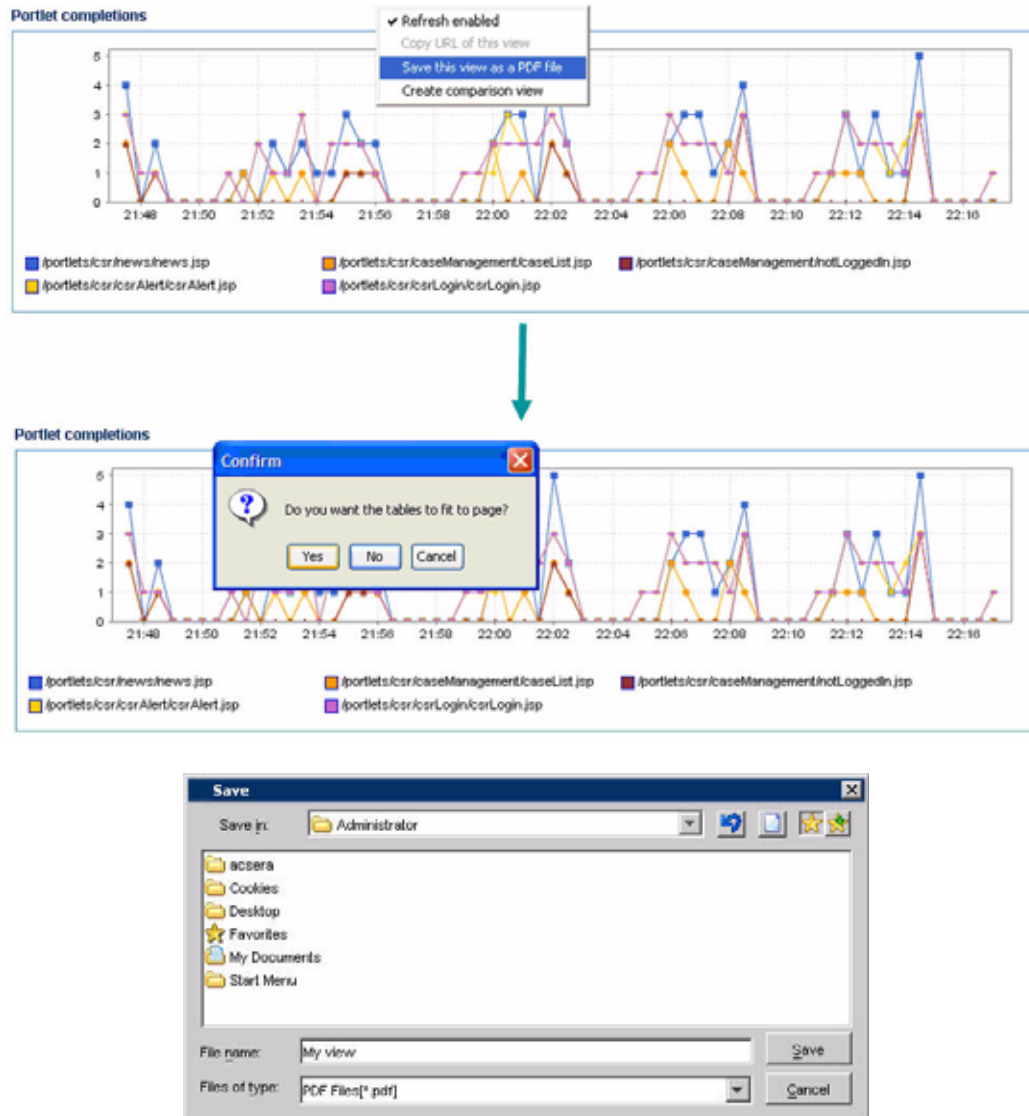


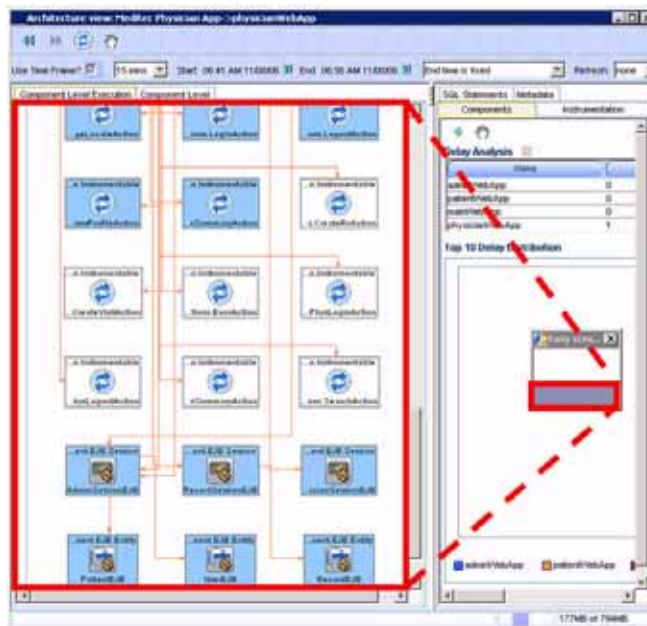
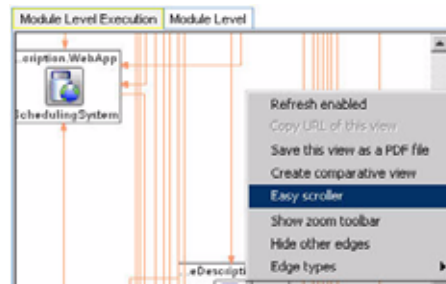
Figure 3-56: File Sharing Improves Collaboration



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 and select the Easy Scroller option if available. Drag the box within Easy Scroller to navigate. See [Figure 3-57](#).

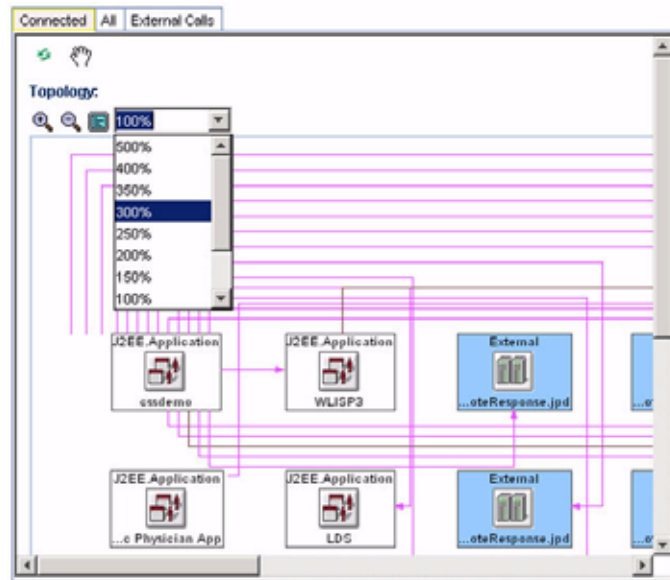
Figure 3-57: Use Easy Scroller to Navigate Complex Views






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. See [Figure 3-58](#).

Figure 3-58: Drop-down Box Enables Quick Zoom In / Zoom Out



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 on the view.

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, custom metrics can be used 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.

Figure 3-59: Configure Custom Metric

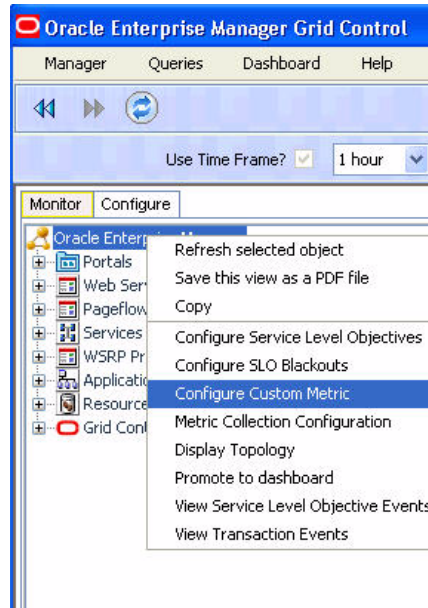


Figure 3-60: Define and Modify Custom Metrics



Custom Metric Configuration window includes the following fields:

Table 3-4: Custom Metrics Configuration Window

Field	Description
Name	This text field is for defining the display name for the custom metric.
Resource Name	This drop-down selection 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 (Optional)	<p>This optional text field is for defining the method name associated with the custom metric.</p> <p>Usage:</p> <ol style="list-style-type: none"> 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. <p>Method field examples:</p> <ol style="list-style-type: none"> 1. <empty string> 2. methodA,methodB,methodC 3. ejb*,*context,methodA 4. !ejb*,*context,methodA

After the custom metrics are defined, the application server instances associated with these customizations should be restarted. The new custom metrics will be listed under the Custom Metrics node in the CAMM™ navigation tree.

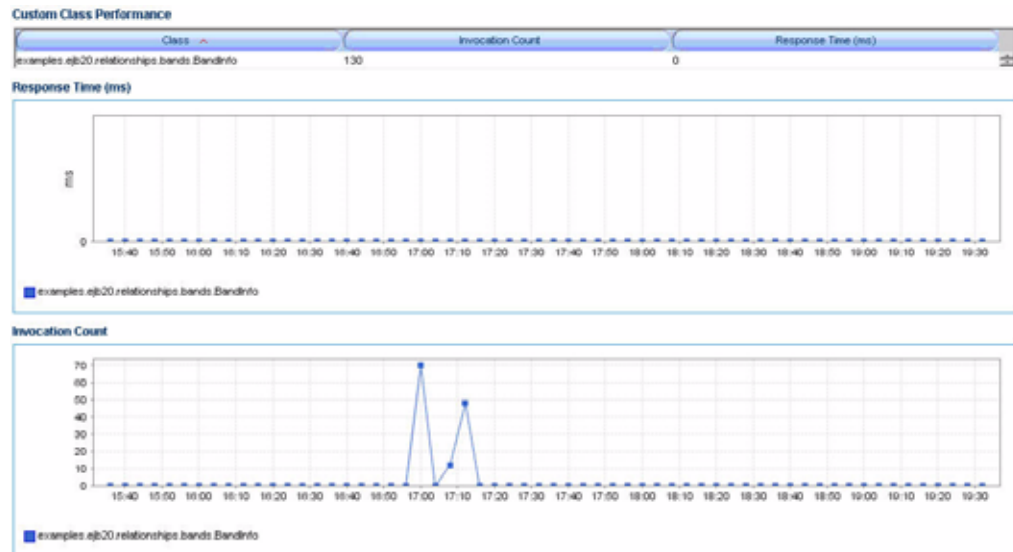
[Figure 3-61](#) shows the newly configured custom metric for the class `examples.ejb20.relationships.bands.BandInfo` is now listed under the Custom Metrics node of the CAMM™ navigation tree. The Custom Metrics node provides a list of pre-configured custom metrics.

Figure 3-61: Custom Metrics Node



Newly configured custom metric reports class level performance data like invocation count and response time.

Figure 3-62: Custom Metric Reports Class Level Performance Data



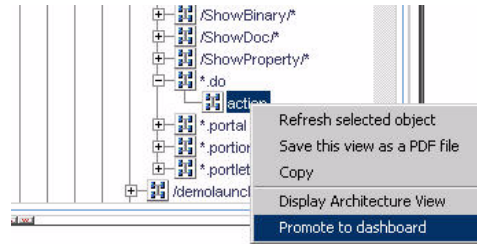
Promote To Dashboard

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

To add new metrics to the operational dashboard:

1. Right-click and select Promote to dashboard to activate the Dashboard Configuration window

Figure 3-63: Dashboard Configuration Window



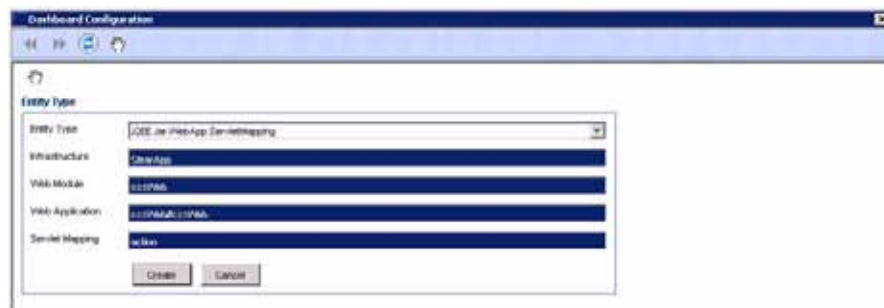
2. Click **Create Dashboard Promotion**.

Figure 3-64: Create Dashboard Promotion



3. Select the metric to promote to Operational Dashboard from the drop-down menu.
4. Click **Create**.

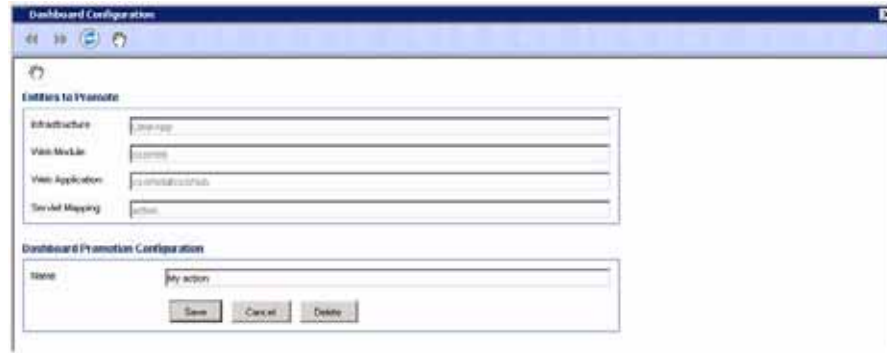
Figure 3-65: Define Display Name



5. Type in a name in the **Name** box.
6. Click **Save**.

7. Close the Dashboard Configuration window.

Figure 3-66: Dashboard Promotion Configuration



8. Click **Yes** to confirm you want to close the window.
Your action is now a new entry on the Operational Dashboard.

Figure 3-67: New Entry on Operational Dashboard



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. The following is a list of functional views currently available in CAMM™:

Table 3-5: 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.
WLI Web Service	WLI Web Service Functional View	This functional view depicts all the public operations associated with the selected WLI Web Service.
Web Service	Web Service Functional View	This functional view depicts all the public operations associated with the selected Web Service.
Pageflow	Pageflow Functional View	This functional view depicts the logical flow associated with a pageflow. It shows all the pages in a pageflow and the relationships among them.

Depending on the type of entity selected, CAMM™ displays different functional views. The following are screen shot examples of these functional views:

Right-click and select Display Functional View to bring up the relevant Functional View associated with the selected entity.

Figure 3-68: Accessing Functional View

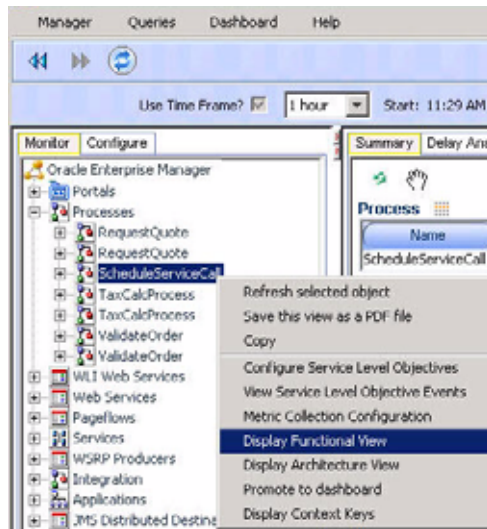


Figure 3-69: Process Functional View

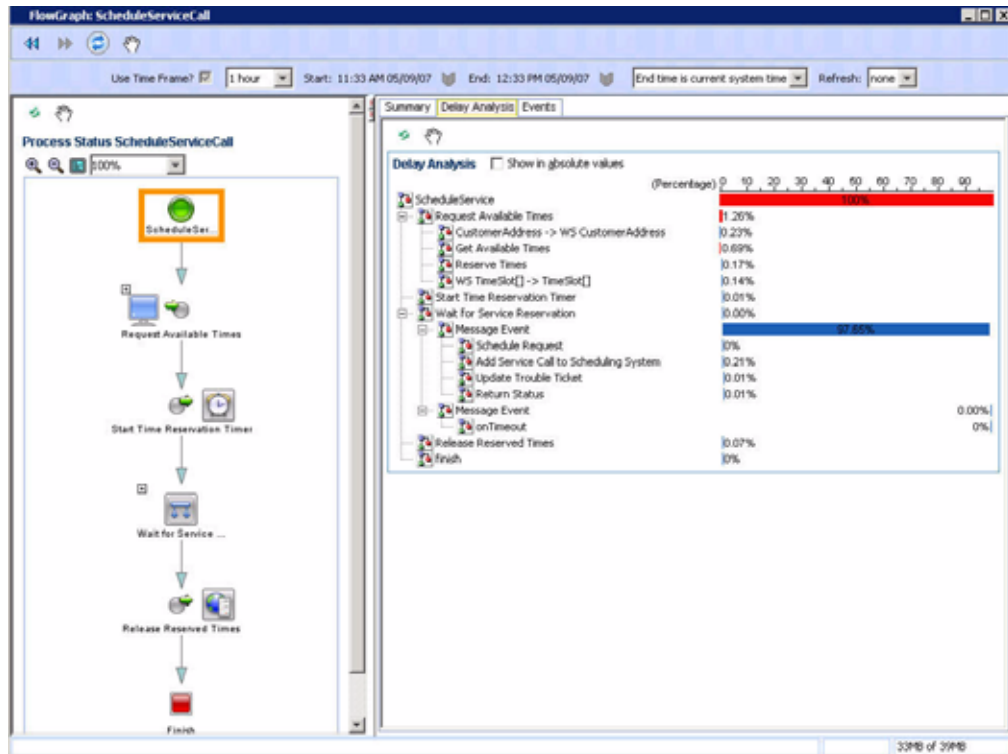
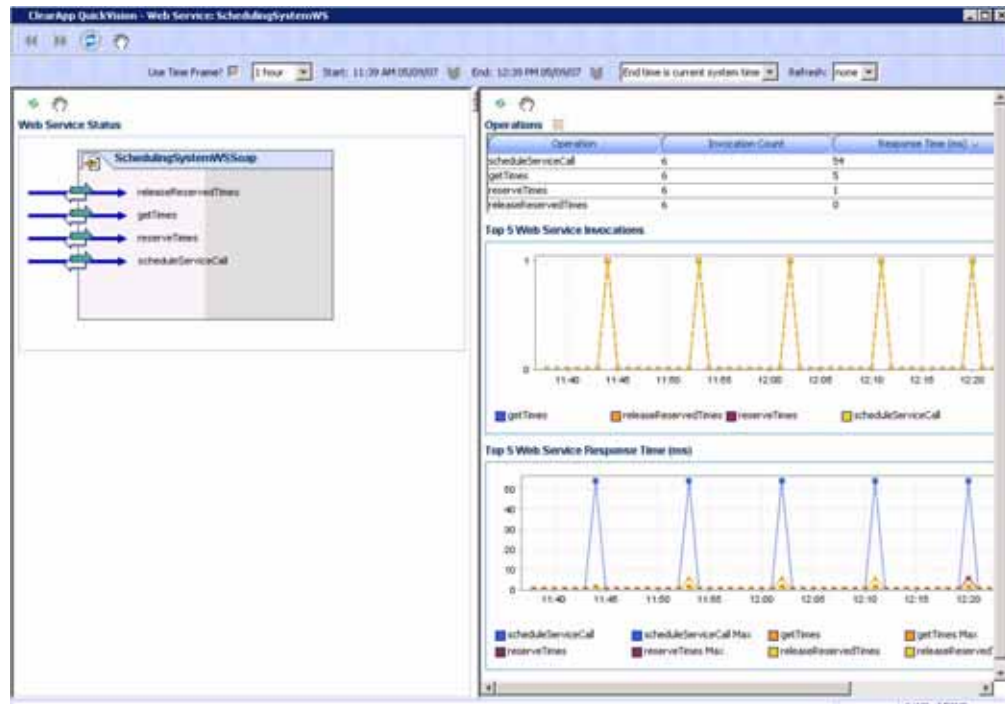


Figure 3-70: Web Service Functional View

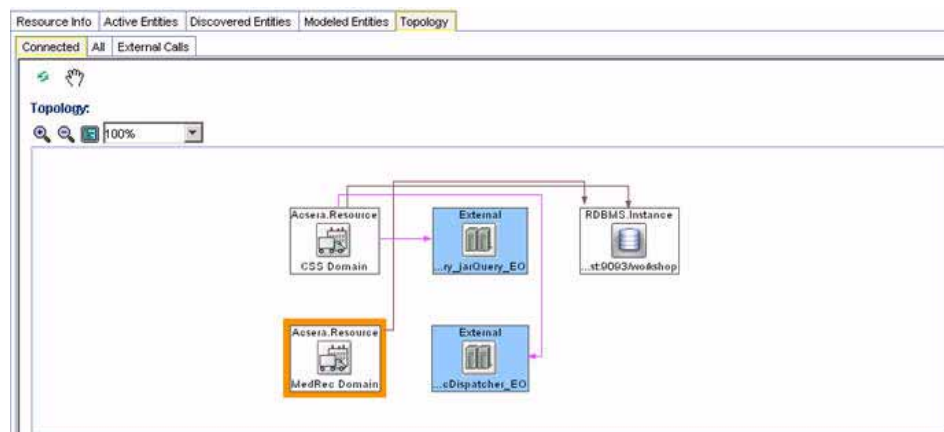


Topology View

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.

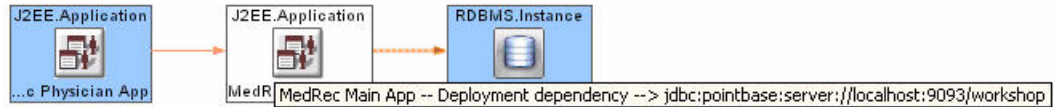
The highest level topology view graphically depicts domains, external resources, and shared database resources. The demo applications used in the following examples are CSS and MedRec demos.

Figure 3-71: Highest level Topology View



[Figure 3-71](#) shows 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. See [Figure 3-72](#).

Figure 3-72: Mouse Over Specific Arrow for Tool Tip



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

Edge Types and Colors

There are two kinds of edge types:

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

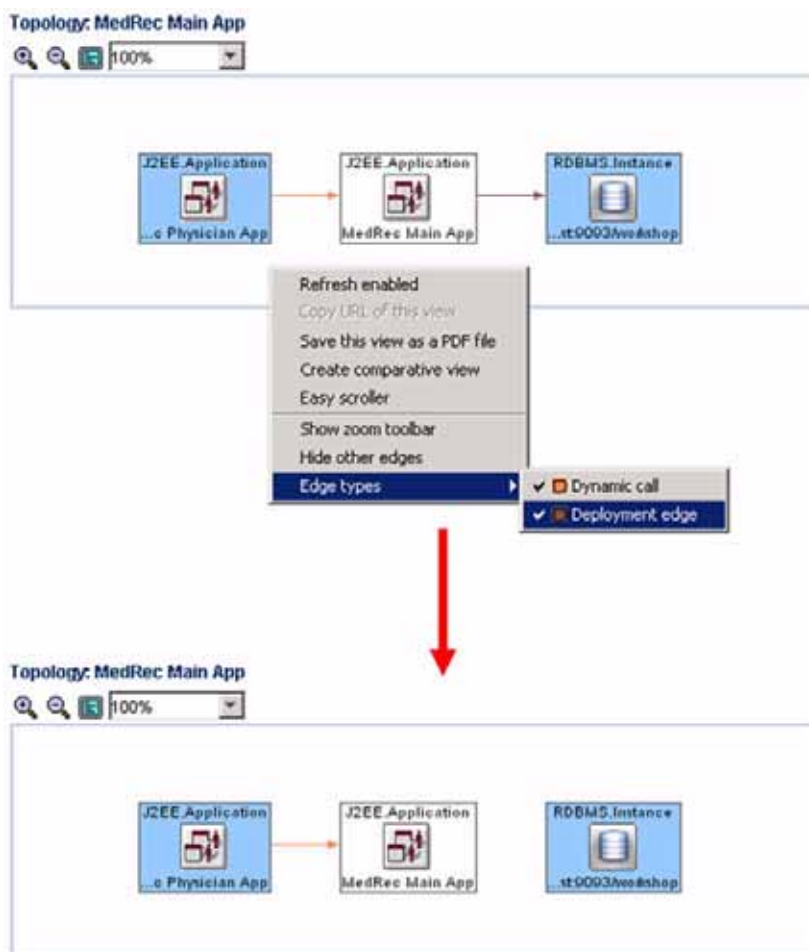
The colors displayed in the topology view are explained in [Table 3-6](#):

Table 3-6: 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.
Gray	Method call.
Light Green	Java Implementation edge. When Class B extends to A, green edge will be from B to A.
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.

Un-checking a specific edge type hides all lines of that type in the Topology View. Checking a specific edge type makes these lines appear. See [Figure 3-73](#).

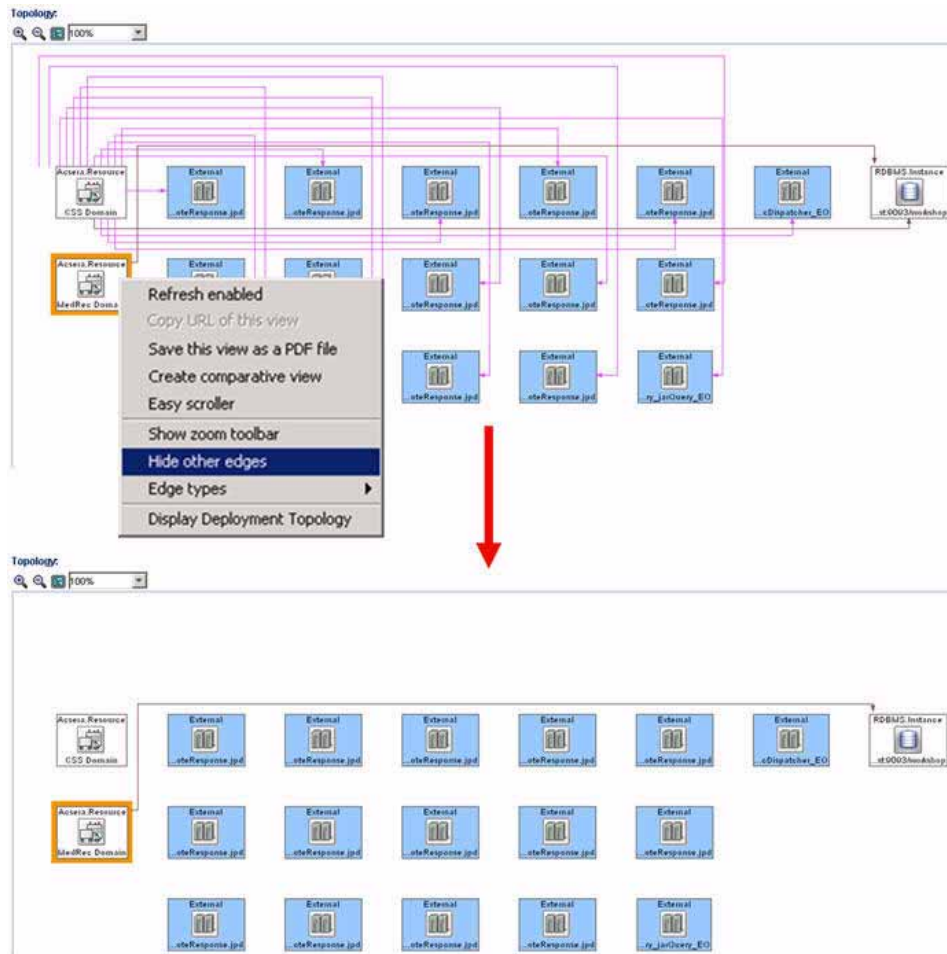
Figure 3-73: Hide Arrows in Topology View



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. See [Figure 3-74](#).

Highlight An Entity and select Hide other edges to hide all arrows not connected to the managed entity.

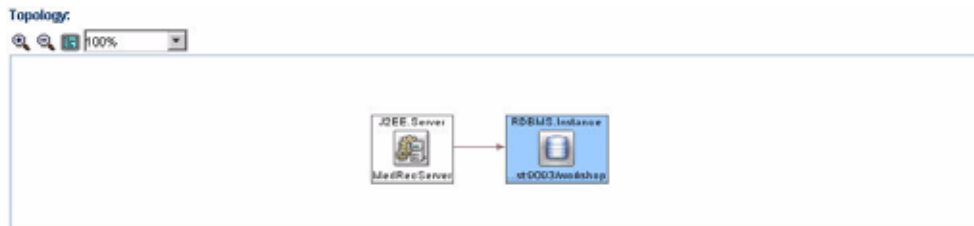
Figure 3-74: Hide Other Edges



Select a specific managed resource and double-click to display the topology specific to the selected managed resource. [Figure 3-75](#) shows the result of double-clicking on the MedRec Domain.

Drilling down on a specific managed resource reveals the relationship among the application server instance and the shared resources it uses.

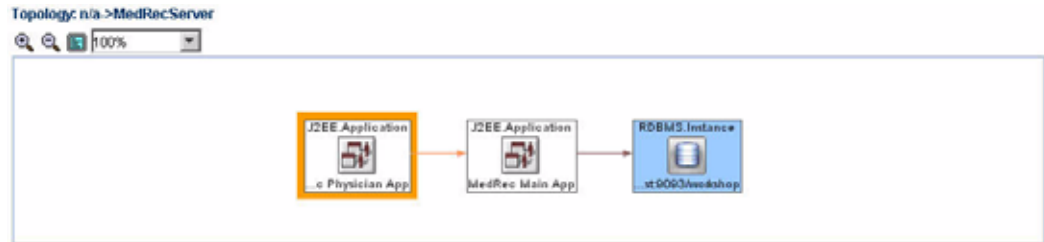
Figure 3-75: Drill Down to Reveal Relationships



[Figure 3-75](#) shows all the server instances currently configured for the MedRec domain. This view helps you understand the topological configuration of their J2EE infrastructure. It is possible to reach this same view by right-clicking on the MedRec Domain entity and selecting Display Deployment Topology option.

Drilling down on specific application server instance reveals calling relationship among J2EE™ applications and shared resources.

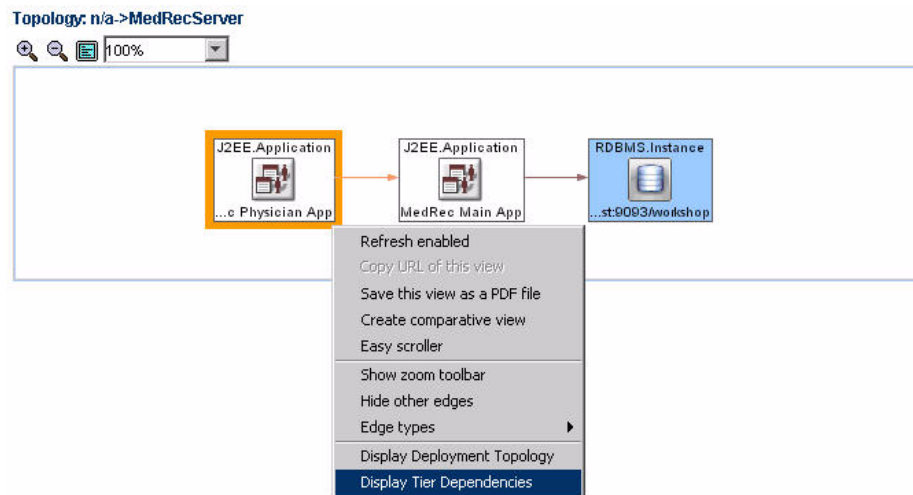
Figure 3-76: Drill Down to Display Relationship

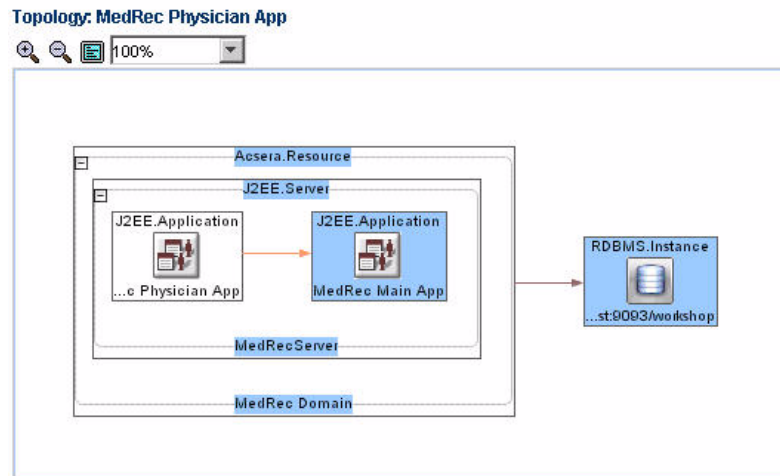


[Figure 3-76](#) shows the applications that are currently active in the MedRecServer server instance and their calling relationships. This information is useful to understand how applications are distributed across the infrastructure.

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. See [Figure 3-77](#).

Figure 3-77: Access Tier Dependency Topology View





CAMM™ brings up the Tier Dependency Topology View on the left side pane and displays the associated External Calls in the Main Display Window. [Table 3-7](#) is a list of metrics in the External Calls table and their descriptions.

Table 3-7: 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 that can be used to determine the types of calls made among various servers, applications, and shared resources. This information can be used to diagnose problems associated with cross-JVM calls. The target URL provides clue as to the type of external call made.

Drill down on a specific application to launch into the Architecture View.

Figure 3-78: Architecture View

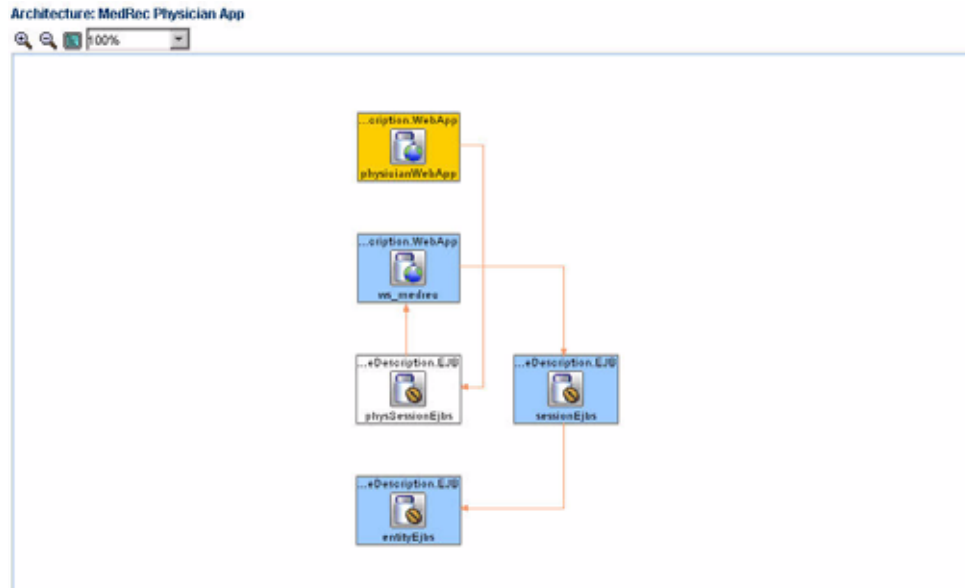


Figure 3-78 demonstrates the logical progression of drilling from high level resource-centric topology view down through application-centric topology view to module-centric architecture view. Using this logical drill down, you can understand the structure of their application runtime environments and diagnose problems.

Note: Topology View is available for nodes under Applications and Resources nodes. The drill down example in Figure 3-78 starts from the Topology View associated with the Resources node. The Topology tab displays the Topology View associated with a specific application.

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 3-8 describes various types of Architecture Views.

Table 3-8: 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, etc.). 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, etc.). 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 entities are color coded in order to provide additional information. The following is a list of colors and their meanings.

Table 3-9: Architecture View Color Codes

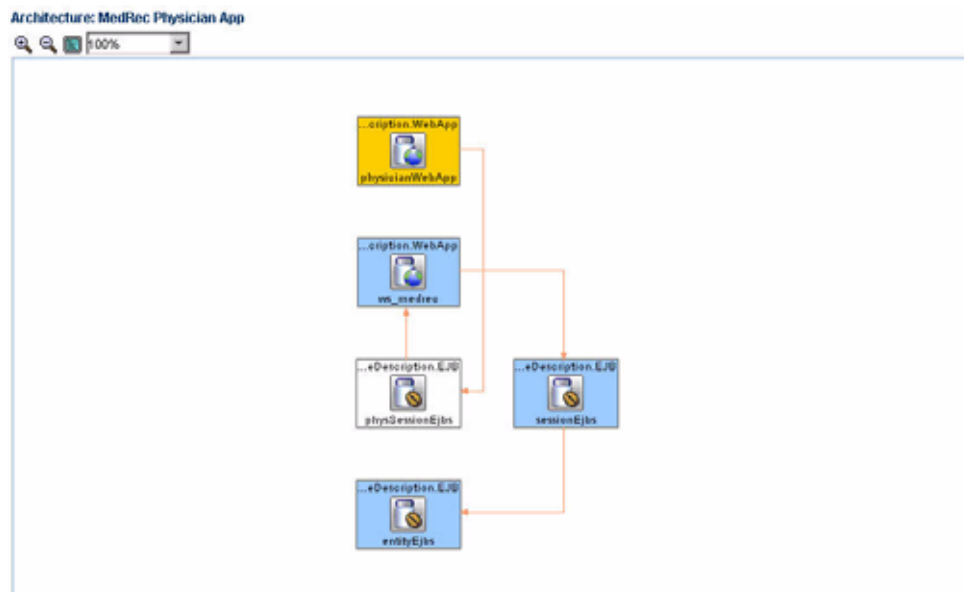
Background Color	Description
Orange	The orange background color represent entry points into the application or module. The orange color also represents these entities belong to the same application or module currently selected (in context).
Green	The green background color represent entry points into the application or module. The green color also represents 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 these entities belong to the same application and module currently selected (in context).

Table 3-9: Architecture View Color Codes

Background Color	Description
Blue	The blue background color represents these entities belong to other applications and modules (out of context). The blue color also represents shared resources.

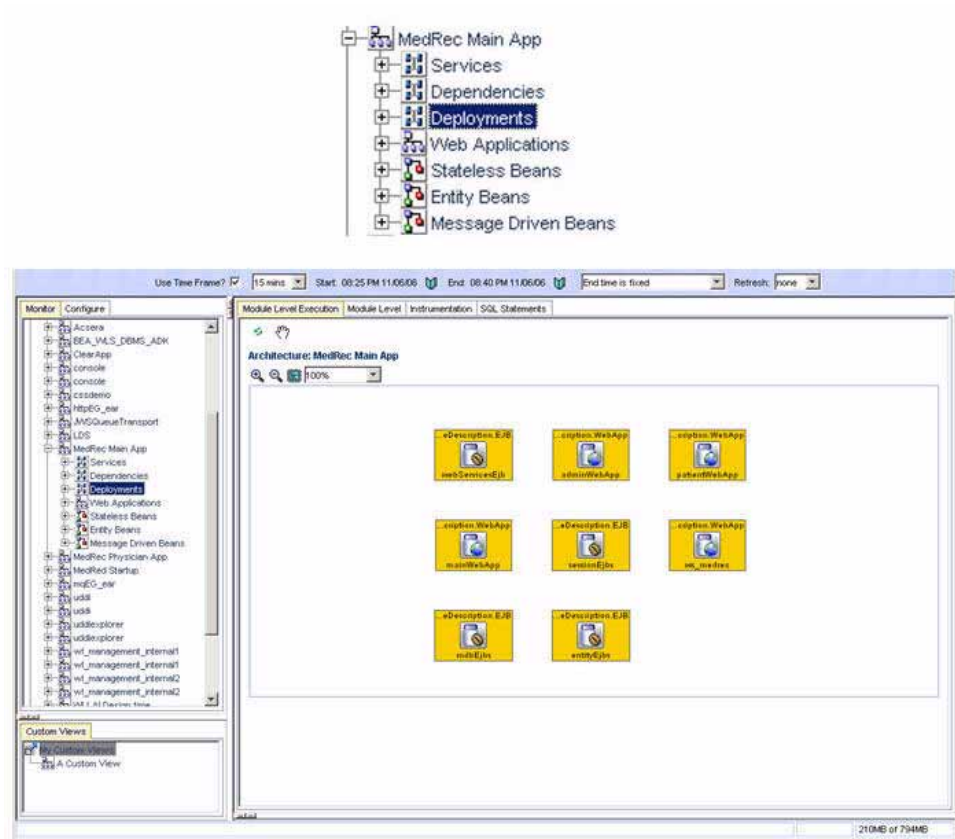
CAMM™ graphically depicts active calling relationships among various J2EE modules and shared resources.

Figure 3-79: Active Calling Relationships



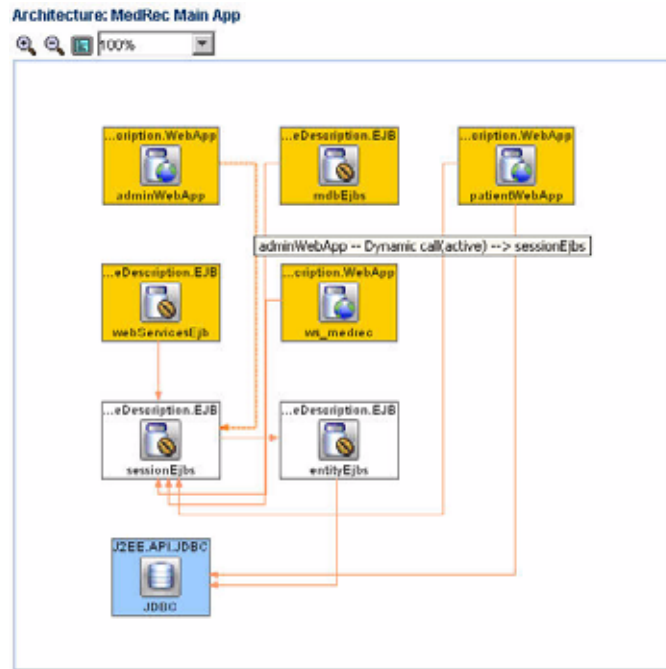
There are several ways to access the Architecture View. The first method is through the Deployments node associated with a specific application under the Application Node. See [Figure 3-80](#). Application specific Architecture View can be accessed via Deployments node on the Oracle Tree.

Figure 3-80: Application Specific Architecture View



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. See [Figure 3-81](#). Mouse over arrow shows the details of a specific call in Architecture View

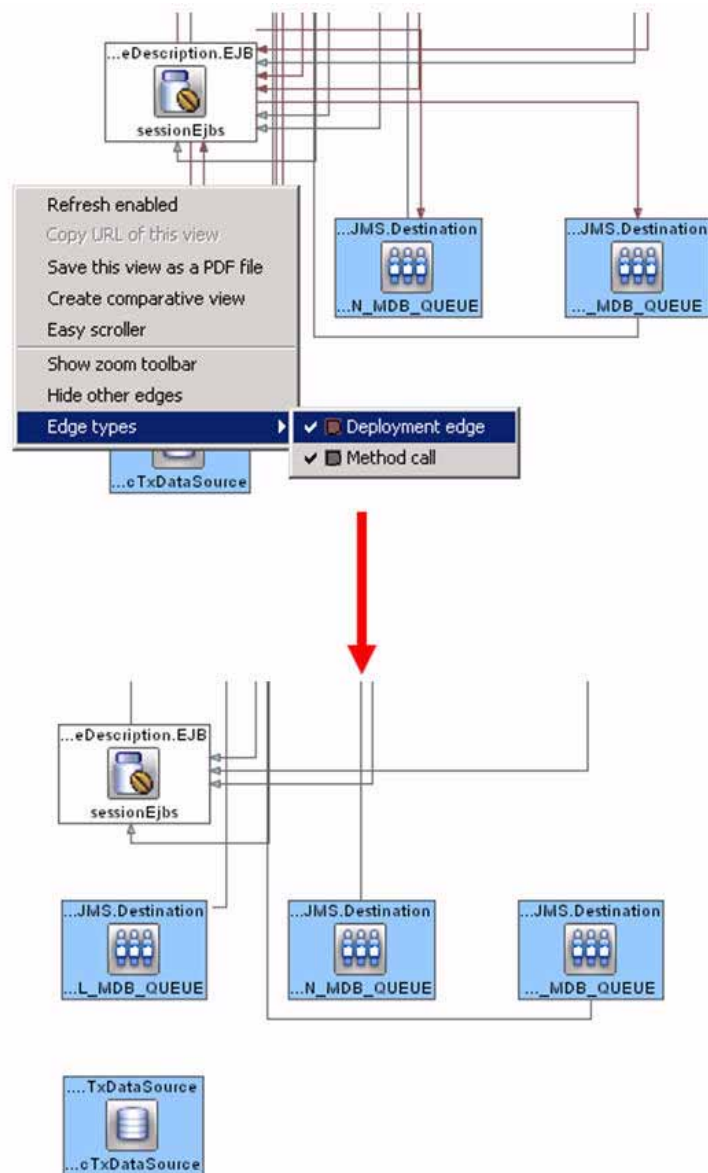
Figure 3-81: Details of a Specific Call in Architecture View



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.

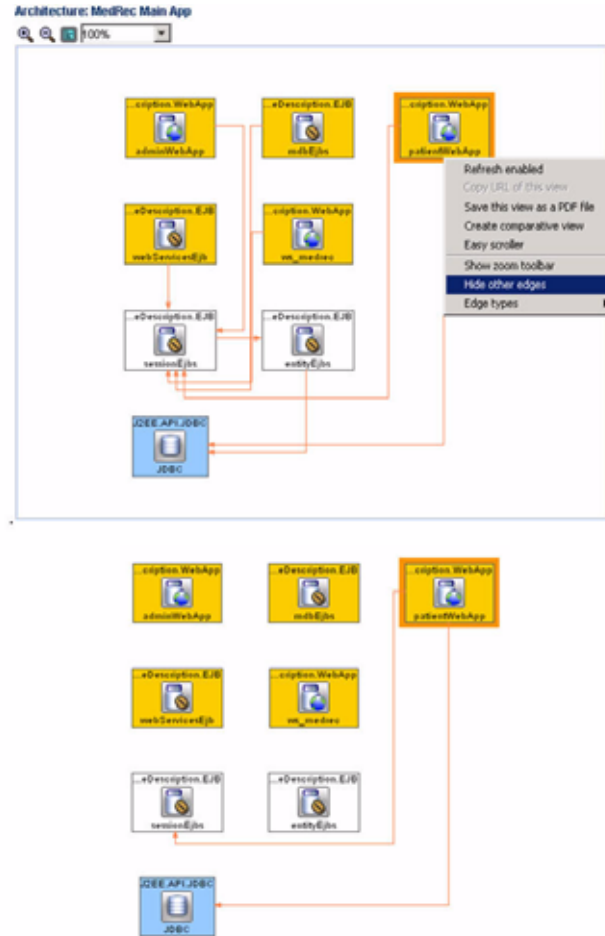
See [Figure 3-82](#). Un-check a specific Edge type to hide arrows in Architecture View.

Figure 3-82: Edge Types



See [Table 3-6](#) for the color descriptions in 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. See [Figure 3-83](#). Highlight an entity and select Hide other edges to hide all arrows not connected to the managed entity.

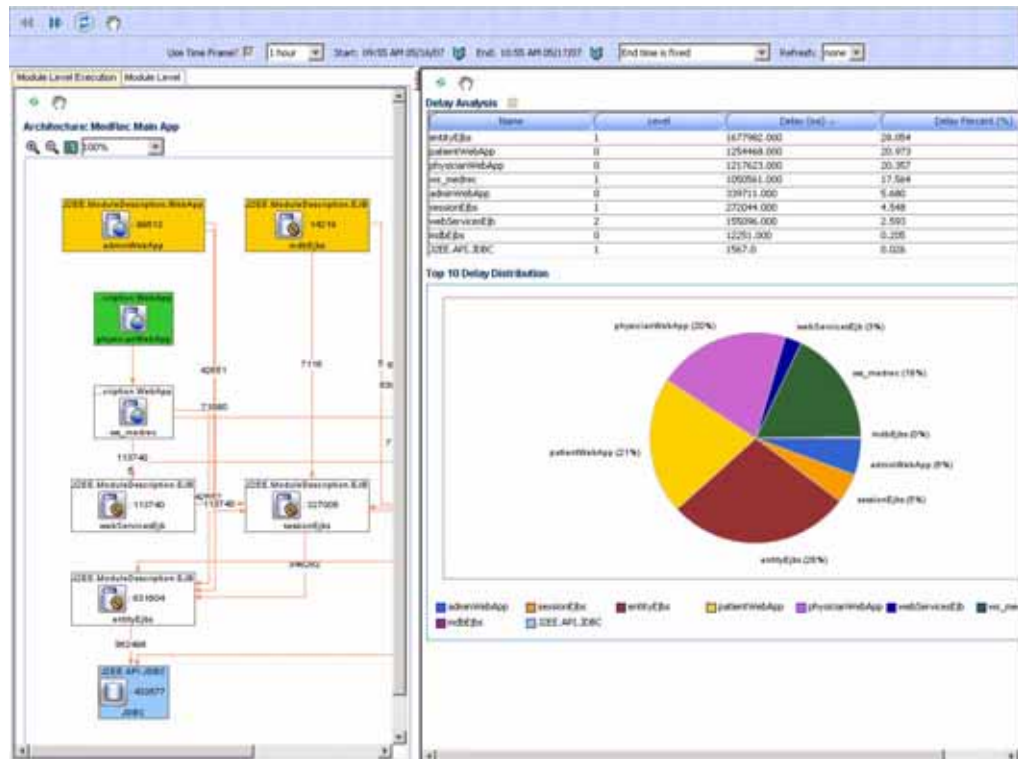
Figure 3-83: Hide Other Edges



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 on a managed entity and selecting the Architecture View. See [Figure 3-84](#). Right-click and select Architecture View to start the drill down process.

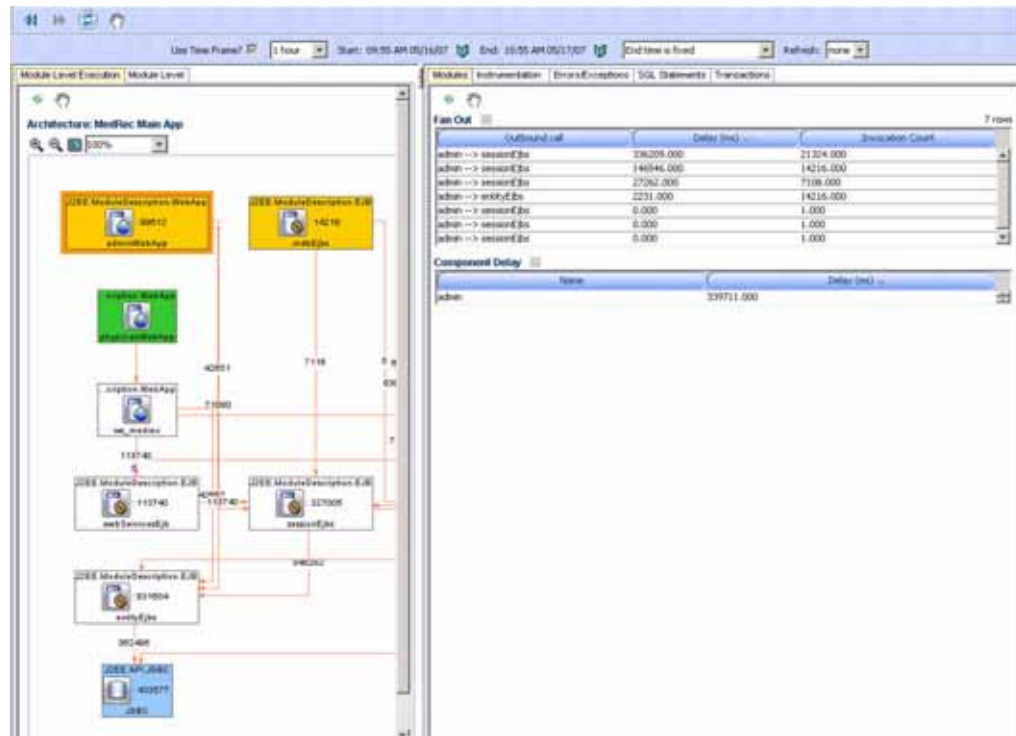
Figure 3-84: Access Architecture View - Summary View



[Figure 3-84](#) shows the Architecture View Summary with 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 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. 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. Refer to [Triage from Dashboard](#) for more information.

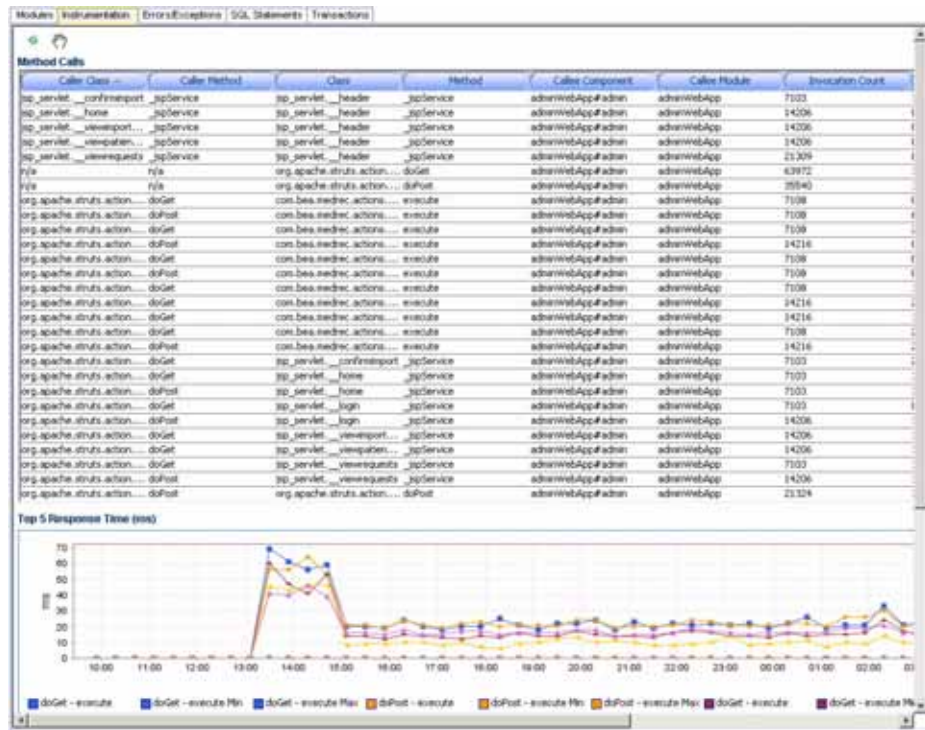
Selecting a specific entity in the active call path view brings up detailed performance data.

Figure 3-85: Detail Performance Data



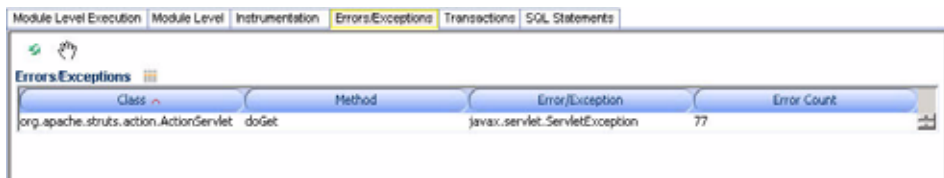
Click the Instrumentation tab to see detailed performance measurements at the method level. Navigate to the Instrumentation tab for more detailed performance information at the method level.

Figure 3-86: Instrumentation Tab



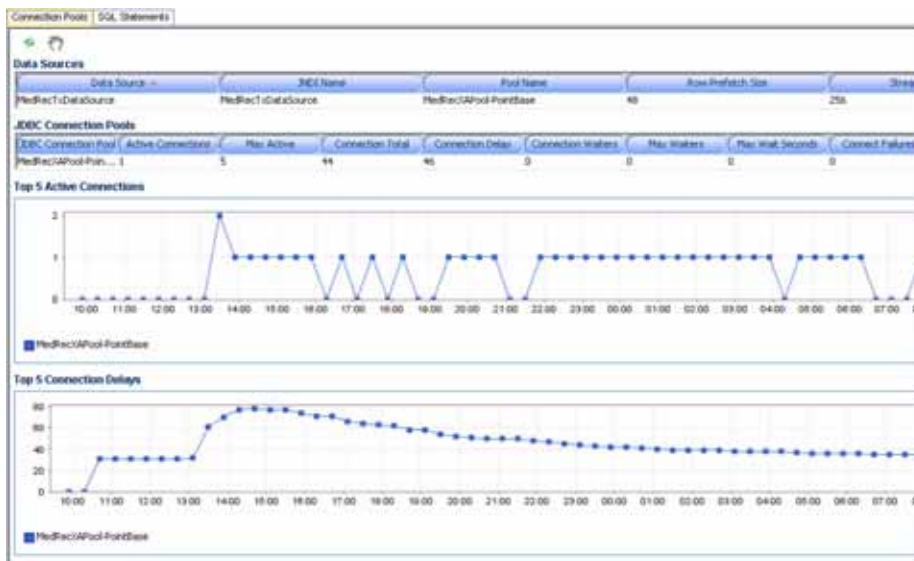
Click the Errors/Exceptions tab to view the errors metrics associated with the selected portal.

Figure 3-87: Errors/Exceptions Tab



Click the SQL Statements tab to see detailed performance data of all SQL statements associated with the selected entity. Navigate to the SQL Statements tab for a list of SQL statements associated with the selected component and their current performance.

Figure 3-88: SQL Statements Tab



Metric Types

[Table 3-10](#) contains various types of metrics provided by CAMM™ and their descriptions.

Table 3-10: Metric Types

Examples	Metric Type	Metric Description
<ul style="list-style-type: none"> 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.
<ul style="list-style-type: none"> 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.

Table 3-10: Metric Types

Examples	Metric Type	Metric Description
Response Time Elapse Time Connection Delay	Average Timing	Calculated every sampling period (default 15 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: <ol style="list-style-type: none"> 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.

4

Exploring the Monitor Workspace

This chapter includes the following topics:

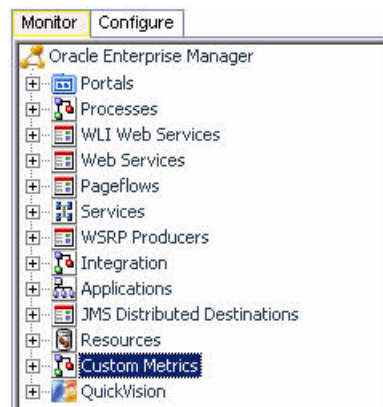
- [Oracle™ Tree](#)
- [WebLogic® Portals](#)
- [WebSphere® Portals](#)
- [Oracle® BPEL Processes](#)
- [Oracle® ESB](#)
- [WLI Web Services](#)
- [Web Services](#)
- [Pageflows](#)
- [Services](#)
- [WSRP Producers](#)
- [Integration](#)

- [Applications](#)
- [WebLogic® Resources](#)
- [WebSphere® Resources](#)
- [Oracle® Resources](#)
- [Custom Metrics](#)
- [Grid Control™](#)

Oracle™ Tree

When Oracle Enterprise Manager - Composite Application Monitor and Modeler (CAMM™) is pointed to a BEA® 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™ displayed this information in the Monitor Workspace under the Oracle™ Tree. [Figure 4-1](#) illustrates the main nodes under the Oracle™ Tree.

Figure 4-1: Main Nodes in the Oracle™ Tree



Each node represents a construct in the platforms monitored by CAMM™. We will cover each in this chapter.

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 panel.

For WebLogic® Portal, this hierarchy contains the following:

Table 4-1: 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.

We will discuss each component in more detail later in this document.

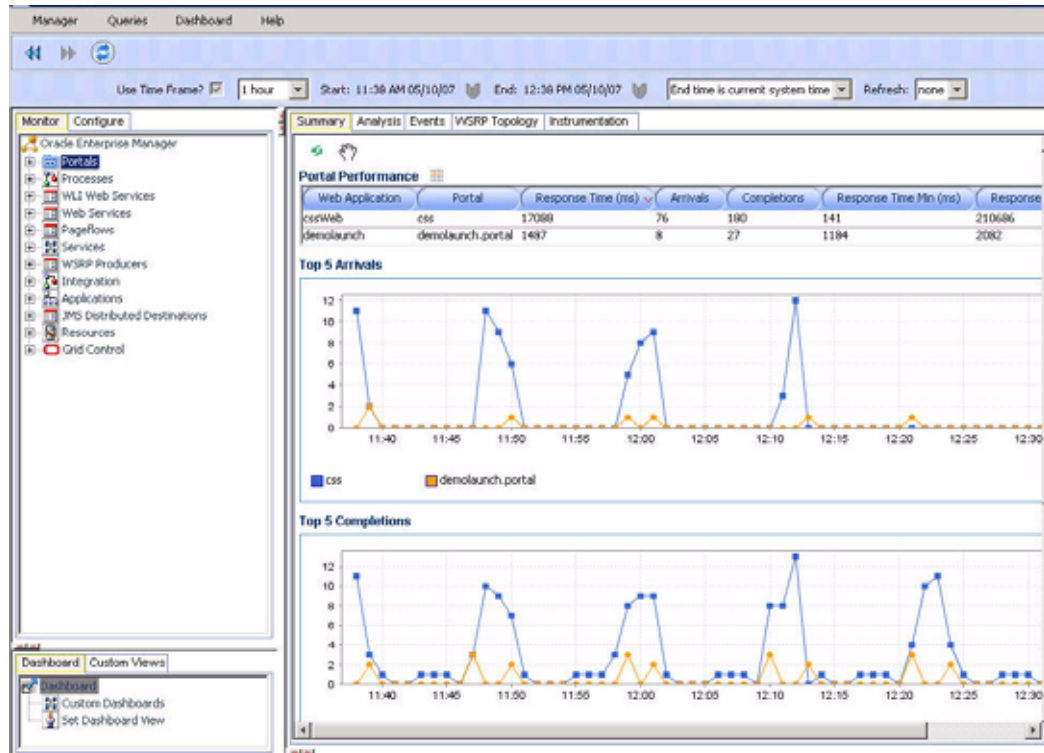
When you click the Portals node under the Oracle™ Tree, CAMM™ displays summary information on active portal applications. This summary includes the following:

Table 4-2: 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.
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.
Portal backing file response time	Average response time of WebLogic® Portal backing files for a specific portal applications.

For Portal web application activity and Portal performance, CAMM™ displays information in both table and graph formats. [Figure 4-2](#) shows a table and corresponding graph for Portal performance of various portal applications.

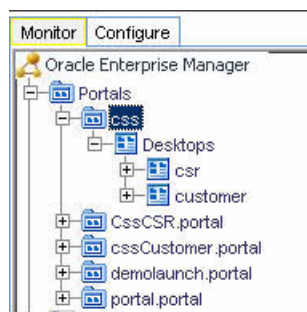
Figure 4-2: Portal Performance Table and Graphs



For the other three metrics, CAMM™ shows the information in graph format.

When you click the + icon next to the Portals node, CAMM™ expands the tree to show all managed portal applications currently deployed on the WebLogic® domain. [Figure 4-3](#) shows this configuration with five portals.

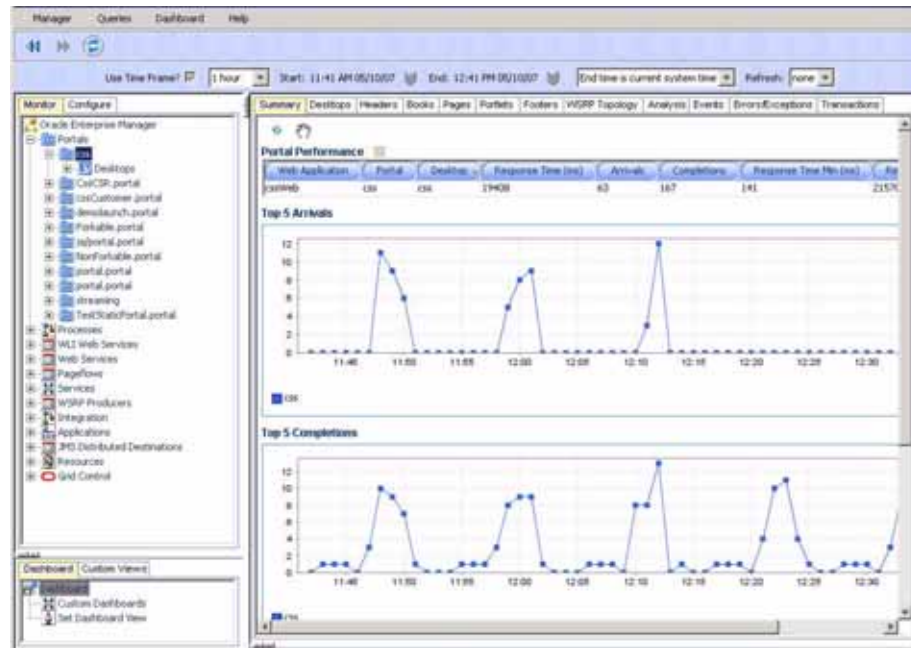
Figure 4-3: Portals Node Expanded



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.

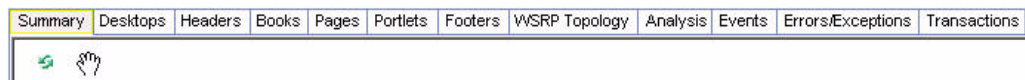
In the [Figure 4-4](#) example, a user selects the *css* portal application under the Portals node. The Main Display Window now only shows information specific to *css* portal application.

Figure 4-4: Display *css* Portal Specific Metrics



At the Portal level, you can navigate to different levels of the portal application by using different tabs. [Figure 4-5](#) shows the tabs available for portal level nodes. Use the tabs available to quickly access lower level components.

Figure 4-5: Portal Level Nodes Tabs



The following is a list of the tabs available for portal level nodes and their descriptions.

Table 4-3: Portal Level Tab Descriptions

Tab	Description
Summary	The performance summary specific to the selected portal.
Desktops	The performance summary for all the desktops associated with the selected portal.
Headers	The performance summary for all the headers associated with the selected portal.
Books	The performance summary for all the books associated with the selected portal.

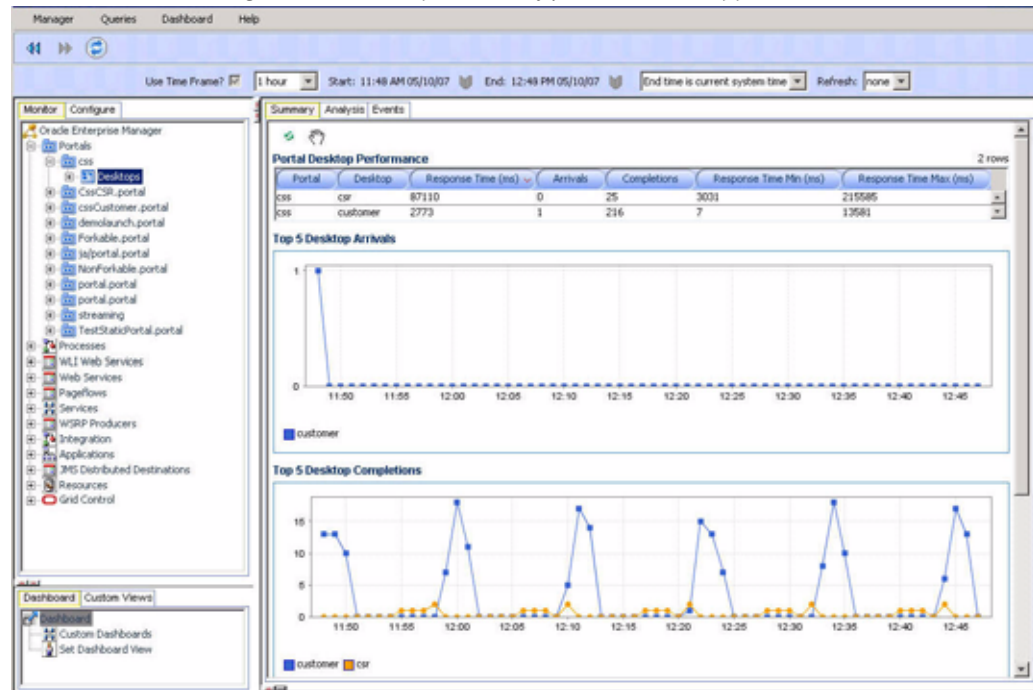
Table 4-3: Portal Level Tab Descriptions (Continued)

Tab	Description
Pages	The performance summary for all the pages associated with the selected portal.
Portlets	The performance summary for all the pages associated with the selected portal.
Footers	The 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	The SLO violation events associated with the selected portal.
Errors/ Exceptions	The errors metrics associated with the selected portal.
Transactions	The transaction events associated with the selected portal and children below.

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. See [Figure 4-6](#) to view the Desktop Summary for the css portal application.

Figure 4-6: Desktop Summary for css Portal Application



This Desktops Summary includes the following metrics:

Table 4-4: 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.

Tip: 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.

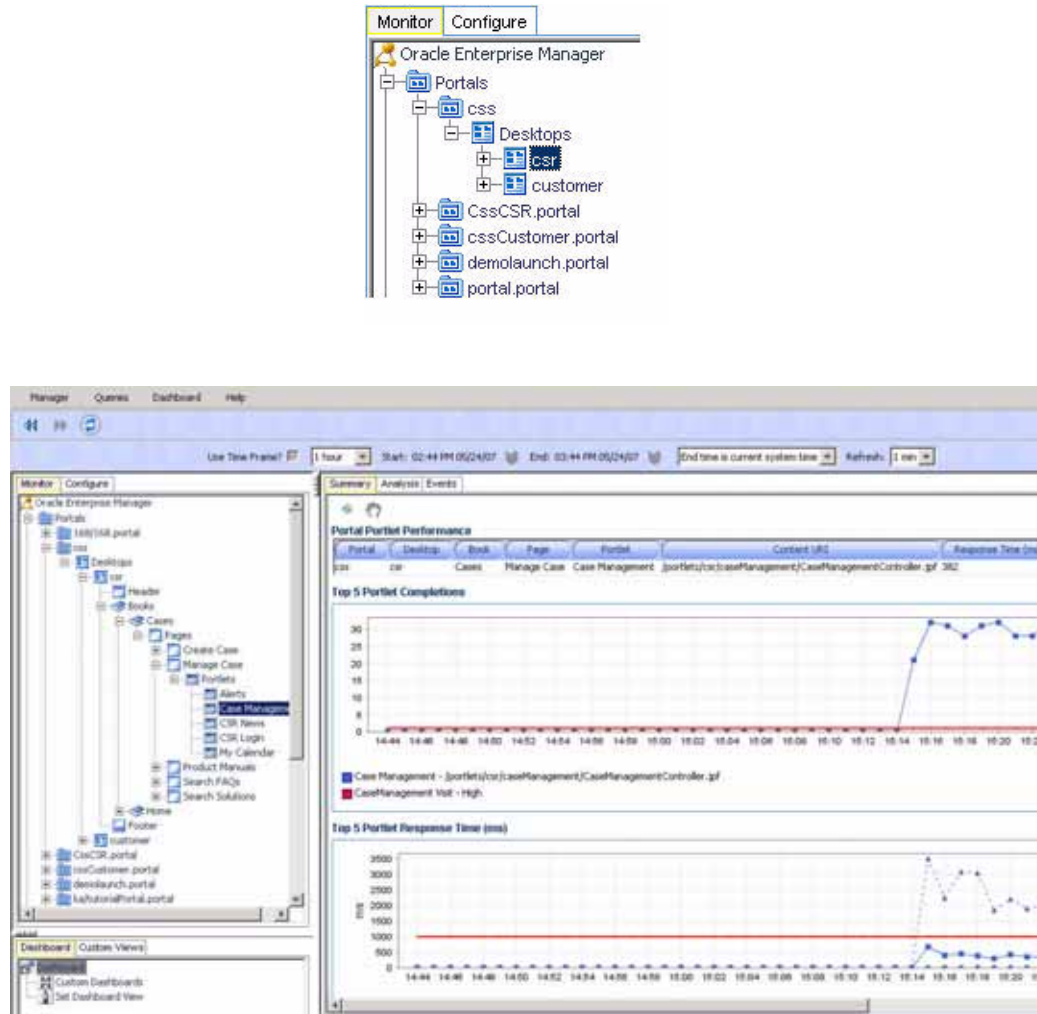
CAMM™ displays these metrics in both table and graph formats.

You can see in our example, there are currently two active desktops for the css portal application. These active desktops, csr and customer, are listed in the table and Plotted in the graphs.

You can drill down further to specific desktop by expanding the Desktops node. Again, clicking on the + icon expands the tree view for you.

[Figure 4-7](#) shows an example of the user selecting the csr node in the expanded tree to get more information specific for that desktop. CAMM™ changes information in the Main Display Window to reflect new context - csr desktop.

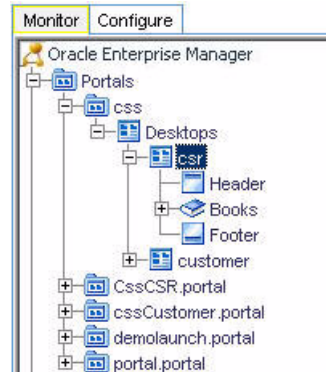
Figure 4-7: csr Desktop Specific Metrics and Thresholds



CAMM™ not only shows the performance metrics associated with a specific node, but it also displays other relevant settings for that node. For the csr desktop, there are two pre-configured Service Level Objectives (SLOs) - CSR Desktop Visit Count Threshold and CSR Desktop Average Response Time Threshold. These SLOs are displayed in these graphs as red lines.

Expand the csr desktop node to see Header, Footer, and Books. You can see detailed information for these components by clicking on the appropriate nodes. See [Figure 4-8](#) for an example of the expanded csr desktop tree.

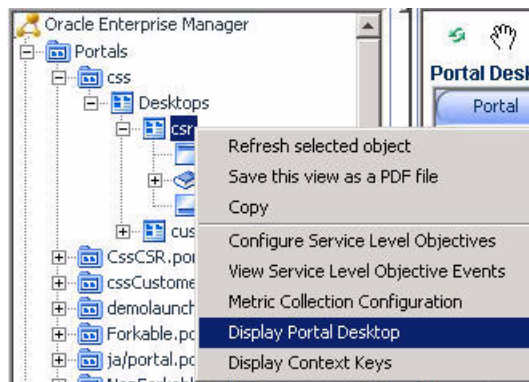
Figure 4-8: Expanded csr Desktop Tree



Display Portal Desktop - Desktop Structure Viewer

One of CAMM™'s unique capabilities 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.

Figure 4-9: Desktop Structure Viewer

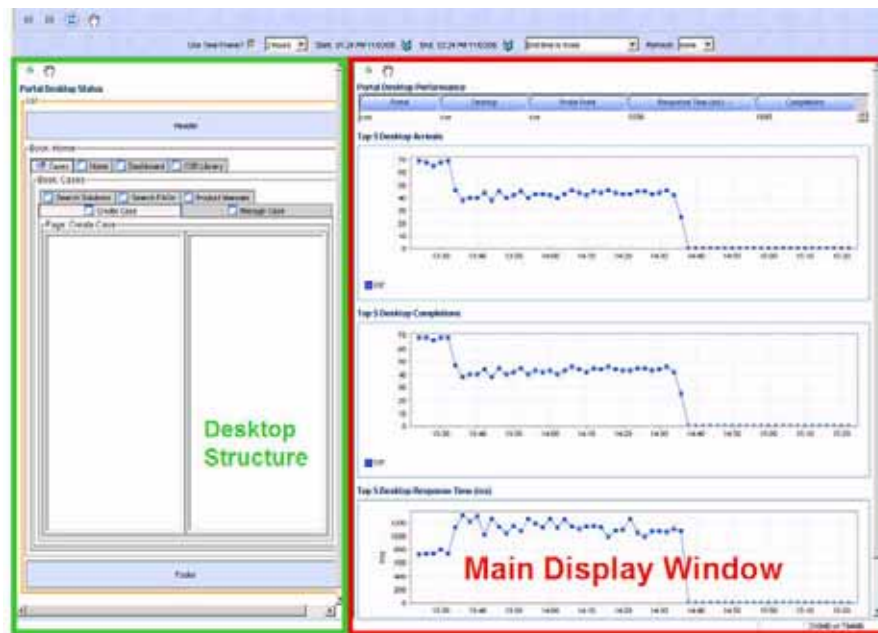


After the Desktop Structure Viewer appears, you can navigate through the portal desktop structure by clicking on the appropriate book, page, or portlet. The ability to see portal desktop structure using the same perspective as portal end-users is a unique value especially for the IT support staff.

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 panels. The panel on the left is the Desktop Structure panel. This panel allows you to graphically navigate the portal desktop. The panel 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 panel. As you navigate through the portal desktop and click different components, the Main Display Window provides information relevant for that selected context. See [Figure 4-10](#).

Figure 4-10: Desktop Structure Viewer



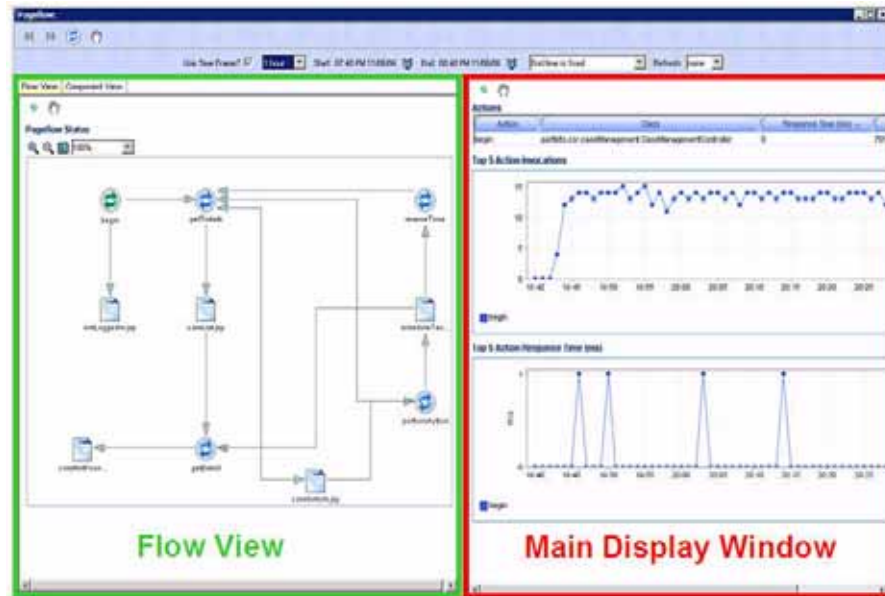
The Main Display Window shows relevant performance metrics for different portal desktop components - desktop, books, pages, and portlets. We will discuss these metrics further in later sections.

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

To activate the Pageflow Viewer, double-click on the interested portlet. As an example, double-click on the Case Management portlet on the Manage Case page. By looking at the information displayed in the Portal portlet performance table for Case Management, you know it is using a pageflow control called CaseManagementController.

Double-click on the Case Management portlet in the Desktop Structure panel to open the appropriate pageflow in the Pageflow Viewer. [Figure 4-11](#) is a screen shot of the CaseManagementController page flow.

Figure 4-11: Pageflow Viewer with Flow View



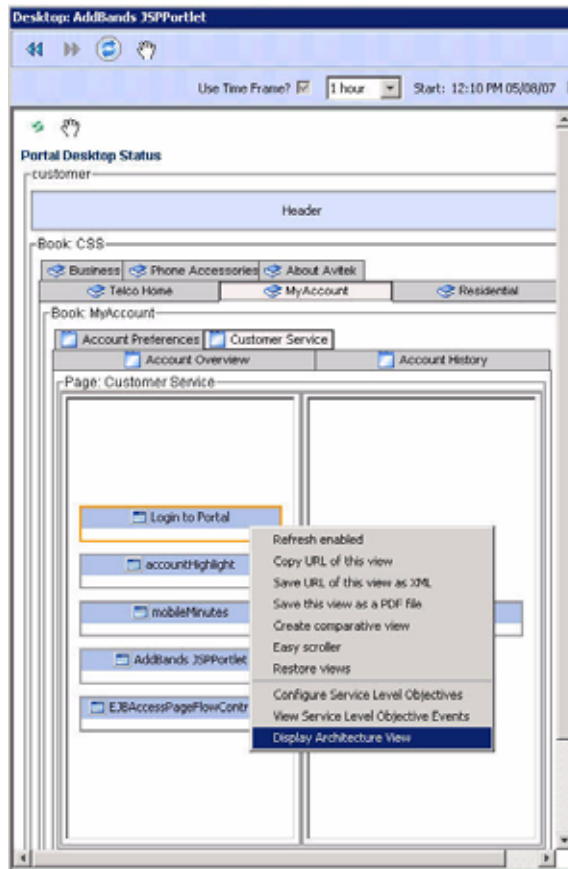
Portlet Drill Down

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

1. Select Consumer Portlets under the WSRP Producers node. See [Figure 4-80](#).
2. Double-click on a consumer portlet name to see the Portal Desktop Status page. See [Figure 4-84](#).

3. In the Portal Desktop Status window right-click on a service box to select Display Architecture View.

Figure 4-12: Drill Down Portlet to Access Architecture View

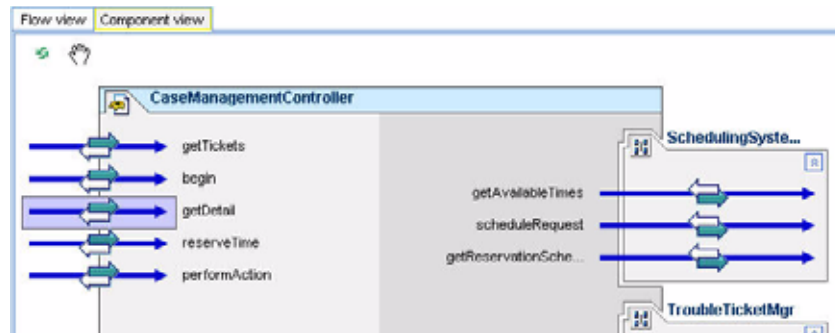


4. See [Drill Down - Bottleneck Analysis](#) on how to use the architecture view.

Pageflow Viewer

[Figure 4-11](#) shows the Pageflow Viewer and its two panels. The panel on the right is the Main Display Window. The Main Display Window shows information corresponding to the item selected in the left panel. The left panel 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.

Figure 4-13: Flow or Component View in Pageflow Viewer



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

Figure 4-14: Pageflow Viewer with Component View



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

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. [Figure 4-15](#) shows the Books Summary for the csr portal desktop.

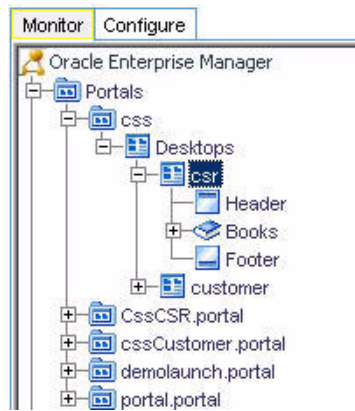
This Books Summary includes the following metrics:

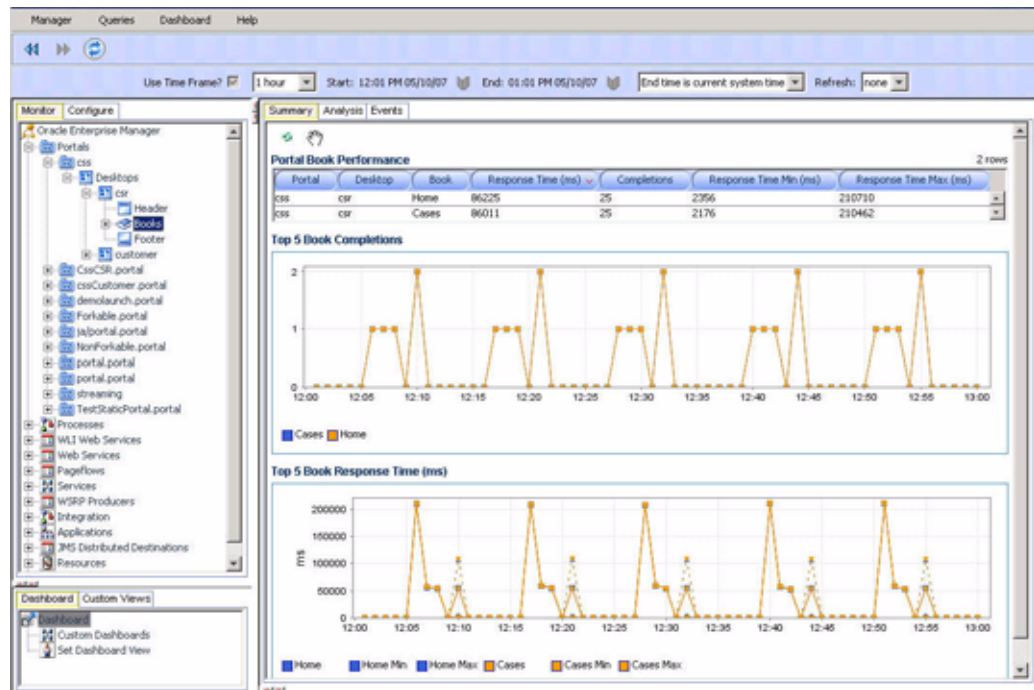
Table 4-5: Book Summary Metrics

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.

Figure 4-15: csr Desktop Expanded to Show Books Node

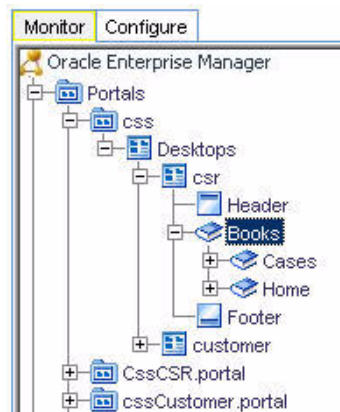




You can see in our example, there are currently two active books for the csr portal desktop. These active books, Cases and Home, are listed in the table and plotted in the graphs.

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

Figure 4-16: Expand Books to View Cases



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

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. [Figure 4-17](#) shows the Pages Summary for the Cases book.

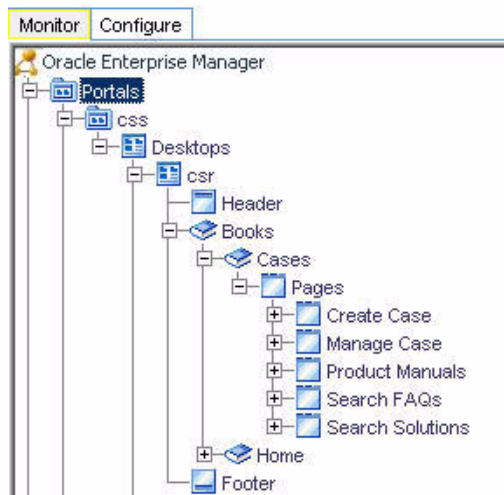
This Pages Summary includes the following metrics:

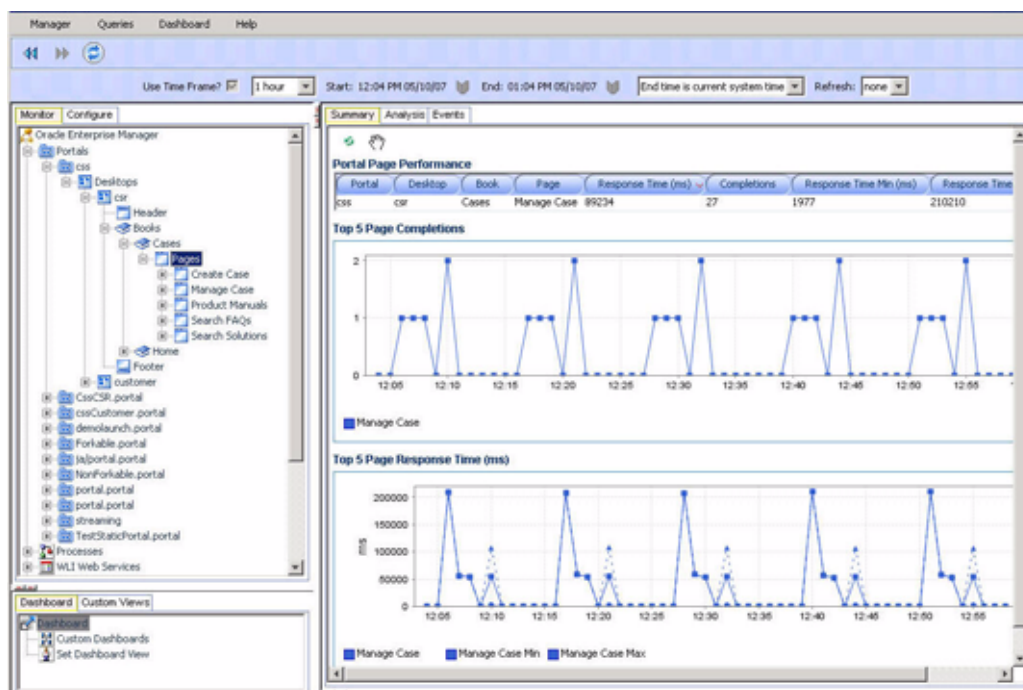
Table 4-6: Pages Summary Metrics

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.

Figure 4-17: Cases Book Node Expanded to Show Pages Node



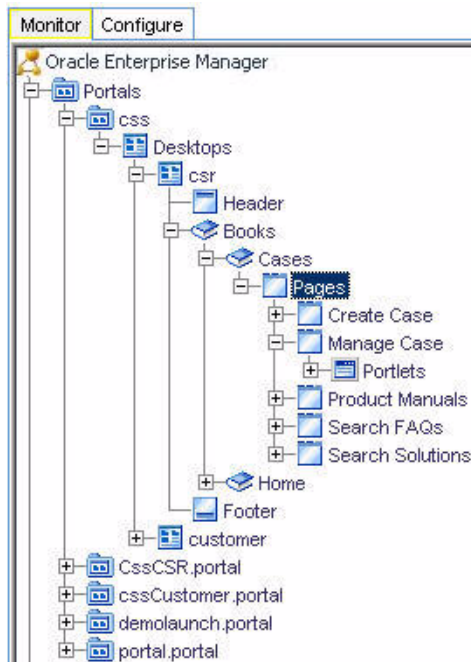


You can see in this example, there is currently one active page for the Cases book. The active page, Manage Case, is listed in the table and plotted in the graphs.

You can drill down further to a specific page by expanding the Pages node. Click the + icon to expand the tree view.

[Figure 4-18](#) shows an example of when a user selects the Manage Case node in the expanded tree to get more information specific to that Page.

Figure 4-18: Expanding the Manage Case Page Reveals the Portlets Node



As shown in [Figure 4-18](#), expanding the Manage Case page reveals the next level of components - Portlets.

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. [Figure 4-19](#) shows the Portlets Summary for the Manage Case page.

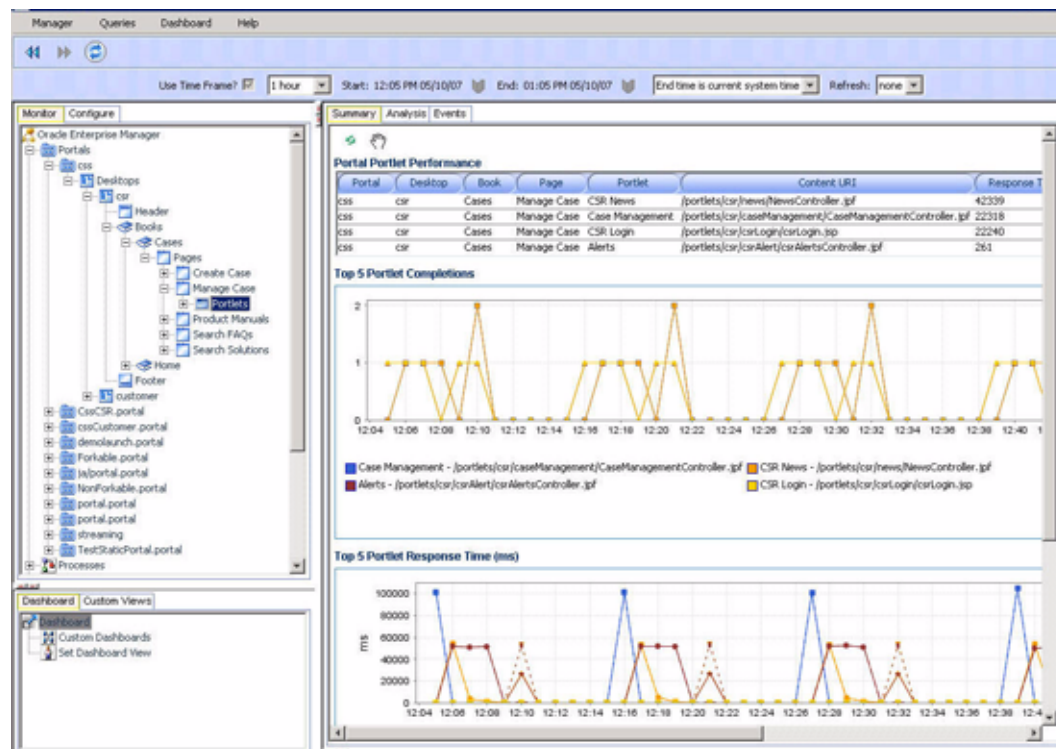
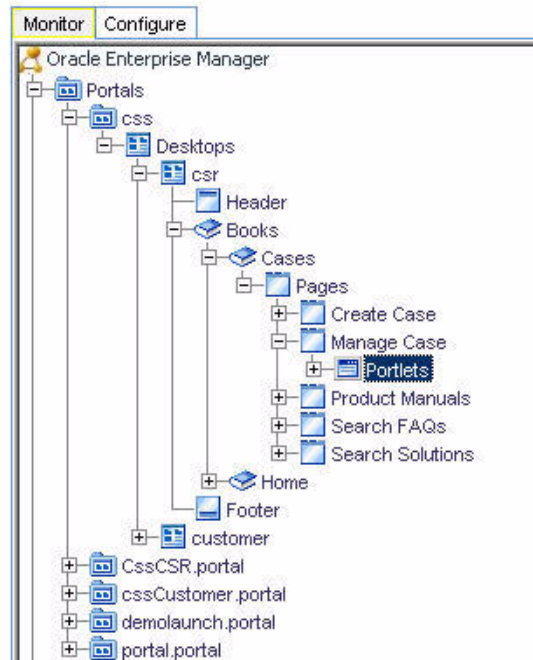
This Portlets Summary includes the following metrics:

Table 4-7: 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.

Figure 4-19: Manage Case Page Node Expanded to Show Portlets Node

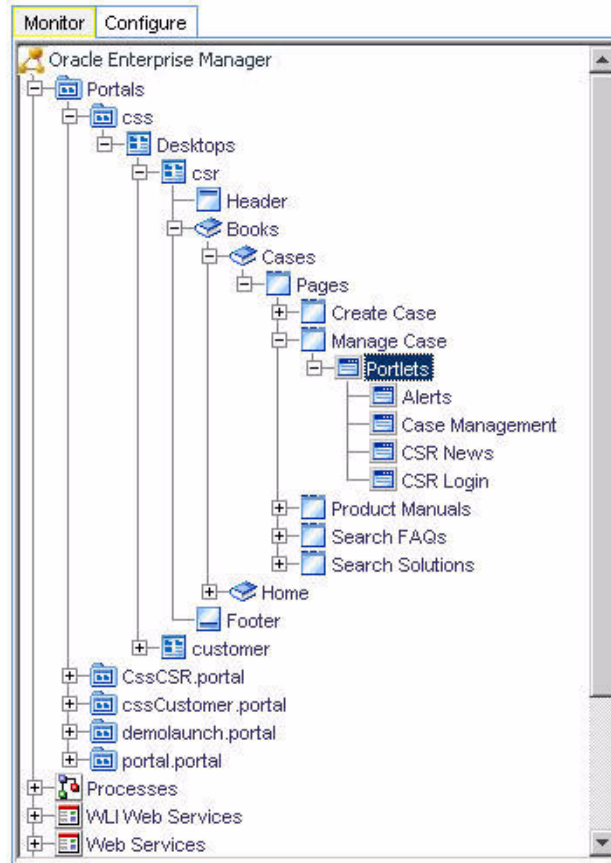


You can see in this example, there are currently four active portlets for the Manage Case page. These active portlets are listed in the table and plotted in the graphs.

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

[Figure 4-20](#) shows when a user selects the Portlets node in the expanded tree to get more information specific to that Page.

Figure 4-20: Expanding the Portlets Node Reveals the Specific Portlet Nodes



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 panel.

For WebSphere® Portal, this hierarchy contains the following:

Table 4-8: 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.

We will discuss each component in more detail later in this document.

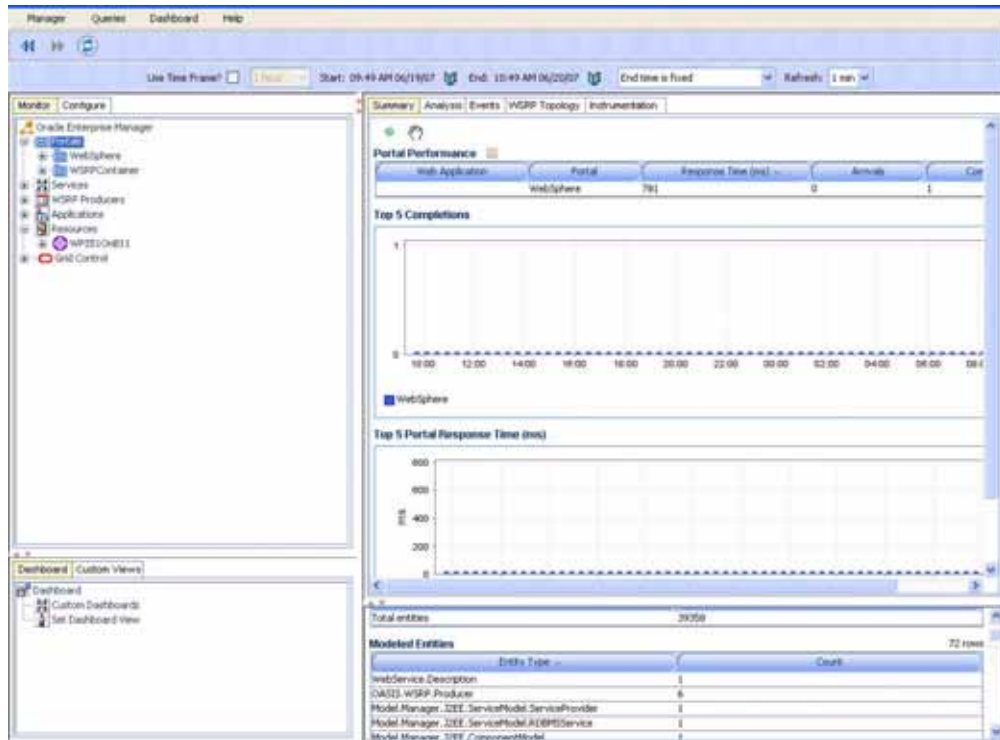
When you click the Portals node under the Oracle™ Tree, CAMM™ displays summary information on active portal applications. This summary includes the following:

Table 4-9: 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. [Figure 4-21](#) shows a table and corresponding graph for Portal performance of various portal applications.

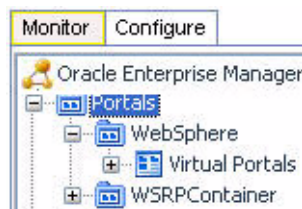
Figure 4-21: WebSphere® Portal Performance Table and Graphs



For the other three metrics, CAMM™ shows the information in graph format.

When you click the + icon next to the Portals node, CAMM™ expands the tree to show all managed portal applications currently deployed on the WebLogic® domain. [Figure 4-22](#) shows this configuration with five portals.

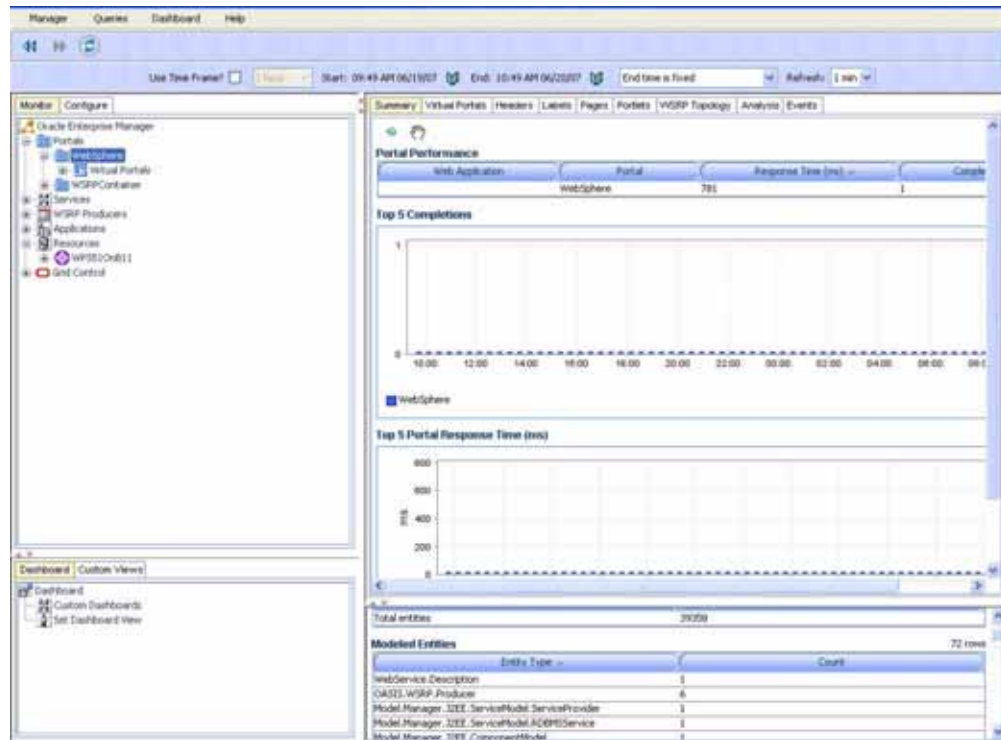
Figure 4-22: WebSphere® Portals Node Expanded



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.

In the [Figure 4-23](#) example, a user selects the *WebSphere* portal application under the Portals node. The Main Display Window now only shows information specific to WebSphere portal application.

Figure 4-23: Display WebSphere Portal Specific Metrics



At the Portal level, you can navigate to different levels of the portal application by using different tabs. [Figure 4-24](#) shows the tabs available for portal level nodes. Use the tabs available to quickly access lower level components.

Figure 4-24: WebSphere® Portal Level Nodes Tabs



The following is a list of the tabs available for portal level nodes and their descriptions.

Table 4-10: Portal Level Tab Descriptions

Tab	Description
Summary	The 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	The SLO violation events associated with the selected portal.

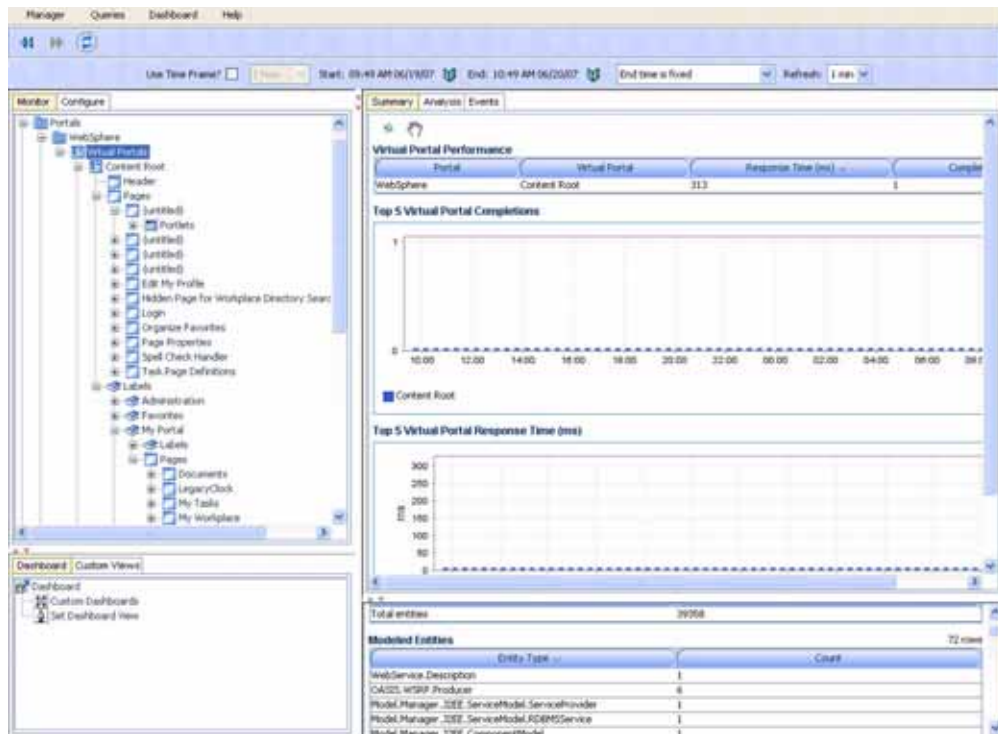
Table 4-10: Portal Level Tab Descriptions (Continued)

Tab	Description
WSRP Topology	View WSRP consumer-producer relationships and WSRP deployment topology.
Errors/ Exceptions	The 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 to Instrumentation for more detail.

Virtual Portals

Expand a particular portal application further to reach the Virtual Portals node. By selecting this node, CAMM™ provides a list of currently active portals associated with that portal application. See [Figure 4-25](#) to view the Summary for the WebSphere portal application.

Figure 4-25: Desktop Summary for WebSphere Portal Application



This Summary includes the following metrics:

Table 4-11: Virtual Portals Summary Metrics

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

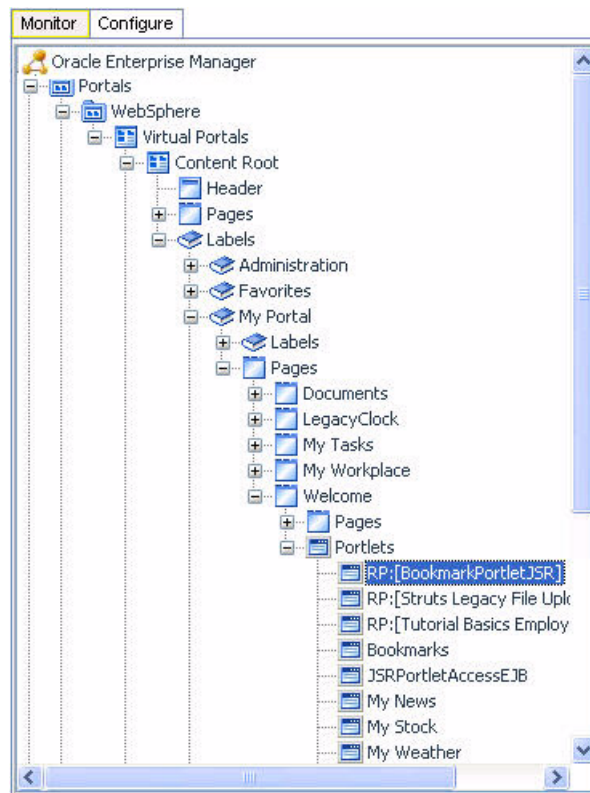
CAMM™ displays these metrics in both table and graph formats.

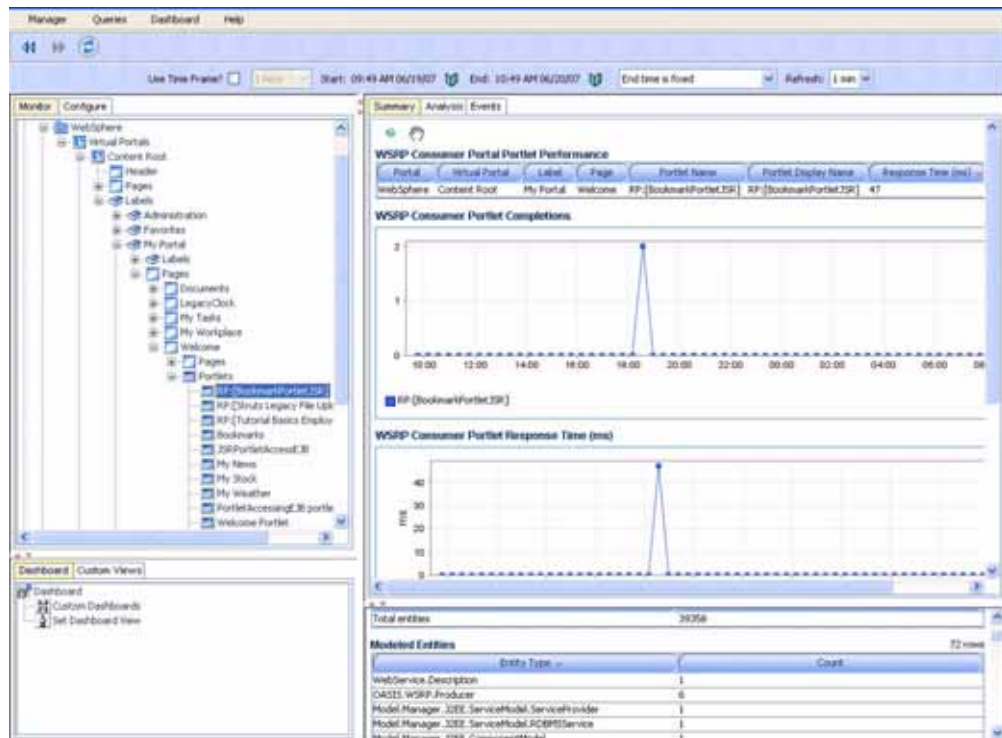
You can see in our example, there are currently one active portal for the WebSphere portal application. The Content Root is listed in the table and Plotted in the graphs.

You can drill down further to specific portlets by expanding the Content Root node. Again, clicking on the + icon expands the tree view for you.

Figure 4-26 shows an example of the user selecting the csr node in the expanded tree to get more information specific for that portal. CAMM™ changes information in the Main Display Window to reflect new context - portlet.

Figure 4-26: Portlet Specific Metrics and Thresholds



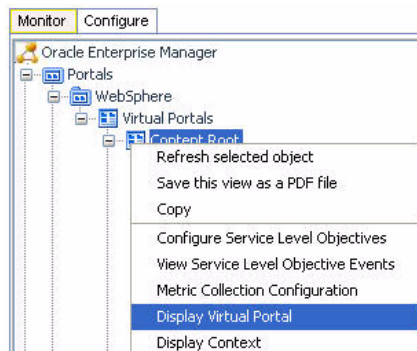


CAMM™ not only shows the performance metrics associated with a specific node, but it also displays other relevant settings for that node.

Display Virtual Portal - Structure Viewer

One of CAMM™'s unique capabilities 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.

Figure 4-27: Display Virtual Portal Structure Viewer

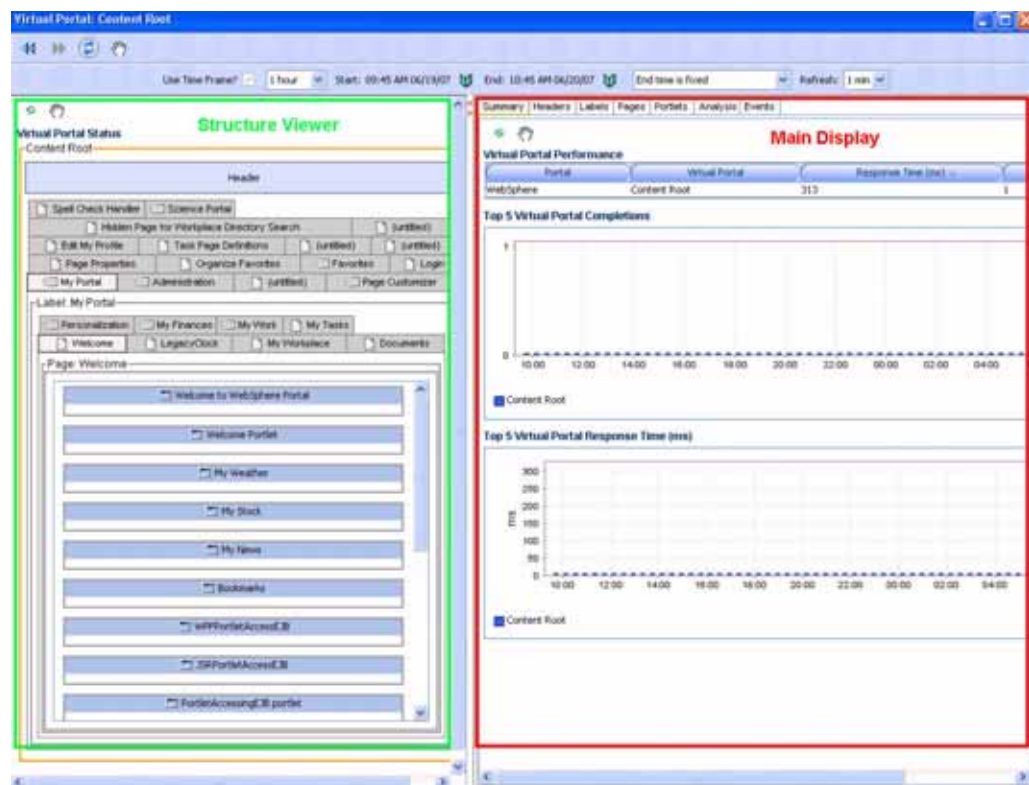


After the Structure Viewer appears, you can navigate through the portal structure by clicking on the appropriate header. The ability to see the portal structure using the same perspective as portal end-users is a unique value especially for the IT support staff.

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 panels. The panel on the left is the Structure Viewer panel. This panel allows you to graphically navigate the portal desktop. The panel 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 panel. As you navigate through the desktop and click different components, the Main Display Window provides information relevant for that selected context. See [Figure 4-28](#).

Figure 4-28: Virtual Portal Structure Viewer



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

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. [Figure 4-29](#) shows the Pages Summary for the Content Root virtual portal.

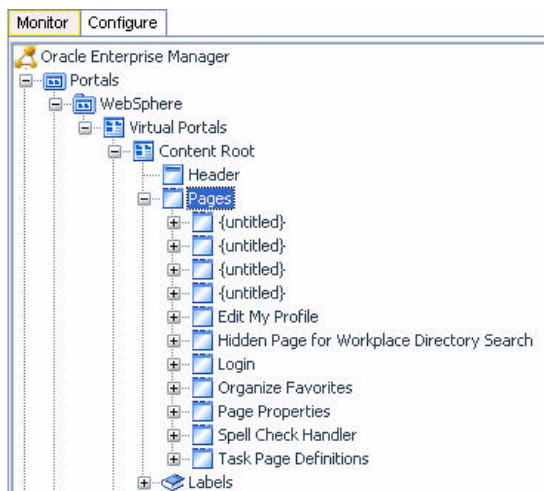
This Pages Summary includes the following metrics:

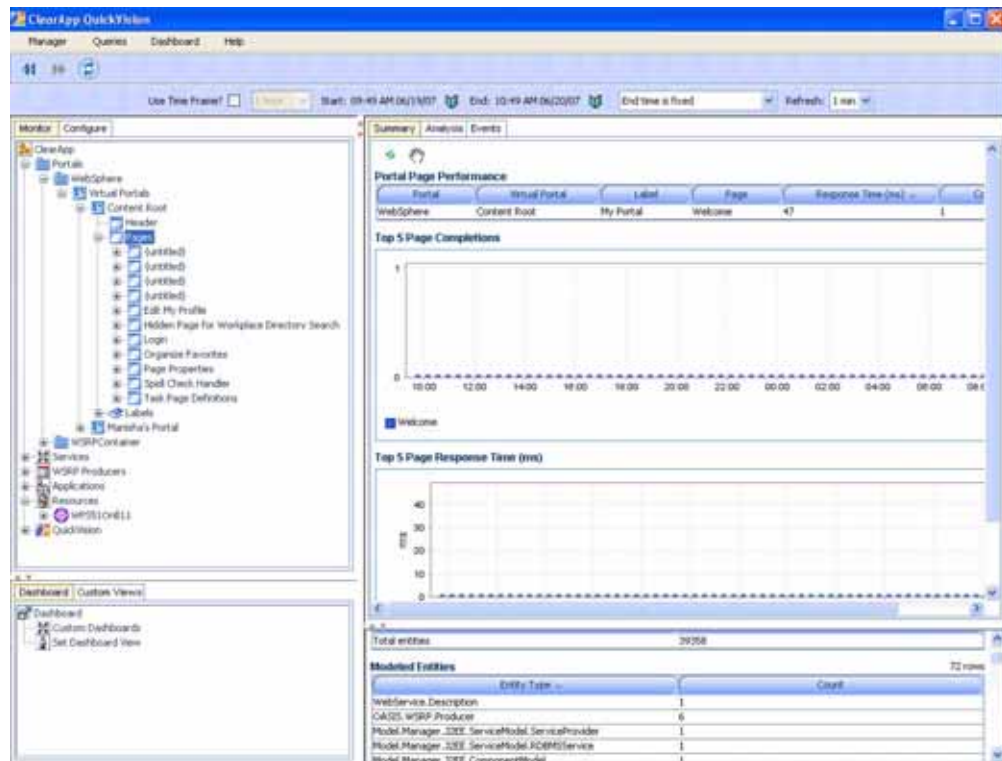
Table 4-12: Pages Summary Metrics

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.

Figure 4-29: Virtual Portal Node Expanded to Show Pages Node





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. [Figure 4-30](#) shows the Portlets Summary for the Manage Case page.

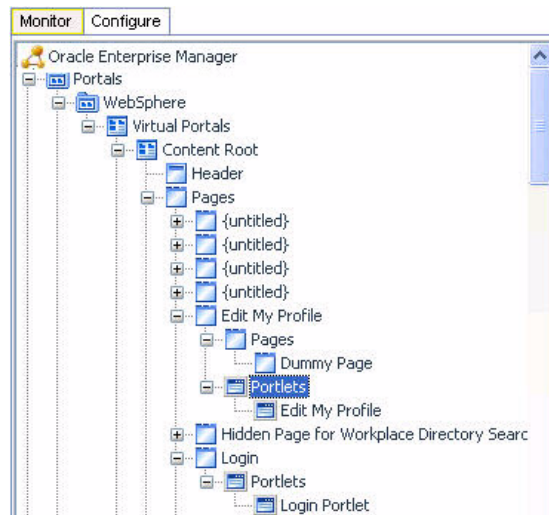
This Portlets Summary includes the following metrics:

Table 4-13: 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.

Figure 4-30: Content Root Node Expanded to Show Portlets Node



You can see in this example, there are currently 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 + icon to expand the tree view.

[Figure 4-30](#) shows when a user selects the Portlets node in the expanded tree to get more information specific to that Page.

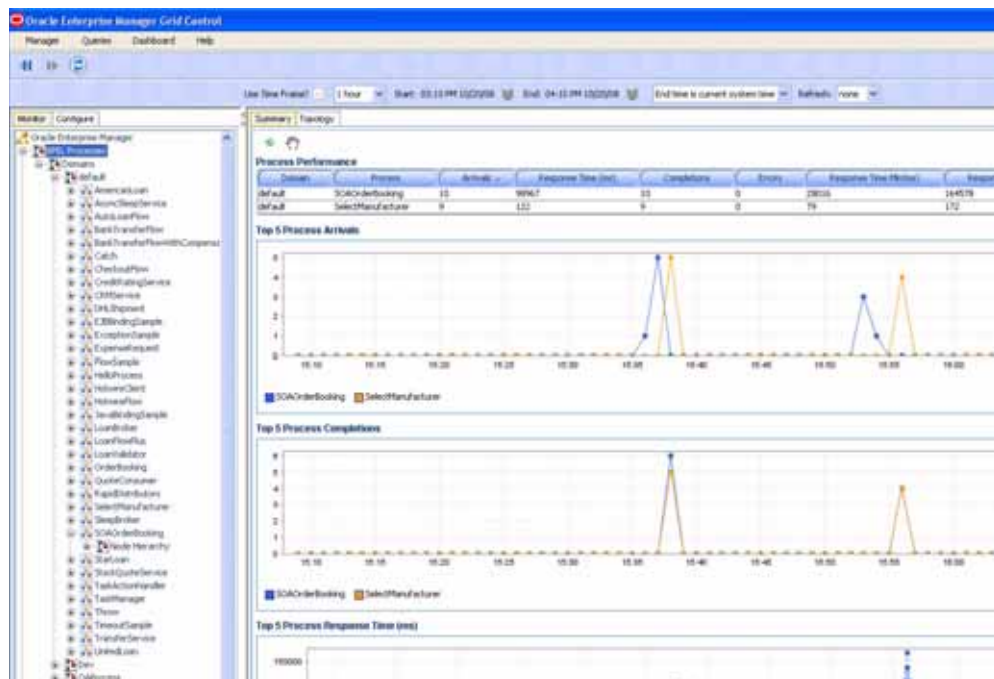
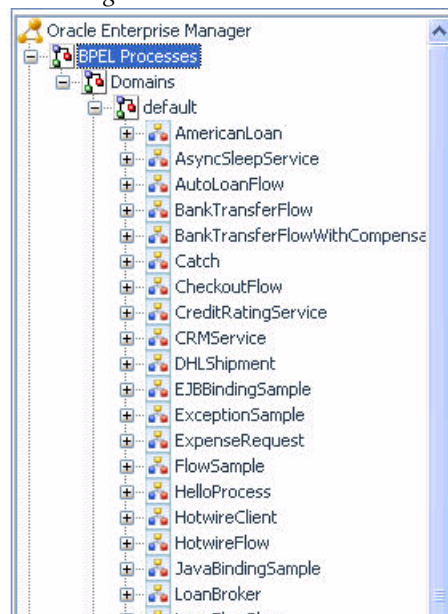
Oracle® BPEL Processes

The BPEL Processes node under Oracle™ 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 are 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 panel.

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

Figure 4-31: BPEL Process Summary



The BPEL Process Summary includes the following:

Table 4-14: BPEL Process Summary Metrics

Metrics	Description
Domain	Name of the OC4J domain container.

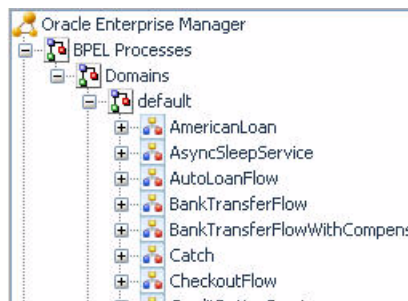
Table 4-14: BPEL Process Summary Metrics (Continued)

Metrics	Description
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 + 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. [Figure 4-32](#) shows that the selected *default* domain has five active BPEL processes.

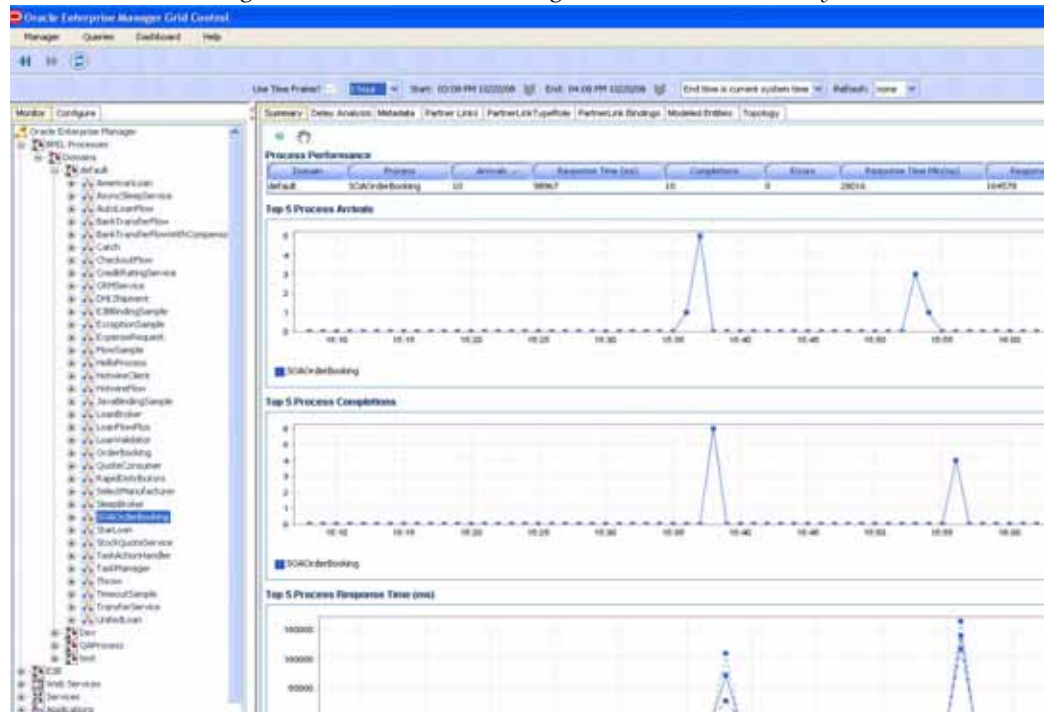
Figure 4-32: BPEL Processes Node Expanded



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.

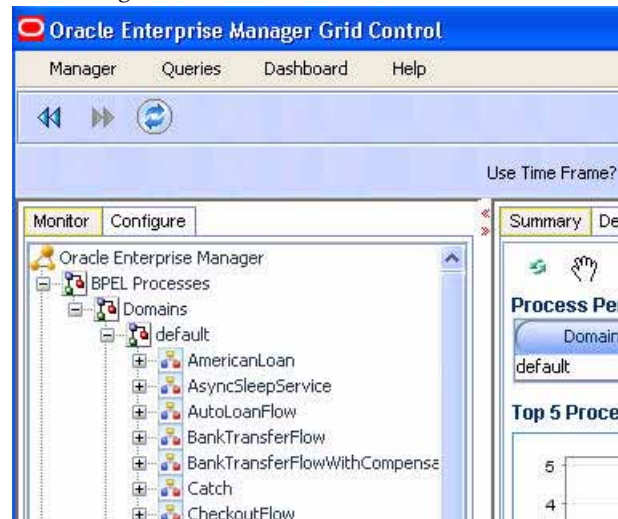
Figure 4-33 shows when a user selects the SOAOrderBooking BPEL process under the BPEL Processes node. The Main Display Window now only shows information specific to SOAOrderBooking BPEL process.

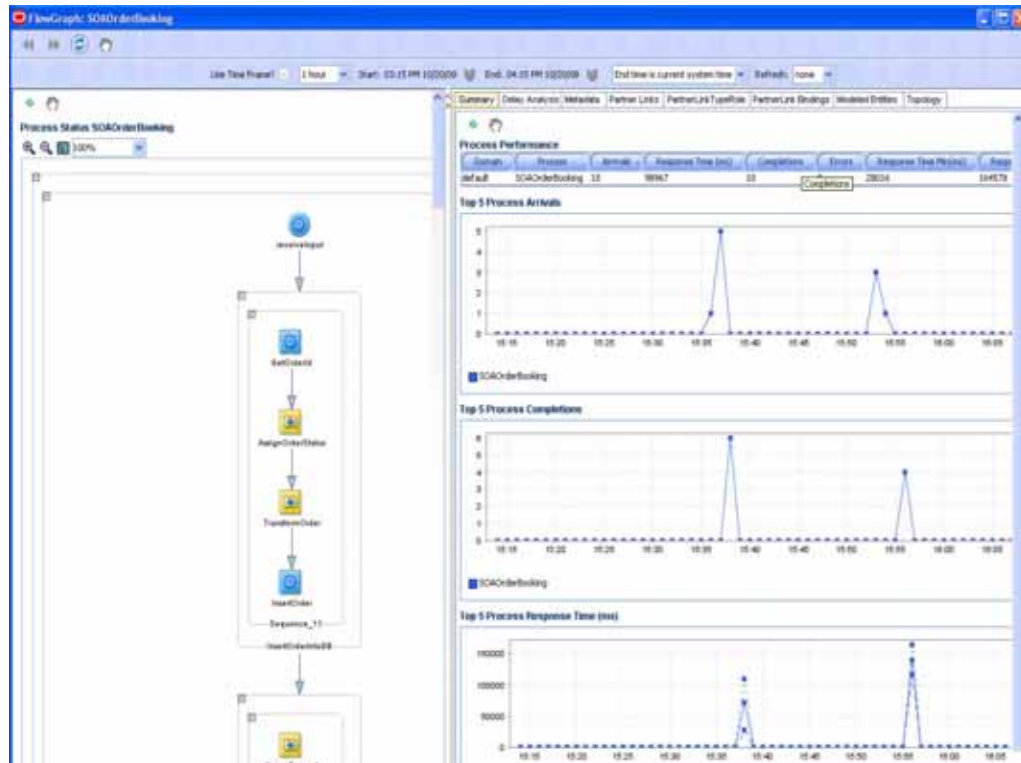
Figure 4-33: SOAOrderBooking BPEL Process Summary



To see the BPEL process work flow associated with the SOAOrderBooking BPEL process, select SOAOrderBooking 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 Figure 4-34.

Figure 4-34: Functional Work Flow View





See [Table 4-15](#) for BPEL Functional View summary.

Table 4-15: 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 find out which BPEL process node is running slowly and whether there are errors.

Besides the summary, there are seven other views available for the SOAOrderBooking node:

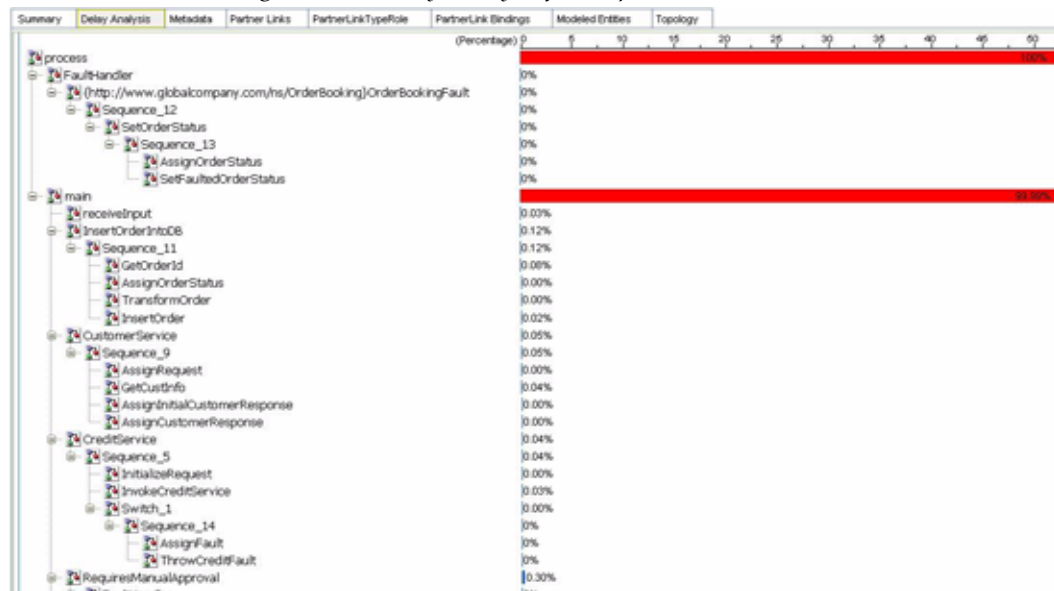
- Delay Analysis view
- Metadata view
- Partner Links view
- Partner Link Type Role view
- Partner Link Bindings view
- Modeled Entities view
- Topology view

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

Delay Analysis View

[Figure 4-35](#) is the 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.

Figure 4-35: Delay Analysis for RequestQuote Process



Metadata View

The Metadata view ([Figure 4-36](#)) displays three 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.

Figure 4-36: Metadata View

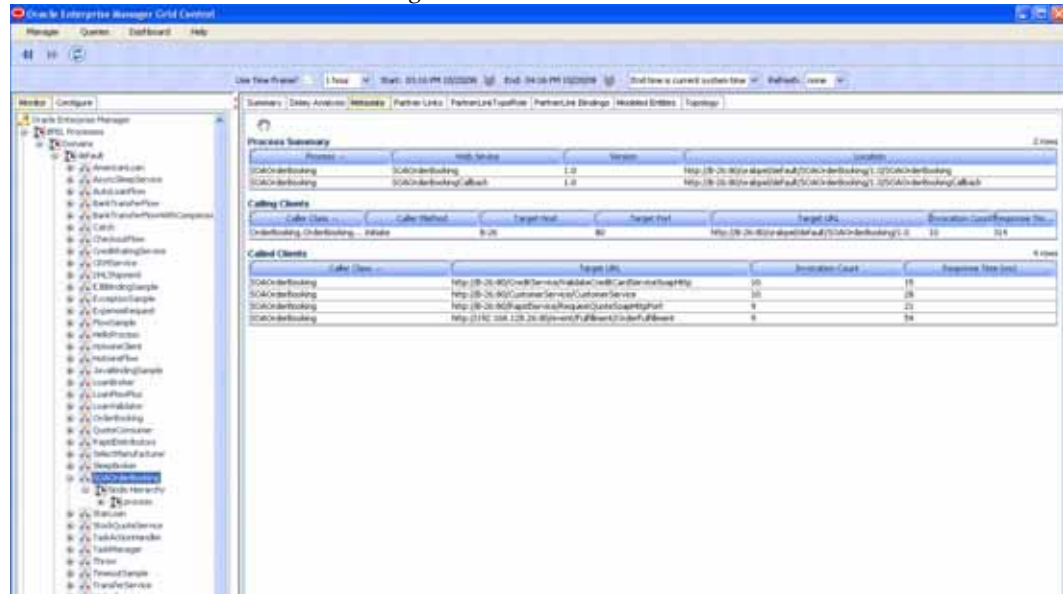


Table 4-16: Metadata View Summary

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	The class name for the caller class that is calling the BPEL process
Caller Table - Caller Method	The class method for the caller class that is calling the BPEL process
Caller Table - Target Host	The target host that the caller class targeted to instantiate the BPEL process
Caller Table - Target Port	The target port that the caller class targeted to instantiate the BPEL process

Table 4-16: Metadata View Summary (Continued)

Column / Metric	Description
Caller Table - Target URL	The target URL that the call class targeted to instantiate the BPEL process
Caller Table - Invocation Count	The number of invocations of the BPEL process instantiated by the caller class.
Caller Table - Response Time	The average response time of the BPEL process instantiated by the caller class.
Called Clients Table - Called Class	The class name of the class that was called by the BPEL process
Called Clients Table - Target URL	The target URL of the class that was called by the BPEL process
Called Clients Table - Invocation Count	The number of invocations made from the BPEL Process to the called class.
Called Clients Table - Response Time	The response time of the called class.

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.

Figure 4-37: Partner Links View

Partner Link	My Role	Partner Role	Partner Link Type
CreditCardService	N/A	ValidateCreditCard_Role	http://www.globacompany.com/credit/validateCreditCard_?L
CustomerService	N/A	CustomerService_Role	http://www.globacompany.com/customer/CustomerService_?L
DecisionService	N/A	DecisionService_Role	http://www.globacompany.com/order/booking/DecisionService/DecisionService_?L
NotificationService	N/A	NotificationServiceProvider	http://links.oracle.com/bpel/adaptor/NotificationService/NotificationServiceLink
Order	N/A	Order_role	http://links.oracle.com/bpel/adaptor/Order/Order_?L
OrderFulfillment	N/A	status_getProvider	http://www.globacompany.com/fulfillment/insert_get?L
OrderSequence	N/A	OrderSequence_role	http://links.oracle.com/bpel/adaptor/OrderSequence/OrderSequence_?L
OrderStatus	N/A	OrderStatus_role	http://links.oracle.com/bpel/adaptor/OrderStatus/OrderStatus_?L
ReportService	N/A	ReportQuery_Role	http://www.globacompany.com/report/ReportQuery_?L
SelectService	SelectServiceRequester	SelectServiceProvider	http://www.globacompany.com/select/service/SelectService_?L
TaskService	TaskServiceCalloutListener	TaskService	http://links.oracle.com/bpel/monitor/TaskService/TaskService
Item	SOAPMessageProvider	SOAPMessageReceiver	http://www.globacompany.com/order/booking/SOAPMessage

Table 4-17: Partner Links View Summary

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

Partner Link Type Role View

Figure 4-38: Partner Link Type Role View

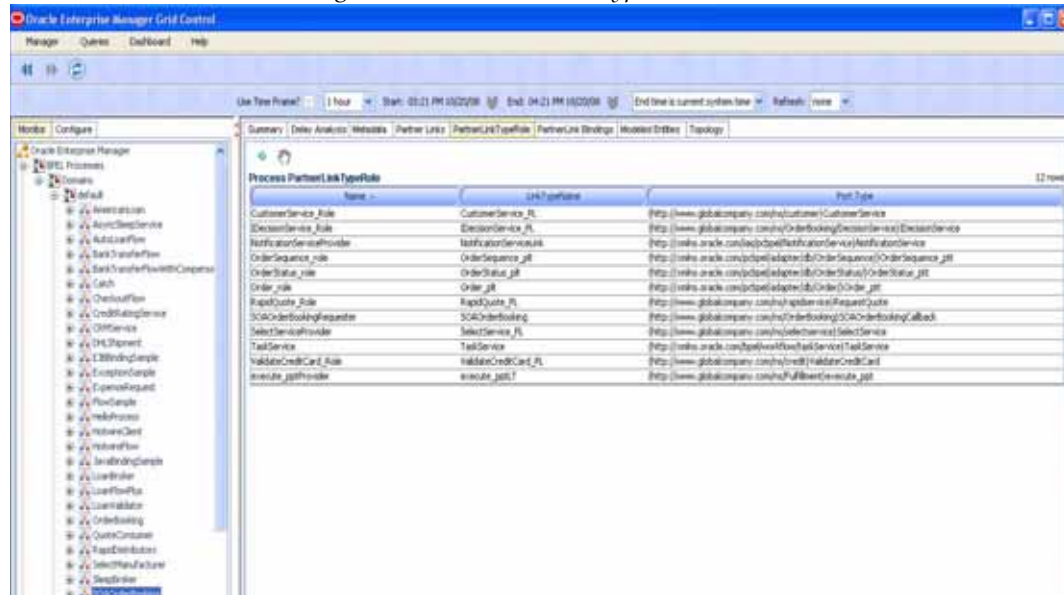


Table 4-18: Partner Link Type Role View Summary

Column / Metric	Description
Name	The name of the partner link
Link Type Name	The category (type) of the partner link.
Port Type	The partner link service URL.

Partner Link Bindings View

The Partner Link Bindings view (Figure 4-39) 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.

Figure 4-39: Partner Link Bindings View

Partner Link Role	Partner Link Type	WebService PortType	WebService Port Namespace ID
CustomerService_Role	CustomerService_FL	CustomerService	http://www.globalcompany.com/nc/customer
ExceptionService_Role	ExceptionService_FL	ExceptionService	http://www.globalcompany.com/nc/OrderBinding/ExceptionService
NotificationServiceRoleProvider	NotificationServiceRole	NotificationService	http://xmlns.oracle.com/bpel/selector/NotificationService
OrderSequence_Role	OrderSequence_FL	OrderSequence_jst	http://xmlns.oracle.com/bpel/selector/OrderSequence
OrderStatus_Role	OrderStatus_FL	OrderStatus_jst	http://xmlns.oracle.com/bpel/selector/OrderStatus
Order_Role	Order_FL	Order_jst	http://xmlns.oracle.com/bpel/selector/Order
RequestQuote_Role	RequestQuote_FL	RequestQuote	http://www.globalcompany.com/nc/requestquote
SOAOrderBindingRequester	SOAOrderBinding	SOAOrderBindingCallBack	http://www.globalcompany.com/nc/OrderBinding
SelectServiceProvider	SelectService_FL	SelectService	http://www.globalcompany.com/nc/selectservice
TaskService	TaskService	TaskService	http://xmlns.oracle.com/bpel/selector/taskService
ValidateCreditCard_Role	ValidateCreditCard_FL	ValidateCreditCard	http://www.globalcompany.com/nc/credit
Invoice_jstProvider	Invoice_jstProvider	Invoice_jst	http://www.globalcompany.com/nc/invoice

Table 4-19: 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

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.

Figure 4-40: Modeled Entities View

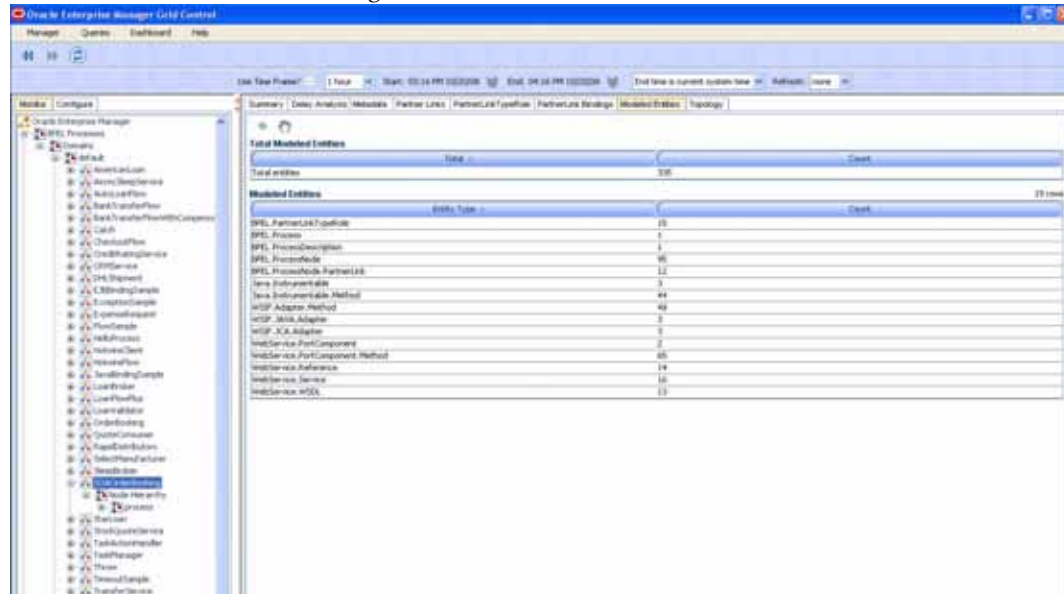


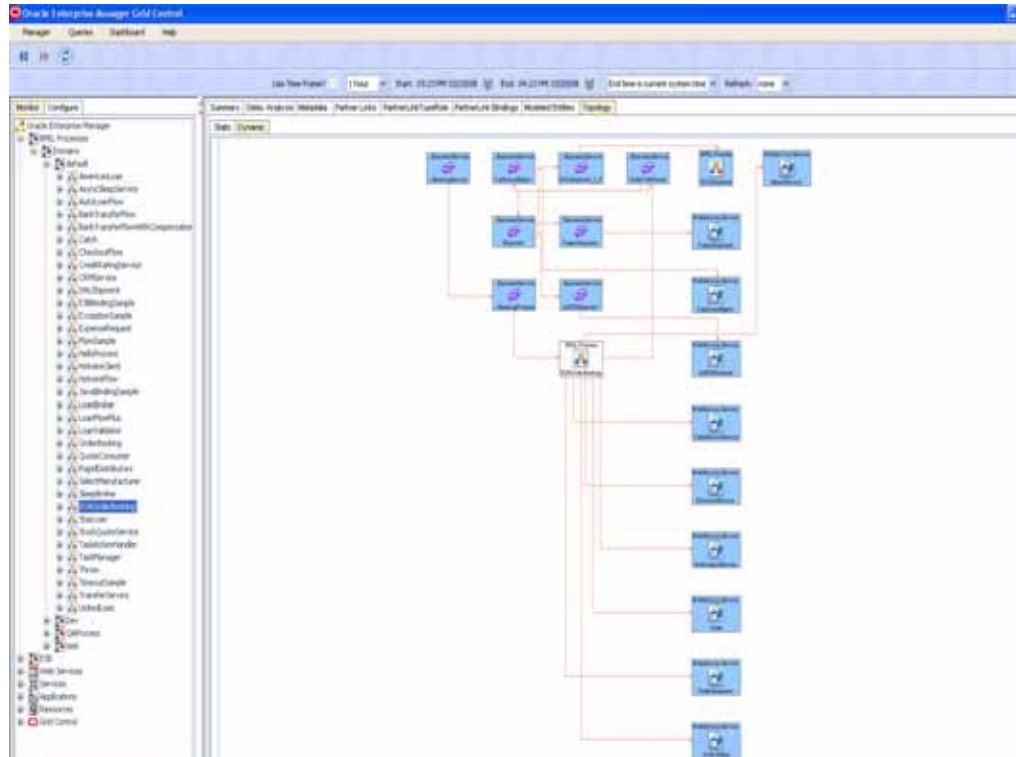
Table 4-20: Modeled Entities Summary

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

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 as can be seen in [Figure 4-41](#) below. The user can toggle between static and dynamic relationship views via the tabs at the top of the Topology pane.

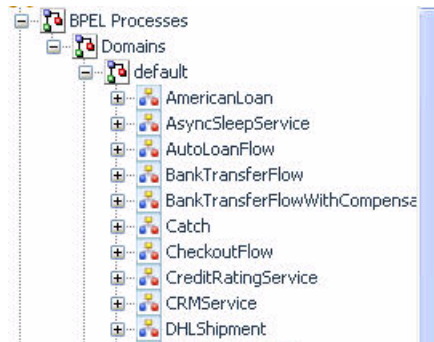
Figure 4-41: BPEL Topology View



Node Hierarchy

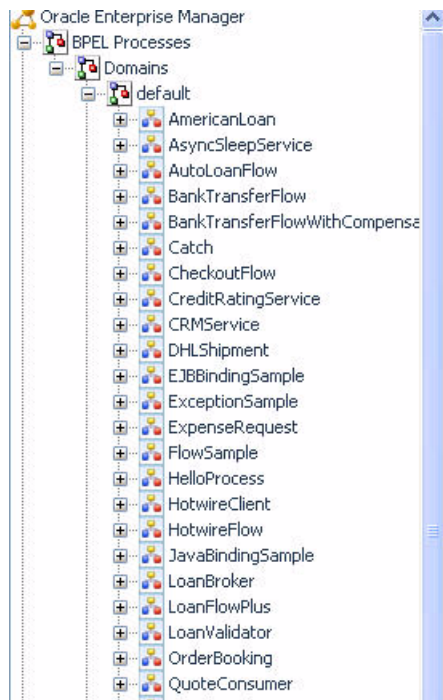
Expanding a particular BPEL process further, the first item you see is the Node Hierarchy node. By selecting the Node Hierarchy node, CAMM™ provides a list of nodes associated with the specific process. [Figure 4-42](#) shows the Node Hierarchy Summary for the SOAOrderBooking process.

Figure 4-42: SOAOrderBooking BPEL Process Node Hierarchy Summary



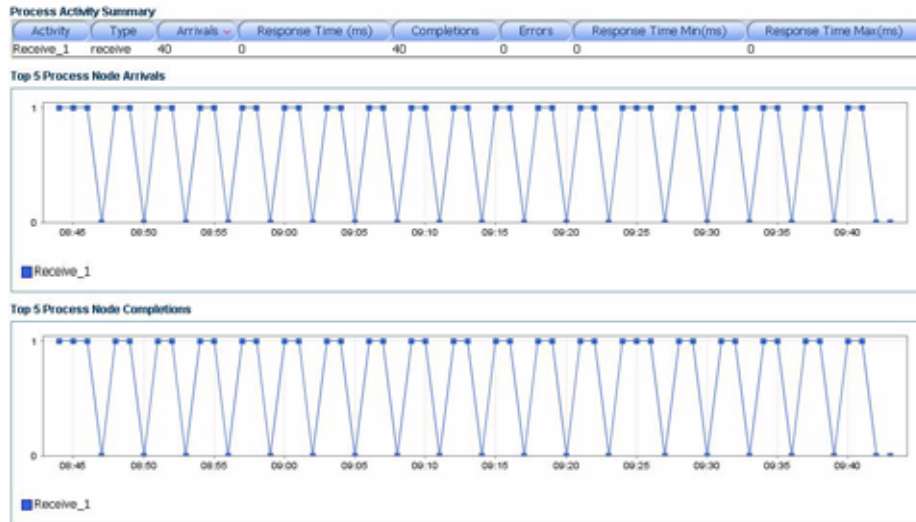
When you click the + icon next to a specific Node Hierarchy node, CAMM™ expands the tree to show BPEL process nodes in the Node Hierarchy. [Figure 4-43](#) shows the entire node hierarchy for SOAOrderBooking process.

Figure 4-43: Drill Down to See the Entire BPEL Node Hierarchy



Click an individual BPEL process node to see the load and performance of the selected node in the Main Display Window. See [Figure 4-44](#) for an example of a specific performance information.

Figure 4-44: Process Node Specific Performance Information



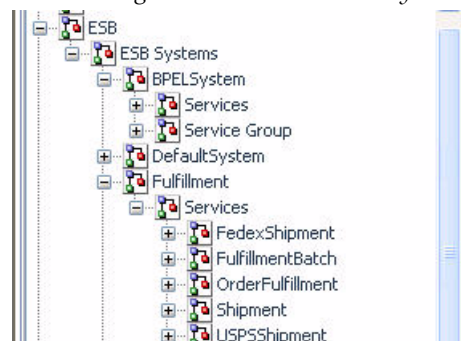
The BPEL process node information also includes the name of the method invoked. This information is displayed as part of the summary table at the top of the main view window.

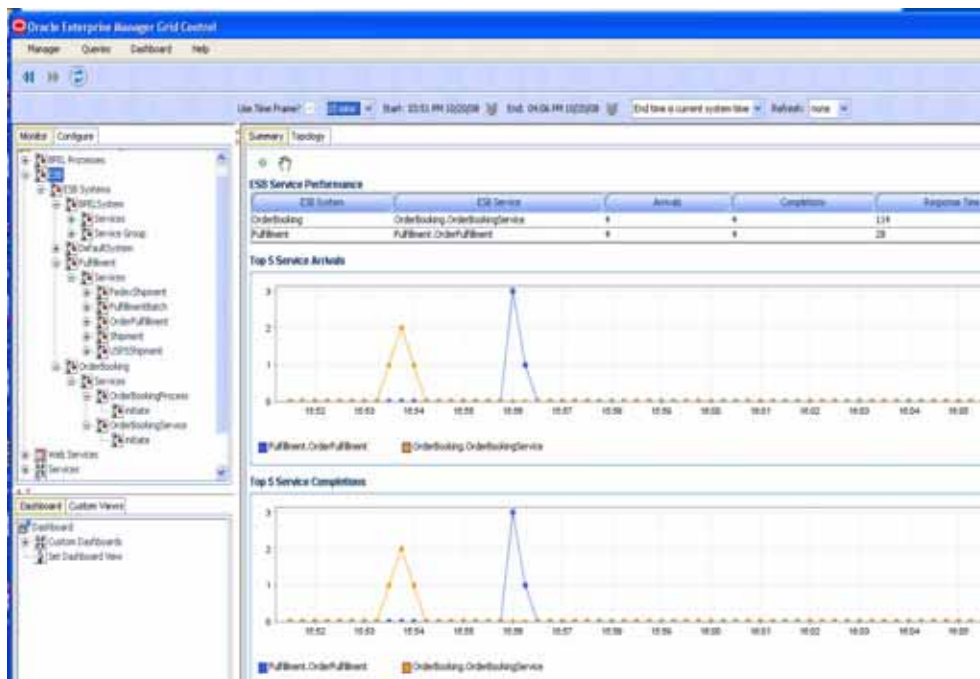
Oracle[®] ESB

The Oracle ESB node under the 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.

Figure 4-45: ESB Summary





The ESB Summary includes the following:

Table 4-21: ESB Summary Metrics

Metrics	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. Graphical representations of two metrics, Running Instances and Average Execution Time, are displayed below the table.

When you click the + 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. [Figure 4-46](#) shows that the selected *ESB Systems* list has three active ESB systems.

Figure 4-46: ESB Systems Node Expanded



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 (see [Figure 4-47](#)) relevant to this new context.

Figure 4-47: ESB Systems Topology

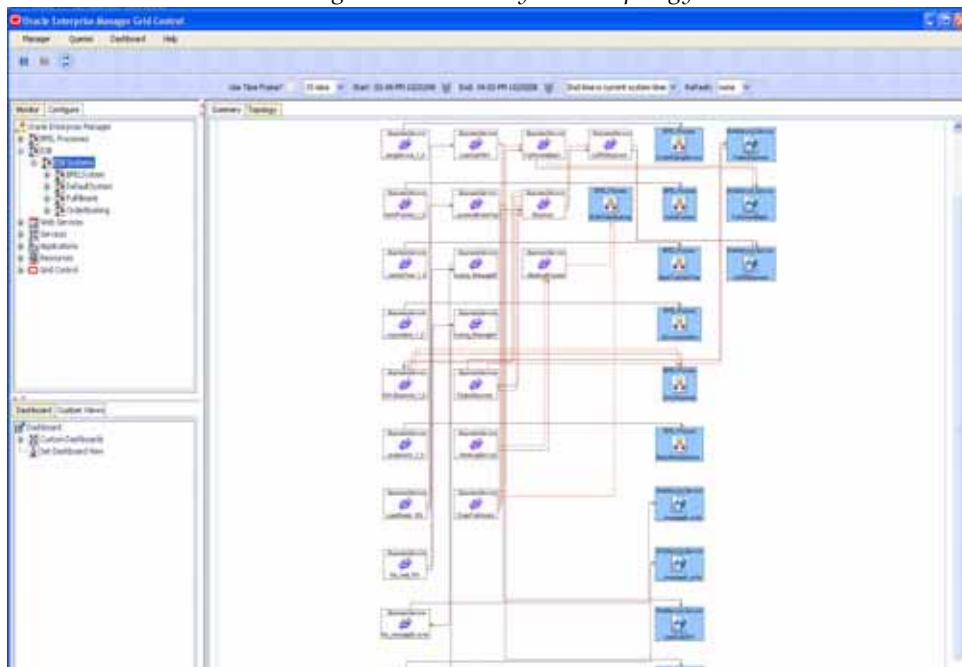
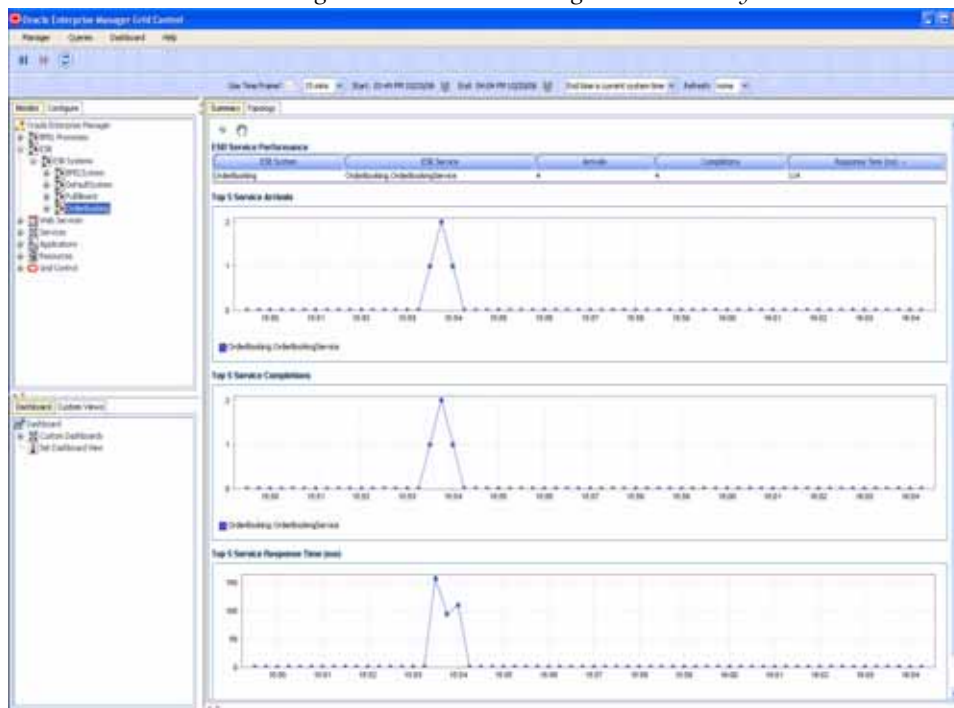


Figure 4-48 shows when a user selects the OrderBooking ESB system under the ESB Systems node. The Main Display Window now only shows information specific to OrderBooking ESB specific details

Figure 4-48: OrderBooking ESB Summary



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

Besides the summary, there are seven other views 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.

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.

Figure 4-49: OrderBooking ESB Service Details View

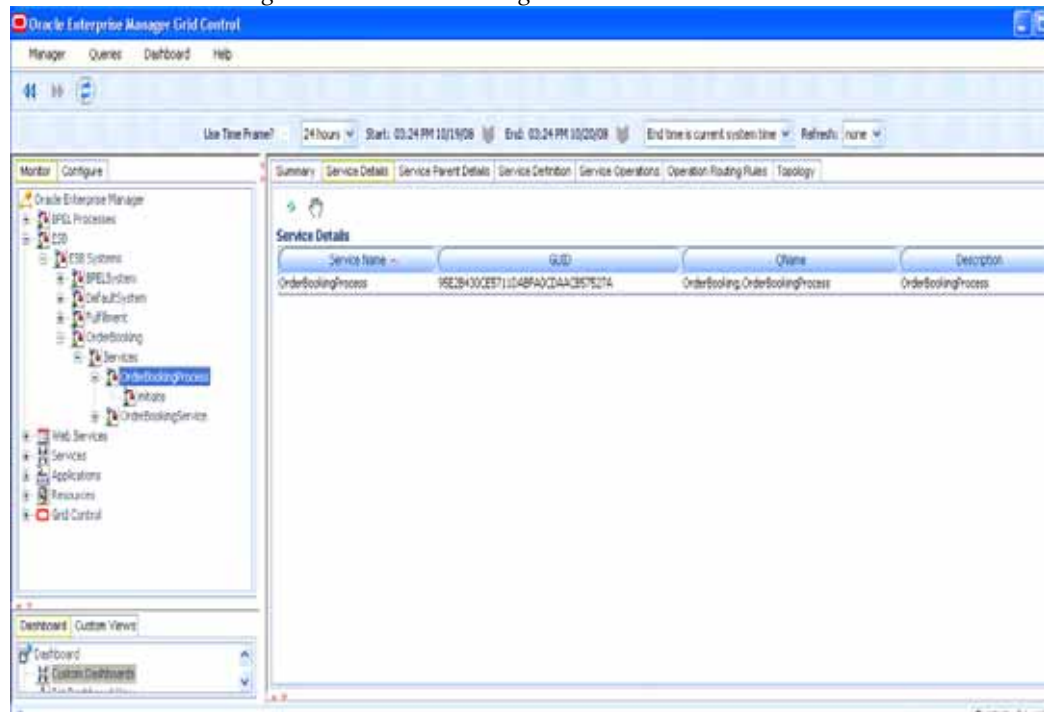


Table 4-22: Service Details View Summary

Column / Metric	Description
Service Name	The name of the ESB service.
GUID	The GUID of the ESB service.
Qname	The queue name for the bound ESB service.
Description	The description of the ESB service.

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.

Figure 4-50: OrderBookingProcess Service Parent Details View

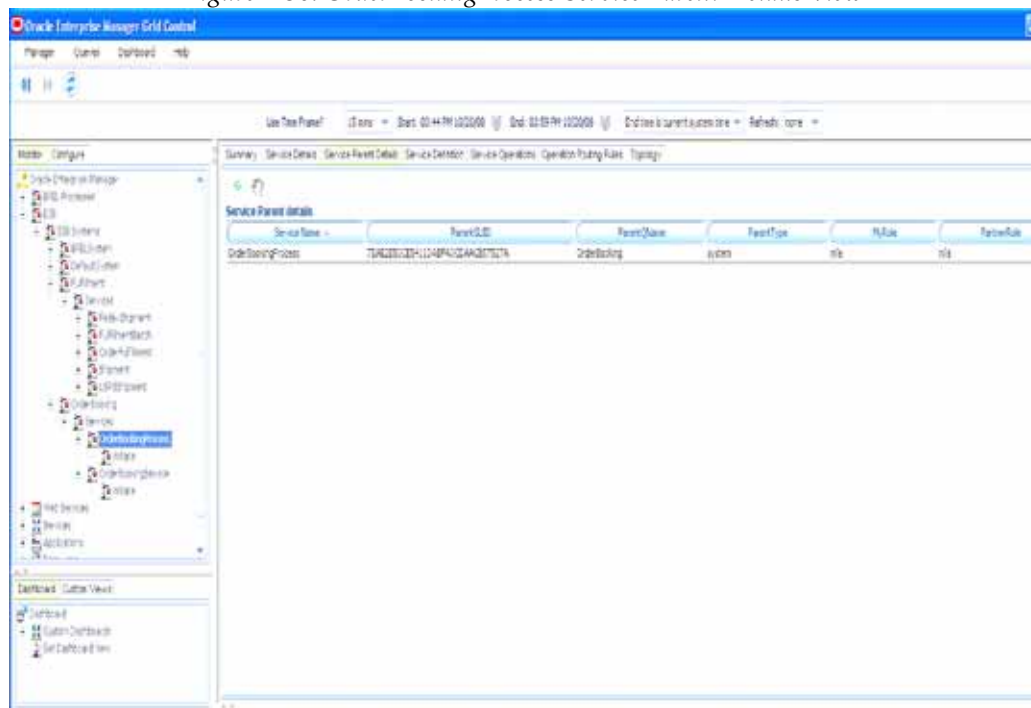


Table 4-23: Service Parent Details View Summary

Column / Metric	Description
Service Name	The name of the parent ESB service.
ParentGUID	The GUID of the parent ESB service.
ParentQname	The queue name for the parent of the bound ESB service.
ParentType	The parent type of the parent ESB service.
MyRole	The role of the caller of the parent ESB service instance.
ParentRole	The role of the callee of the parent ESB service instance.

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.

Figure 4-51: OrderBookingProcess Service Definition View

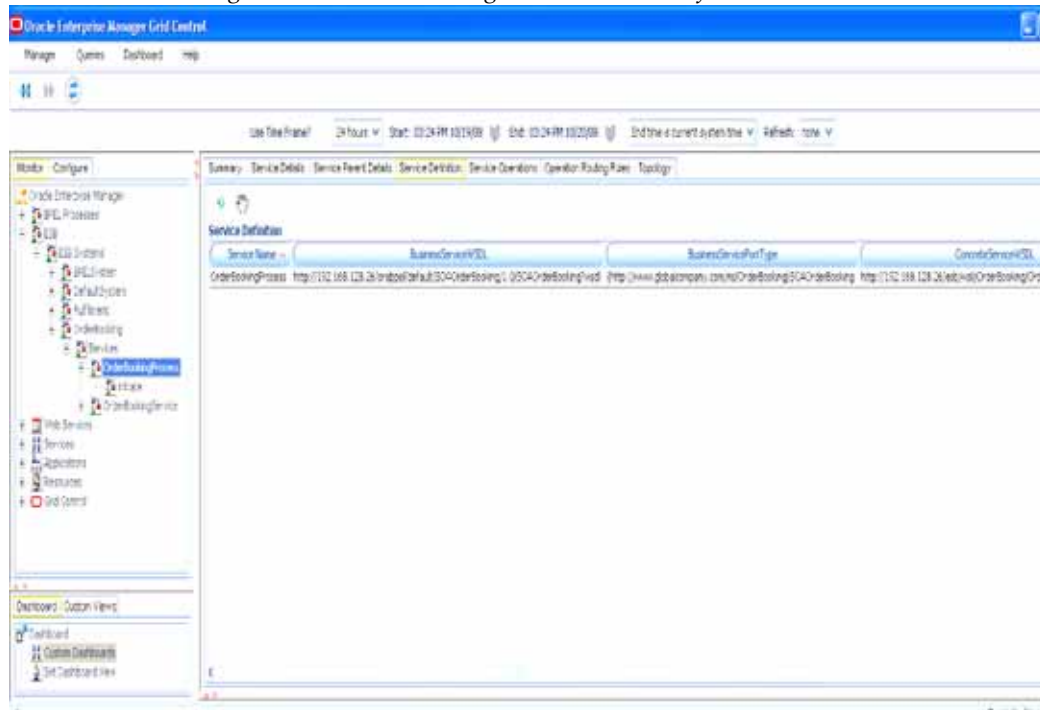


Table 4-24: Service Definition View Summary

Column / Metric	Description
Service Name	The name of the ESB service.
BusinessService WSDL	The URL of the Business Service WSDL.
BusinessService PortType	The port type of the Business Service.
ConcreteService WSDL	The URL of the Concrete Service WSFL.
ConcreteService URI	The URI for the concrete service.

Service Operations View

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

Figure 4-52: OrderBookingProcess Service Operations View

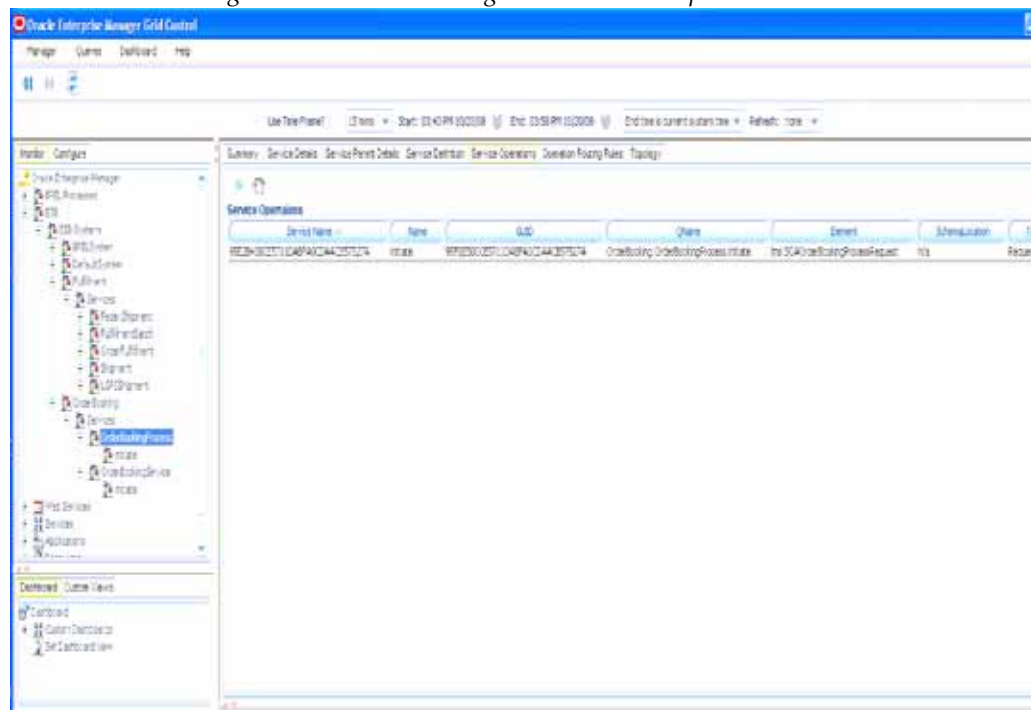


Table 4-25: Service Operations View Summary

Column / Metric	Description
Service Name	The name of the ESB service.
Name	The service operation name being executed.
GUID	The GUID of the ESB service.
Qname	The queue name for the bound ESB service.
Element	The associated element within the ESB Service.
SchemaLocation	The schema location for the associated ESB service.
Type	The type of ESB service operation.

Operation Routing Rules View

The Operation Routing Rules view provides various details in relation to the operation routing rules for Business Service operations. This includes the specific instanced business service names being utilized for operations.

Figure 4-53: OrderBookingService Operation Routing Rules View

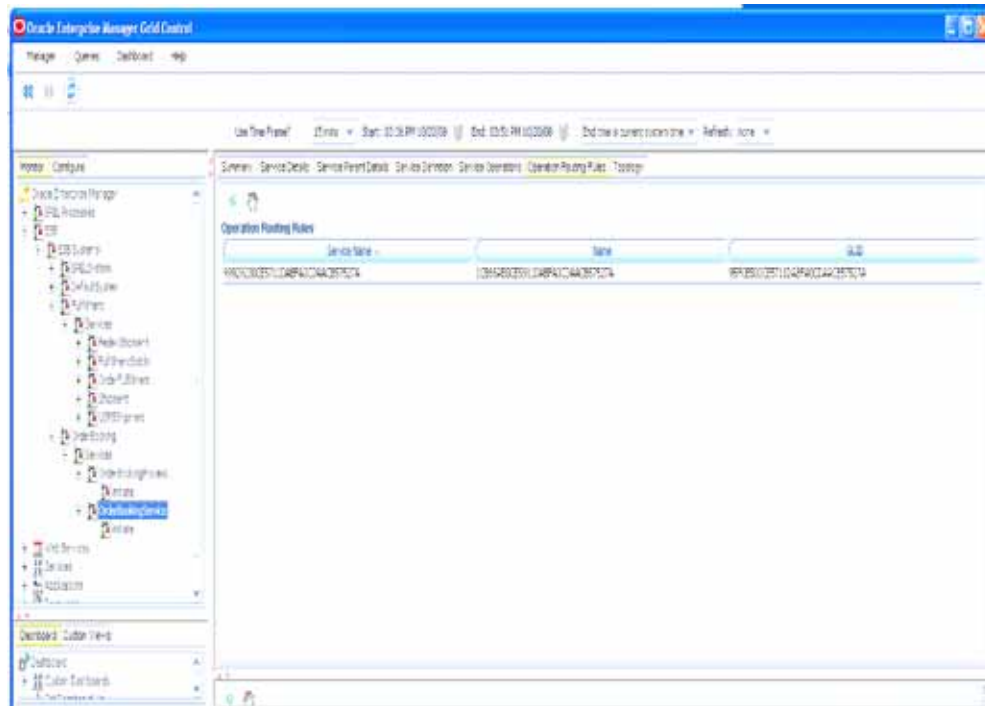


Table 4-26: Operation Routing Rules View Summary

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

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 three major categories:

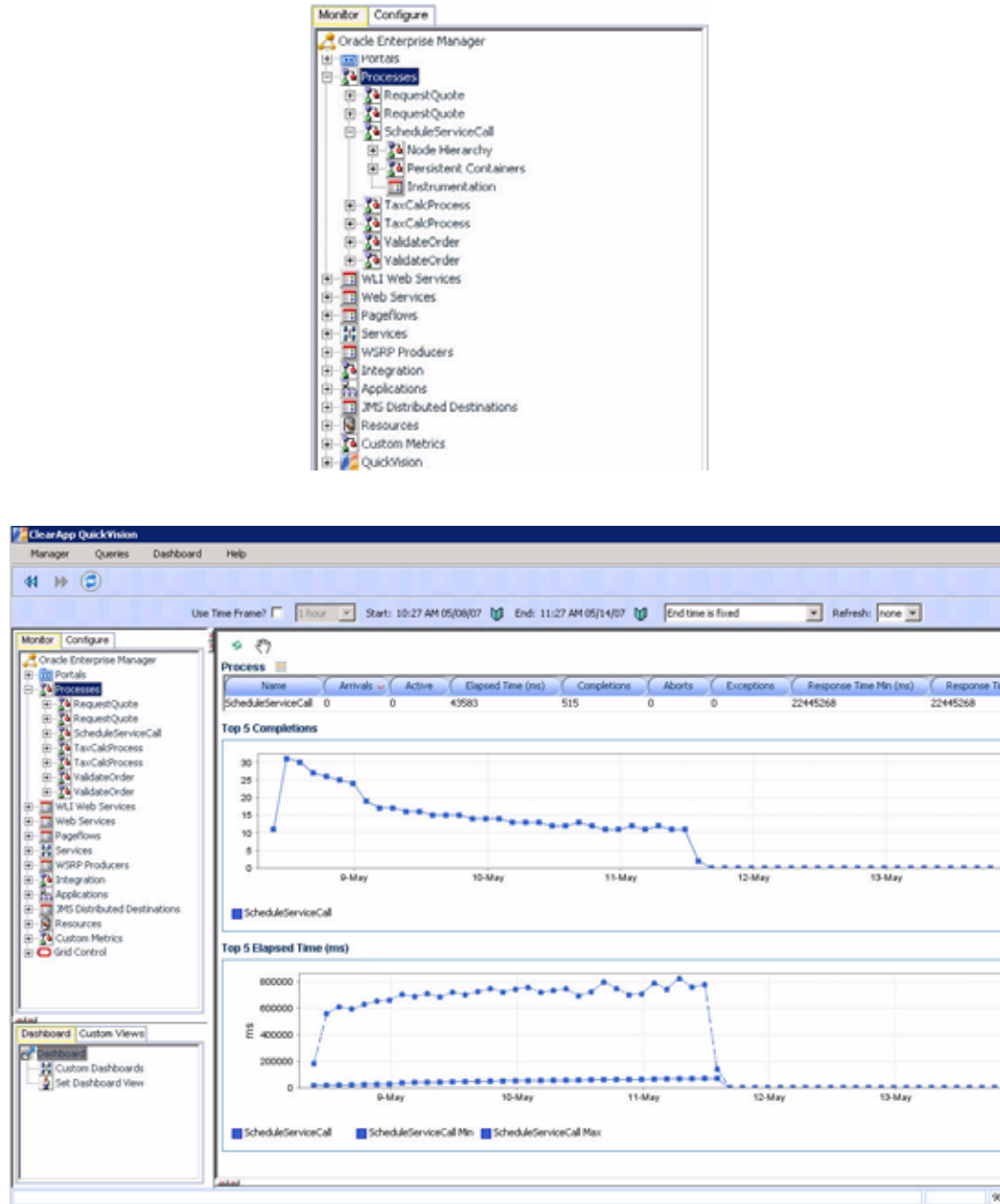
- Node Hierarchy

- 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 panel.

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

Figure 4-54: Processes Summary



The Process Summary includes the following:

Table 4-27: Process Summary Metrics

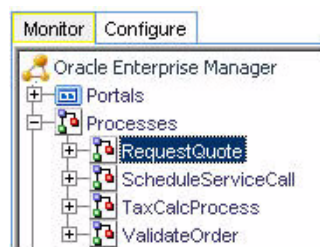
Metrics	Description
Process	Name of process
Running	Total number of currently running instances for a specific process
Suspended	Total number of suspended instances for a specific process. A Suspended request from a user is a common cause for a process instance to go into a Suspended state.
Frozen	Total 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.
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 number of process instances with Terminated, Aborted, and Frozen states can indicate abnormal in 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 Processes node. Graphical representations of two metrics, Running Instances and Average Execution Time, are displayed below the table.

When you click the + icon next to the Processes node, CAMM™ expands the tree to show all managed processes currently deployed on the WebLogic® domain. [Figure 4-55](#) shows that this domain has four active processes.

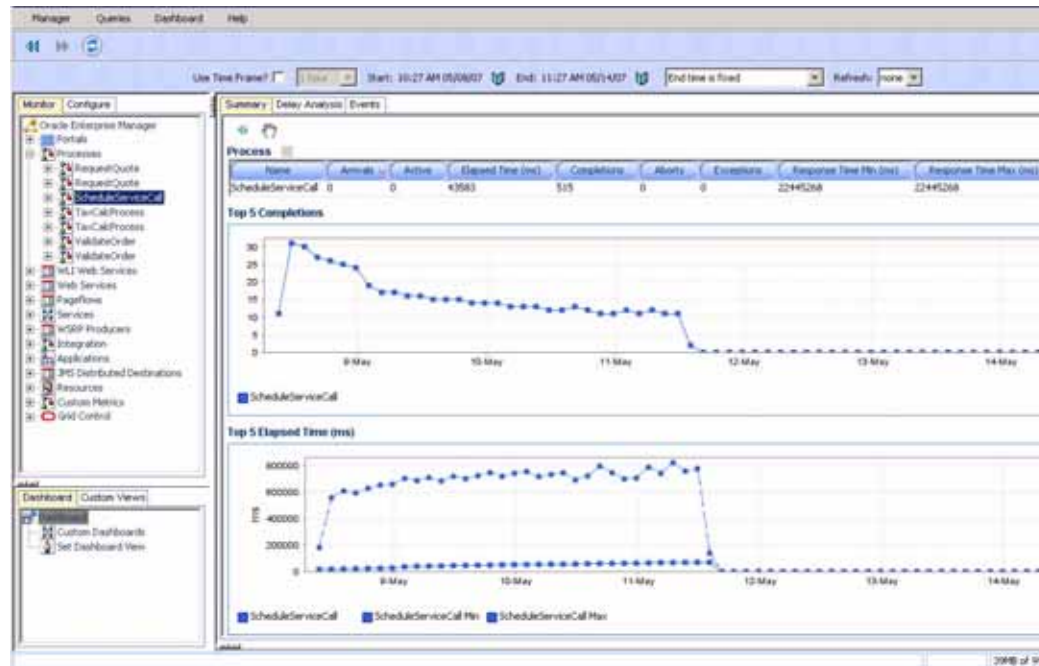
Figure 4-55: Processes Node Expanded



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.

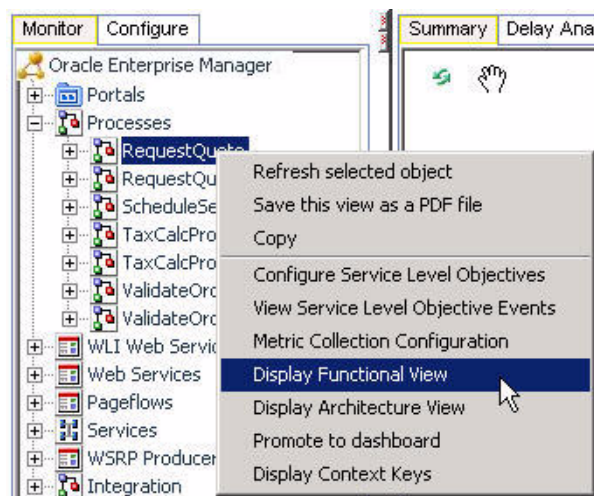
[Figure 4-56](#) shows when a user selects the RequestQuote process under the Processes node. The Main Display Window now only shows information specific to RequestQuote process.

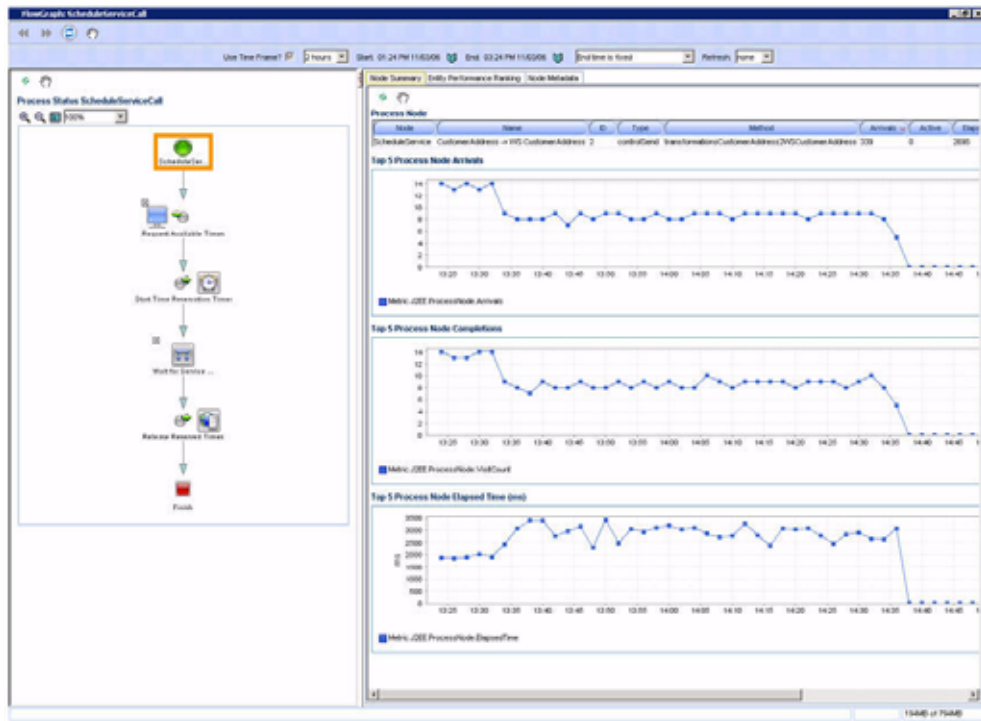
Figure 4-56: RequestQuote Process Specific Metrics



To see the process work flow associated with RequestQuote process, select RequestQuote 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 [Figure 4-57](#).

Figure 4-57: Functional Work Flow View

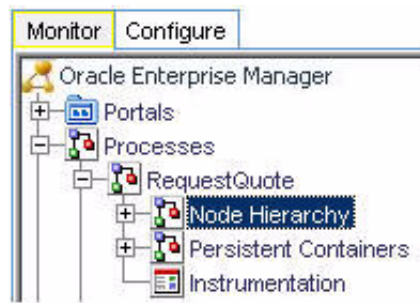


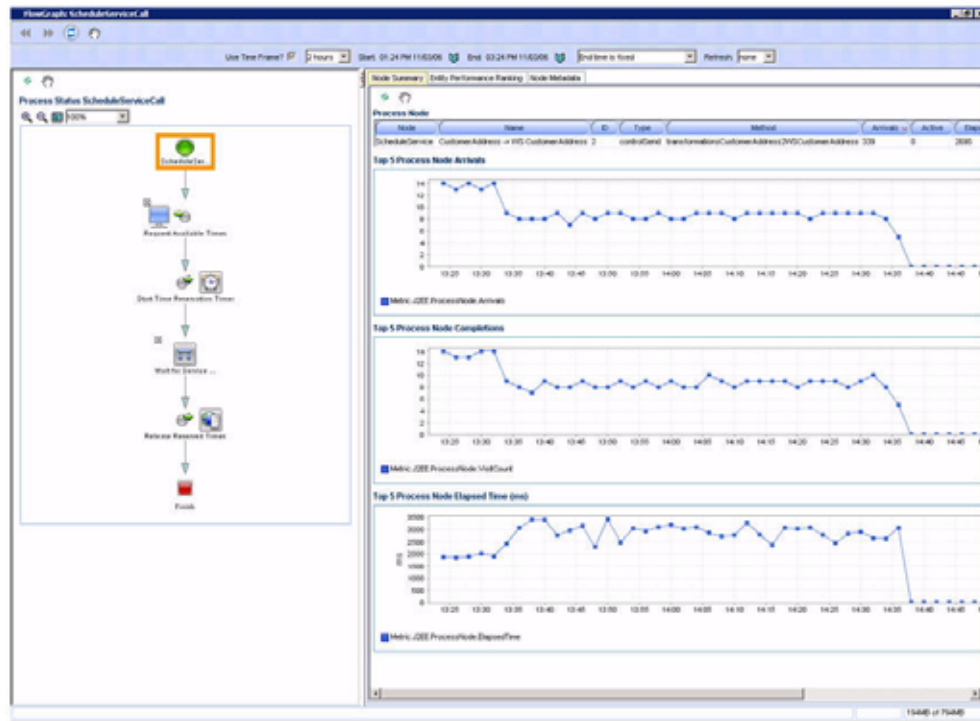


Node Hierarchy

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. [Figure 4-58](#) shows the Node Hierarchy Summary for the RequestQuote process.

Figure 4-58: RequestQuote Process Node Hierarchy Summary





See [Table 4-28](#) for Node Hierarchy summary.

Table 4-28: 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 node.

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

There are two other views available for the Node Hierarchy node:

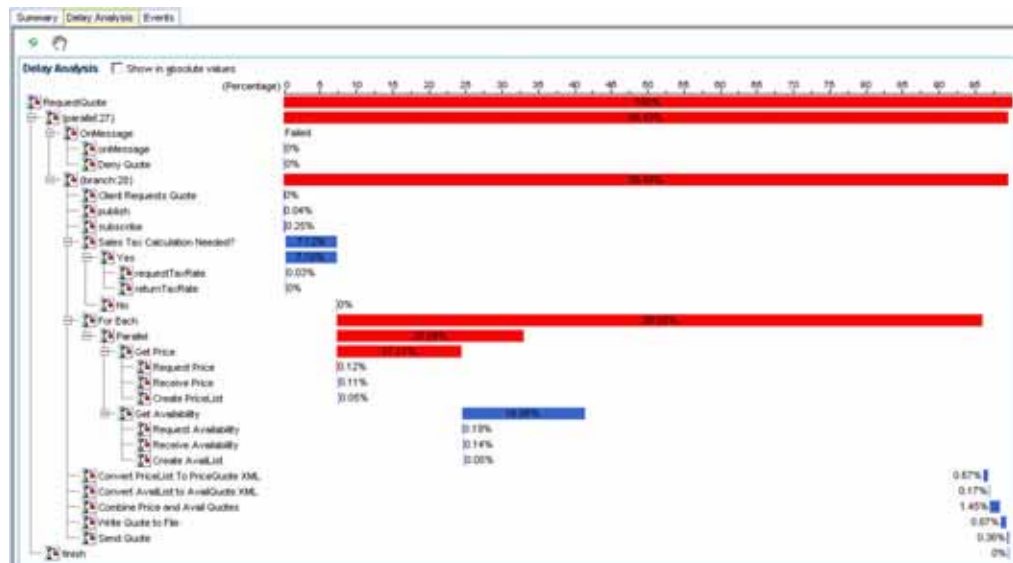
- Delay Analysis view
- Events view

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

Delay Analysis View

[Figure 4-59](#) is the 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.

Figure 4-59: Delay Analysis for RequestQuote Process



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 4-29: Events View Summary

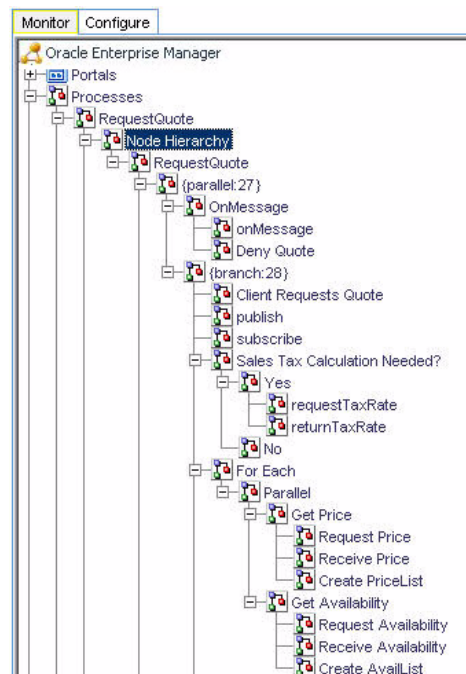
Column / Metric	Description
Start Time	Start time for the process instance that violated a SLO.
Entity Name	Name of the process node that violated a SLO.
SLO Name	Name of the violated SLO.
Service URI	URI of the process that violated a SLO.

Table 4-29: Events View Summary (Continued)

Column / Metric	Description
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.

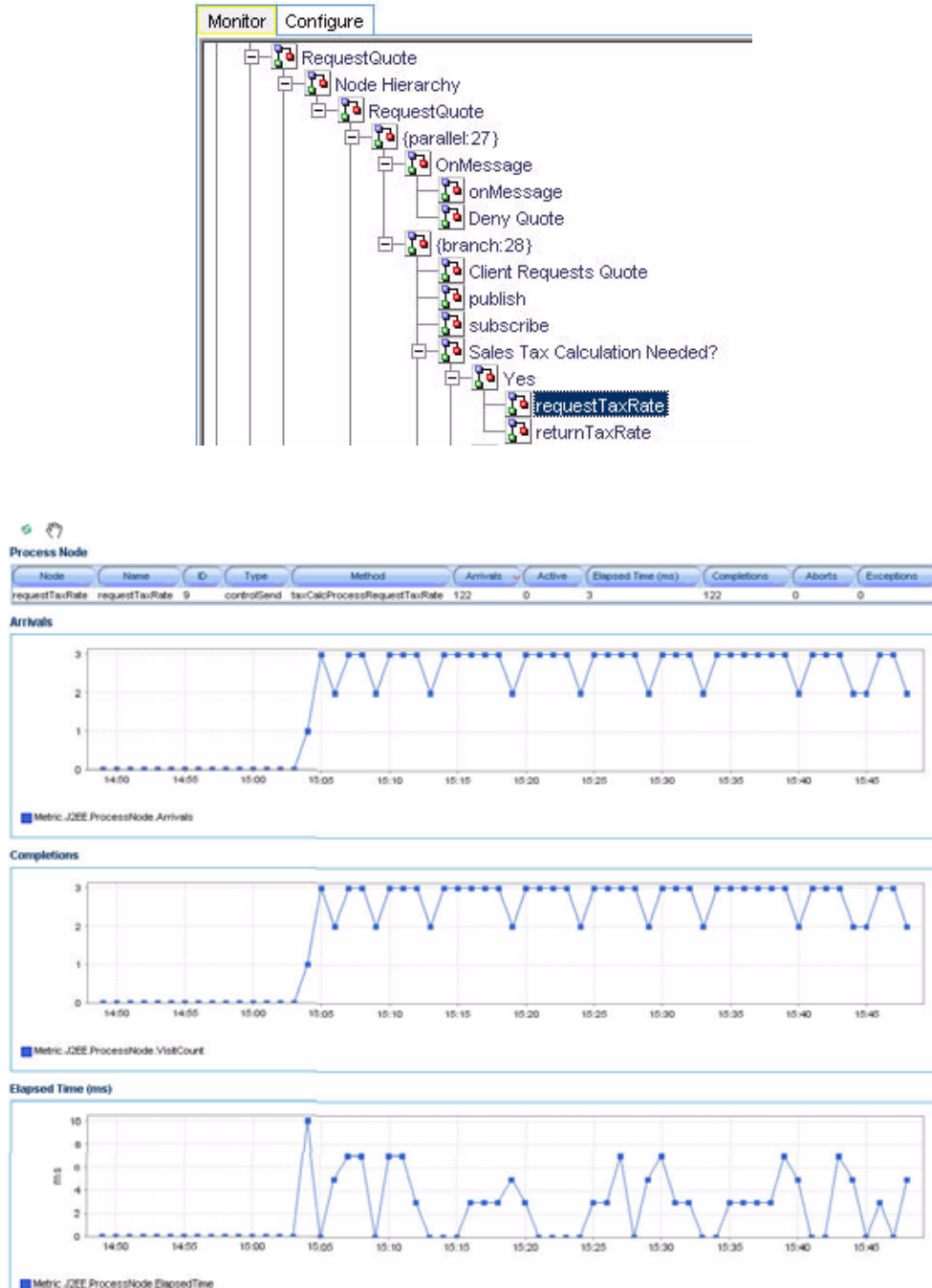
When you click the + icon next to a specific Node Hierarchy node, CAMM™ expands the tree to show process nodes in the Node Hierarchy. [Figure 4-60](#) shows the entire node hierarchy for RequestQuote process.

Figure 4-60: Drill Down to See the Entire Node Hierarchy



Click an individual process node to see the load and performance of the selected node in the Main Display Window. See [Figure 4-61](#) for an example of a specific performance information.

Figure 4-61: Process Node Specific Performance Information

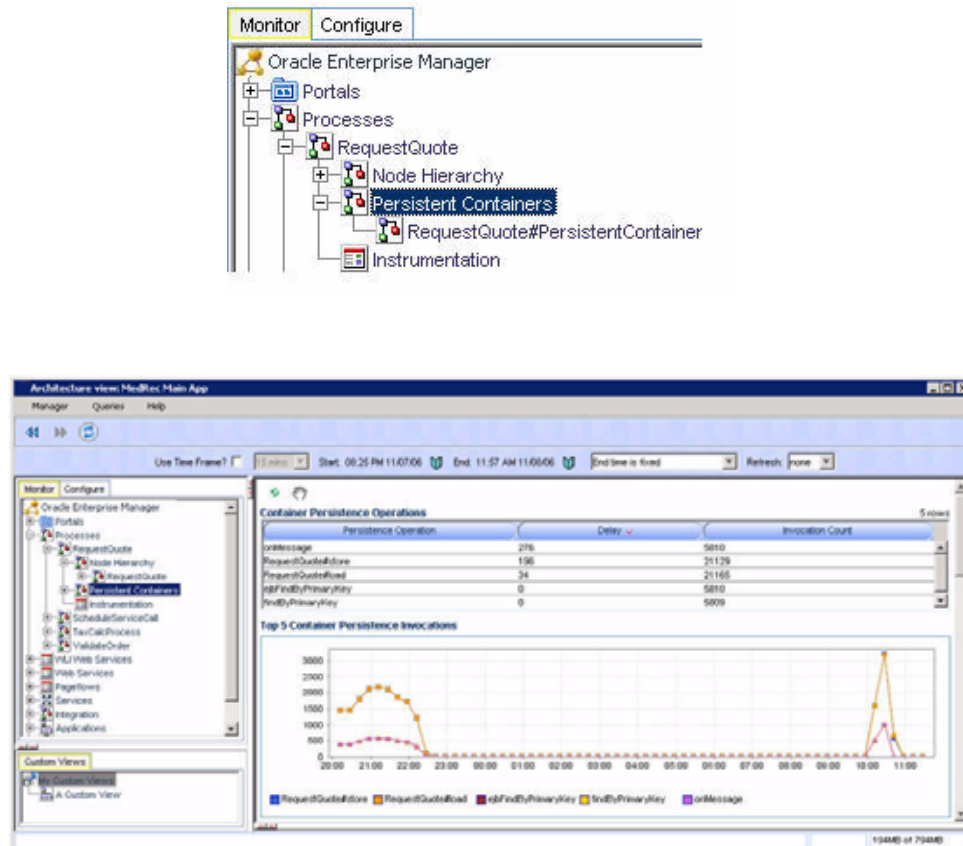


The process node information also includes the name of the method invoked. This information is displayed as part of the summary table at the top of the main view window.

Persistent Containers

When you expand a particular process further, the third item is the Persistent Containers node. By selecting the Persistent Containers node, CAMM™ provides a list of persistence performance statistics relevant to the selected process. [Figure 4-62](#) shows the Persistent Containers relevant to RequestQuote.

Figure 4-62: Persistent Containers Summary



As you select the root of the Persistent Containers tree, a summary of all Persistent Containers relevant to the selected process is presented. In our example, the summary is the RequestQuote. This summary contains 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. See [Figure 4-63](#) and [Figure 4-64](#).

Figure 4-63: Persistence Containers Summary Information

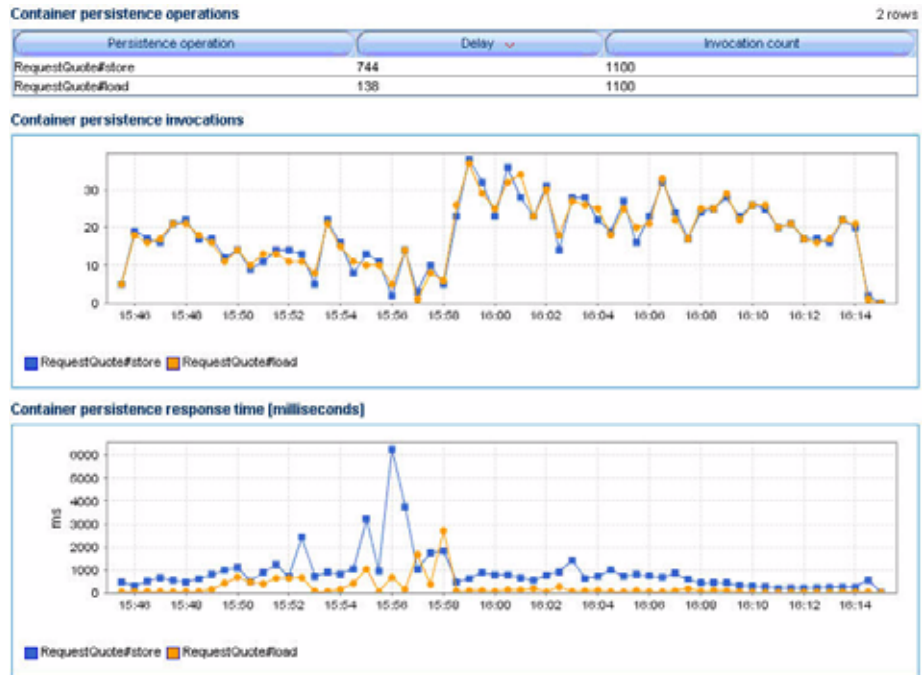


Figure 4-64: Persistence Containers Summary Information Continued



The Persistent Containers Summary includes four different tables:

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

Entity EJB Activity Table

Entity EJB Activity table includes the following information:

Table 4-30: 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.

Entity EJB Cache Table

Entity EJB Cache table includes the following information:

Table 4-31: 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 (reconstitute EJB state information from disk) are resource intensive operations. Ideally, we would like to see low level of activity in these metrics.

Entity EJB Transactions Table

Entity EJB Transactions table includes the following information:

Table 4-32: 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.

Entity EJB Locking Table

Entity EJB Locking table includes the following information:

Table 4-33: 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 two 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.

Instrumentation

Expanding a particular process further, the last item you see is the Instrumentation node. Click the + icon next to Instrumentation to expand the tree to reveal four categories of instrumentation:

- Class
- Methods
- Errors/Exceptions
- Transactions

The Class node in the Instrumentation tree provides the following information:

Table 4-34: 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 4-35: 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 the [Architecture View](#) section.

WLI Web Services

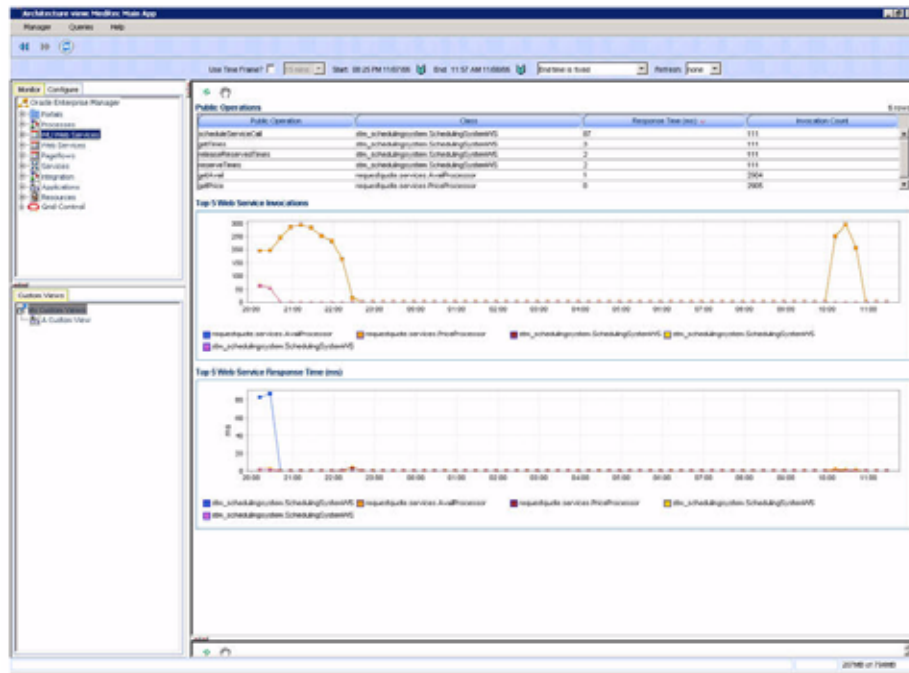
The WLI Web Services node under Oracle™ Tree contains information about all deployed Web Services defined as part of WebLogic® Integration applications in the managed domains. CAMM™ organizes information for various WLI Web Services node is into tree major categories:

- Public Operations
- 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 panel.

When you select the root of the WLI Web Services tree, CAMM™ displays the WLI Web Services Summary in the Main Display Window.

Figure 4-65: WLI Web Services Summary



The WLI Web Services Summary includes the following:

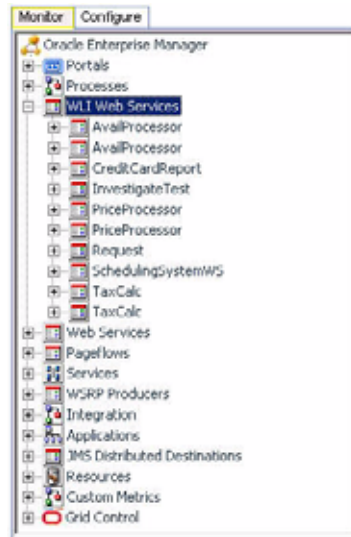
Table 4-36: WLI Web Services Summary

Metrics	Description
Public Operation	Name of the web service public operation.
Response Time (ms)	The average response time for a specific web service public operation.
Invocation Count	Total number of times a specific web service public operation has been called.

CAMM™ presents these metrics in a table format in the Main Display Window when you select the WLI Web Services node. Graphical representations of two metrics, Web service invocation and Web service response time, are displayed below the table.

Click the + icon next to the WLI Web Services node to expand the tree to show all monitored web services currently deployed on the WebLogic® domain. [Figure 4-66](#) shows this domain has four deployed web services.

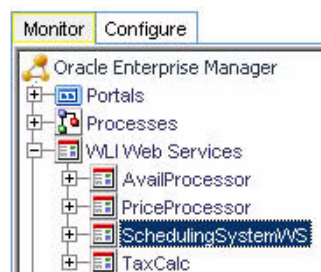
Figure 4-66: Web Services Node Expanded

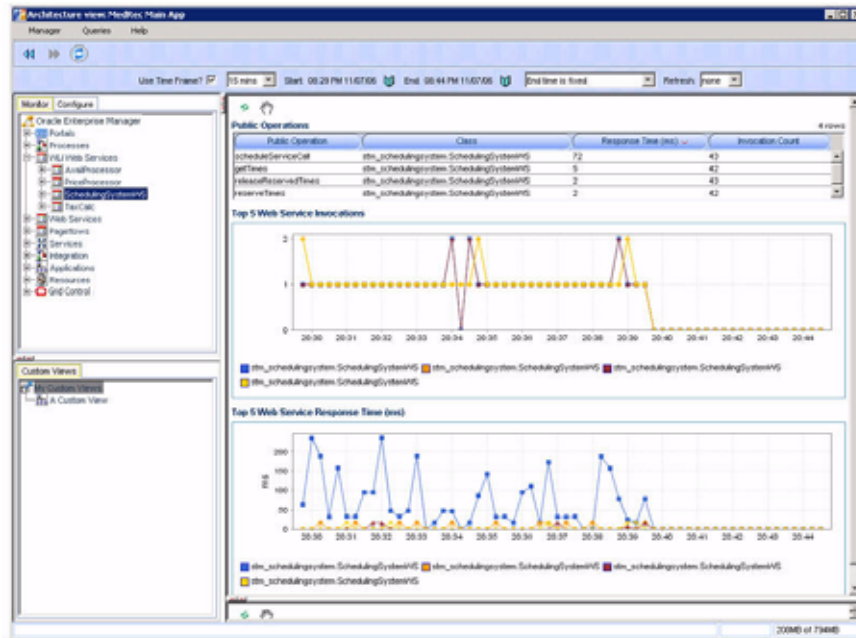


You can also see information specific to a particular web service. By selecting a specific web service, all information displayed in the Main Display Window changes to only show data relevant to this new context.

[Figure 4-67](#) shows when the SchedulingSystemWS web service under the Web Services node is selected. The Main Display Window only shows information specific to SchedulingSystemWS web service.

Figure 4-67: SchedulingSystemWS Web Service Specific Metrics

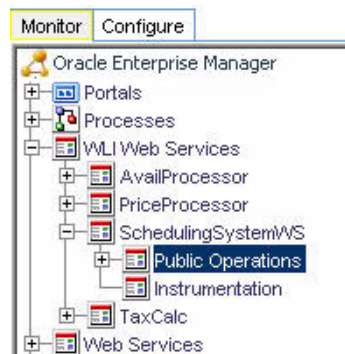


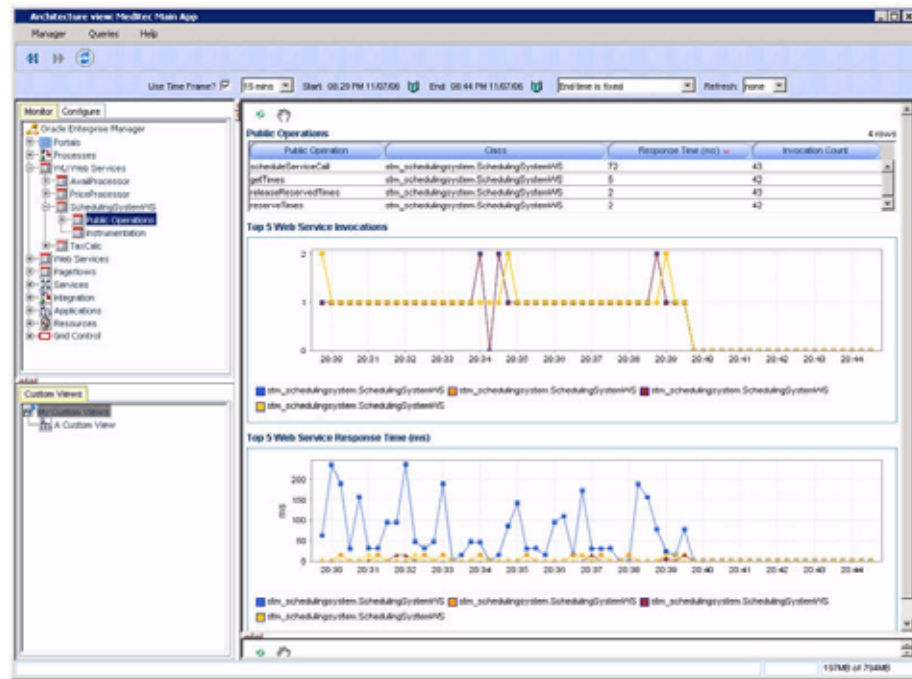


Public Operations

Expanding a particular process further, the first item you see is the Public Operations node. By selecting the Public Operations node, CAMM™ provides a list of public operations associated with the specific web service. [Figure 4-68](#) shows the Public Operations Summary for the SchedulingSystemWS web service.

Figure 4-68: SchedulingSystemWS Public Operations Summary





The Node Hierarchy Summary displays as a table that includes the following:

Table 4-37: Public Operations Node Hierarchy Summary

Metrics	Description
Public Operations	Name of the web service public operation.
Response Time (ms)	The average response time for a specific web service public operation.
Invocation Count	Total number of times a specific web service public operation has been called.

CAMM™ presents these metrics in a table format in the Main Display Window when you select the Web Services node. Graphical representations of two metrics, Web service invocation and Web service response time, are displayed below the table.

Instrumentation

When you expand a particular process further, the last item you see is the Instrumentation node. Click the Instrumentation node to show the Main Display Window containing two panels with instrumentation data relevant at the class and method levels.

Figure 4-69: Class Panel Displays Instrumentation Data at the Class Level



The Class panel provides the following information:

Table 4-38: Instrumentation Class Panel Summary

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.

Figure 4-70: Instrumentation Data at the Method Level



The Method panel provides the following information:

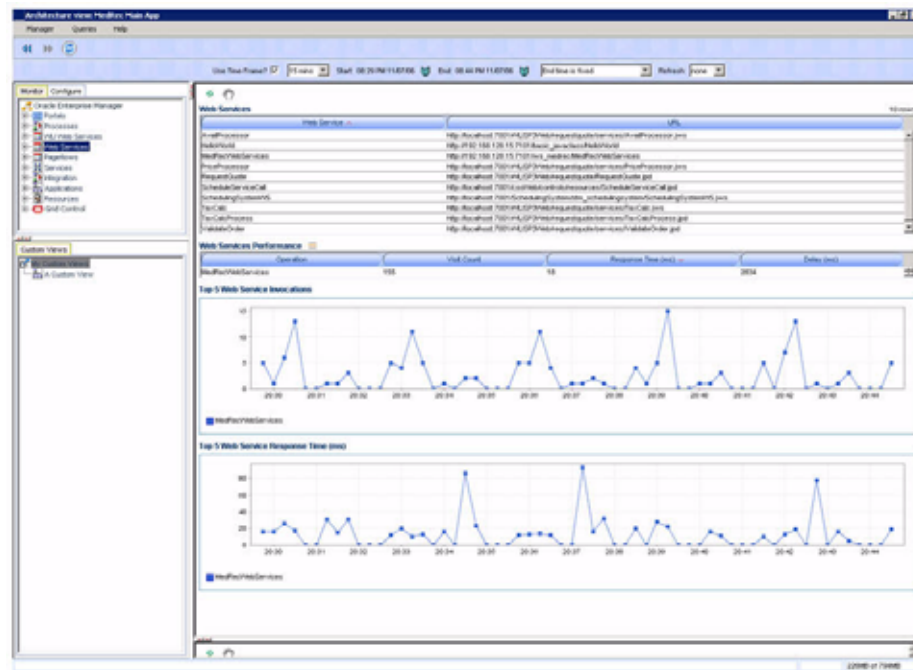
Table 4-39: Instrumentation Method Panel Summary

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.

Web Services

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

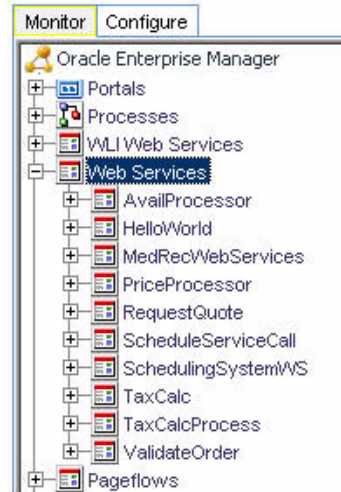
Figure 4-71: Web Services Summary in CAMM™



This summary view lists out 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 + icon next to the Web Services node, CAMM™ expands the tree to show all monitored web services currently deployed on the WebLogic® domain. [Figure 4-72](#) shows this domain has several deployed web services.

Figure 4-72: Web Services Node Expanded



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

[Figure 4-73](#) shows the current performance data for MedRecWebServices and the expanded tree node with seven discovered public operations.

CAMM™ displays the performance data associated with the selected MedRecWebServices node.

Figure 4-73: MedRecWebServices Performance Data



The Operations table provides the following information:

Table 4-40: Operations Table

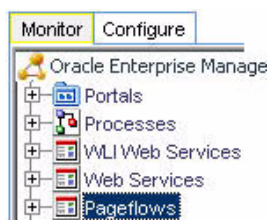
Column / Metric	Description
Operation	Name of the web service operation.
Invocation Count	Number of times the operation is called.
Response Time (ms)	Average response time for the operation in milliseconds.
Delay (ms)	Overall delay contributed by the operation in milliseconds.

[Figure 4-74](#) shows two graphs with time-based trending for web service invocation and response time.

Pageflows

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

Figure 4-74: Pageflows Summary





Services

The Services node under Oracle™ Tree contains information about all external entry points into the managed domain. CAMM™ currently monitors three different types of services:

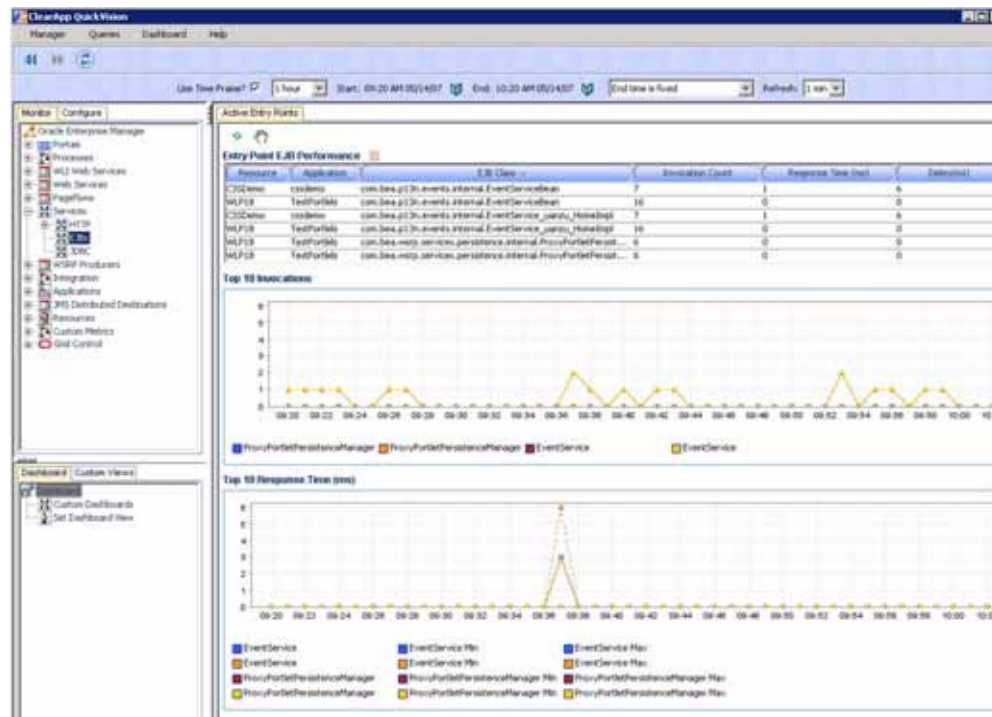
- HTTP
- EJBs
- JDBC

Selecting each service type reveals service summary in the Main Display Window. [Figure 4-75](#) shows this view when the EJB service is selected.

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 panel.

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

Figure 4-75: EJB Service Entry Point Activity Summary



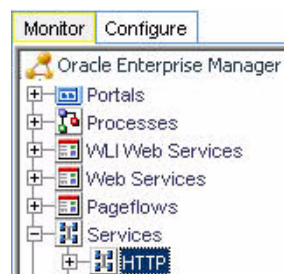
Tip: Setting thresholds at some of these entry points enables CAMM™ to monitor the performance of key business services. When a violation event occurs, you can begin investigating from the Service node.

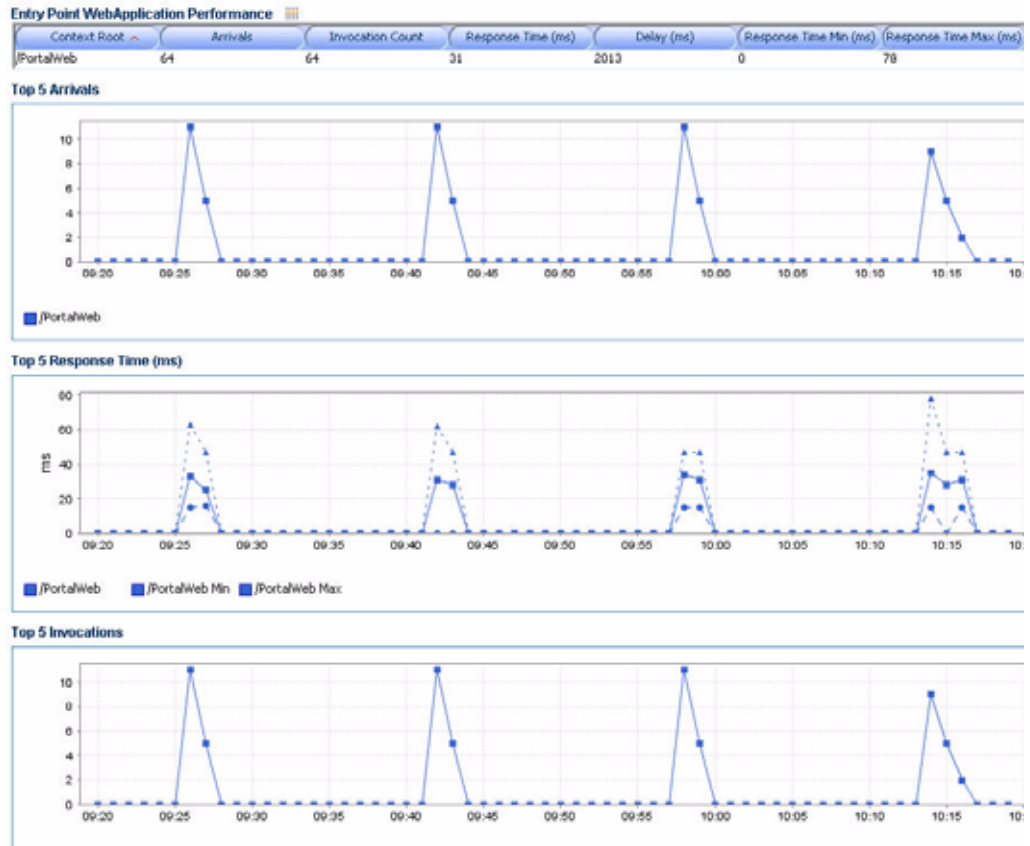
HTTP

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 web services. 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. See [Figure 4-76](#).

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

Figure 4-76: 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 + 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. See [Figure 4-77](#).

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

Figure 4-77: Method Level Performance Data

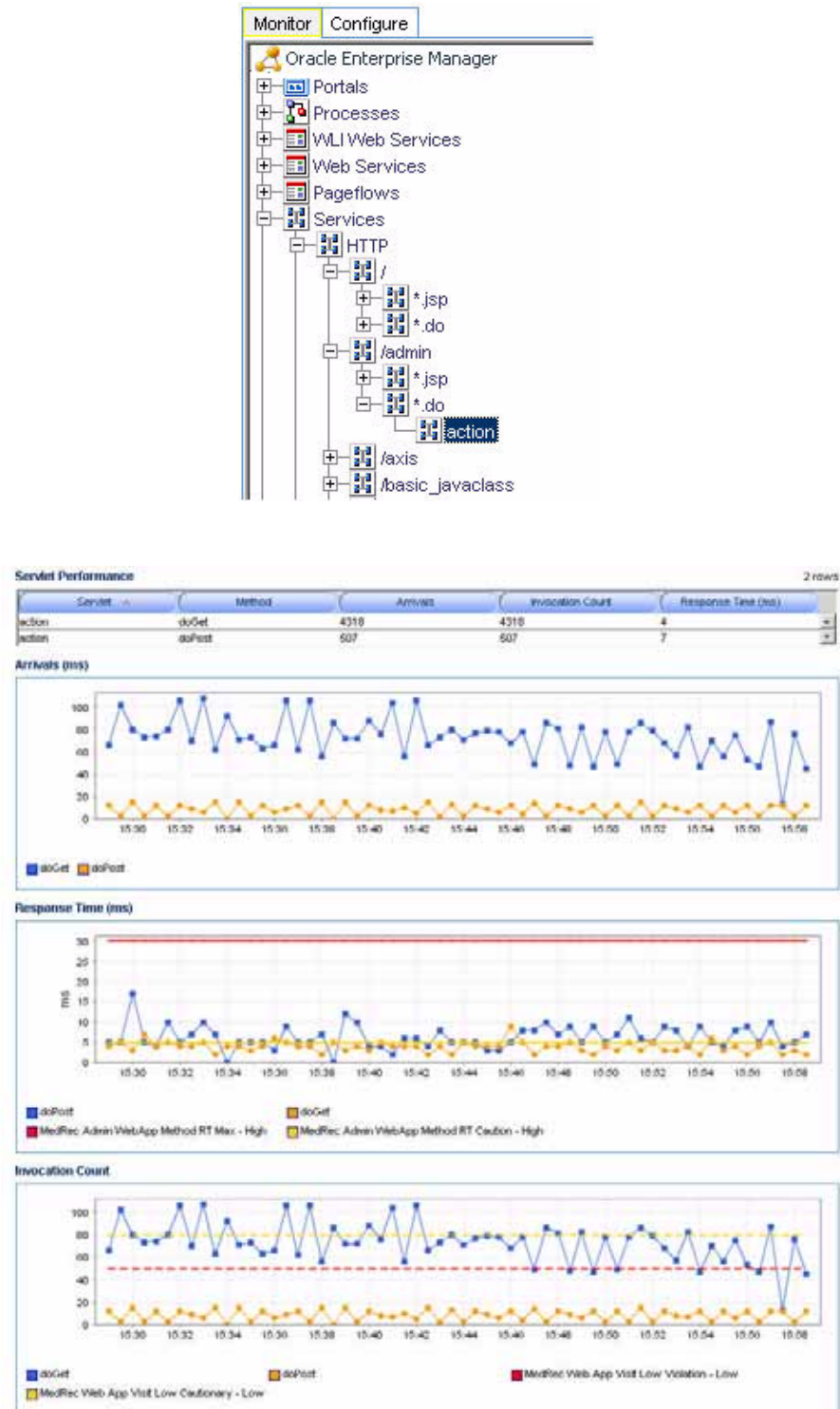


Table 4-41: HTTP Performance Summary

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

EJBs

Click the EJBs node to bring up the performance summary for EJBs invoked from outside of the JVM.

Table 4-42: 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.

JDBCs

Click the JDBCs note to bring up the performance summary for JDBC operations invocated from outside of the JVM.

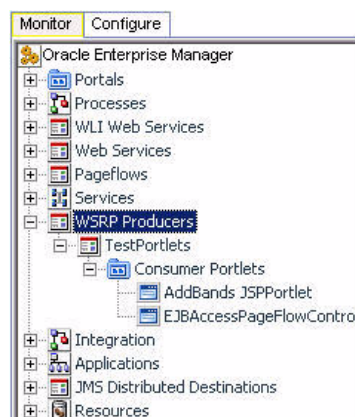
Table 4-43: JDBC Performance Summary

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.

WSRP Producers

The Web Services Remote Portlet (WSRP) Producers node under the Oracle™ 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.

Figure 4-78: WSRP Producers



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 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 panel.

WSRP Summary

To view the WSRP Producers Summary:

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

Figure 4-79: WSRP Producers Summary



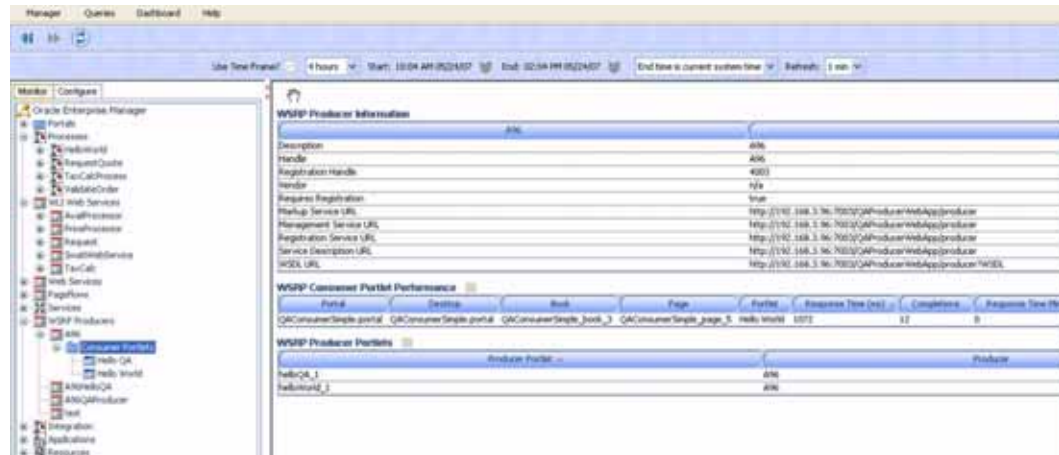
The WSRP Producers summary includes the following table:

Table 4-44: WSRP Producers Summary

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

- To view the portlet details, click the Consumer Portlets node under the WSRP Producers. See [Figure 4-78](#).

Figure 4-80: WSRP Details



There are three tables in this view:

- WSRP Producer Information
- WSRP Consumer Portlet Performance

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

- WSRP Producer Portlets

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

Table 4-45: 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 4-46: 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.

Table 4-46: WSRP Consumer Portlet Performance (Continued)

Column	Description
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 4-47: WSRP Producer Portlets

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

- Click on a portlet name in the tree view to see the performances associated with the consumer and producer portlets:

Figure 4-81: WSRP Consumer Portlet Performance

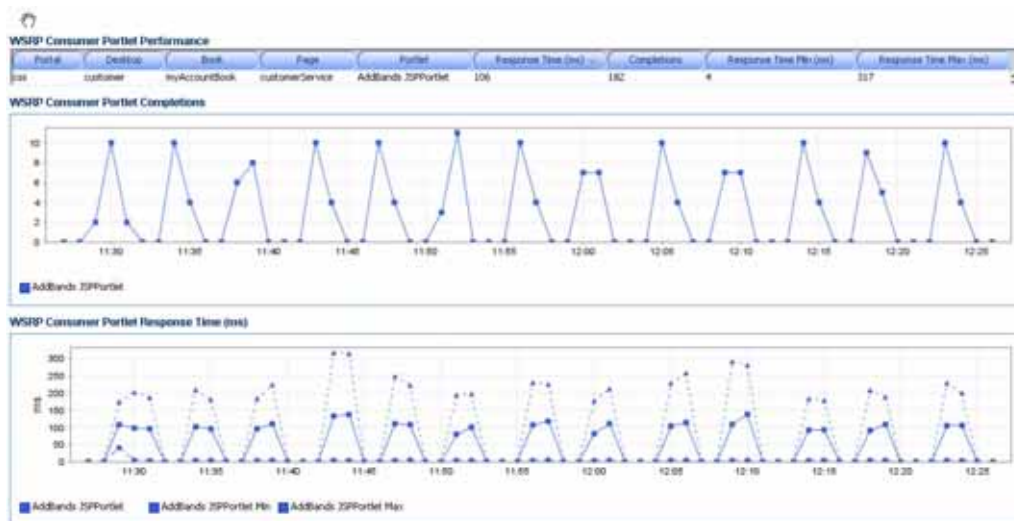
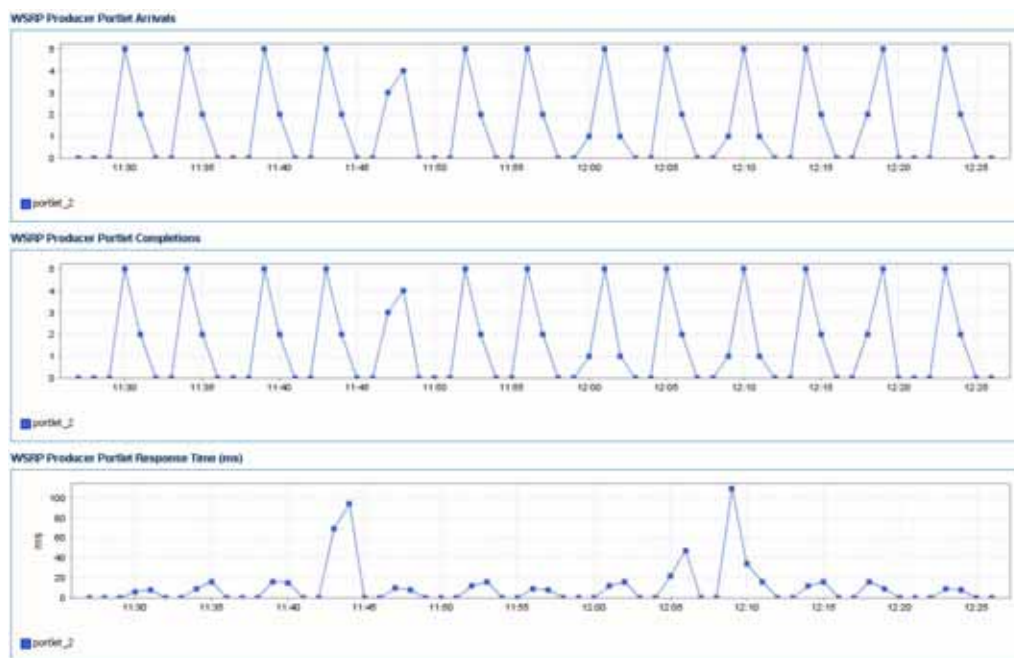


Figure 4-82: WSRP Producer Portlet Performance



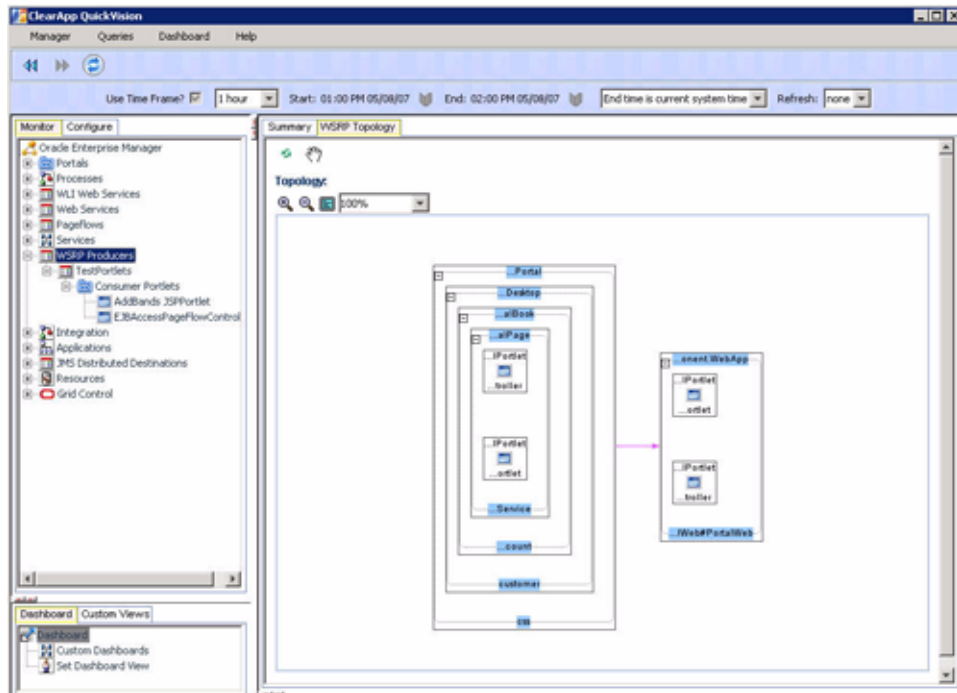
WSRP Topology

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.

Figure 4-83: WSRP Topology



2. Click the WSRP Topology tab to view the details.

Display Portal Desktop

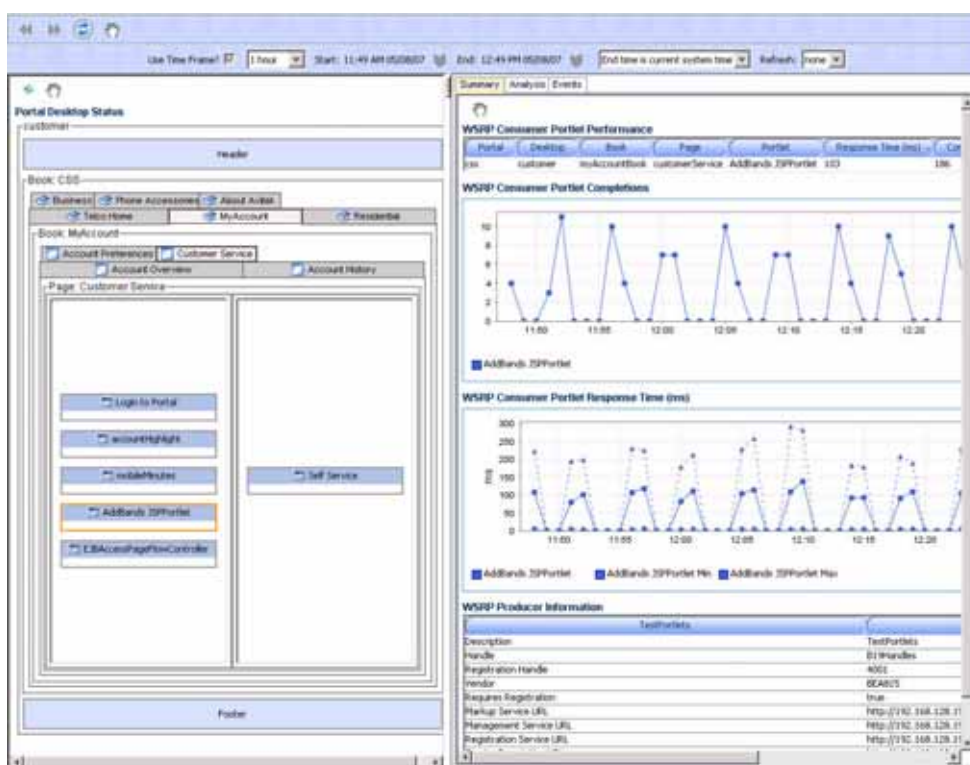
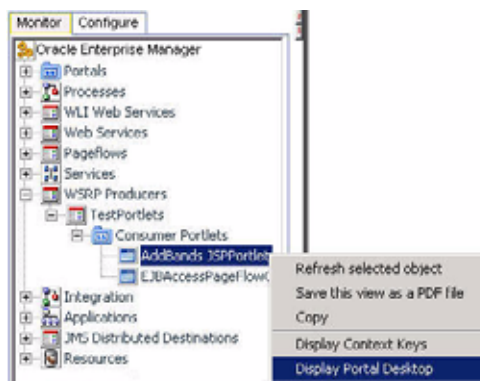
The portal desktop is described in [Display Portal Desktop - Desktop Structure Viewer](#) section in this guide.

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.

Figure 4-84: Display Portal Desktop



3. You can drill-down to view the Architecture View from this view. See the instructions in [Portlet Drill Down](#) section for details.

Integration

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

Figure 4-85: Integration Summary



The Integration Summary includes the following:

Table 4-48: Integration Summary

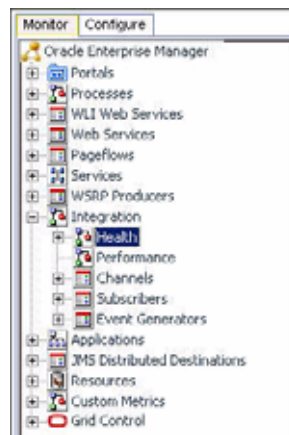
Metrics	Description
Process	Name of process.
Running	Total number of currently running instances for a specific process.
Suspended	Total number of suspended instances for a specific process.
Frozen	Total number of frozen instances for a specific process.
Completed	Total number of completed instances 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 number of process instances with Terminated, Abort, and Frozen states can indicate abnormal in 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 three metrics, Running Instances, Completed Instances, and Average Execution Time, are displayed below the table.

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

Figure 4-86: Expanded Integration Tree



The expanded Integration tree allows you to look at various components of WebLogic® Integration and help identify performance bottlenecks. In this section, we discuss the nodes under the Integration Tree further.

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:

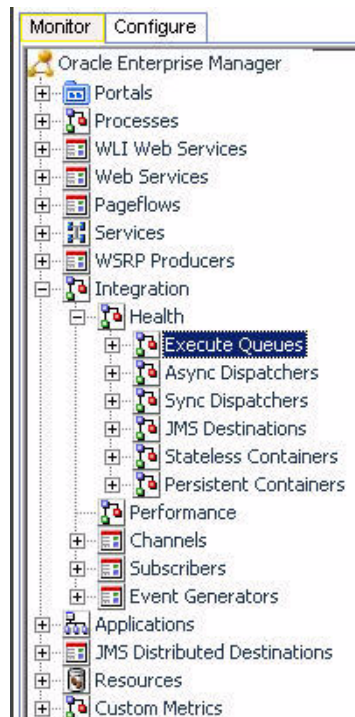
- Execute Queues
- Async Dispatchers
- Sync Dispatchers
- JMS Destinations
- Stateless Containers
- Persistent Containers

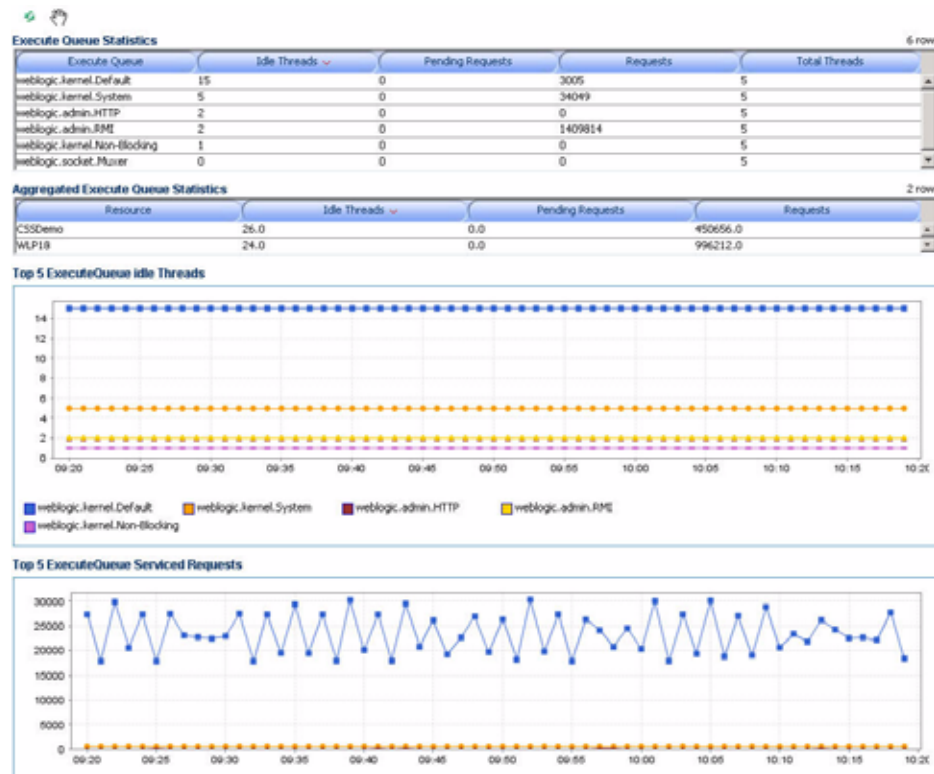
You can get to the health information specific to each of these subsystems by clicking on the appropriate node. You can even get to a particular instance of a subsystem.

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.

Figure 4-87: Execute Queues Summary





The Execute Queues Summary provides the following information:

Table 4-49: Execute Queues Summary

Metrics	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 4-50: 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.

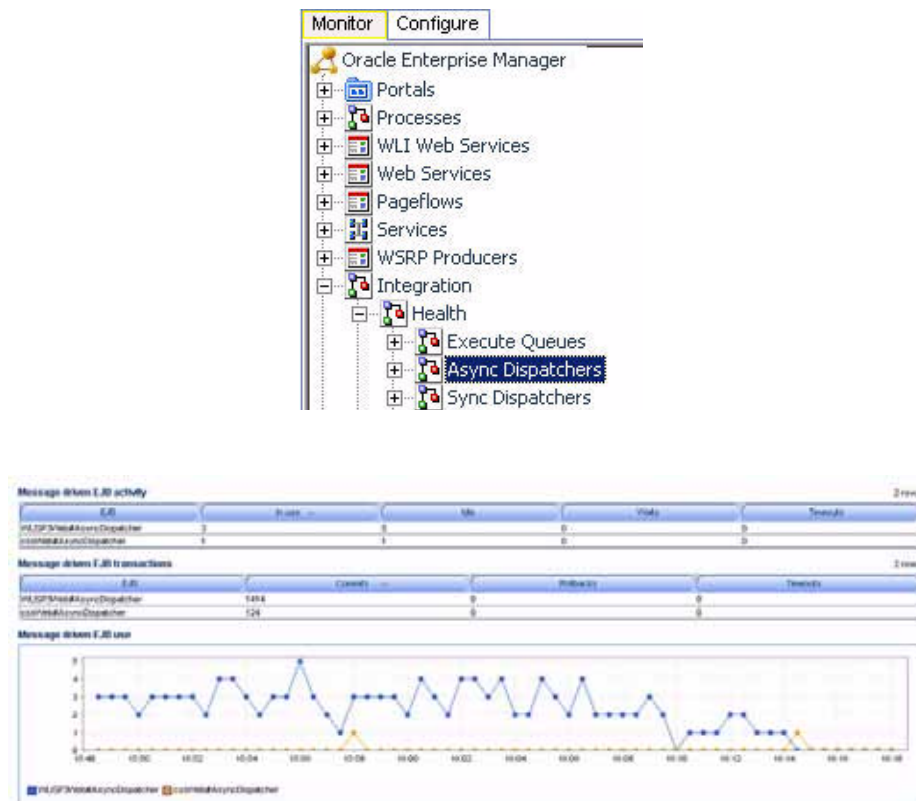
CAMM™ presents these metrics in a table format in the Main Display Window when you select the Health node. Graphical representations of three metrics, Idle Treads, Pending Threads, and Requests, are displayed below the table.

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

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.

Figure 4-88: Async Dispatcher Summary



The Async Dispatcher Summary includes the following information:

Table 4-51: Async Dispatcher Summary

Metrics	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 (Transactions)	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.

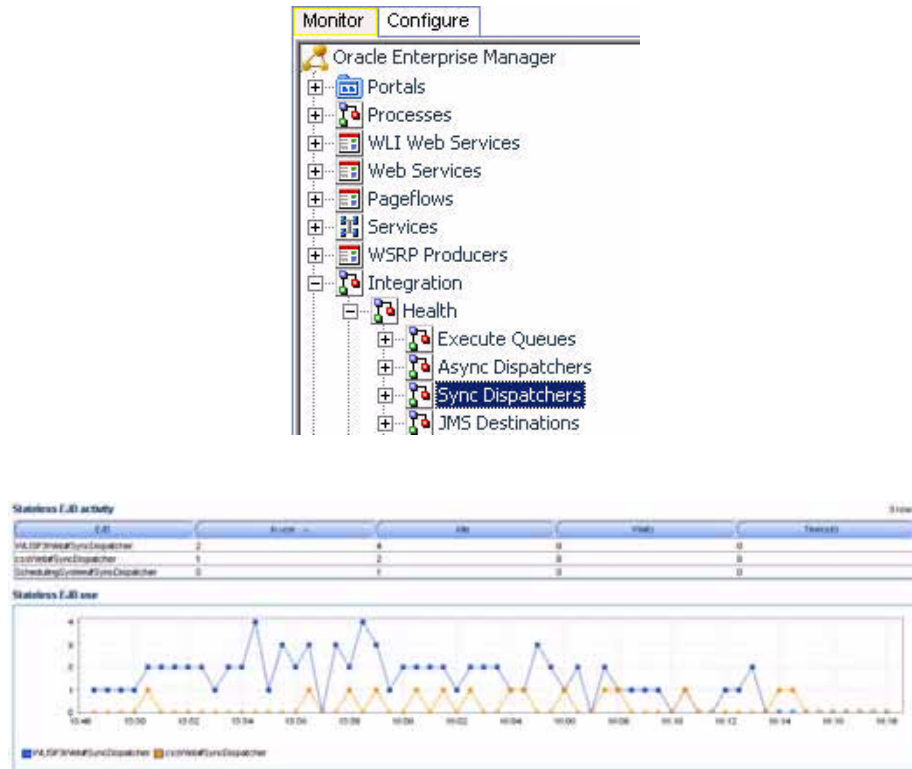
CAMM™ presents these metrics in a table format in the Main Display Window when you select the Async Dispatchers node. Graphical representation of one metrics, Message Driven EJB in use, is displayed below the table.

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

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.

Figure 4-89: Sync Dispatchers Summary



The Sync Dispatcher Summary includes the following information:

Table 4-52: Sync Dispatcher Summary

Metrics	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.

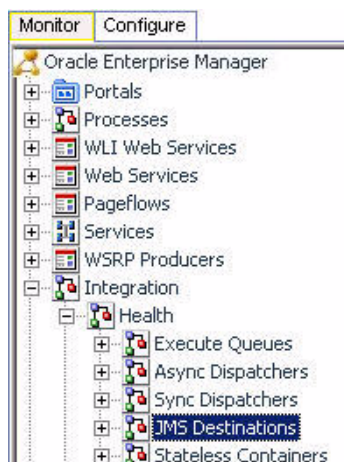
CAMM™ presents these metrics in a table format in the Main Display Window when you select the Sync Dispatchers node. Graphical representation of one metrics, Stateless EJB in use, is displayed below the table.

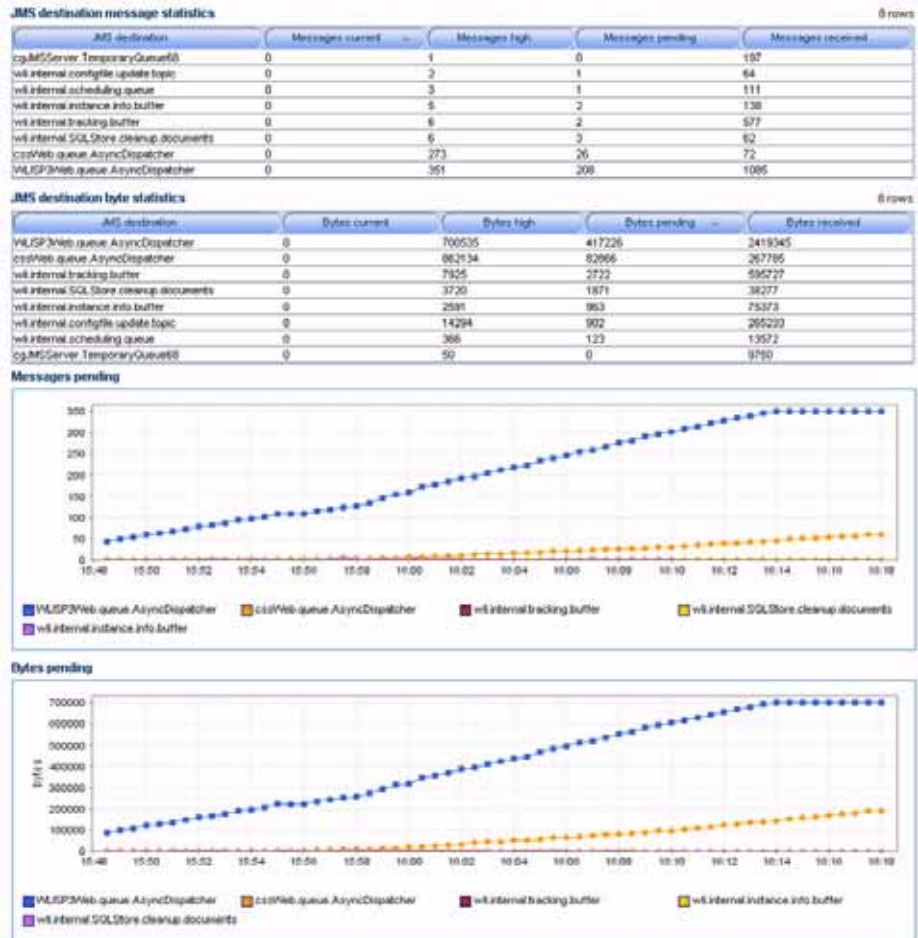
Expand the Sync Dispatchers tree by clicking on the + icon next to Sync Dispatchers node. You can get the same summary as described above for a specific sync dispatcher.

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.

Figure 4-90: JMS Destinations Summary





JMS Destination Summary includes two tables - JMS destination message statistics and JMS destination byte statistics. The JMS destination message statistics table includes the following information.

Table 4-53: 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 4-54: 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 two metrics, Message pending and Byte pending, are displayed below the table.

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

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.

Figure 4-91: Stateless Containers Summary



The Stateless Containers Summary includes the following information:

Table 4-55: Stateless Container Summary

Metrics	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]

Table 4-55: Stateless Container Summary (Continued)

Metrics	Description
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]

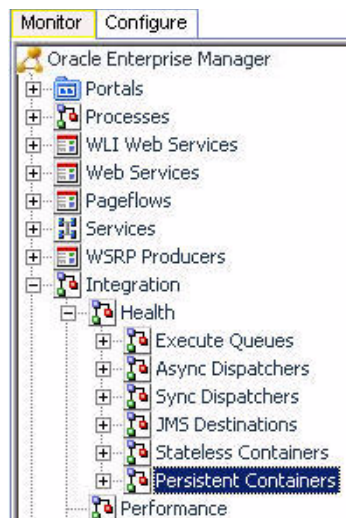
CAMM™ presents these metrics in a table format in the Main Display Window when you select the Stateless Containers node. Graphical representation of one metrics, Stateless EJB in use, is displayed below the table.

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

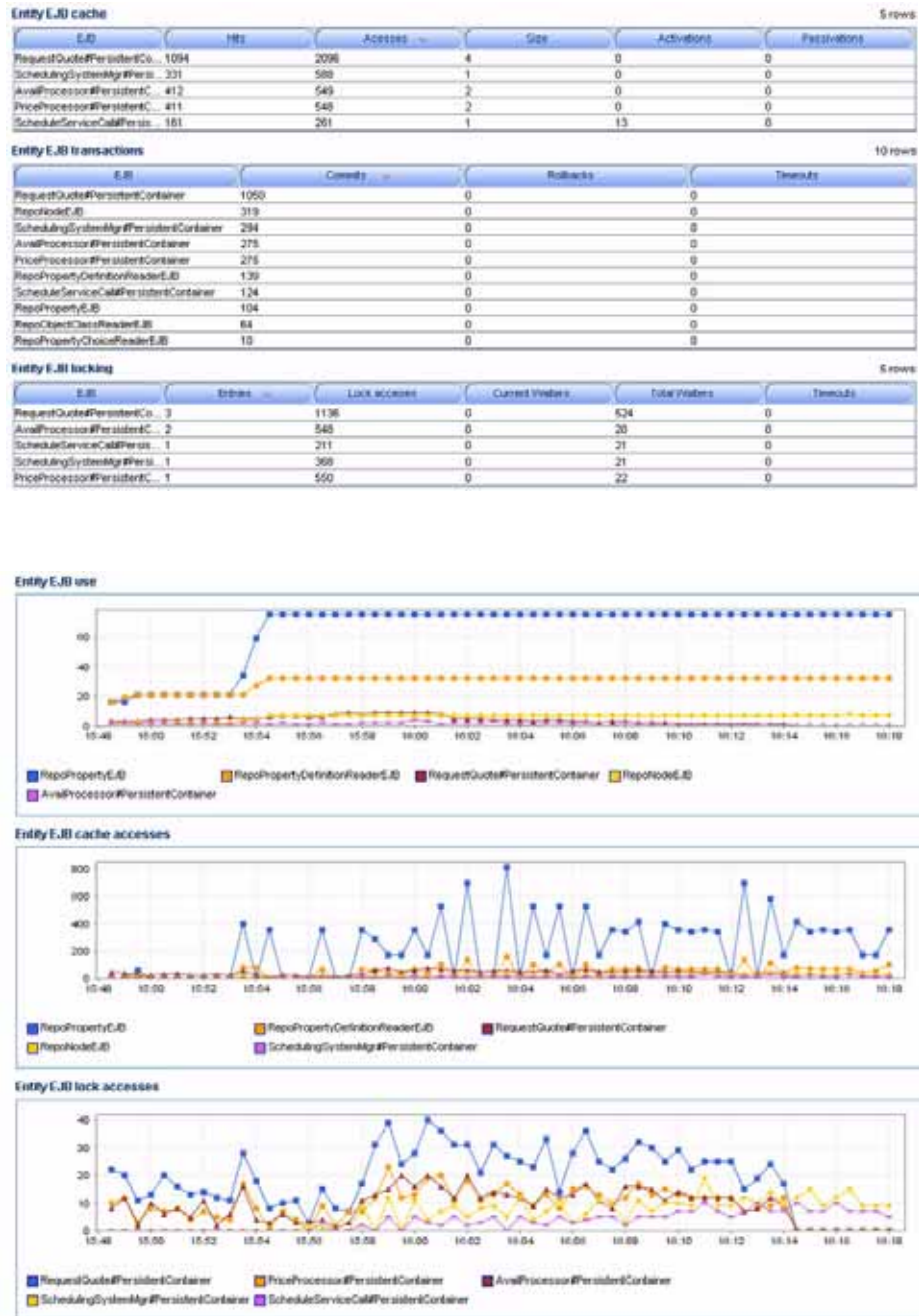
Persistent Containers

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.

Figure 4-92: Persistent Containers Summary



EJB	In Use	Idle	Waits	Timeouts
RequestQueue#PersistentContainer	4	0	0	0
AvailProcessor#PersistentContainer	2	4	0	0
PriceProcessor#PersistentContainer	2	0	0	0
ScheduleServiceCall#PersistentCo...	1	2	0	0
SchedulingSystemMy#Persistent...	1	2	0	0



The Persistent Containers Summary includes four different tables:

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

Entity EJB Activity table includes the following information:

Table 4-56: 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 4-57: 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, we would like to see low level of activity in these metrics.

Entity EJB Transactions table includes the following information:

Table 4-58: Entity EJB Transactions

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.

Entity EJB Locking table includes the following information:

Table 4-59: Entity EJB Locking

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 two metrics can indicate possible performance problems caused by EJB Locking. Ideally, 0s should be displayed for these metrics.

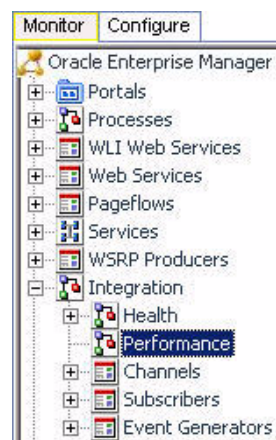
CAMM™ presents these metrics in a table format in the Main Display Window when you select the Persistent Containers node. Graphical representations of three metrics, Entity EJB in use, Entity EJB cache access, and Entity EJB lock access, are displayed below the table.

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

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.

Figure 4-93: Performance Summary



Node	ServiceID	Arrivals	Active	Elapsed Time (ms)	Completions	Aborts	Exceptions
ScheduleService	AcstWebcontrols/resources/ScheduleServiceCall.jpd	129	0	1822	129	0	0
Message Event	AcstWebcontrols/resources/ScheduleServiceCall.jpd	258	0	1055	129	258	0
Request Available Times	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	420	129	0	0
Get Available Times	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	282	129	0	0
Add Service Call to Scheduling System	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	92	129	0	0
Reserve Times	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	53	129	0	0
Customer Address -> WFS CustomerAddress	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	51	129	0	0
WFS TimeSlot[] -> TimeSlot[]	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	33	129	0	0
Release Reserved Times	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	17	129	0	0
Update Trouble Ticket	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	6	129	0	0
Return Status	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	4	129	0	0
Start Time Reservation Timer	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	3	129	0	0
Wait for Service Reservation	AcstWebcontrols/resources/ScheduleServiceCall.jpd	129	0	0	0	258	0
Finish	AcstWebcontrols/resources/ScheduleServiceCall.jpd	129	0	0	129	0	0
Schedule Request	AcstWebcontrols/resources/ScheduleServiceCall.jpd	0	0	0	129	0	0

The Performance Summary includes two 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 4-60: 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.
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 node.
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 vs. Elapsed Time (arrival rate vs. 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 4-61: 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.

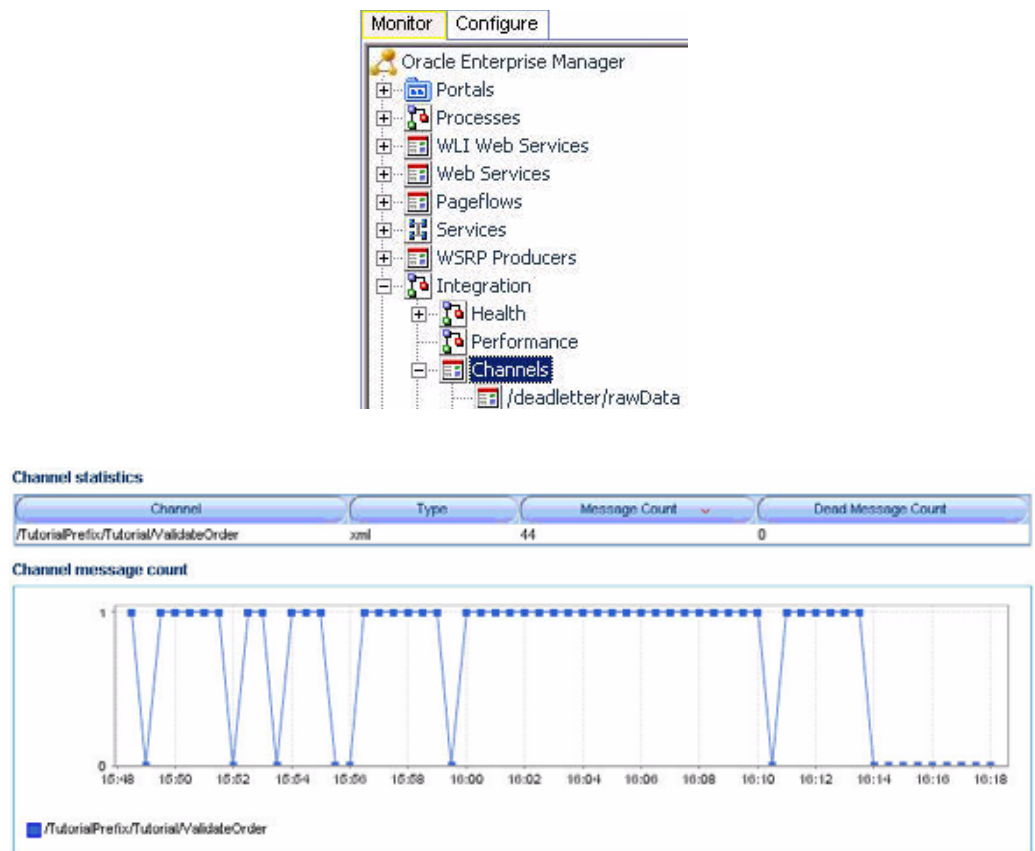
Table 4-61: Performance - Events Node Summary (Continued)

Column / Metric	Description
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.

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.

Figure 4-94: Channels Summary



The Channels Summary includes the following information:

Table 4-62: 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.

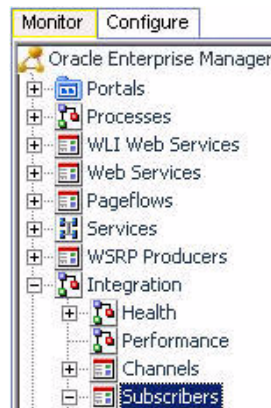
Tip: Increasing count in the Dead Message Count metric may indicate a configuration issue. When the Message Broker is unable to determine the URI to send a message to, the message is sent to the appropriate deadletter channel. Check to make sure the URI configured for the channel is reachable.

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

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.

Figure 4-95: Subscribers Summary



Subscriber information 4 rows

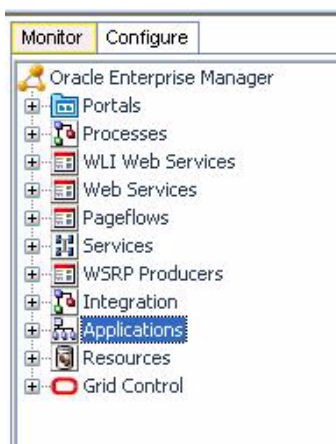
/TutorialPrefix/Tutorial/StopQuote	
Control Name	mbSubValidate
Filter String	n/a
Method Name	onMessage
Subscriber URI	/WLISP3Web/requestquote/RequestQuote.jspd
/TutorialPrefix/Tutorial/ValidateOrder	
Control Name	n/a
Filter String	n/a
Method Name	subscription
Subscriber URI	/WLISP3Web/requestquote/services/ValidateOrder.jspd

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

Applications

The Applications node under Oracle™ Tree contains information about all deployed applications in the managed domain. By selecting the Applications node under the Oracle™ Tree, CAMM™ displays the Applications Summary.

Figure 4-96: Applications Summary





The Applications Summary includes the following information:

Table 4-63: Applications Summary

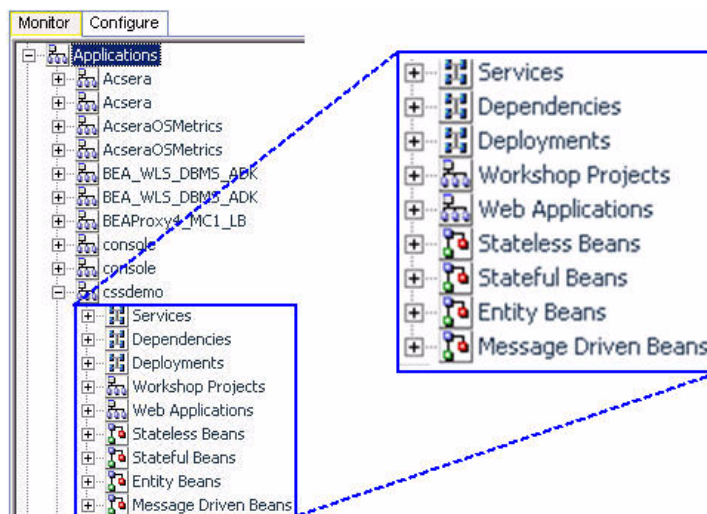
Column / Metric	Description
Application	Name of application.
Status	Operations status for a specific application.
Response Time (ms)	Average response time in milliseconds for a specific application. This is the average of response times of all JSPs and servlets contained in the deployment archive.
Invocation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

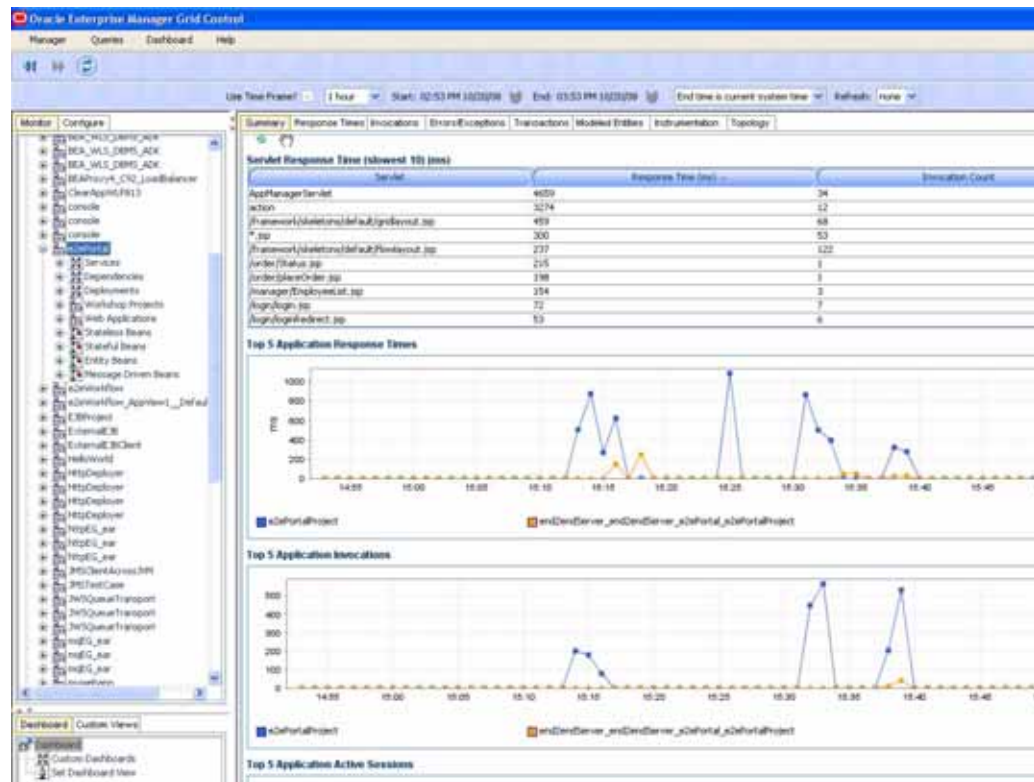
Tip: Application is a packaging unit in J2EE™. Each EAR, WAR, and JAR files deployed to the application server is considered an individual application. These metrics track performance and arrival rate of these entities.

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

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

Figure 4-97: Expanded Tree for cssdemo Application and cssdemo Summary





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

The Applications Summary includes the following tabs:

Table 4-64: 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 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	The errors metrics associated with the selected portal.

Table 4-64: Applications Summary Tabs (Continued)

Tab Name	Description
Transactions	The transaction events associated with the selected portal and children below. Refer to Triage from Dashboard for more information.
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 on the children nodes under `cssdemo`, the relevant data is displayed in the Main Display Window. Application response time and invocations measurements can be reached by clicking on the panels in the Main Display Window.

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

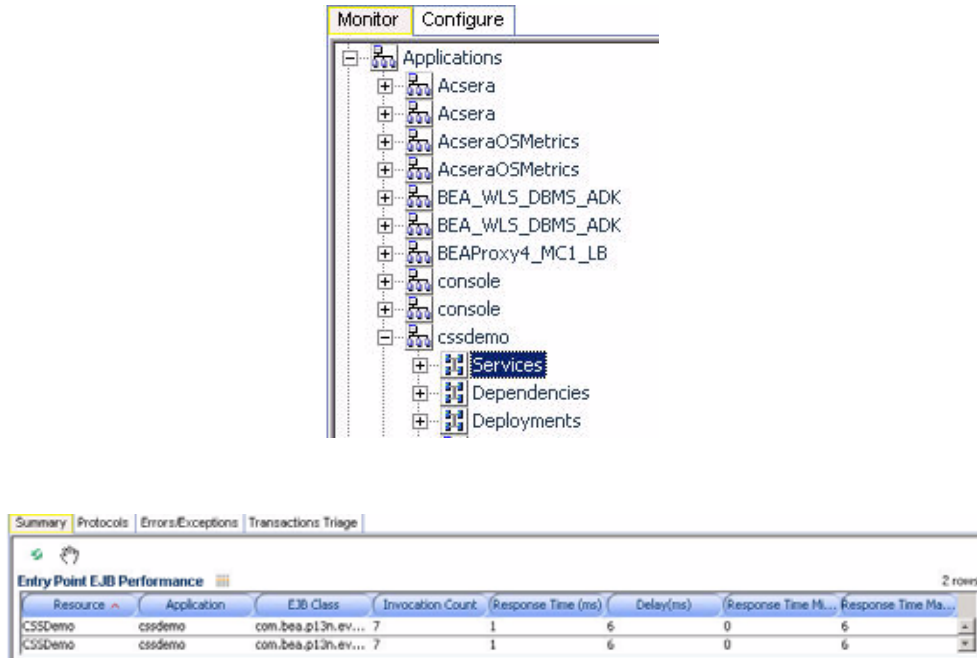
- Services
- Dependencies
- Deployments
- Workshop Projects
- Web Applications
- Stateless Beans
- Stateful Beans
- Entity Beans
- Message Driven Beans

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.

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. See [Figure 4-98](#). CAMM™ displays the performance data associated with various entry points associated with the selected application.

Figure 4-98: Services - Summary View



Tip: The children nodes under the Services node include entry point specific performance data. To understand the meaning of these metrics, please refer to [Services on page 180](#).

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 4-65: Dependencies Column Descriptions

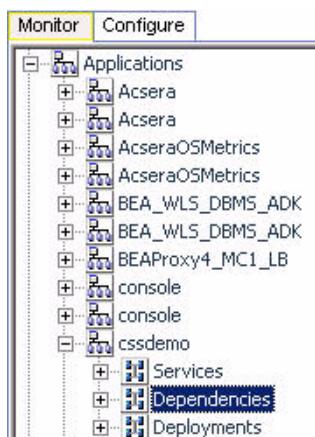
Column / Metric	Description
Name	Display name of the component or resource used by the application. If this is undefined in the Deployment Descriptor, the reference name for the component is used.

Table 4-65: Dependencies Column Descriptions (Continued)

Column / Metric	Description
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.

Figure 4-99: Dependencies - Associated Components



Resource Dependencies

Name	Reference	
/cssWeb/controls/resources/cssWeb/controls/resources/ScheduleService	/cssWeb/controls/resources/cssWeb/controls/resources/ScheduleService	com.bea.control.Process
ConversationDataSource	ConversationDataSource	javax.sql.DataSource
ConversationDataSource	ConversationDataSource	javax.sql.DataSource
ConversationDataSource	ConversationDataSource	javax.sql.DataSource
ConversationDataSource	ConversationDataSource	javax.sql.DataSource
ConversationDataSource	ConversationDataSource	javax.sql.DataSource
ConversationDataSource	ConversationDataSource	javax.sql.DataSource
ConversationDataSource	ConversationDataSource	javax.sql.DataSource
ConversationDataSource	ConversationDataSource	javax.sql.DataSource
ConversationDataSource	ConversationDataSource	javax.sql.DataSource
SchedulingSystem.bean.SyncDispatcher	SchedulingSystem.bean.SyncDispatcher	Session
bean.SyncDispatcher	bean.SyncDispatcher	Session
bean.SyncDispatcher	bean.SyncDispatcher	Session
bean.SyncDispatcher	bean.SyncDispatcher	Session
bean.SyncDispatcher	bean.SyncDispatcher	Session
bean.SyncDispatcher	bean.SyncDispatcher	Session
bean.SyncDispatcher	bean.SyncDispatcher	Session
bean.SyncDispatcher	bean.SyncDispatcher	Session
bean.SyncDispatcher	bean.SyncDispatcher	Session
com.bea.control.TimerControl.bean.PersistentContainer	com.bea.control.TimerControl.bean.PersistentContainer	Entity
com.bea.control.TimerControl.bean.StatelessContainer	com.bea.control.TimerControl.bean.StatelessContainer	Session
com.bea.p13n.controls.ejb.events.EventServiceEJBControl.jcix	com.bea.p13n.controls.ejb.events.EventServiceEJBControl.jcix	Session
com.bea.p13n.controls.ejb.events.EventServiceEJBControl.jcix	com.bea.p13n.controls.ejb.events.EventServiceEJBControl.jcix	Session
com.bea.p13n.controls.ejb.property.EntityPropertyManagerEJBControl.jcix	com.bea.p13n.controls.ejb.property.EntityPropertyManagerEJBControl.jcix	Session

The Dependencies node can be further expanded by clicking on the + icon. The children nodes of the Dependencies node are organized by type. Here are the list of dependency types and their descriptions:

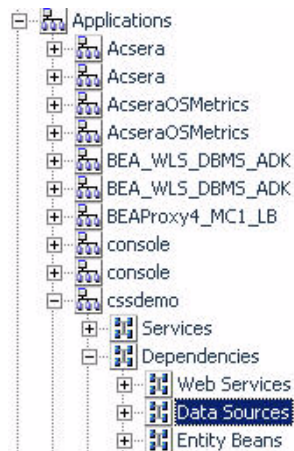
Table 4-66: Dependency Types

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 on the + icons. The expanded tree includes specific components and share resources used by the application. See [Figure 4-100](#).

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

Figure 4-100: Dependency - Data Sources





For more information on the metric description, please refer to [Metric Types on page 103](#).

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. The following is a list of tabs available as part of this summary view and their descriptions.

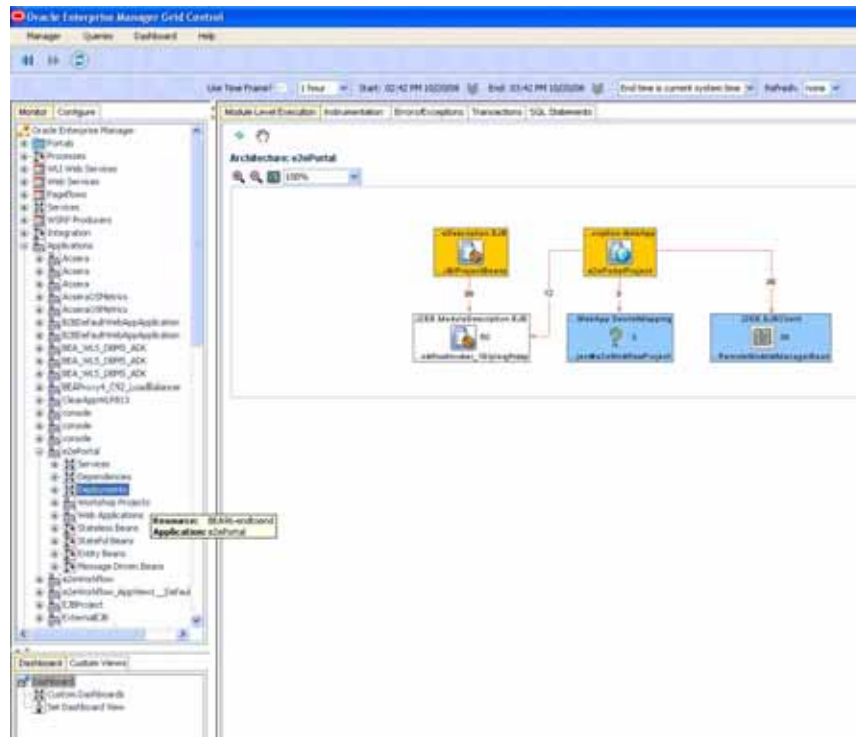
Table 4-67: Deployment Tabs

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.

[Figure 4-101](#) depicts the Deployment node for a specific application in CAMM™.

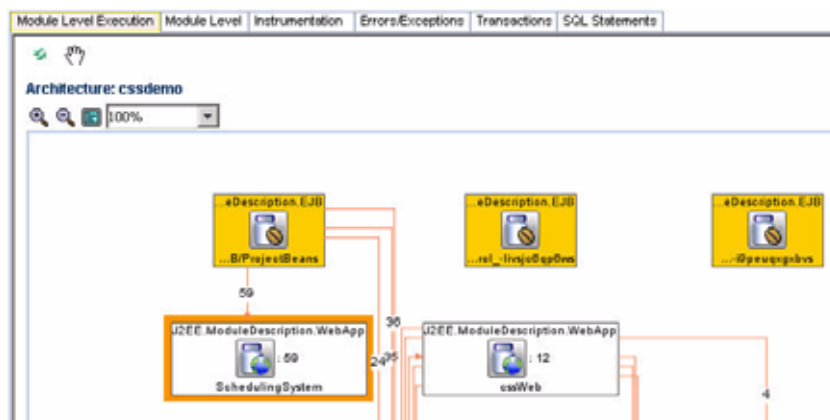
Active module-level call path is displayed as the default view for the Deployments node of a selected application.

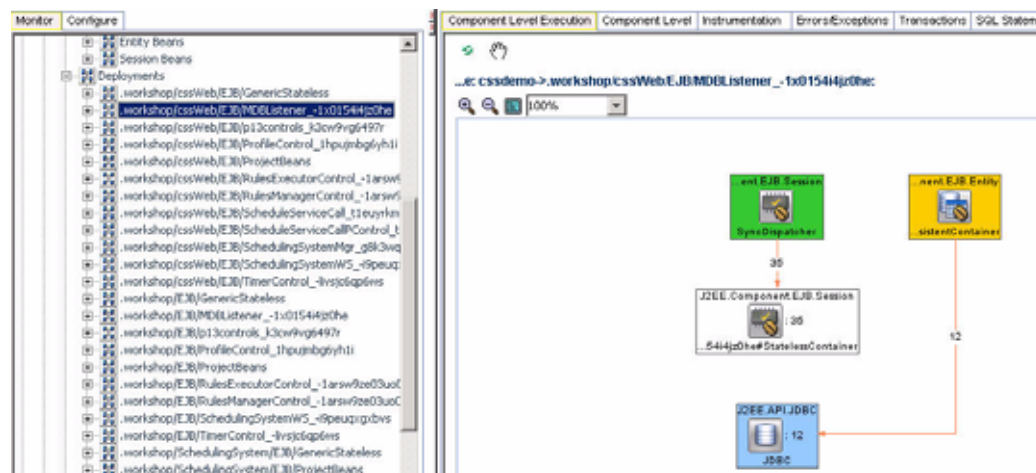
Figure 4-101: Deployment Node



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

Figure 4-102: Deployment - Module Detail





Expand the Deployments node by clicking on the + 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.

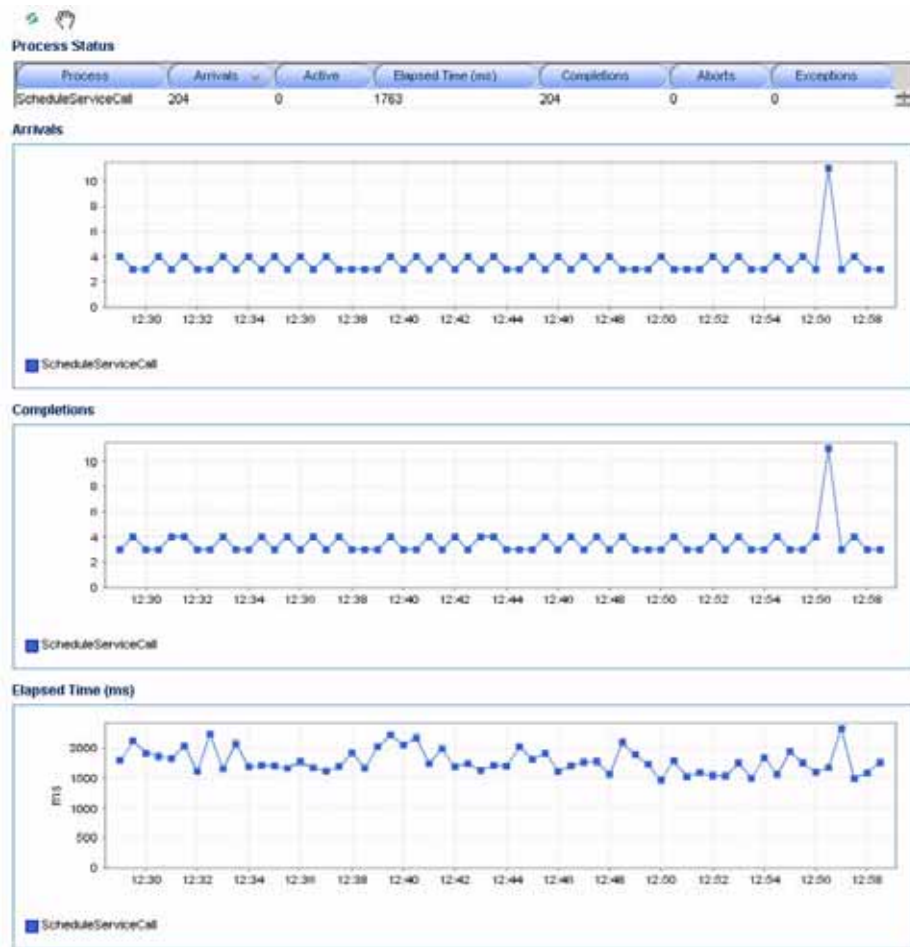
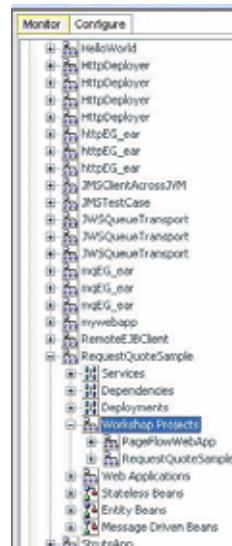
Tip: Use the active call path diagram as a guide to identify entities with performance data. If an entity does not have performance data, CAMM™ displays *No data available for the selected time frame* in the Main Display Window.

Workshop Projects

The Workshop Projects node includes performance information about modules and components created using the BEA WebLogic® Workshop. These modules and components include WebLogic® Integration processes, WebLogic® Integration web services, and WebLogic® Portal pageflows. [Figure 4-103](#) shows the Workshop Projects node and its children nodes associated with the cssdemo application:

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

Figure 4-103: Workshop Projects Nodes and It's Children



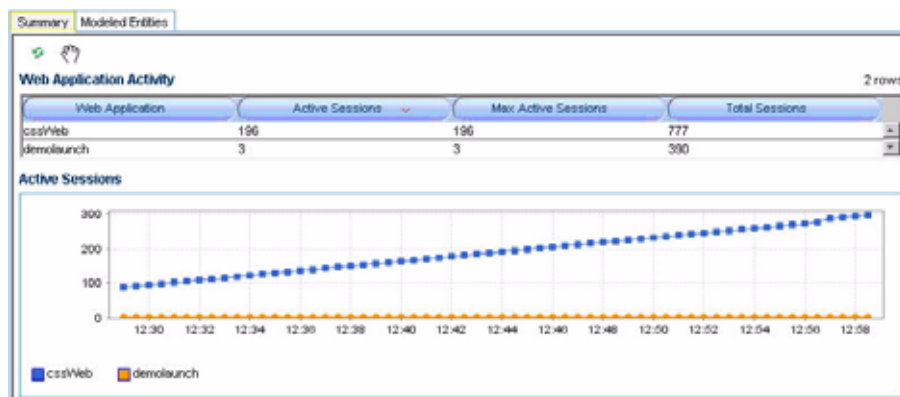
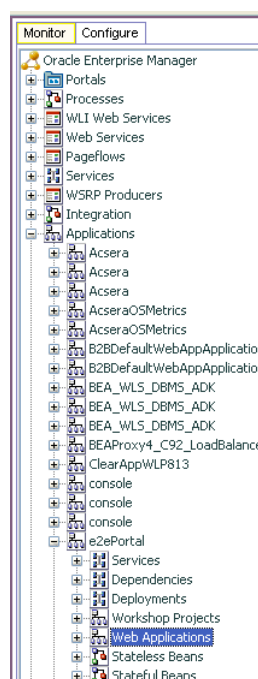
When you select a specific children node, CAMM™ displays detailed performance information.

For metric descriptions, please refer to [Table 4-28](#) and [Table 4-36](#) associated with WLI Processes, WLI Web Services, and WebLogic® Portal Pageflows in this guide.

Web Applications

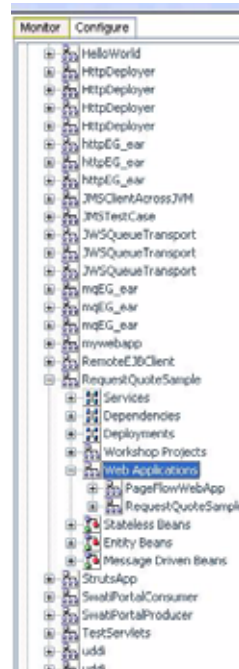
The Web Applications node includes performance information related to the Web Applications modules and components associated with the selected application. Click the Web Applications node to reveal a performance summary in the Main Display Window. Click the + icon to expand the Web Applications node to reveal various web modules deployed as part of this application. See [Figure 4-104](#).

Figure 4-104: Expanded Web Applications Node with Performance Summary



Click the + icon to expand on a specific web module and reveal different groupings for web components. For example, Pageflows, Struts Modules and Servlets. Clicking on 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 + icon to reveal more detailed information as shown in [Figure 4-105](#). Fully expanded Web Applications node contains all web modules organized by type.

Figure 4-105: Fully Expanded Web Applications Node

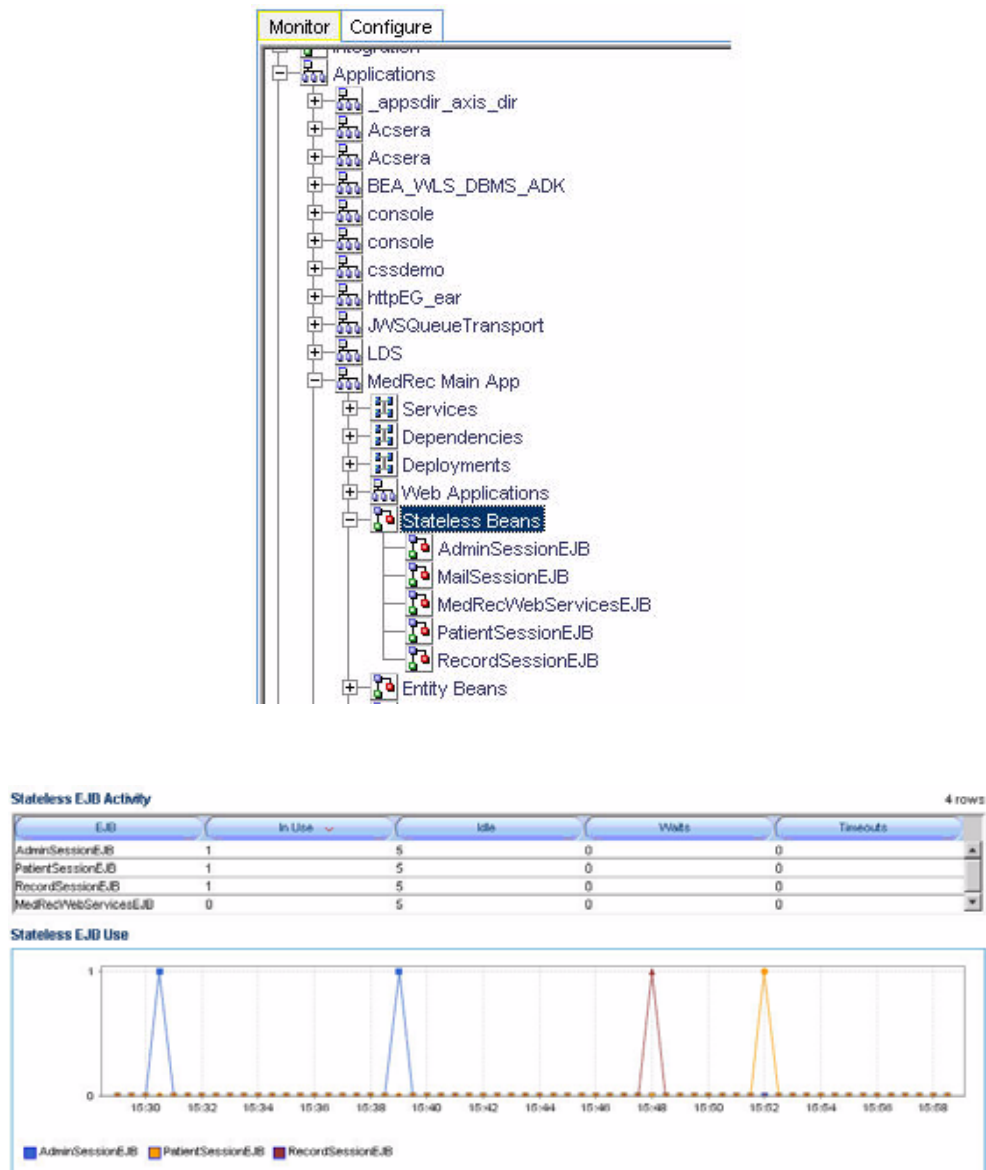


Detailed performance information at the individual pageflow, struts action, and servlet levels will be displayed when you click the lowest level nodes.

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 + icon to expand the Stateless Beans node to reveal various stateless EJBs deployed as part of this application. See [Figure 4-106](#).

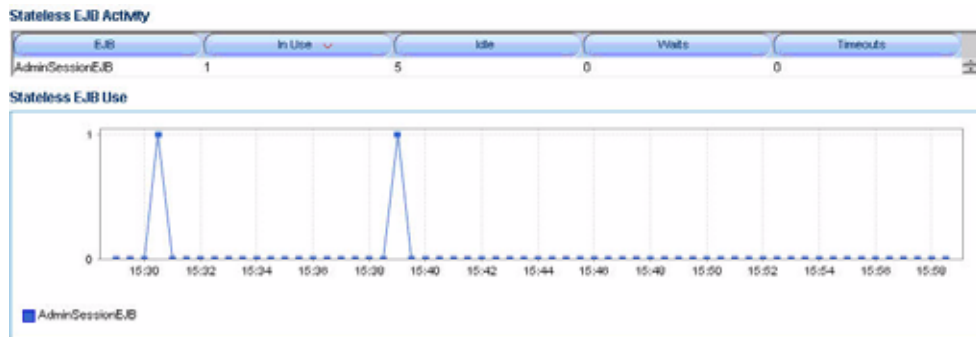
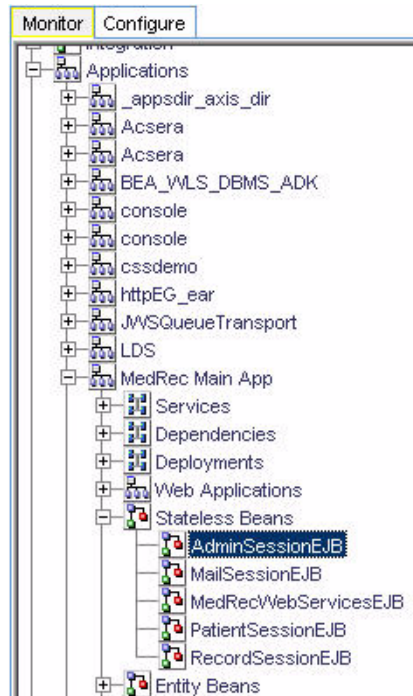
Figure 4-106: Expanded Stateless Beans Node with Activity Summary



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

Selecting a specific Stateless Bean node triggers CAMM™ to display detailed activity metrics.

Figure 4-107: Further Expand Stateless Beans



The detailed view contains the following activity metrics:

Table 4-68: Stateless Beans Detail View

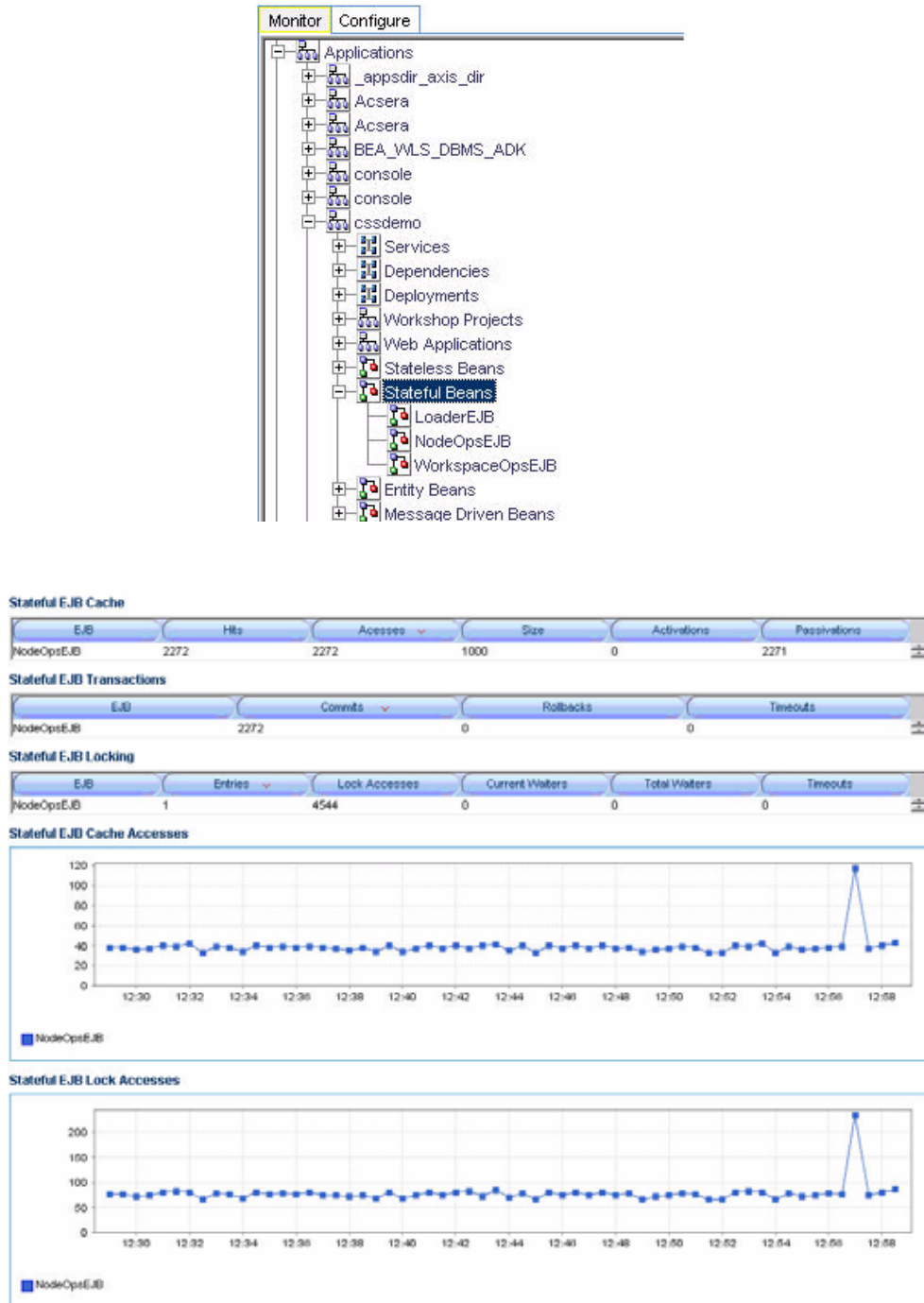
Column / Metric	Description
EJB	Name of the stateless EJB.
In Use	Number of instances for a specific stateless EJB currently being used from the free pool. [Snapshot Count]
Idle	Number of instances for a specific stateless EJB currently in the idle state in the free pool. These bean instances are available 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.

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 + icon to expand the Stateful Beans node to reveal various stateful EJBs deployed as part of this application. See [Figure 4-108](#).

Figure 4-108: Expanded Stateful Beans Node With Activity Summary



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

The Stateful EJB Summary includes three different tables:

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

Stateful EJB Cache

Stateful EJB Cache table includes the following information:

Table 4-69: 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, we would like to see low level of activity in these metrics.

Stateful EJB Transactions

Stateful EJB Transactions table includes the following information:

Table 4-70: 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.

Stateful EJB Locking

Stateful EJB Locking table includes the following information:

Table 4-71: Stateful EJB Locking

Metrics	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 two 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.

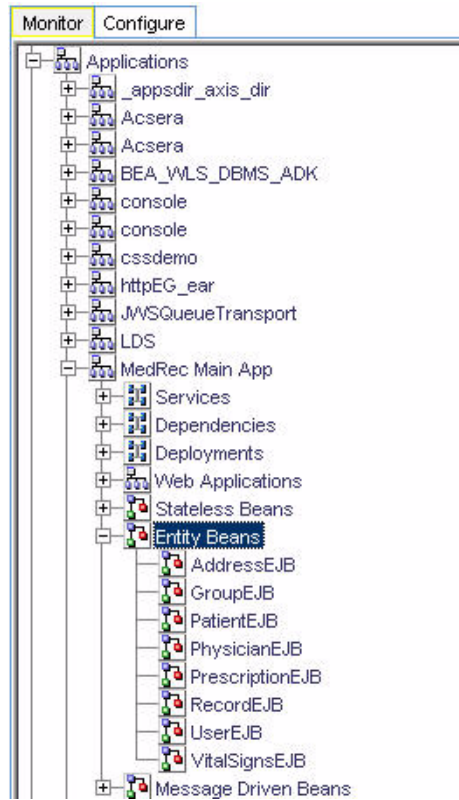
By looking at the activities related to Stateful EJBs, you can determine if there any abnormal activities associated with Stateful EJBs.

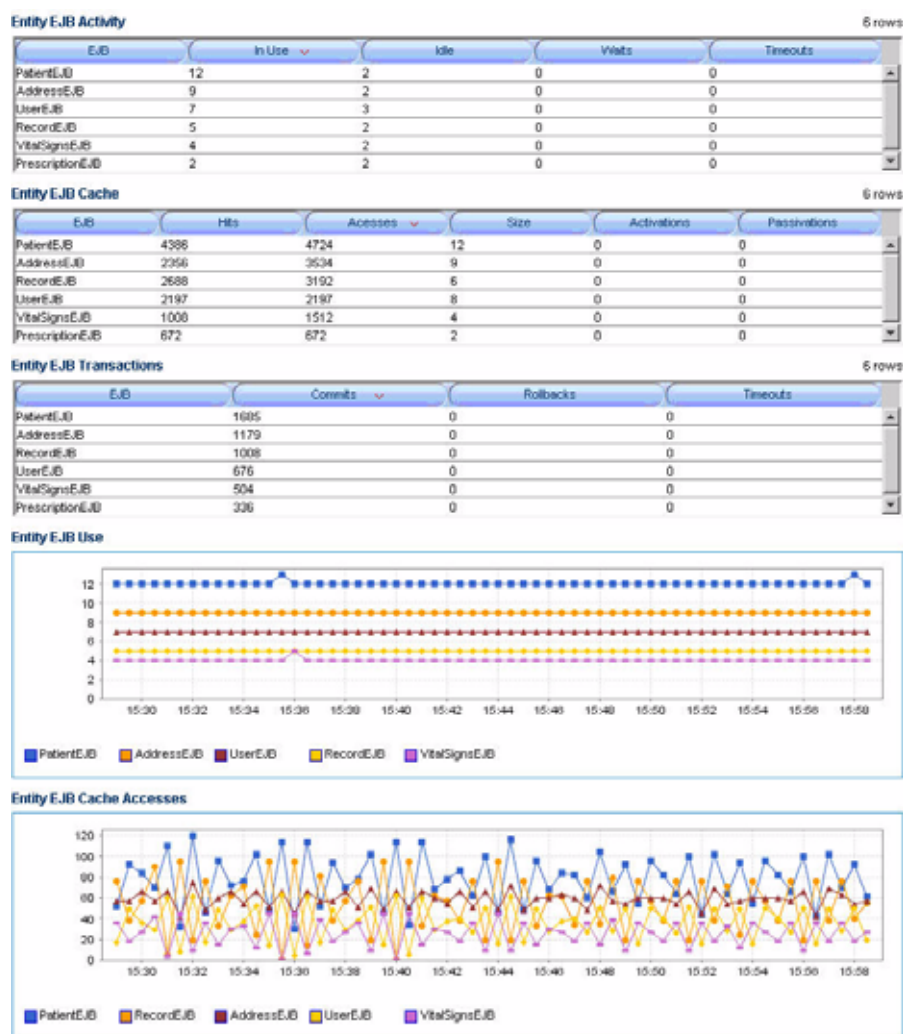
Note: The metrics reported in the Stateful Beans node are reported by the MBean (Management Bean) of the EJB container. These activity metrics can be used for checking the overall health of the EJB container. When the EJB container is restarted, these metrics are reset.

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 + icon to expand the Entity Beans node to reveal various Entity EJBs deployed as part of this application. See [Figure 4-109](#).

Figure 4-109: Expanded Entity Beans Node With Activity Summary

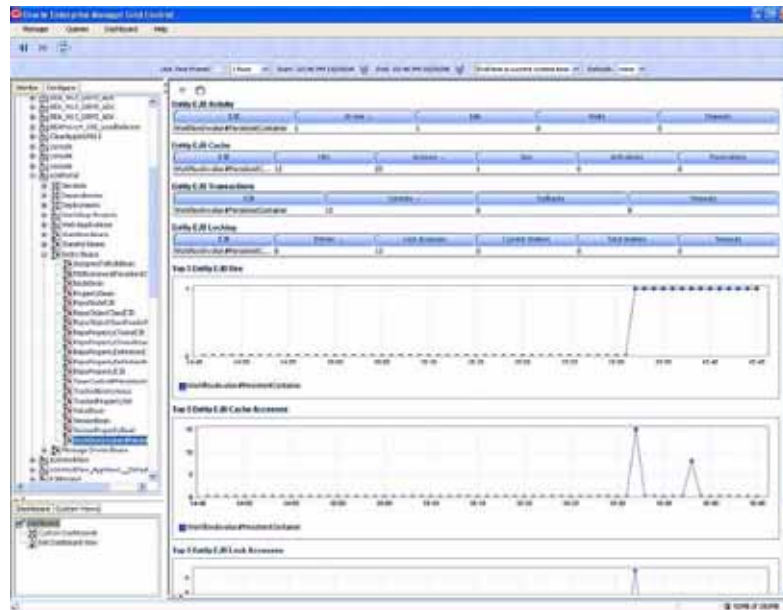




You can further select individual nodes to obtain detailed activity information.

Selecting a specific Entity Bean node triggers CAMM™ to display detailed activity metrics.

Figure 4-110: Entity Beans - Detail Activity



The Entity EJB Summary includes four different tables:

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

Entity EJB Activity

Entity EJB Activity table includes the following information:

Table 4-72: 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 waiting to be serviced by the EJB container. Ideally, 0 should be indicated for these metrics.

Entity EJB Cache

Entity EJB Cache table includes the following information:

Table 4-73: 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, we would like to see a low level of activity in these metrics.

Entity EJB Transactions

Entity EJB Transactions table includes the following information:

Table 4-74: Entity EJB Transactions

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 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.

Entity EJB Locking

Entity EJB Locking table includes the following information:

Table 4-75: Entity EJB Locking

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 two 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 three 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 on the + icon next to Entity Beans node. You can get the same summary as described above for a specific Entity EJB.

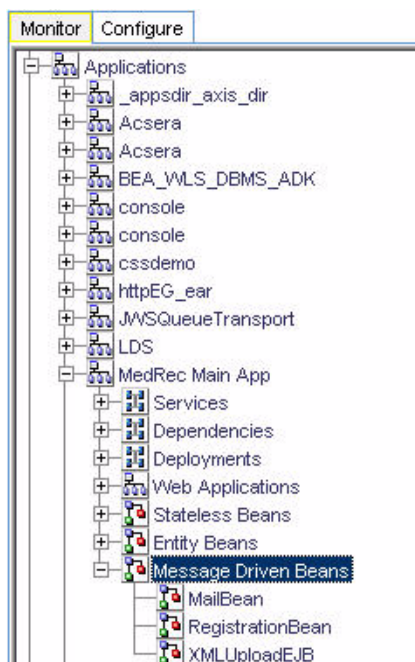
By looking at the activities related to Entity EJBs, you can determine if there any abnormal activities associated with Entity EJBs.

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.

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 + icon to expand the Message Driven Beans node to reveal various message driven EJBs deployed as part of this application. See [Figure 4-111](#).

Figure 4-111: Expanded Message Driven Beans Node With Activity Summary



Message Driven EJB Activity					2 rows
EJB	In Use	Idle	Visits	Timeouts	
RegistrationBean	0	5	0	0	-
XMLUploadEJB	0	1	0	0	-

Message Driven EJB Transactions				2 rows
EJB	Commits	Rollbacks	Timeouts	
RegistrationBean	169	0	0	-
XMLUploadEJB	169	0	0	-

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

The Message Driven EJB Summary includes two different tables:

- Message Driven EJB Activity
- Message Driven EJB Transactions

Message Driven EJB Activity

Message Driven EJB Activity table includes the following information:

Table 4-76: Message Driven EJB Activity

Metrics	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 waiting to be serviced by the EJB container. Ideally, 0 should be indicated for these metrics.

Message Driven EJB Transactions

Message Driven EJB Transactions table includes the following information:

Table 4-77: Message Driven EJB Transactions

Metrics	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.

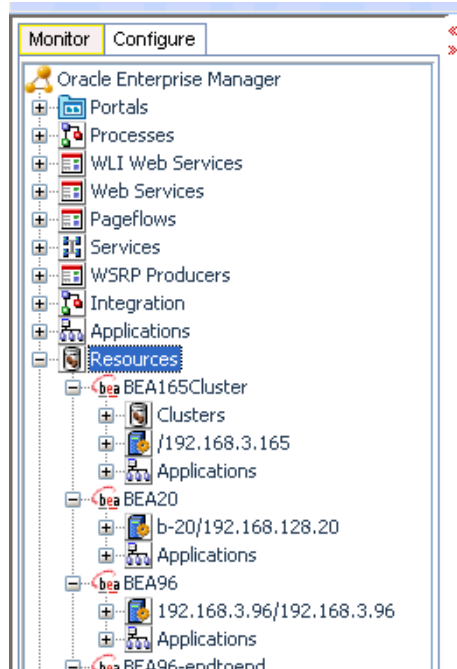
By looking at the activities related to Message Driven EJBs, you can determine if there are any abnormal activities associated with Message Driven EJBs.

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.

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.

Figure 4-112: Expanded Resources Tree for WebLogic®



The Resources tree includes the following nodes:

Table 4-78: WebLogic® Resources Tree

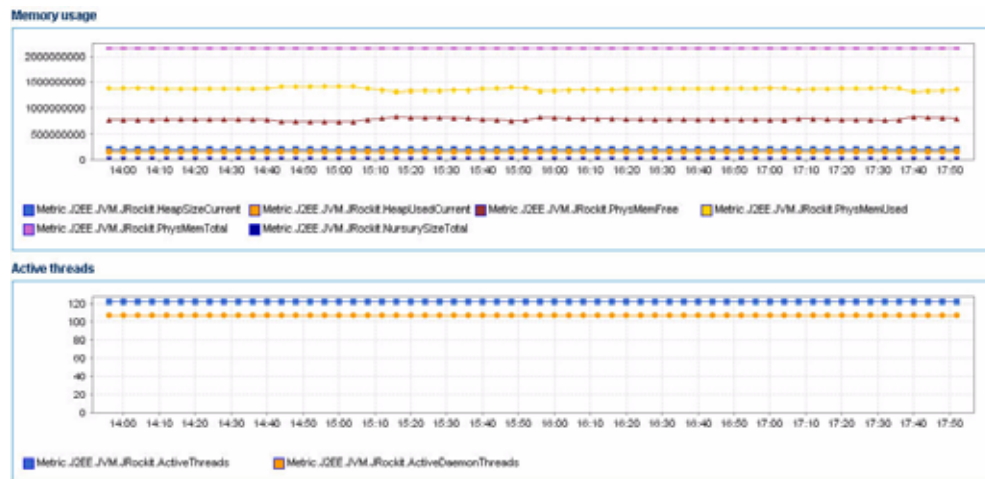
Example Node	Description
CSS Domain	Name of the WebLogic® Domain configured.
b-15/ 192.168.128.15	ID of the physical machine.
OS Metrics	Summary view of three OS metrics collected by OS Agent.
CPU	CPU usage data.
Memory	Memory usage data.
Disk	Disk usage data.
cgServer	Name of the WebLogic® Server configured.

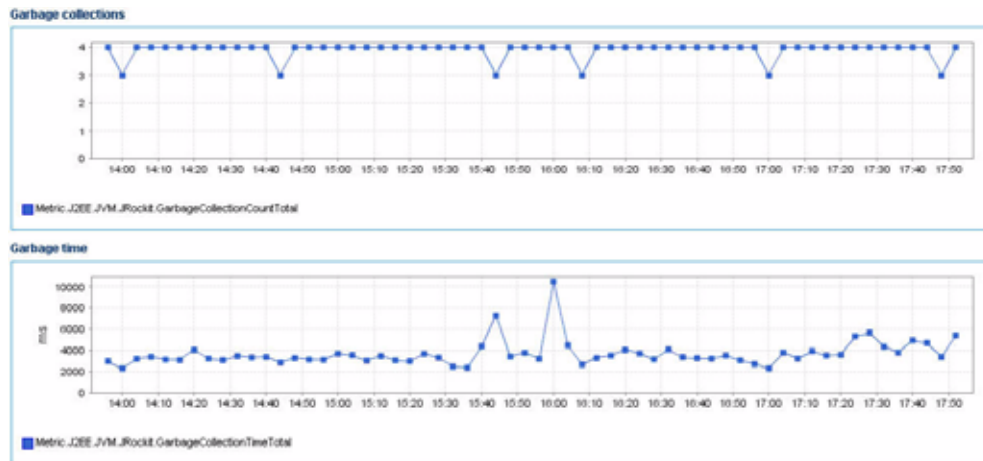
Table 4-78: WebLogic® Resources Tree (Continued)

Example Node	Description
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.
JRokit	JRokit information including Heap Size for this server.
Modeling Status	Entities modeled by CAMM™ for this server.
CAMM™ Modules	Status of the CAMM™ JAgent Module for this server.

Expand these nodes by clicking on the + icon next to the node name to get more information. See examples in [Figure 4-113](#):

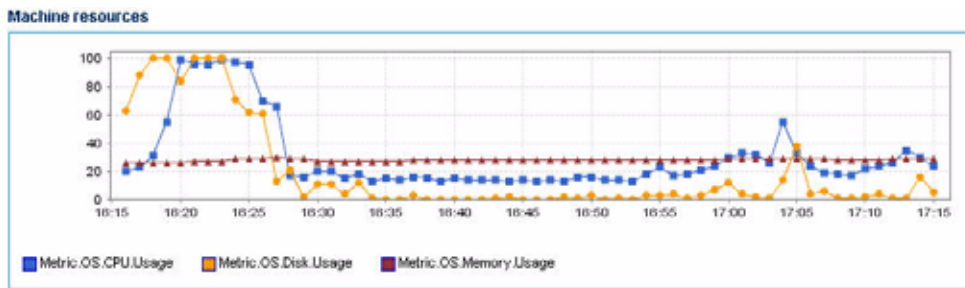
Figure 4-113: JRokit related graphs under JRokit Node





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.

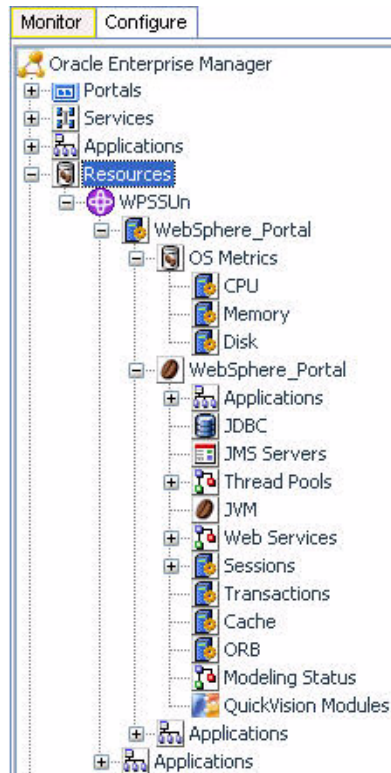
Figure 4-114: OS Agent Metrics



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.

Figure 4-115: Expanded Resources Tree for WebSphere®



The Resources tree includes the following nodes:

Table 4-79: WebSphere® Resources Tree

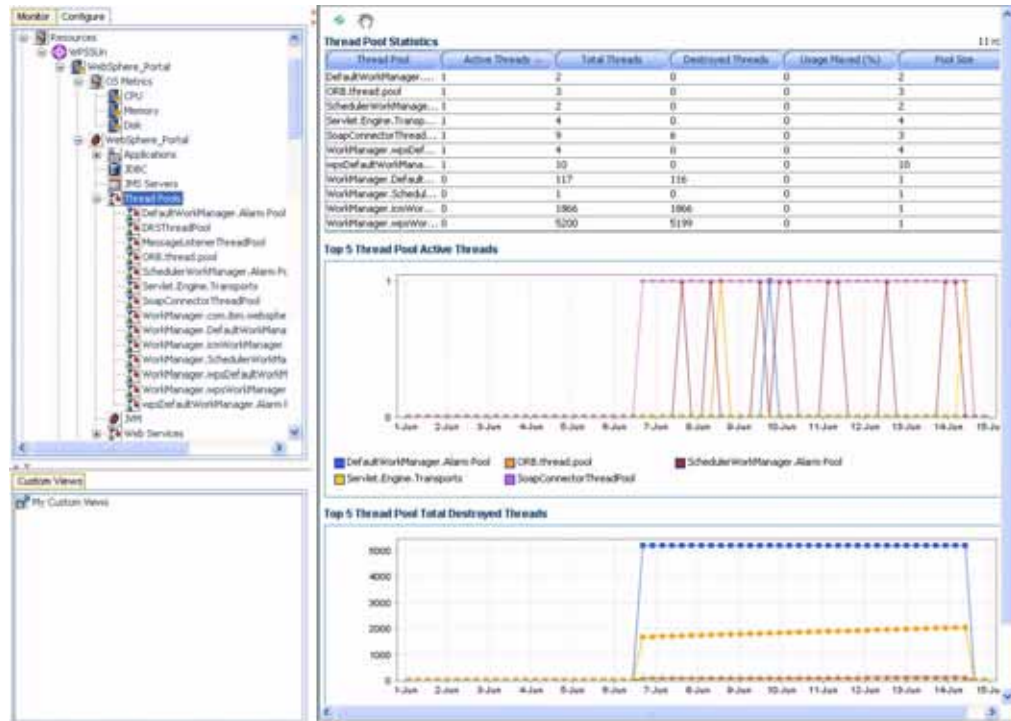
Example Node	Description
WPSUn	Resource name. For example, WPSUn.
WebSphere Portal	Machine name. For example, WebSphere_Portal.
OS Metrics	Summary view of three OS metrics collected by OS Agent.
CPU	CPU usage data.
Memory	Memory usage data.
Disk	Disk usage data.

Table 4-79: WebSphere® Resources Tree (Continued)

Example Node	Description
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 requests.
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	The modeled entities for the container.
CAMM™ Modules	Status of the CAMM™ JAgent Module for this server.
Applications	Performance information about the applications deployed in the container.

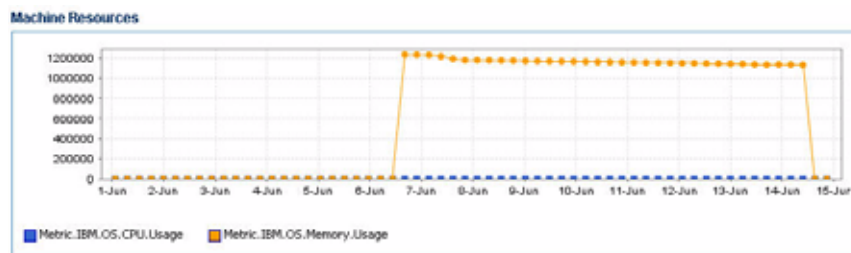
Expand these nodes by clicking on the + icon next to the node name to get more information. See examples in [Figure 4-116](#):

Figure 4-116: Thread Pools Expanded Node



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

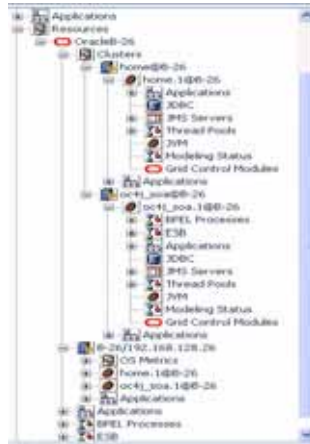
Figure 4-117: OS Metrics



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.

Figure 4-118: Expanded Resources Tree for Oracle®



The Resources tree includes the following nodes:

Table 4-80: WebSphere® Resources Tree

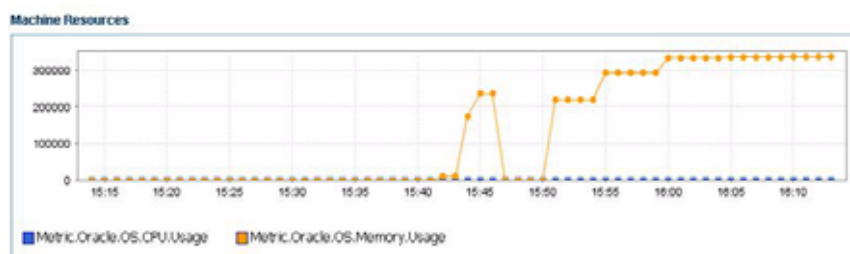
Example Node	Description
Managed System Resource Name	Top-level Resource name. For example, oc4j_soa.
Oracle AS Server	Machine name which can be navigated to both within or outside a cluster. For example, oc4j_soa@192.168.1.119 which includes both the server name and the host server IP address.
OS Metrics	Summary view of three OS metrics collected by OS Agent.
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.

Table 4-80: WebSphere® Resources Tree (Continued)

Example Node	Description
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	The modeled entities for the container.
CAMM™ Modules	Status of the CAMM™ JAgent Module for this server.
Applications	Performance information about the applications deployed in the container.

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

Figure 4-119: OS Metrics



S

Custom Metrics

The Custom Metrics node under Oracle™ tree contains all custom metrics defined by you. 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.

Figure 4-120: Custom Metrics - Expanded

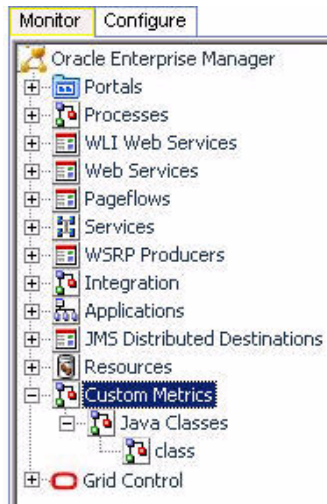
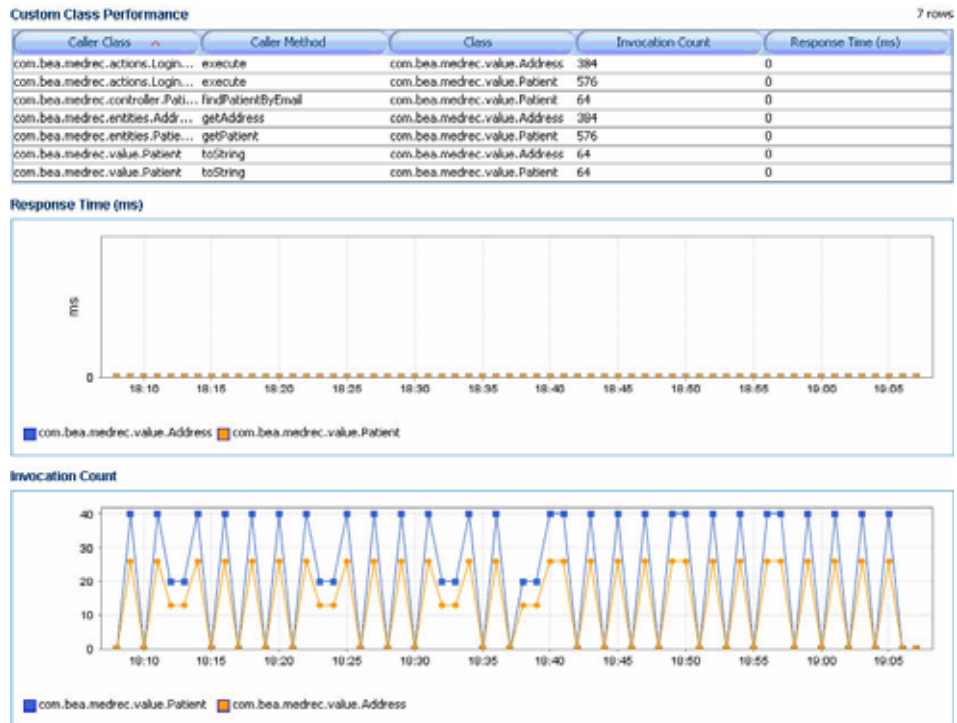


Figure 4-121: Custom Metrics Node Performance Summary



The following is a list of columns in the Custom Class Performance table and their descriptions:

Table 4-81: Custom Class Performance

Column / Metric	Description
Caller Class	The fully qualified name of the class that is making the inbound call.
Caller Method	The method name in the class that is making the inbound call.
Class	The fully qualified name of the class that is the destination of the inbound call.
Invocation Count	The total number of times the inbound call is made.
Response Time (ms)	The average response time of the inbound call in milliseconds.

Grid Control™

The Grid Control™ node under Oracle™ tree contains information for CAMM™ environment for the monitored WebLogic® domain, WebSphere® cell, or Oracle AS® cluster. Select the Grid Control™ node to see the CAMM™ JAgent status for the WebLogic® domain. The CAMM™ JAgent status includes the following:

Table 4-82: CAMM™ JAgent Status

Column / Metrics	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™ JAgent installation status.
Agent Activated	CAMM™ JAgent activation status.
Agent Status	CAMM™ JAgent operational status.

Table 4-82: CAMM™ JAgent Status (Continued)

Column / Metrics	Description
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™'s RMI registry.
Manager RMI Registry Port	Port number of the CAMM™'s 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™ JAgent major version.
Agent Minor Version	CAMM™ JAgent minor version.
Agent Build ID	CAMM™ JAgent 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.

5

Exploring Configuration Workspace

This chapter includes the following topics:

- [Resource Configuration](#)
- [User Configuration](#)
- [Service Level Objectives by Name](#)
- [Service Level Objectives by Metrics](#)
- [Service Level Objectives by Entity Type](#)
- [Action Configuration](#)

Resource Configuration

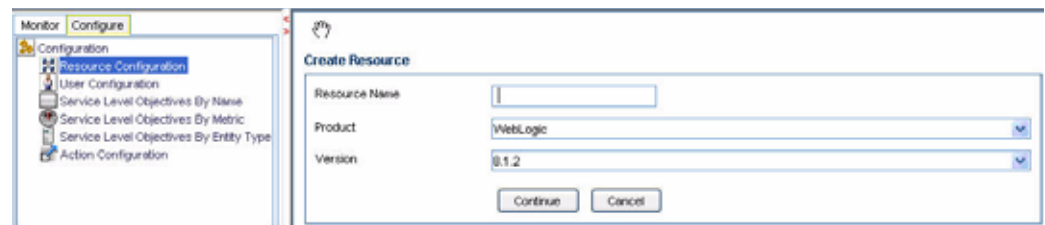
Setting up a performance monitoring environment with Oracle Enterprise Manager Grid Control - Composite Application Monitor and Modeler (CAMM™) requires minimal effort. The application requires that you provide the location of the WebLogic® Domain, Oracle® Cluster, or WebSphere® Cell Administration Server to CAMM™. This is done in the Resource Configuration tab in the configuration tree.

Figure 5-1: Resource Configuration



1. Click **Create New Resource** to add new resources into CAMM™'s monitored environment. You must fill out a few forms to complete the addition.

Figure 5-2: New Resource Creation Form



The form allows you to specify basic resource information. Refer to [Table 5-1](#) for a description of the fields.

Table 5-1: Resource Creation Form

Input	Description
Resource Name	A name that uniquely identifies the monitored resource.
Product	Type of the product monitored.
Version	Version of the product monitored.

2. Click **Continue** to complete the resource configuration.

Figure 5-3: Resource Configuration Form for WebLogic®

3. Click **Configure**.

Refer to [Table 5-2](#) to see an example configuration of a BEA WebLogic® Resource Configuration Form. It should be noted that Oracle® and IBM® resource configuration requirements are almost identical with the exception of some dehydration store authentication requirements for Oracle® SOA Suite clusters.

Table 5-2: BEA WebLogic® Based Resource

Property	Description
Resource Enabled	Enable / disable monitoring of this resource.
BEA JMX Metrics	Enable / disable monitoring of performance measurements from BEA MBeans via JMX. Additional configuration required to connect to BEA MBeans.
Instrumentation Metrics	Enable / disable Java Byte Code Instrumentation metrics.
OS Metrics	Enable / disable OS metrics.

4. Fill out the form as shown in [Figure 5-4](#) to configure the connection to the BEA Admin Server and Collect JMX Metrics.

Figure 5-4: Resource Configuration Form for BEA Admin Server

The figure shows two configuration forms. The first, titled "Resource Configuration", has four text input fields: "Resource" (WebLogic_D2), "Product" (WebLogic), "Version" (8.1.3), and "Metric Collector" (BEA JMX Metrics). The second, titled "Metrics Collection Configuration", has a dropdown for "Protocol" (j3), text inputs for "Host" (localhost), "Port" (7001), "Admin Server User ID" (weblogic), "New Password" (masked with asterisks), and "Retype New Password" (masked with asterisks). It also includes "OK" and "Cancel" buttons.

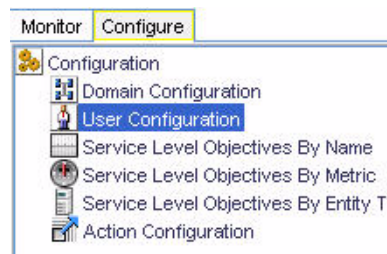
5. Click **OK** and then **Save**.

After this information is saved, CAMM™ automatically discovers the WebLogic® domain, its configuration, and all deployed applications. Next, CAMM™ analyzes the applications and calculates what metrics to use for monitoring.

User Configuration

Select the **User Configuration** node in the Configuration tree to manage user roles, create new users, and delete users. See [Figure 5-5](#).

Figure 5-5: User Configuration



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 three different types of roles:

- Admin
- Operator
- User

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.

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.

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 as shown in [Figure 5-6](#).
2. Select a user and double-click to change the configuration of an existing user account.

Figure 5-6: Change User Account Configuration

The screenshot shows a 'User Configuration' dialog box. At the top left is a 'Delete User' button. Below it is a 'Roles' section with a list box containing 'admin', 'operator', and 'user'. The 'user' role is selected. Below the roles list are input fields for 'First Name' (containing 'Test'), 'Last Name' (containing 'Test'), 'New Password' (masked with asterisks), and 'Retype New Password' (masked with asterisks). There is a 'Must Change Password' checkbox which is checked. At the bottom of the dialog are 'Previous' and 'Save' buttons.

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.

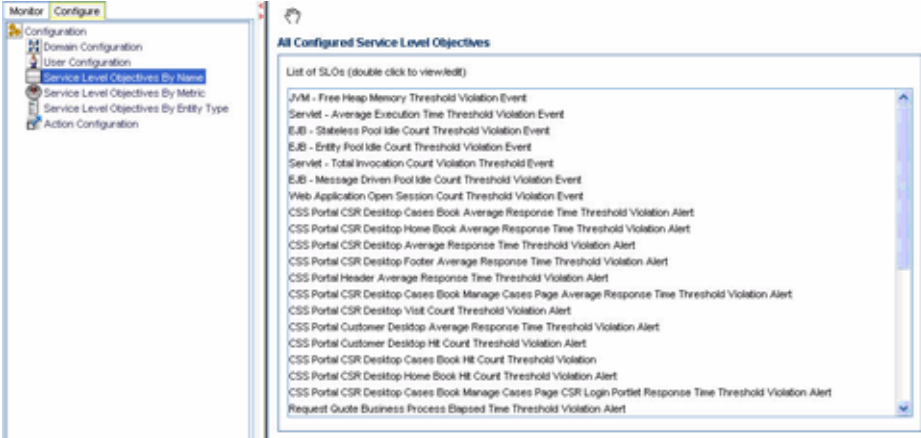
- Password length check
- Password complexity check
- Password expiration check

For more information, please refer to the *CAMM™ Configuration Guide*.

Service Level Objectives by Name

Service Level Objectives by Name node in the Configuration tree allows you to manage SLOs by SLO names. See [Figure 5-7](#).

Figure 5-7: Service Level Objectives by Name



On this window you can:

- Double-click on a specific SLO to open the Service Level Objective Editor.

- View or edit the selected SLO. See [Figure 5-8](#).

Figure 5-8: Service Level Objective Editor

Entities Included in SLO

Domain	
Node	
Server	
Application	cssdemo
Web Application	cssWeb
Portal	css
Portal desktop	csr

SLO Parameters

Name	CSS Portal CSR Desktop Visit Count Threshold Violation Alert
Severity	Violation
Metric	Metric:BEA.PortalDesktop.VisitCount
Window Size	15 seconds
Trigger on High	<input checked="" type="checkbox"/>
High Threshold	10
Trigger on Low	<input type="checkbox"/>
Low Threshold	0
Actions	Administrator - Portal Average Response Time Threshold Caution Email Action Administrator - Portal Average Response Time Threshold Violation Email Action Administrator - Process Average Response Time Threshold Caution Email Action Administrator - Process Average Response Time Threshold Violation Email Action Administrator - Web Service Average Response Time Threshold Caution Email Action Administrator - Web Service Average Response Time Threshold Violation Email Action Engineering - Portal Average Response Time Threshold Caution Email Action Engineering - Portal Average Response Time Threshold Violation Email Action Engineering - Process Average Response Time Threshold Caution Email Action Engineering - Process Average Response Time Threshold Violation Email Action

Save Cancel Delete

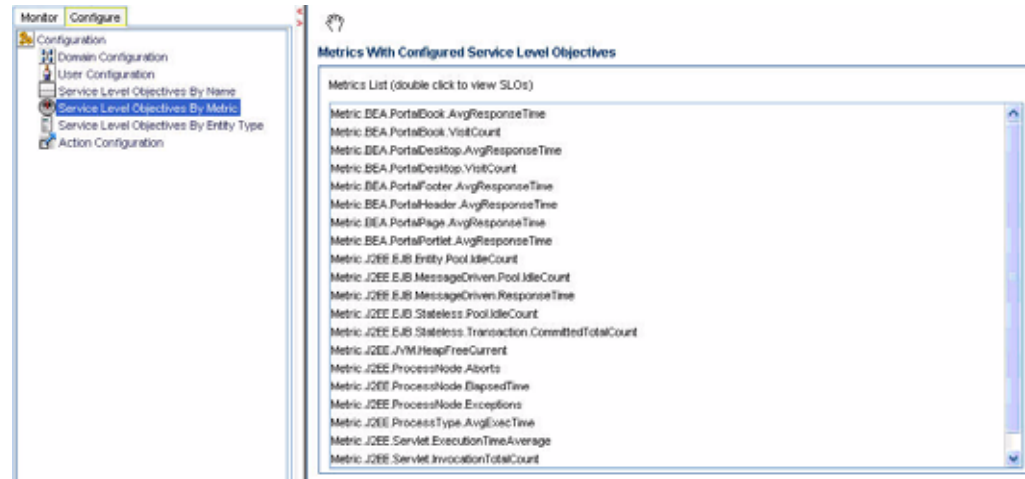
Service Level Objectives by Metrics

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 Heal Free Current
- Process Node Aborts
- Servlet Invocation Total Count
- Servlet Execution Time Average

See [Figure 5-9](#) for examples of Service Level Objectives by Metric.

Figure 5-9: Service Level Objectives by Metrics



On this 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.

Service Level Objectives by Entity Type

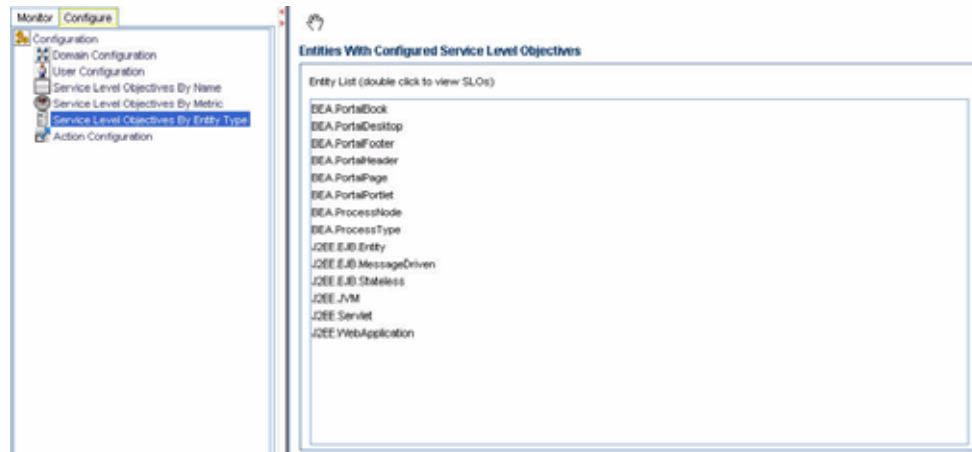
Service Level Objectives by Entity Type node in the Configuration tree allows you to manage SLOs by modeled entity type.

The following are examples of performance metrics:

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

See [Figure 5-10](#) for examples of Service Level Objectives by Entity Type:

Figure 5-10: Service Level Objectives by Entity Type



On this 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.

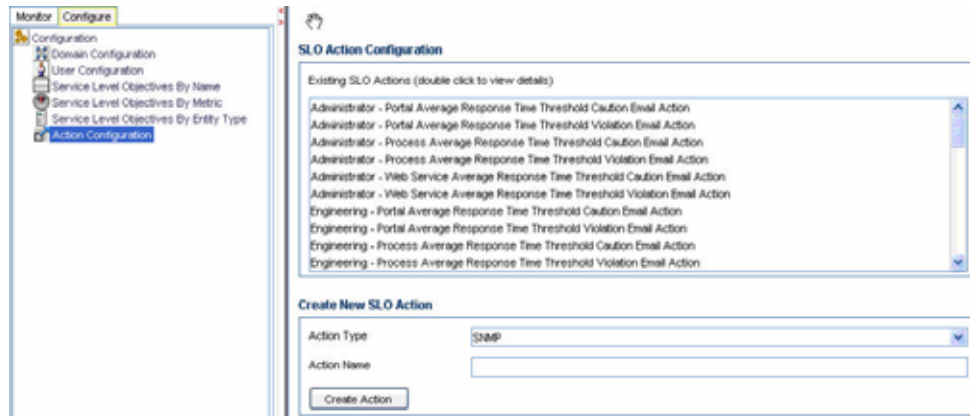
Action Configuration

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

See [Figure 5-11](#) for an example of Action Configuration.

Figure 5-11: Action Configuration



On this window you can:

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

See [Figure 5-12](#) to [Figure 5-15](#) for various action configuration forms.

Figure 5-12: SNMP Trap Action Configuration

SNMP Trap Configuration


Name	<input type="text"/>
Description	<input type="text"/>
SNMP Version	1 <input type="button" value="v"/>
IP Address	<input type="text"/>
Port	162 <input type="text"/>
Protocol	UDP <input type="button" value="v"/>
Community Name	public <input type="text"/>
Optional Authentication Information	
Authentication Protocol	MD5 <input type="button" value="v"/>
New Authentication Passphrase	<input type="text"/>
Retype Authentication Passphrase	<input type="text"/>
Privacy Protocol	DES <input type="button" value="v"/>
New Privacy Passphrase	<input type="text"/>
Retype Privacy Passphrase	<input type="text"/>
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 5-13: Send E-Mail Action Configuration

Send E-Mail Configuration

Action Name	<input type="text"/>
Description	<input type="text"/>
SMTP Server	<input type="text"/>
From E-Mail	<input type="text"/>
To E-Mail	<input type="text"/>
Subject	<input type="text"/>
Message	<input type="text"/>
Optional SMTP Authentication Info	
SMTP Username	<input type="text"/>
SMTP Password	<input type="text"/>
Retype Password	<input type="text"/>
<input type="button" value="Save"/>	<input type="button" value="Cancel"/>

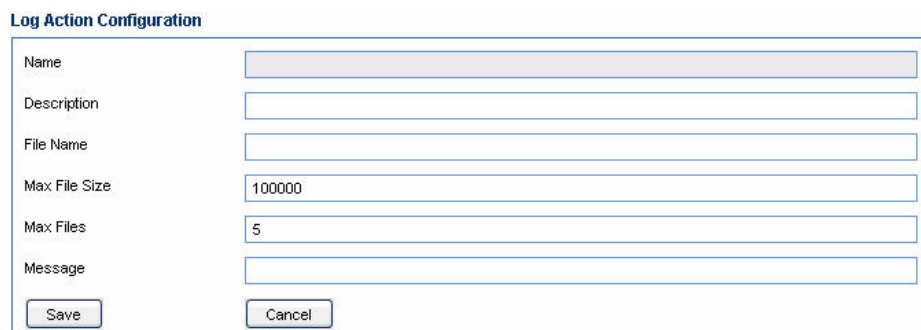
Figure 5-14: Script Action Configuration



The 'Script Action Configuration' dialog box contains the following fields and buttons:

- Name:
- Description:
- Command:
- Username:
- Group:
- New Password:
- Retype New Password:
- Buttons: Save, Cancel

Figure 5-15: Log Action Configuration

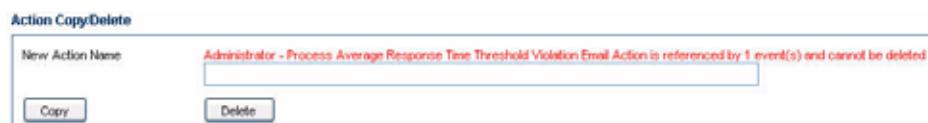


The 'Log Action Configuration' dialog box contains the following fields and buttons:

- Name:
- Description:
- File Name:
- Max File Size:
- Max Files:
- Message:
- Buttons: Save, Cancel

In the Action Configuration window, double-click on a specific action to see its configuration information. You can edit, copy, and edit selected action. CAMM™ automatically enforces referential integrity during the deletion process. [Figure 5-16](#) is an example of a violation of the referential integrity.

Figure 5-16: Enforcing Referential Integrity



The 'Action Copy/Delete' dialog box shows a red error message: "Administrator - Process Average Response Time Threshold Violation Email Action is referenced by 1 event(s) and cannot be deleted".

Fields and buttons:

- New Action Name:
- Buttons: Copy, Delete

Important: 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 5-3](#) for a list of SLO variables.

Table 5-3: 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	End 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;
```

6

Performance Analytics

This chapter includes the following topics:

- [Entity Performance Ranking](#)
- [Performance Characterization](#)
- [Memory Leak Detection](#)
- [Drill Down - Bottleneck Analysis](#)
- [Drill Out - Impact Analysis](#)

Entity Performance Ranking

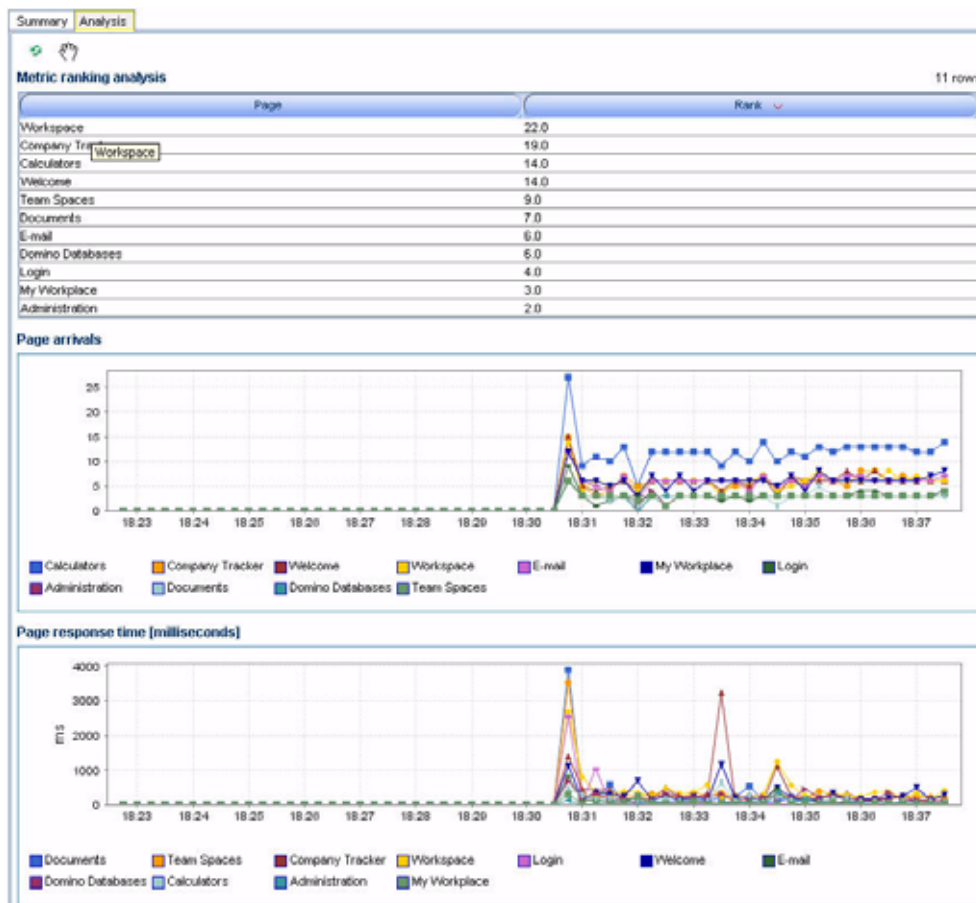
When performance metrics with contextual information are used throughout the monitoring environment, accurate statistical analysis of these performance measurements then becomes possible. This section describes how CAMM™'s Entity Performance Ranking can be used to accelerate bottleneck isolation.

As CAMM™ collects measurements, it processes incoming data and stores resulting information in CAMM™'s embedded database. CAMM™'s performance analytics feature then queries against this database to create statistical models and perform mathematical calculations. One of the performance analytics is called *Entity Performance Ranking* and is designed to assist you to quickly identify performance bottlenecks. [Figure 6-1](#) shows the Entity Performance Ranking view in CAMM™.

Entity Performance Ranking Analysis accelerates the identification of performance bottlenecks.

Click on a node with children entities to view this feature.

Figure 6-1: Entity Performance Ranking View

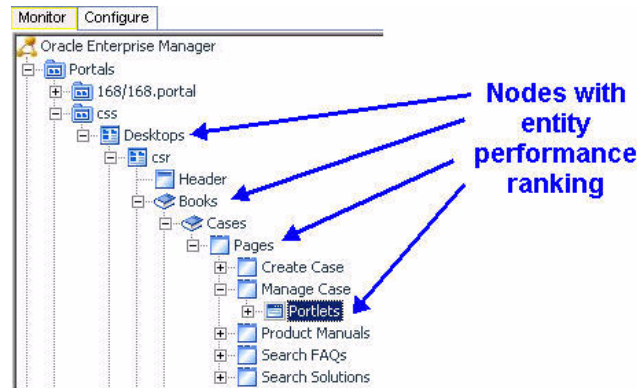


To create the Entity Performance Ranking, CAMM™ uses patent-pending algorithm to normalize a pair of performance metrics, arrivals and response times, from a single monitored entity. As shown in the [Figure 6-1](#) example, CAMM™ normalizes metrics from eleven page entities. This data normalization process enables direct comparison and ranking. CAMM™ uses this technique to rank compare monitored entities at the same level. In this example, CAMM™ compares the performance of eleven pages (children entities) organized under a single book (parent entity). You can use this approach to isolate bottlenecks efficiently.

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. [Figure 6-2](#) shows a diagram to illustrate this feature with a WebSphere® Portal example.

CAMM™ provides Entity Performance Ranking on Nodes with Children Entities.

Figure 6-2: Entity Performance Ranking with a WebSphere® Portal



Tip: To identify an entity that is performing abnormally, compare current Entity Performance Ranking to the baseline Entity Performance Ranking. To obtain the baseline ranking, simply change CAMM™'s time frame to a period that has a typical performance.

Performance Characterization

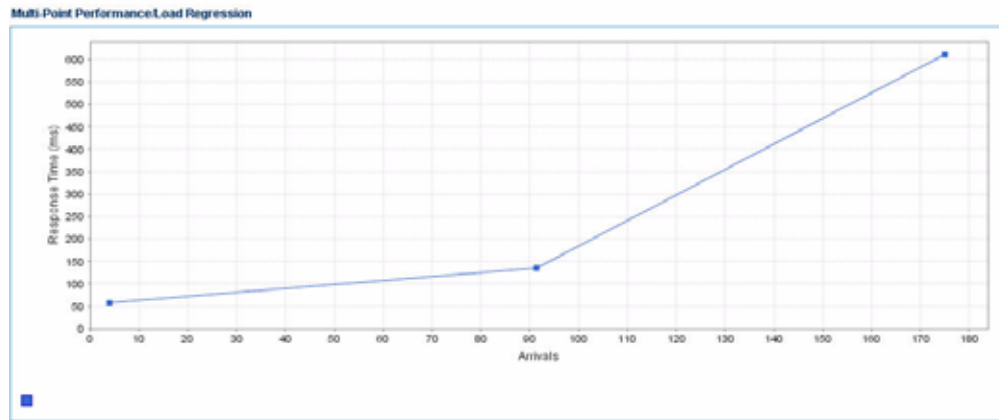
Performance Characterization is a set of performance analytics provided by CAMM™. Use the Performance Characterization analytics to visualize actual performance of a monitored entity in a given time frame. The following is a list of Performance Characterization analytics provided by CAMM™:

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.

Figure 6-3: Multi-point Regression



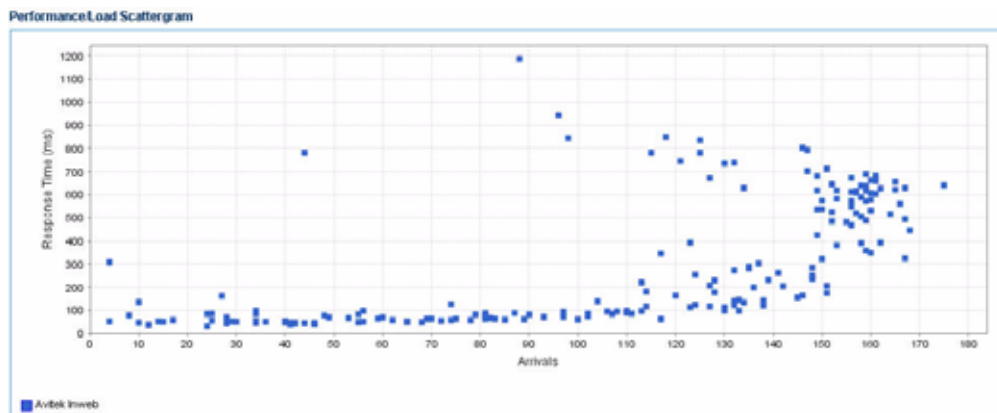
As a performance analytical tool, the Multi-Point Performance/Load Regression is extremely useful. It enables you to characterize the performance of a system or a component for a certain load range. This analysis can be used to capture the performance baseline of a system with a specific infrastructure configuration such as deployed applications, application server configuration, network topology and hardware capacity and load their characteristics. It 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.

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.

Figure 6-4: Performance Scattergram

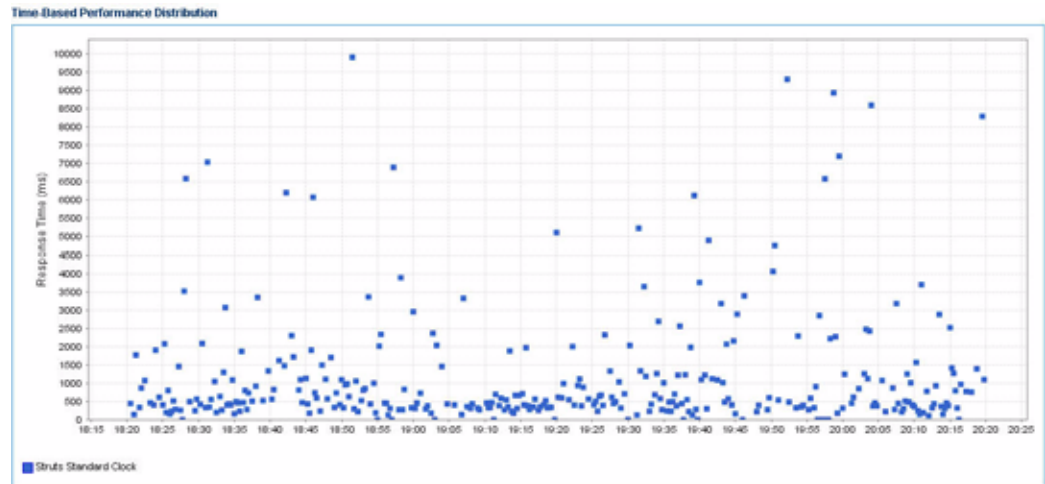


Time-Based Performance Distribution

This feature automatically plots the performance (response time) of a specific monitored entity during a time frame specified by you. In [Figure 6-5](#) plot, 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.

Figure 6-5: Time-Based Performance Distribution

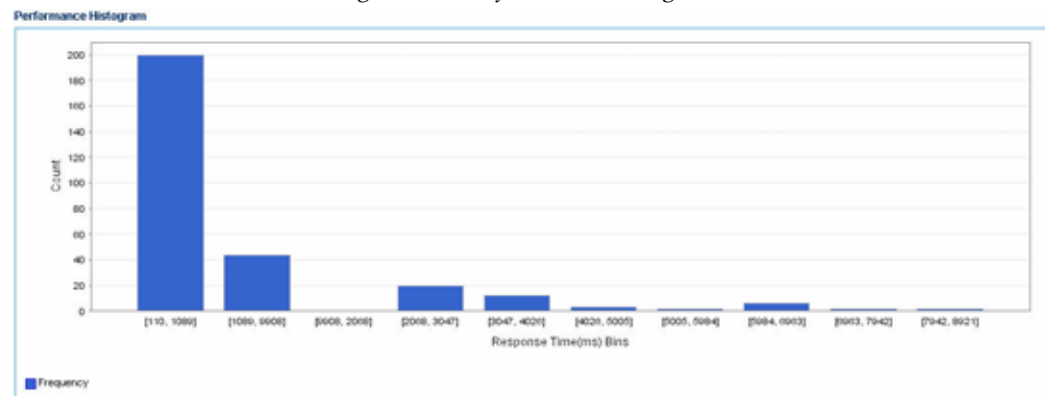


Performance Histogram

This feature automatically creates a histogram with ten bins (performance ranges) and plots the occurrences of response time that falls in each bin. In [Figure 6-6](#) histogram, 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.

Figure 6-6: Performance Histogram

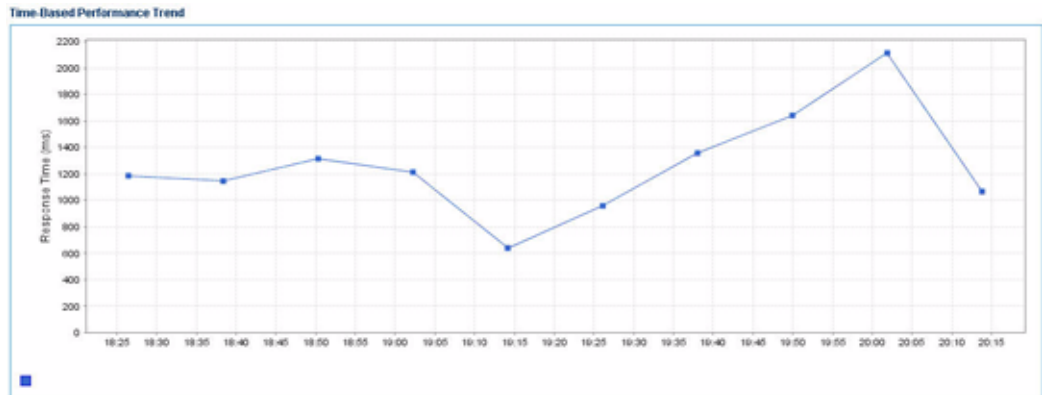


Time-based Performance Trend

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. In [Figure 6-7](#) plot, 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.

Figure 6-7: Time-Based Performance



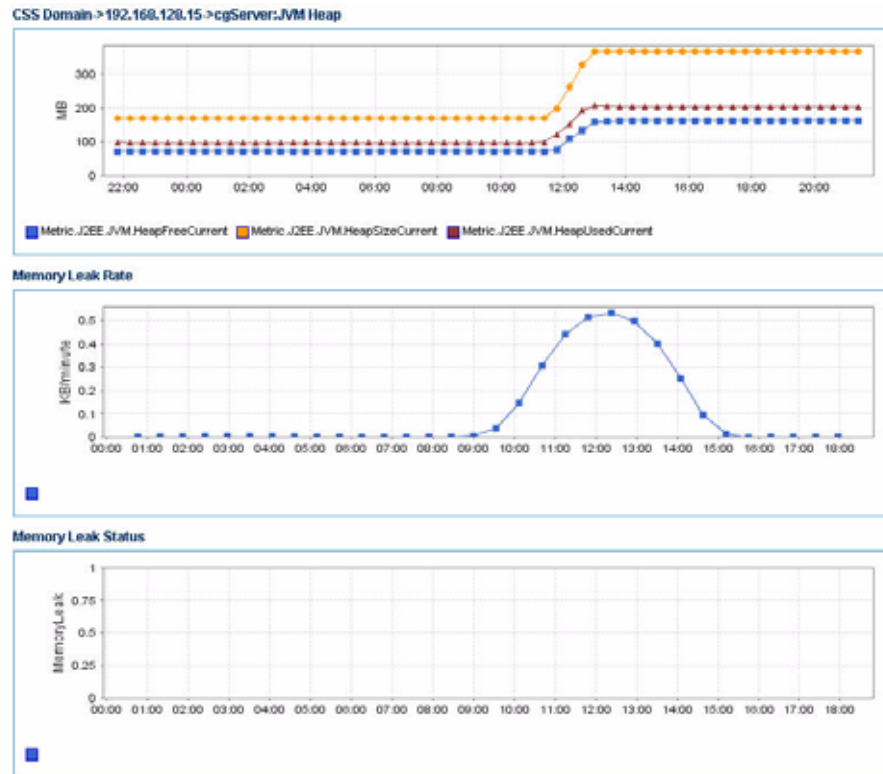
Memory Leak Detection

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.

Figure 6-8: Memory Leak Detection



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: Persisted 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 is a leak. When there is a severe leak, the value of the Memory Leak Status graph remains at 1.

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 6-1: 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.

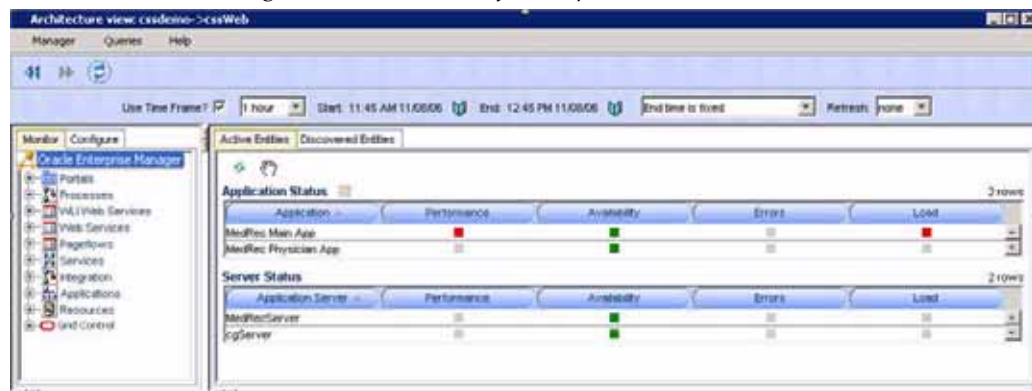
Whenever possible, use the built-in Delay Analysis to isolate performance bottlenecks. Delay Analysis is available under the Process node as well as in the Architecture View. CAMM's™ Delay Analysis shows the delay contribution associated with a specific component in a process or an active call path to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

Entity Performance Ranking can also be used to identify abnormally behaving components and performance bottlenecks.

[Figure 6-9](#) is a drill down example from the Operational Dashboard.

A red light on the Operational Dashboard indicates that the performance threshold for MedRec Main App has been violated.

Figure 6-9: Drill Down from Operational Dashboard



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

Figure 6-10: Drill Down Details



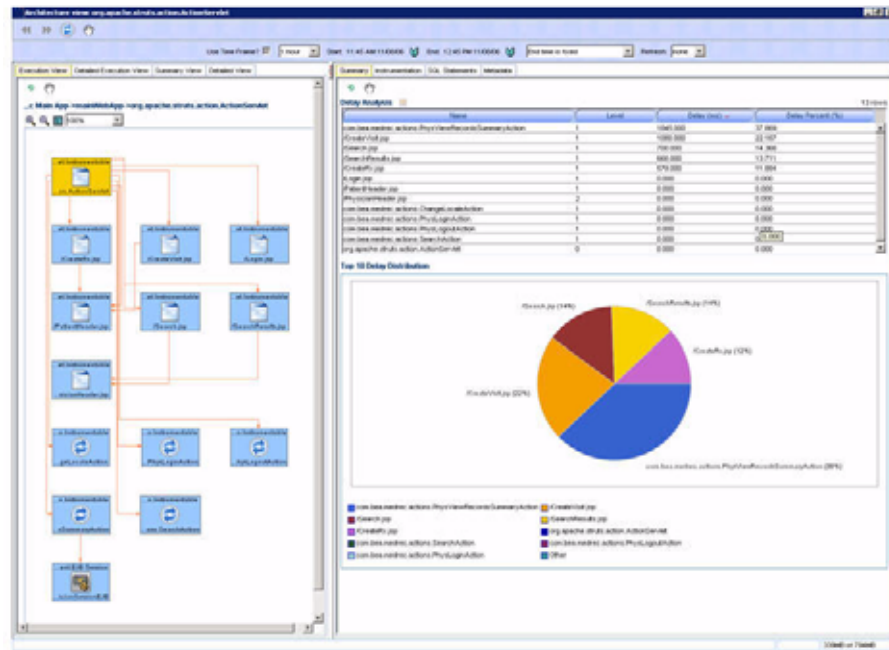
Use the detailed performance data to further isolate the performance problem. This pop-up window contains the following information:

- 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]
- Top five JDBC Response Time (ms) [Graph]
- Top five JDBC Invocation Count [Graph]

Double-clicking on a specific row in this pop-up window brings up the Architecture View associated with the selected row. [Figure 6-11](#) shows what happens after you double-click on a row in the Servlet Performance table.

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

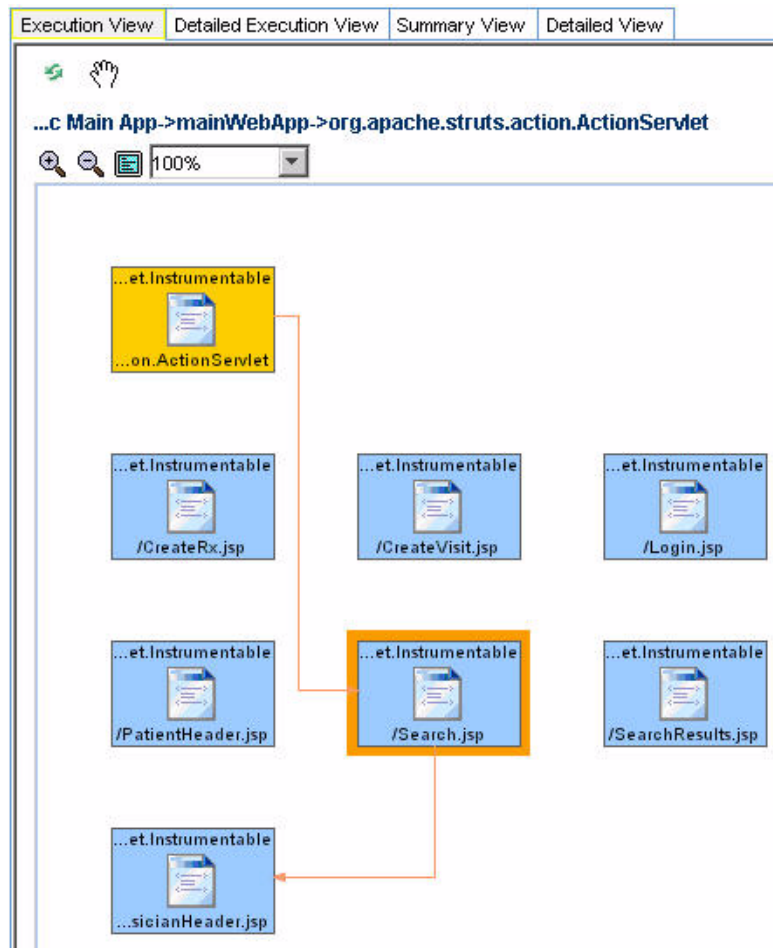
Figure 6-11: Architecture View with Delay Analysis



The delay analysis indicates 38% of the overall delay was contributed by `com.bea.medrec.actions.PhysViewRecordsSummaryAction`. Selecting the `com.bea.medrec.actions.PhysViewRecordsSummaryAction` brings up detailed performance data as shown in [Figure 6-12](#).

ActionServlet was identified as the biggest delay contributor in the delay analysis and was selected for more detailed performance data.

Figure 6-12: Detailed Performance Data on ActionServlet

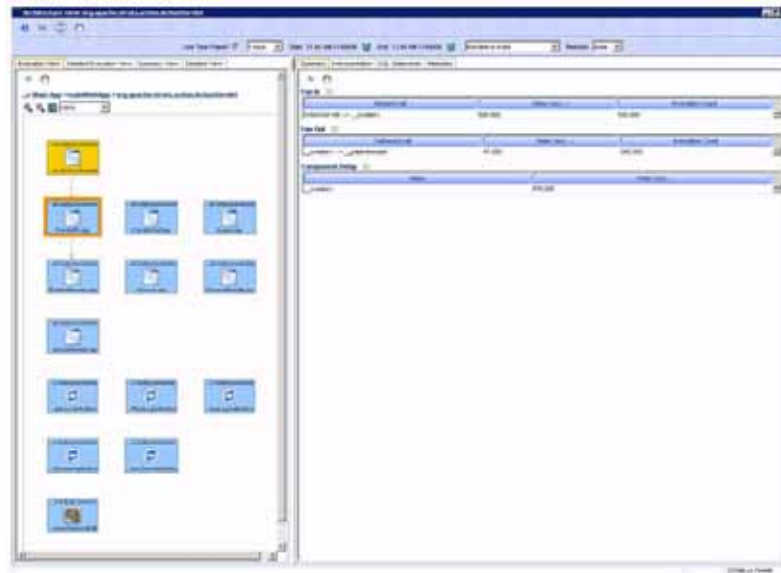
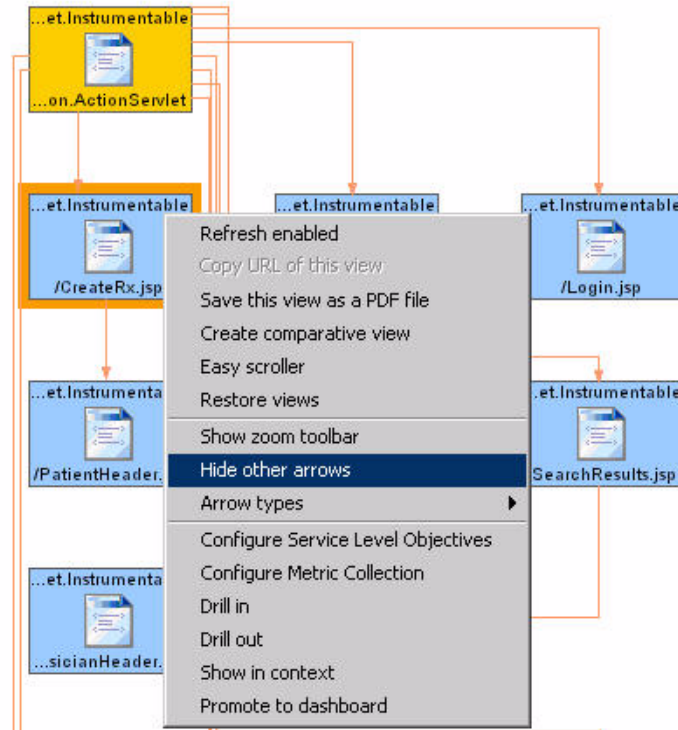


Summary Instrumentation SQL Statements Metadata			
Fan In			
Inbound call		Delay (ms)	Invocation Count
ActionServlet --> __search	700.000		1393.000
ActionServlet --> __search	283.000		271.000
Fan Out			
Outbound call		Delay (ms)	Invocation Count
__search --> __physicianheader	283.000		1625.000
Component Delay			
Name		Delay (ms)	
__search		700.000	

Tip: Right-click and check the **Hide other arrows** option to remove irrelevant arrows. Follow the slowest Inbound calls to the slowest next level component. Repeat this method to find the bottleneck.

Follow the ViewRequestAction call to the ViewRequestAction class and ultimately to the AdminSessionEJB.

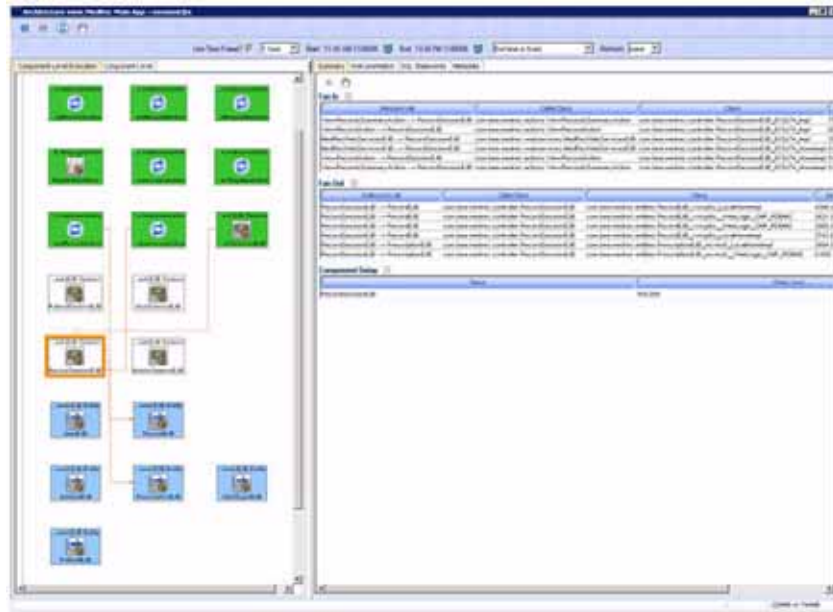
Figure 6-13: Hide Other Arrows

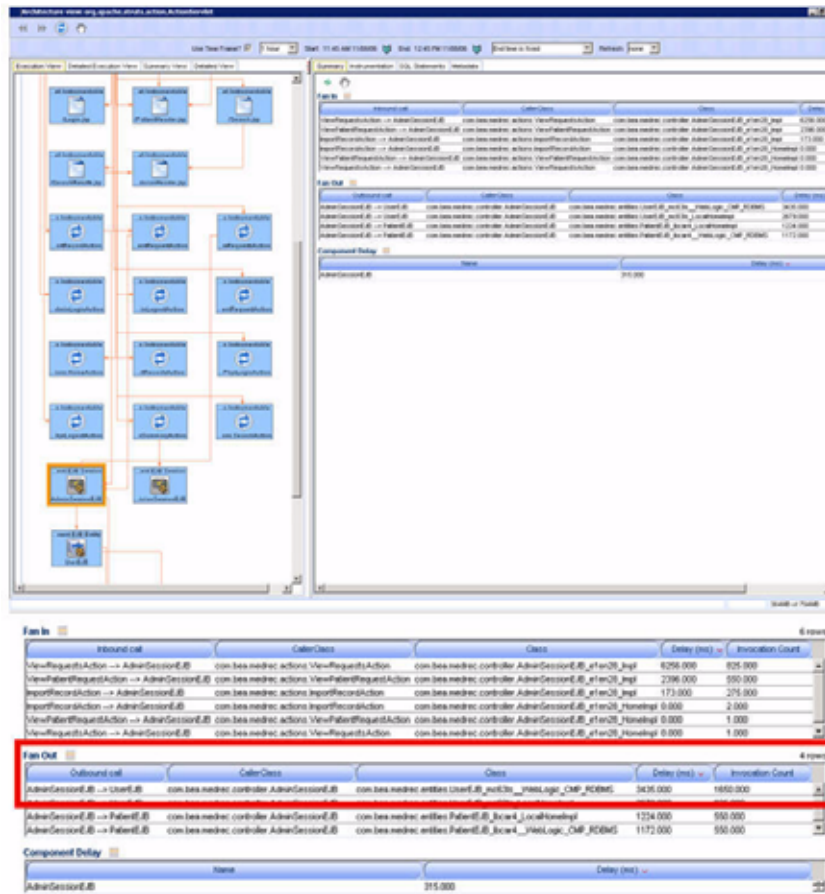


Look at the Fan Out table first to see if any outbound calls are performing poorly. If a specific call is performing poorly, follow the call to the destination component and evaluate further. See [Figure 6-14](#).

Evaluate all the Fan Out calls for AdminSessionEJB and follow the slowest Fan Out call to UserEJB.

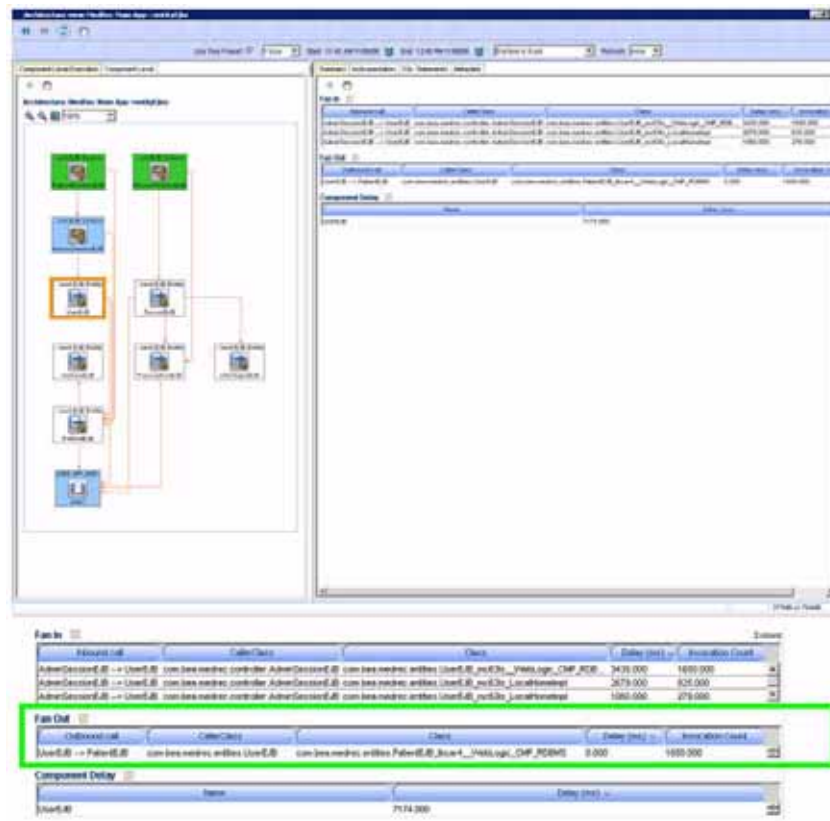
Figure 6-14: Fan Out Call to UserEJB





Since the Fan Out call for the UserEJB has 0 delay, the performance bottleneck is located in the UserEJB.

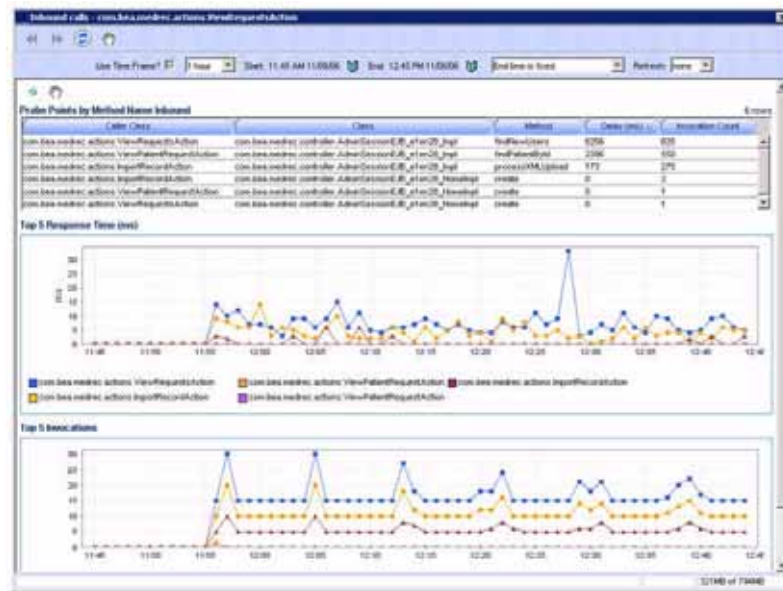
Figure 6-15: Bottleneck Detected



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. In [Figure 6-15](#), it is clear the performance bottleneck is in the UserEJB component because there is no delay associated with the outbound call. 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. See [Figure 6-16](#).

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

Figure 6-16: Fan In Call Detail



As shown in [Figure 6-16](#), CAMM™ guides you from the Operational Dashboard to the performance bottleneck that exists at the method level. In the end, the `getPatientObj` method in `com.bea.medrec.entities.UserJEB` class was identified as a performance bottleneck and the appropriate team was alerted to remedy the problem.

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

Drill Out - Impact Analysis

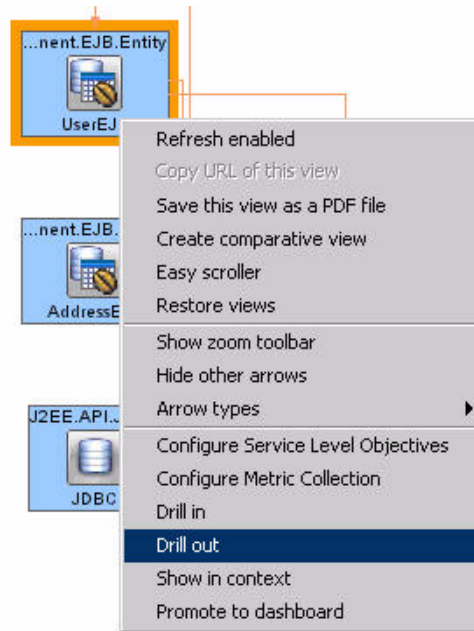
After performance bottleneck is identified, you can use CAMM™'s unique drill out ability to determine the impact of the bottleneck. The impact analysis can be done in the Architecture View.

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.

- Use a combination of Drill Out and Show in Context navigation techniques to determine the scope of impact.

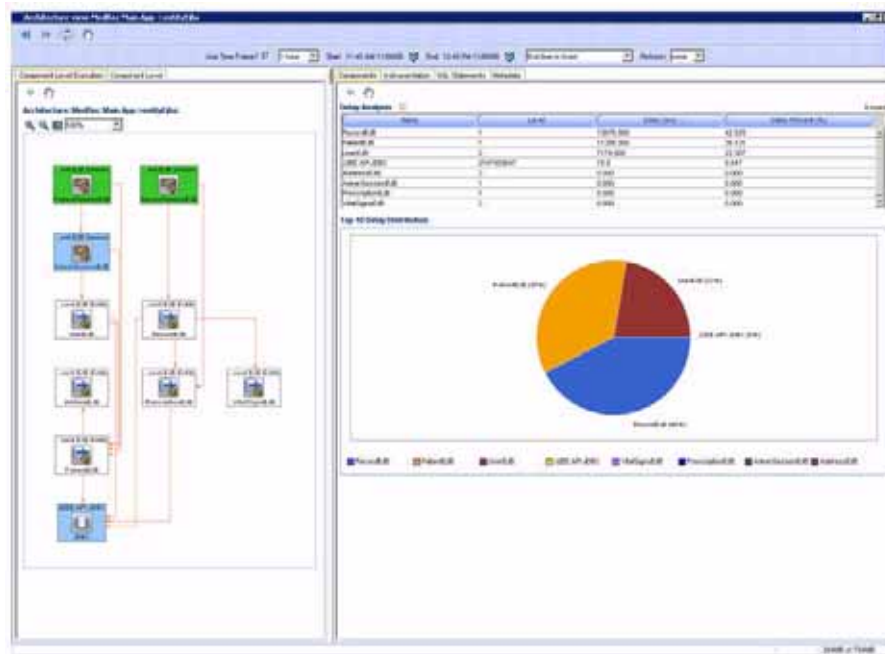
Figure 6-17: Drill Out



We continue with the example in the previous section to determine the impact of the performance bottleneck found in UserEJB class.

Drilling out from the UserEJB refocuses the Architecture View and the module that contains UserEJB in context (white background color).

Figure 6-18: Drill Out UserEJB



On this window you can:

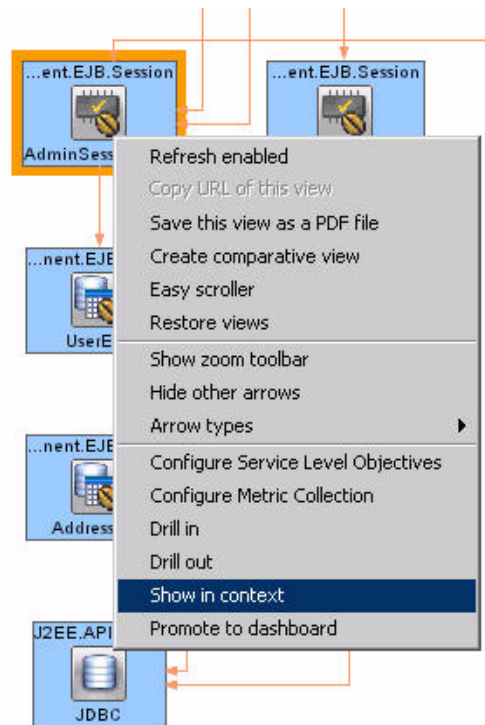
- Check the Hide other arrows option in the right-click menu to eliminate irrelevant arrows and improve visual navigation.
- Use the Architecture View or the Fan In table to identify the upstream components that make inbound calls into the UserEJB. The impact analysis would follow these inbound calls to their origins to further determine the impact of the performance bottleneck.

For this example, there is only one inbound call for the UserEJB and its origin is the AdminSessionEJB. The blue background color of the AdminSessionEJB indicates this entity belongs to a module that is different from current view context.

To continue the impact analysis:

1. Highlight the AdminSessionEJB in the Architecture View.
2. Right-click and select Show in context to switch the view context to focus on the module that contains the AdminSessionEJB. See [Figure 6-19](#).

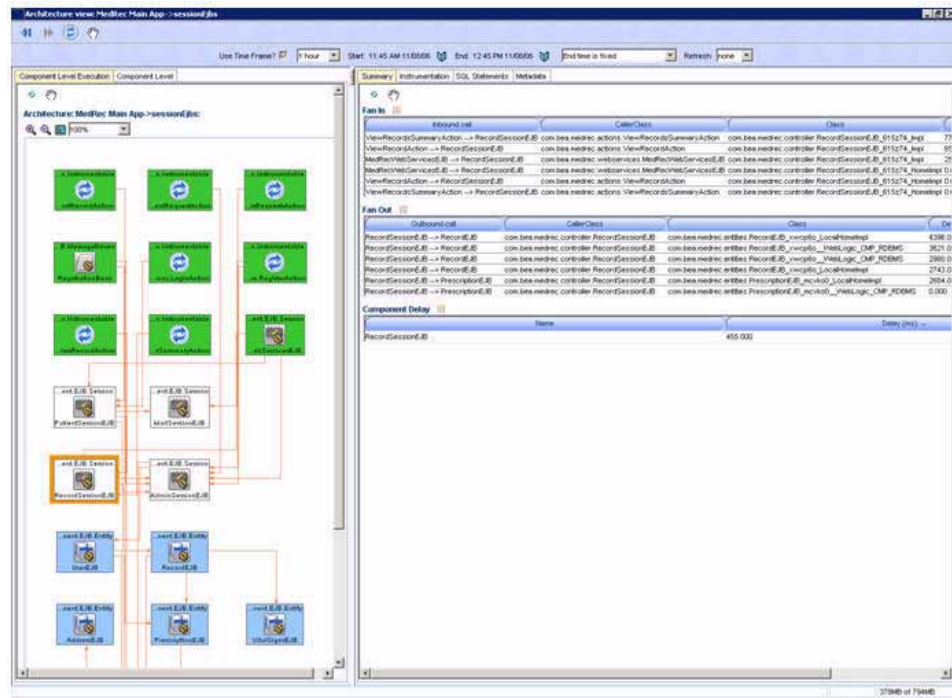
Figure 6-19: Drill Out to Show in Context



The resulting Architecture View reveals that there are four upstream components that invoke the AdminSessionEJB. More information on each of these upstream components can be obtained.

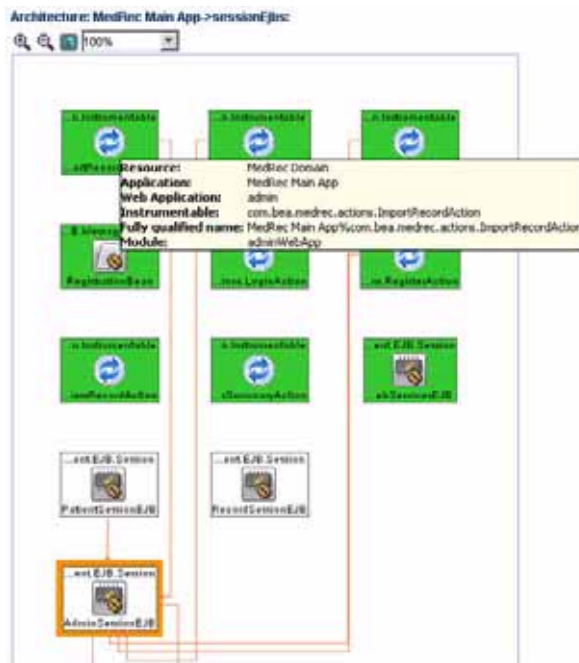
3. Select a specific component for detailed performance data.

Figure 6-20: Detailed Performance Data



4. Mouse over relevant upstream components to obtain detailed information.

Figure 6-21: Mouse Over to Obtain More Information



5. Look at the Fan In table associated with AdminSessionEJB to determine if any inbound calls are suffering from a performance problem.

6. Select AdminSessionEJB and evaluate various inbound calls to determine if the performance bottleneck has impacted other upstream components.

Figure 6-22: AdminSessionEJB

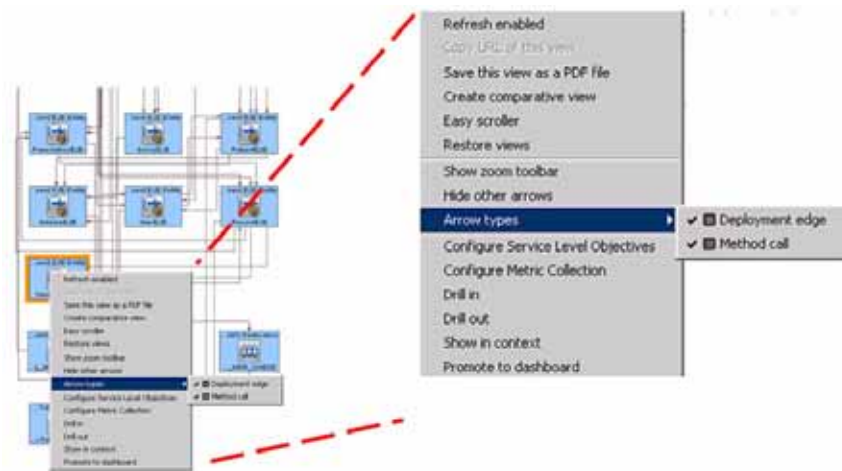
Inbound call	Caller Class	Class	Delay (ms)	Invocation Count
ViewRequestsAction -> AdminSessionEJB	com.bea.medrec.actions.ViewRequestsAction	com.bea.medrec.controller.AdminSessionEJB_etier20_impl	6256.000	825.000
ViewPatientRequestAction -> AdminSessionEJB	com.bea.medrec.actions.ViewPatientRequestAction	com.bea.medrec.controller.AdminSessionEJB_etier20_impl	2396.000	550.000
ImportRecordAction -> AdminSessionEJB	com.bea.medrec.actions.ImportRecordAction	com.bea.medrec.controller.AdminSessionEJB_etier20_impl	173.000	275.000
ViewPatientRequestAction -> AdminSessionEJB	com.bea.medrec.actions.ViewPatientRequestAction	com.bea.medrec.controller.AdminSessionEJB_etier20_HomeImpl	0.000	2.000
ViewRequestsAction -> AdminSessionEJB	com.bea.medrec.actions.ViewRequestsAction	com.bea.medrec.controller.AdminSessionEJB_etier20_HomeImpl	0.000	1.000
ViewRequestsAction -> AdminSessionEJB	com.bea.medrec.actions.ViewRequestsAction	com.bea.medrec.controller.AdminSessionEJB_etier20_HomeImpl	0.000	1.000

In addition to looking at the call paths active for the specified time frame, it may be valuable to look at potential call paths captured in CAMM™'s AppSchema model.

To access the potential call path view:

1. Click the Component Level tab in the Architecture View.
2. If the potential call path view has too many arrows and is too complex, right-click to remove the arrows. See [Figure 6-23](#).

Figure 6-23: Uncheck Less Relevant Arrows

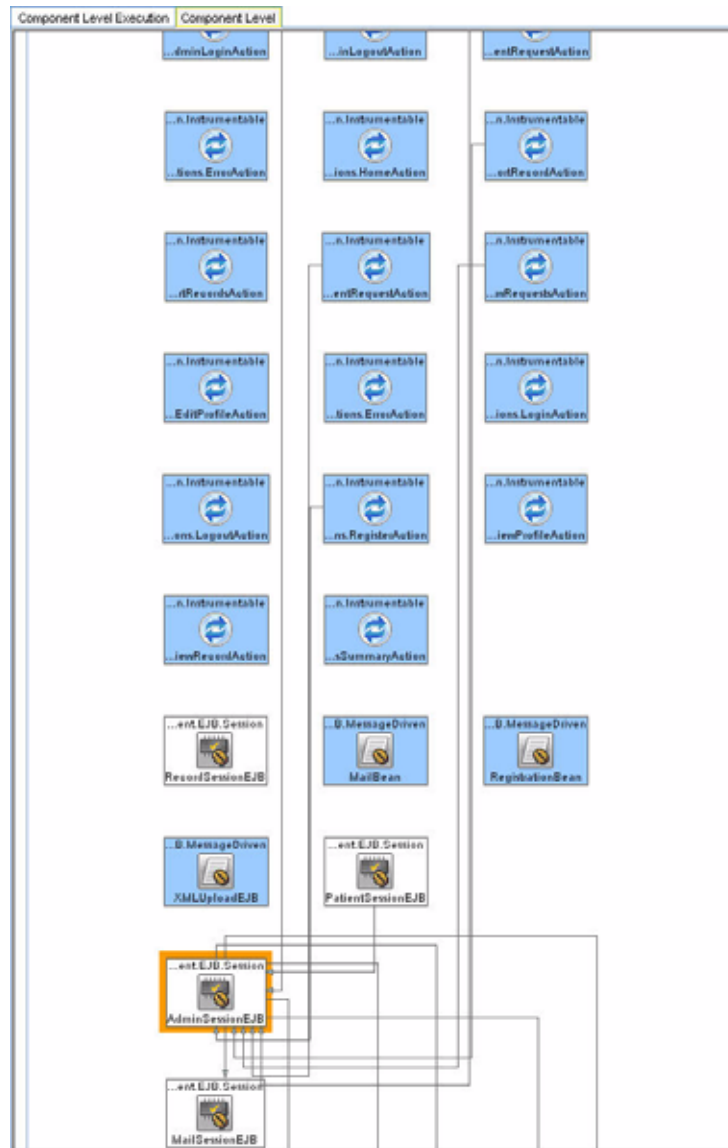


[Figure 6-23](#) shows the Deployment edge option is used to hide all arrows of this type in the potential call path diagram. Combining this with checking the Hide other arrows option results in a much simpler view.

3. Evaluate all potential call paths associated with AdminSessionEJB using the potential call path diagram in the Architecture View.

Figure 6-24 shows the resulting potential call path diagram with the AdminSessionEJB selected.

Figure 6-24: Potential Call Path Diagram Without Unnecessary Arrows



Tip: Use the potential call path diagram to evaluate *what if* scenarios and determine potential impact resulting from the performance bottleneck.

7

Exporting Data

This chapter includes the following topics:

- [Remote Log Retrieval](#)
- [Data Export Modes](#)
- [Data Export Configuration](#)
- [Example of Exported Data for WebLogic](#)

Remote Log Retrieval

Use this feature to fetch logs from remote server via running scripts. This section describes the scripts that are provided out-of-the-box by Oracle Enterprise Grid Control - Composite Application Monitor and Modeler (CAMM™).

Lardemo.sh File

Run this script first.

Code 7-1: Lardemo.sh File

```
#!/bin/bash
```

Code 7-1: *Lardemo.sh* File (Continued)

```
export ACSERA_HOME=/cygdrive/c/Oracle39_0599

#./lar.sh -V 8.1.3 -U t3://192.168.3.151:7001/ -f outnik.tar.gz -u
weblogic -p weblogic -cvqad

# -q MySQL mysqldump database export

# -a Include agent files in the output archive (specified by -f)

# -d Include all darchive files in the output archive

./lar.sh -V 8.1.3 -U t3://192.168.128.12:7001/ -f outnik.tar.gz -u
weblogic -p weblogic -cvqad

#

# The generated .tar file will be saved in the same directory as the
lardemo.sh file.
```

This script runs the *Lar.sh* shell script. Use [Table 7-1](#) to learn about the commands imbedded in this file. The generated .tar file will be saved in the same directory as the *Lardemo.sh* file. See [Code 7-3](#) for the directory structure of the .tar file.

[Table 7-1](#) lists the commands and their functions.

Table 7-1: *Commands and their Functions*

Command	Function
C	<customer name>. This command denotes the customer name. It needs the parameter.
V	<WebLogic version>. This command denotes version number. It needs the parameter example 8.1.3,8.1.4...
T	<target names> (optional). This command denotes the target. It needs the target name as it's parameter.
U	<admin URL>. It is an admin URL of the machine.
f	<file> (for Compile or X-Fer action). This command denotes the file. It needs the parameter which will be considered as filename.
u	<WebLogic username>. This command denotes username. It needs a parameter which will be considered as username.
p	<WebLogic password>. This command denotes the password. It needs a parameter which is considered as password.
x	X-Fer. This command denotes the action to be performed. It is set to true.
c	Compile. This command denotes the compile Action. It is set to true. This will assemble all the log files in one location.
t	This command denotes the action tell. It is set to true.

Table 7-1: Commands and their Functions (Continued)

Command	Function
d	Archive. This command checks the availability of the archive to collect files from.
q	MySQL database. This command denotes the existence of SQL database. It is set to true.
v	Verbose has a Boolean parameter set to true. This command is used to ensure that all the environment variables are set correctly.

Lardemo.sh File

This file has the command to run the `lardemo.sh` script.

See [Code 7-2](#) for the contents of the file.

Code 7-2: Lar.sh File

```
#!/bin/bash

_LAR_ECHO="echo -e" ;
_LAR_MKDIR="mkdir" ;
_LAR_RM="rm" ;
_LAR_MV="mv" ;
_LAR_WGET="wget" ;
_LAR_MKTEMP="mktemp" ;
_LAR_TAR="tar" ;
_LAR_FIND="find" ;
_LAR_CYGPATH="/usr/bin/cygpath.exe" ;
#
# command dependencies: echo, rm, tar, gawk, find, getopt (alias), mktemp,
#   cygpath, acseraenvlegacy.sh, runlegacyquiet.sh, cat
#
if [ -x "$ACSERA_HOME/bin/deployer.sh" ]; then
    PATH_SEPARATOR=\;
    export ACSERA_HOME PATH_SEPARATOR ;

    LAR_ACSERA_DB=acsera ;
    LAR_DATE=`date +%m%d%Y.%H%M%S` ;
    LAR_ACSERA_HOME_UNIX=`$_LAR_CYGPATH -a --unix "$ACSERA_HOME"` ;
```

Code 7-2: Lar.sh File (Continued)

```
LAR_ACSERA_HOME_WINDOWS=`$LAR_CYGPATH -a --windows "$ACSERA_HOME"` ;  
  
#  
# then, the the options from the command line.  
#  
while getopts "f:cqadxtvu:p:V:T:U:C:" LAR_OPTION ; do  
  case $LAR_OPTION in  
    C) LAR_CUSTOMER=$OPTARG ;;  
    V) LAR_VERSION=$OPTARG ;;  
    T) LAR_TARGETS=$OPTARG ;;  
    U) LAR_ADMINURL=$OPTARG ;;  
    f) LAR_FILE=$OPTARG ;;  
    u) LAR_USERNAME=$OPTARG ;;  
    p) LAR_PASSWORD=$OPTARG ;;  
    x) LAR_ACTION_XFER="true" ;;  
    c) LAR_ACTION_COMP="true" ;;  
    t) LAR_ACTION_TELL="true" ;;  
    d) LAR_COMP_DARC="true" ;;  
    a) LAR_COMP_AGENT="true" ;;  
    q) LAR_COMP_SQL="true" ;;  
    v) LAR_VERBOSE="yes" ;;  
  esac  
done  
  
if [ ! -z "$LAR_VERBOSE" ]; then  
  $_LAR_ECHO "running lar for environment";  
  $_LAR_ECHO "LAR_DATE=$LAR_DATE";  
  $_LAR_ECHO "LAR_CUSTOMER=$LAR_CUSTOMER";  
  $_LAR_ECHO "LAR_FILE=$LAR_FILE";  
  $_LAR_ECHO "LAR_VERSION=$LAR_VERSION";  
  $_LAR_ECHO "LAR_ADMINURL=$LAR_ADMINURL";  
  $_LAR_ECHO "LAR_USERNAME=$LAR_USERNAME";  
  $_LAR_ECHO "LAR_ACSERA_HOME_UNIX=$LAR_ACSERA_HOME_UNIX";  
  $_LAR_ECHO "LAR_ACSERA_HOME_WINDOWS=$LAR_ACSERA_HOME_WINDOWS";  
fi  
  
#  
# If a customer was specified, then name the temp locations after then,
```

Code 7-2: Lar.sh File (Continued)

```
# otherwise use a default
#
if [ ! -z "$LAR_CUSTOMER" ]; then
    LAR_TEMP=`$_LAR_MKTEMP -dt $LAR_CUSTOMER.$LAR_DATE.XXXXXXXX`;
    LAR_TEMP_ARCHIVE="$LAR_TEMP/$LAR_CUSTOMER.$LAR_DATE.tar.gz";
    LAR_TEMP_ROOT="$LAR_TEMP/root";
else
    LAR_TEMP=`$_LAR_MKTEMP -dt lar.$LAR_DATE.XXXXXXXX`;
    LAR_TEMP_ARCHIVE="$LAR_TEMP/compilation.$LAR_DATE.tar.gz";
    LAR_TEMP_ROOT="$LAR_TEMP/root";
fi
( $_LAR_MKDIR $LAR_TEMP_ROOT );
if [ $? != 0 ]; then
    $_LAR_ECHO "cannot make directory \"$LAR_TEMP_ROOT\"";
    exit 1;
fi
#
# compile action will assemble all the log files into one location,
#
if [ ! -z "$LAR_ACTION_COMP" ]; then
    #
    # starting here we will perform remote requests.  At the very least we will
    # need a version to for the "acseraenv" call, we will need username, password,
    # etc... later.
    #
    if [ ! -z "$LAR_VERSION" ] && [ ! -z "$LAR_COMP_AGENT" ]; then
        . "$ACSERA_HOME/bin/acseraenvlegacy.sh" -version $LAR_VERSION 2> /dev/null;
        #
        # to get the remote logs we will need adminurl, username, password, etc...
        # if we don't have these, we'll just move on (and inform the user)
        #
        if [ ! -z "$LAR_ADMINURL" ] && [ ! -z "$LAR_USERNAME" ] && [ ! -z "$LAR_PASSWORD" ];
then
            #
            # We check for blank LAR_TARGETS because the user might have specified
```

Code 7-2: Lar.sh File (Continued)

```
# the targets in the environemnt or on the command-line.
#
if [ -z "$LAR_TARGETS" ]; then
    "$ACSERA_HOME/bin/deployer.sh" \
        -version $LAR_VERSION \
        -adminurl $LAR_ADMINURL \
        -username $LAR_USERNAME \
        -password $LAR_PASSWORD \
        -listservers 1> "$LAR_TEMP_ROOT/servers.lst" 2>> "$LAR_TEMP_ROOT/$i/
deployer.err";
    if [ $? == 0 ]; then
        LAR_TARGETS=`/usr/bin/gawk '{ print $1 }' < $LAR_TEMP_ROOT/servers.lst`
2> /dev/null;
    else
        $_LAR_ECHO "ERROR: failed to retrieve agent targets from WebLogic.";
        exit 1;
    fi
    if [ ! -z "$LAR_VERBOSE" ]; then
        $_LAR_ECHO "found remote servers \"$LAR_TARGETS\".";
    fi
fi;
#
# everything is OK if we got a server list back ...
#
if [ ! -z "$LAR_TARGETS" ]; then
    #
    # If the user asked for the agent logs, then archive it ...
    #
    for i in $LAR_TARGETS; do
        if [ ! -z "$LAR_VERBOSE" ]; then
            $_LAR_ECHO "retrieving remote logs for \"$i\".";
        fi
        $_LAR_MKDIR -p "$LAR_TEMP_ROOT/$i/logs";
        "$ACSERA_HOME/bin/deployer.sh" \
            -version $LAR_VERSION \
            -adminurl $LAR_ADMINURL \
```


Code 7-2: Lar.sh File (Continued)

```
        -username $LAR_USERNAME \  
        -password $LAR_PASSWORD \  
        -targets $i \  
        -copylogs \  
        -localpath `$_LAR_CYGPATH --windows "$LAR_TEMP_ROOT/$i/logs" ` 1\  
"$LAR_TEMP_ROOT/$i/deployer.log" 2>> "$LAR_TEMP_ROOT/$i/deployer.err";  
        done;  
    else  
        $_LAR_ECHO "ERROR: cannot find remote targets for log retrieval." 1>&2;  
        exit 1;  
    fi;  
else  
    $_LAR_ECHO "ERROR: remote server logs not uploaded because of missing username  
(-u), password (-p) and admin url (-U) arguments." 1>&2;  
    exit 1;  
fi;  
elif [ -z $LAR_VERSION ] && [ ! -z $LAR_COMP_AGENT ]; then  
    $_LAR_ECHO "ERROR: remote server logs not uploaded because of missing version option  
(-V) required for agent log download (-a)." 1>&2;  
    exit 1;  
fi;  
#  
# look for any local .csv files.  
#  
if [ -d "$LAR_ACSERA_HOME_UNIX" ]; then  
    if [ ! -z "$LAR_VERBOSE" ]; then  
        $_LAR_ECHO "adding all log files found under $LAR_ACSERA_HOME_UNIX.";  
    fi;  
    $_LAR_FIND "$LAR_ACSERA_HOME_UNIX" -name "*.csv" -print 1>> "$LAR_TEMP/targets.lst"  
2>> "$LAR_TEMP_ROOT/targets.err";  
else  
    $_LAR_ECHO "WARNING: CAMM installation not found at \"$LAR_ACSERA_HOME_UNIX\"."  
1>&2;  
fi;  
#  
# If the user asked for the darchive contents, then archive it too ...  
#
```

Code 7-2: Lar.sh File (Continued)

```
if [ ! -z "$LAR_COMP_DARC" ]; then
    #
    # If there is a darchive available, then collect logs from the darchive
    #
    if [ -d "$LAR_ACSERA_HOME_UNIX/darchive" ]; then
        if [ ! -z "$LAR_VERBOSE" ]; then
            _LAR_ECHO "adding list of files in darchive $LAR_ACSERA_HOME_UNIX/
darchive.";
        fi;
        _LAR_FIND "$LAR_ACSERA_HOME_UNIX/darchive" -type f -print 1>> "$LAR_TEMP/
targets.lst" 2>> "$LAR_TEMP_ROOT/targets.err";
    else
        _LAR_ECHO "WARNING: CAMM darchive not found at \"$LAR_ACSERA_HOME_UNIX/
darchive\"." 1>&2;
    fi;
fi;
#
# If the user asked for the MySQL export, then archive it too ...
#
if [ ! -z "$LAR_COMP_SQL" ]; then
    #
    # If there is a darchive available, then collect logs from the darchive
    #
    if [ -d "$LAR_ACSERA_HOME_UNIX/database" ]; then
        MYSQL_HOME="$LAR_ACSERA_HOME_UNIX/database"
        #PATH="$MYSQL_HOME/bin:$PATH"
        #export MYSQL_HOME PATH;
        export MYSQL_HOME;
        _LAR_MYSQLDUMP="$MYSQL_HOME/bin/mysqldump";
        _LAR_MYSQLDUMP="mysqldump";
        if [ ! -z "$LAR_VERBOSE" ]; then
            _LAR_ECHO "dumping database using $MYSQL_HOME/bin/mysqldump.";
        fi;
        _LAR_MYSQLDUMP --complete-insert $LAR_ACSERA_DB > "$LAR_TEMP_ROOT/
$LAR_ACSERA_DB.sql";
    else

```

Code 7-2: Lar.sh File (Continued)

```
        $_LAR_ECHO "WARNING: CAMM darchive not found at \"$_LAR_ACSERA_HOME_UNIX/
darchive\"." 1>&2;

        fi;

        fi;

        #

        # tar the entire contents of the lar temp directory

        #

        ( $_LAR_FIND "$LAR_TEMP_ROOT" -type f -print ) >> "$LAR_TEMP/targets.lst" 2>>
"$LAR_TEMP_ROOT/archive.err";

        ( cd "$LAR_TEMP_ROOT"; $_LAR_TAR -czT "$LAR_TEMP/targets.lst" -f - ) 1>
"$LAR_TEMP_ARCHIVE" 2>> "$LAR_TEMP_ROOT/archive.err";

        $_LAR_RM -rf "$LAR_TEMP_ROOT";

        if [ -z "$LAR_VERBOSE" ]; then

            _LAR_WGET_OPTS='-q';

        else

            _LAR_WGET_OPTS='--progress=dot';

        fi

        if [ ! -z "$LAR_FILE" ] && [ -z "$LAR_ACTION_XFER" ]; then

            $_LAR_MV "$LAR_TEMP_ARCHIVE" "$LAR_FILE";

        elif [ -z "$LAR_FILE" ] && [ ! -z "$LAR_ACTION_XFER" ] && [ ! -z "$LAR_CUSTOMER" ]; then

            $_LAR_WGET $_LAR_WGET_OPTS -P"$LAR_TEMP_ROOT/wget.log" --post-
file="$LAR_TEMP_ARCHIVE" "http://cannes.acsera.com/~$LAR_CUSTOMER/upload/
rawuploader.php?file=$LAR_TEMP_ARCHIVE";

        else

            $_LAR_ECHO "ERROR: conflicting arguments for compile. Cannot specify xfer and file
for compile action.";

            exit 1;

        fi

        $_LAR_RM -rf "$LAR_TEMP"

        if [ ! -z "$LAR_ACTION_XFER" ] && [ ! -z "$LAR_ACTION_FILE" ]; then

            $_LAR_WGET $_LAR_WGET_OPTS -P"$LAR_TEMP_ROOT/wget.log" --post-file="$LAR_FILE"
"http://cannes.acsera.com/~$LAR_CUSTOMER/upload/rawuploader.php?file=$LAR_FILE";

            $_LAR_RM -rf "$LAR_TEMP_ROOT";

        fi

    else

        echo ""

        echo "usage: $0 <options> <export directory>";

        echo ""

        echo "    -C <customer name>"
```

Code 7-2: Lar.sh File (Continued)

```
echo "    -V <WebLogic version>"
echo "    -U <admin URL>"
echo "    -u <WebLogic username>"
echo "    -p <WebLogic password>"
echo "    -T <target names> (optional)"
echo "    -f <file> (for Compile or X-Fer action)"
echo ""
echo "    -x X-Fer"
echo "    -c Compile"
echo ""
echo "    -a agent"
echo "    -d darchive"
echo "    -q MySQL database"
echo ""
echo "    -v Verbose"
echo ""
echo "examples:"
echo ""
echo "Compile all the logs from the CAMM installation within \$ACSERA_HOME, with the
related agents from \"t3://localhost:7001\", into the TAR file \"outnik.tar\";";
echo "$0 -V 8.1.3 -U t3://localhost:7001/ -u weblogic -p weblogic -cadf outnik.tar"
echo ""
echo "Compile all the logs from the CAMM installation within \$ACSERA_HOME, with the
related agents from \"t3://localhost:7001\", and transfer them to the support site as customer
\"sample\".:";";
echo "$0 -V 8.1.3 -U t3://localhost:7001/ -C sample -u weblogic -p weblogic -cxad"
echo ""
echo "Transfer \"somelog.log\" to the support site as customer \"sample\".:";";
echo "$0 -C sample -xf somelog.log"
echo ""
echo ""
fi
else
echo "ACSERA_HOME has not been set properly. Please set the ACSERA_HOME environment variable
to your target CAMM installation."
fi
```

After running the `Lar.sh` file, you will see the generated `.tar` file in the same directory where you launched the script.

[Code 7-3](#) is an example of the contents of the `.tar` file. The numbers inside brackets in the example are used to show the directory level in the hierarchy structure. These numbers do not show in the actual file. Note the hierarchy of the directories as they were set up in the script to extract the files.

Code 7-3: Directory Structure of the Extracted .tar Log File

```

foo
  (1)cygdrive
    c
      Oracle39_0599(build high level directory)
        (2)bin
          foo(higher level directory)
            (3)cygdrive
              c
                Oracle39_0599
                  (4)container
                    webapp
                      qvadmin
                        WEBINF
                          log
                            Web-log file
                    (4)darchive
                      Oracle
                        A25(Machine number)
                          Apps
                            (Application Name)
                              Metadata
                                Analysis.csv
                                Arch-Analysis.csv
                                Fileindex.csv
                    (4)log
                      All DeployerShell log files
            (3)temp
              lar.(Date)
  
```

Code 7-3: Directory Structure of the Extracted .tar Log File (Continued)

```
    root
      (Server Name)
        Logs
          (Server Name)
            Agent log files
(2)container
  webapps
    qvadmin
      WEBINF
        log
          Web log file
(2)darchive
  Oracle
    Machine Number(A25)
      Apps
        Application name
          Metadata
            Analysis.csv
            Arch-Analysis.csv
            Fileindex.csv
(2)log
  deployershell log files
(1)tmp
  lar.date(timestamp)
    root
      acsera.sql
      archive.err
      target.err
```

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](#)

Export to File

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

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.

Aggregation Export to File

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

Data Export Configuration

Please refer to the Data Export configuration section of the *CAMM™ Configuration Guide* for more information on how to set up and configure this feature. The data export feature is exactly the same between all supported platforms from BEA, IBM, and Oracle.

Example of Exported Data for WebLogic

Table 7-2: Export File Name: *metricBEA_ChannelInstance.csv*

Field	Description
StartTime	clock time (long) at data insertion.
EntityID	CAMM™'s 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.

Table 7-2: Export File Name: *metricBEA_ChannelInstance.csv* (Continued)

Field	Description
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
containerID	Name of the BEA WebLogic® Server instance.
nodeID	Name of the physical machine.
domainID	Name of the BEA 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 7-3: Export File Name: *metricBEA_ProcessType.csv*

Field	Description
StartTime	Clock time (long) at data insertion.
EntityID	CAMM™'s 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 BEA WebLogic® Server instance.
nodeID	Name of the physical machine.
domainID	Name of the BEA WebLogic® domain.
entityTypeID	Type of the monitored entity.

Table 7-3: Export File Name: metricBEA_ProcessType.csv (Continued)

Field	Description
applicationID	Name of the Application.
infrastructureID	ID of the monitoring infrastructure. Oracle is the only value at this time.
deploymentID	Unique ID used by BEA to track application deployments.
resourceID	Name of the monitored resource as configured by the user.
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 7-4: Export File Name: metricBEA_TimerEventGenerator.csv

Field	Description
StartTime	Clock time (long) at data insertion.
EntityID	CAMM™'s 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 BEA WebLogic® domain.

Table 7-4: Export File Name: metricBEA_TimerEventGenerator.csv (Continued)

Field	Description
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 7-5: Export File Name: metricJ2EE_Dispatcher.csv

Field	Description
StartTime	Clock time (long) at data insertion.
EntityID	CAMM™'s 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 BEA WebLogic® Server instance.
nodeID	Name of the physical machine.
domainID	Name of the BEA 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.
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 7-6: Export File Name: metricJ2EE_EJB_Entity.csv

Field	Description
StartTime	Clock time (long) at data insertion.
EntityID	CAMM™'s unique identifier for the monitored entity.
methodID	Name of the EJB method executed.
domainID	Name of the BEA 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 BEA 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.

Table 7-6: Export File Name: metricJ2EE_EJB_Entity.csv (Continued)

Field	Description
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.
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 7-7: Export File Name: metricJ2EE_EJB_Stateless.csv

Field	Description
StartTime	Clock time (long) at data insertion.

Table 7-7: Export File Name: metricJ2EE_EJB_Stateless.csv (Continued)

Field	Description
EntityID	CAMM™'s 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 BEA WebLogic® Server instance.
domainID	Name of the BEA 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.
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 7-8: Export File Name: *metricJ2EE_JDBC_ConnectionPool.csv*

Field	Description
StartTime	Clock time (long) at data insertion.
EntityID	CAMM™'s 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 BEA WebLogic® Server instance.
nodeID	Name of the physical machine.
domainID	Name of the BEA 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 7-9: Export File Name: metricJ2EE_JMS_Destination.csv

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

Table 7-9: Export File Name: metricJ2EE_JMS_Destination.csv (Continued)

Field	Description
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 7-10: Export File Name: metricJ2EE_JMS_Service.csv

Field	Description
StartTime	Clock time (long) at data insertion.
EntityID	CAMM™'s 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 BEA WebLogic® Server instance.
nodeID	Name of the physical machine.
domainID	Name of the BEA 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 7-10: Export File Name: metricJ2EE_JMS_Service.csv (Continued)

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

Table 7-11: Export File Name: metricJ2EE_JVM.csv

Field	Description
StartTime	Clock time (long) at data insertion.
EntityID	CAMM™'s 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 BEA WebLogic® Server instance.
nodeID	Name of the physical machine.
domainID	Name of the BEA 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.JRokit.HeapSizeCurrent	JMX metric.
Metric.J2EE.JVM.JRokit.HeapFreeCurrent	JMX metric.
Metric.J2EE.JVM.JRokit.PhysMemTotal	JMX metric.
Metric.J2EE.JVM.JRokit.PhysMemUsed	JMX metric.
Metric.J2EE.JVM.JRokit.GarbageCollectionCountTotal	JMX metric.
Metric.J2EE.JVM.JRokit.GarbageCollectionTimeTotal	JMX metric.

Table 7-11: Export File Name: metricJ2EE_JVM.csv (Continued)

Field	Description
Metric.J2EE.JVM.HeapFreeCurrent	JMX metric.
Metric.J2EE.JVM.JRokit.PhysMemFree	JMX metric.
Metric.J2EE.JVM.JRokit.NursurySizeTotal	JMX metric.
Metric.J2EE.JVM.JRokit.ActiveDaemonThreads	JMX metric.
Metric.J2EE.JVM.JRokit.ActiveThreads	JMX metric.
Metric.J2EE.JVM.HeapSizeCurrent	JMX metric.
Metric.J2EE.JVM.JRokit.HeapUsedCurrent	JMX metric.

Table 7-12: Export File Name: metricJ2EE_Server.csv

Field	Description
StartTime	Clock time (long) at data insertion.
EntityID	CAMM™'s 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.
containerID	Name of the J2EE server instance.
nodeID	Name of the physical machine.
domainID	Name of the BEA 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 7-13: Export File Name: metricJ2EE_Servlet.csv

Field	Description
StartTime	Clock time (long) at data insertion.
EntityID	CAMM™'s 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 BEA WebLogic® Server instance.
nodeID	Name of the physical machine.
domainID	Name of the BEA 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.

8

Creating Custom Views

This chapter includes the following topics:

- [Custom View Creation](#)
- [Instance Specific Custom Views](#)

Custom View Creation

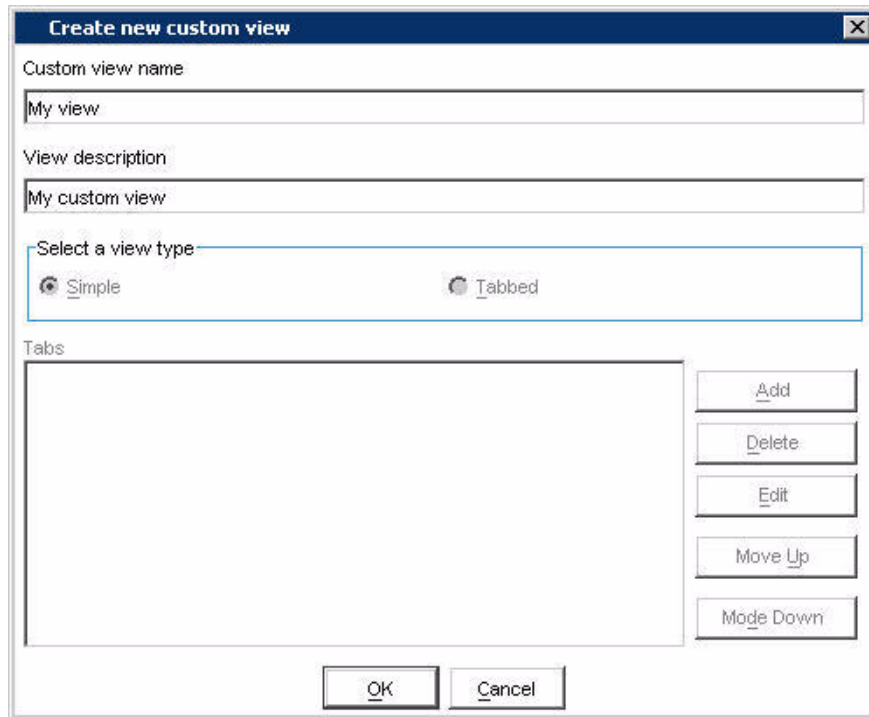
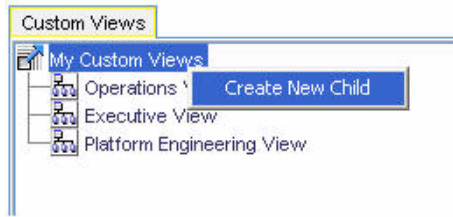
You can create custom views in CAMM™ easily by dragging and dropping visual components. The following are steps on how to create a custom view.

To create a custom view:

1. Right-click on the My CustomViews node in the Custom Views panel.

2. Select Create New Child to start the Custom View creation process. See [Figure 8-1](#).

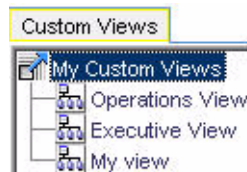
Figure 8-1: Creating New Custom View



In this example, MyView and My custom view for the name and description is selected.

3. Click **Ok** and the new custom view adds to the My Custom View tree. See [Figure 8-2](#).

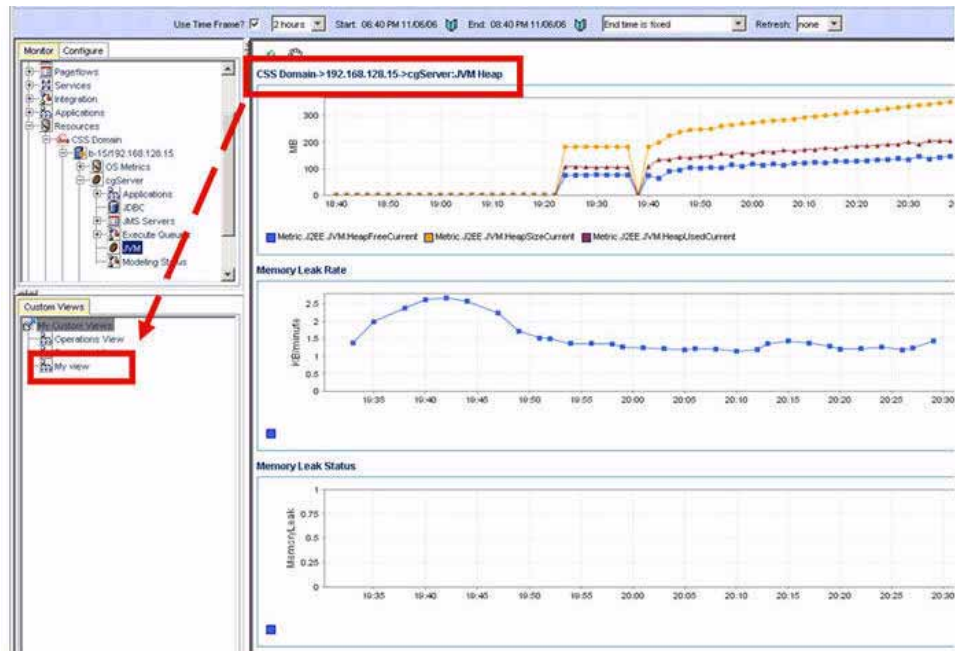
Figure 8-2: New Custom View is Added to My Custom Views Tree



4. Drag and drop view elements to create your custom views. In this example, we add the bottom graph in [Figure 8-3](#) to your custom view.

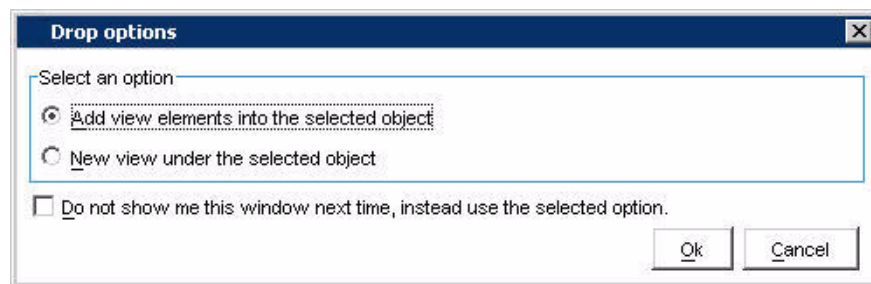
5. Click the graph name and drag it to MyView in the My Custom Views tree.
6. When you click the graph name and start dragging, make sure a + icon appears next to your mouse cursor.
7. As you move your mouse cursor to MyView, the same + icon should reappear when MyView is highlighted. This is an indication that the drag and drop operation is successful.

Figure 8-3: Add a View Element to a Custom View by Drag and Drop



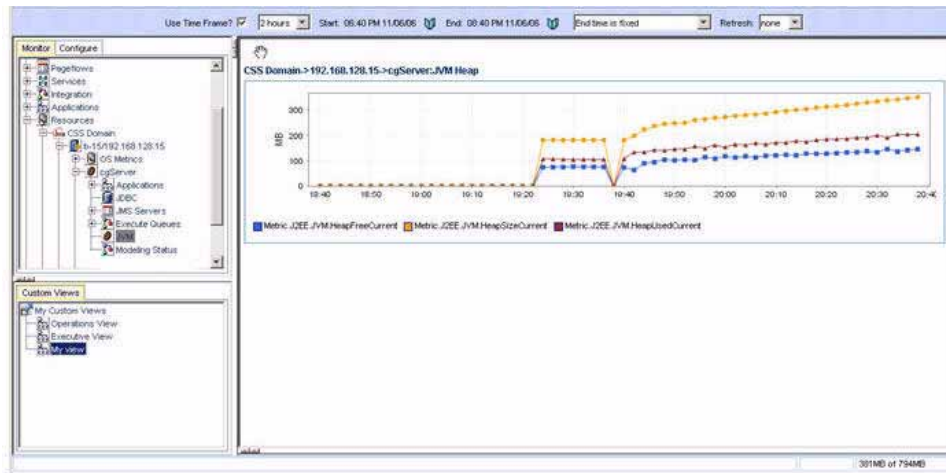
8. Release the mouse and the Drop Options window shows. You can select to add this view element or create a new view.

Figure 8-4: Drop Options





9. Click **Ok**. Your custom view now contains the new view element.

Figure 8-5: New View Element Added to Custom View



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

To add all elements displayed in the Main Display Window:

1. Click the  icon in the Main Display Window then perform a drag and drop operation.
2. When you click the  icon and start dragging, make sure a + icon shows next to your mouse cursor.
3. As you move your mouse cursor to MyView, the same + 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 shows. 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.

Figure 8-6: New View Elements Added to Custom View



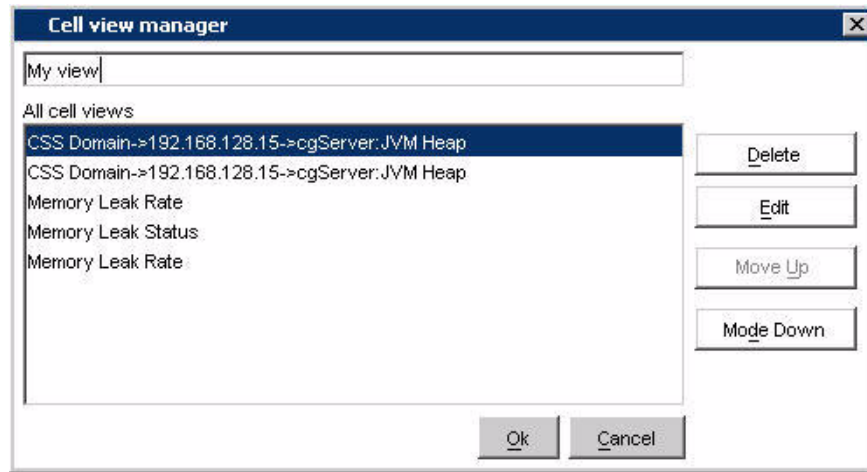
6. To edit or delete a custom view, right-click on the interested custom view.

Figure 8-7: Right-Click Menu for Custom Views



7. Select Edit to open the Cell view manager window. On this window you can edit custom view, delete view elements, and reorder view elements.

Figure 8-8: Cell View Manager



Instance Specific Custom Views

You can bring up multiple CAMM™ UI instances to work with multiple CAMM™ Manager instances. Since CAMM™ stores custom view information locally on the client machine, it creates and manages multiple custom view definitions.

By default, CAMM™ will prefix the custom view definition file with the name or IP address of the CAMM™ Manager. These custom view definition files are located on the client machine - location of the file depends on the OS.

For Windows, these custom view definition files are stored under the Documents and Settings directory of the client machine. Here is an example:

```
C:\Documents and Settings\user_abc\.acsera_preferences\192.168.3.167_userpref.ser
```

For Linux and UNIX, these custom view definition files are stored under the user's home directory on the client machine. Here is an example:

```
~/ .acsera_preferences/192.168.3.167_userpref.ser
```

Since the `acsera_preferences` directory starts with a period, you must issue the `ls` command with `-al` arguments to see the full list.

Optionally, you may assign unique identifiers for various CAMM™ instances. To do so, you may add the `ServiceController.ManagerID` property to the `Acsera.properties` file. You can set the value of `ServiceController.ManagerID` to any string. The value of `ServiceController.ManagerID` will be used as the prefix for the associated custom view definition file.

9

Custom Dashboards

This chapter includes the following topics:

- [Overview](#)
- [Working with Custom Dashboards](#)
- [Create Custom Dashboard](#)
- [Create Tabbed Dashboard](#)
- [Share Views](#)
- [Preview dashboards](#)

Overview

The Custom Dashboards feature is designed in order to allow users that are looking for a more customized display for 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 users to create their own dashboards and then utilize those dashboards as either the default display or as their own customized viewpoints. They are much more powerful than custom views as the user has 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.

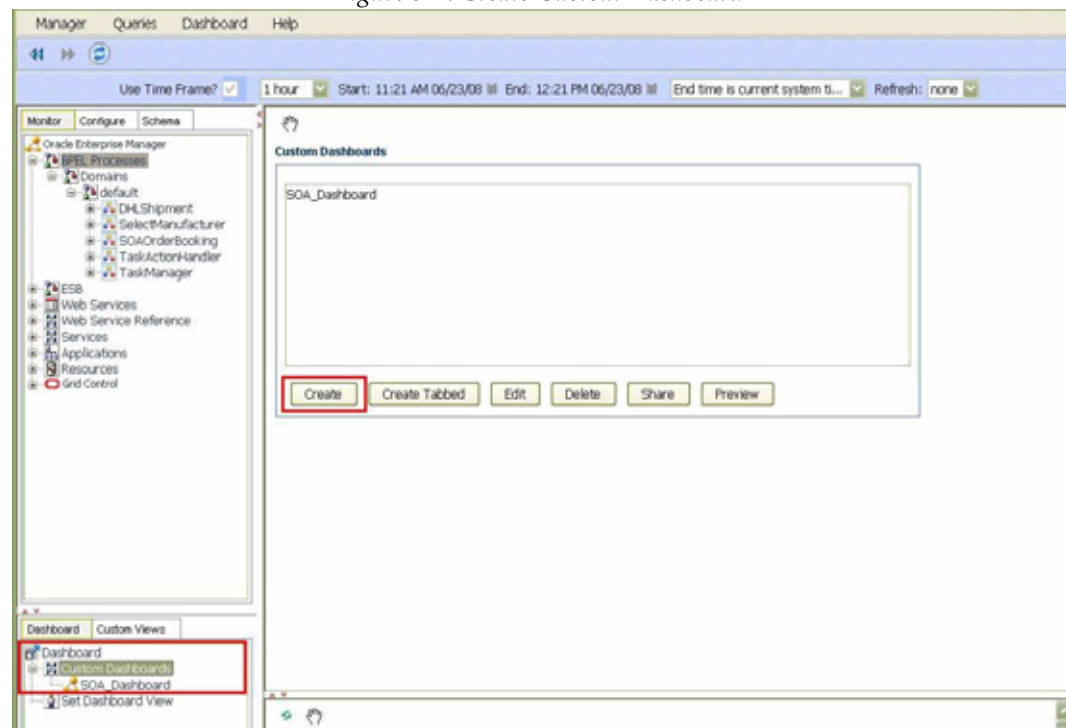
Working with Custom Dashboards

In order to illustrate how to take advantage of this feature, the following sections will walk the user through the creation of a custom dashboard as well as the various layout templates that can be utilized in its creation.

Create Custom Dashboard

In order to initiate the creation of a new custom dashboard, the user should click on the Custom Dashboards node in the tree that can be found in the lower-left hand pane of the CAMM™ UI as shown in [Figure 9-1](#) below.

Figure 9-1: Create Custom Dashboard

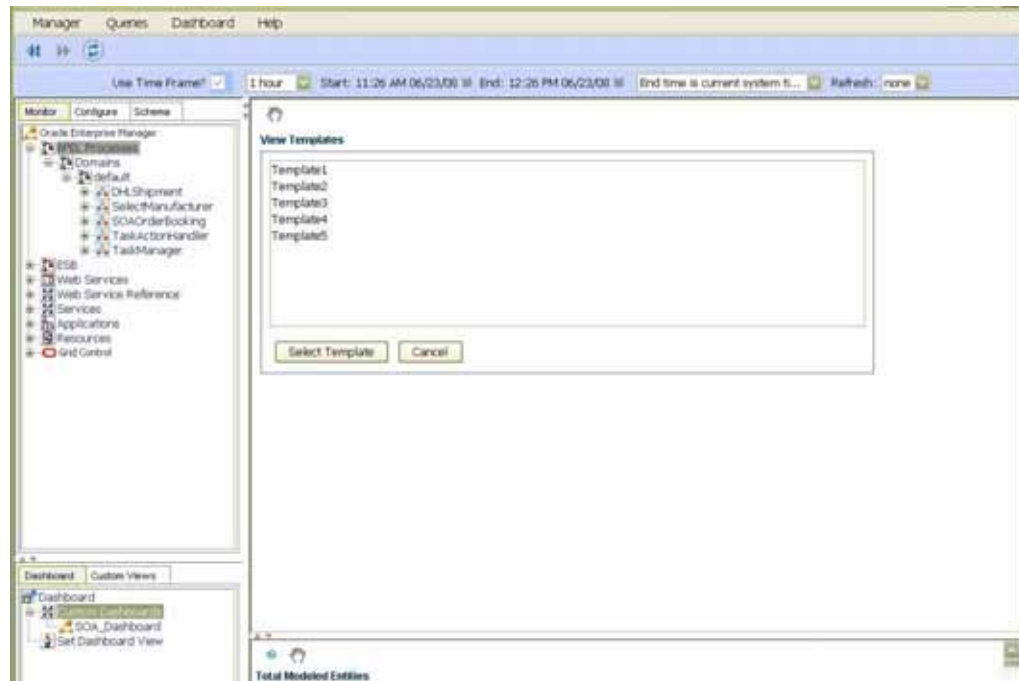


Layout templates

There are five different out-of-the-box layout templates to help format custom dashboards. The user should select the one with the appropriate layout for the custom dashboard they are looking for. If additional layouts are necessary, Oracle Professional Services can assist in the creation of new layout templates. However, in most cases, the following layout templates should suffice.

1. Click the **Custom Dashboards** option in the bottom left pane to view the templates.

Figure 9-2: Custom Dashboard Layout Templates



Each template has a different layout for components. See [Figure 9-3](#) to [Figure 9-7](#) to view the layouts in each template.

Figure 9-3: Custom Dashboard Template 1

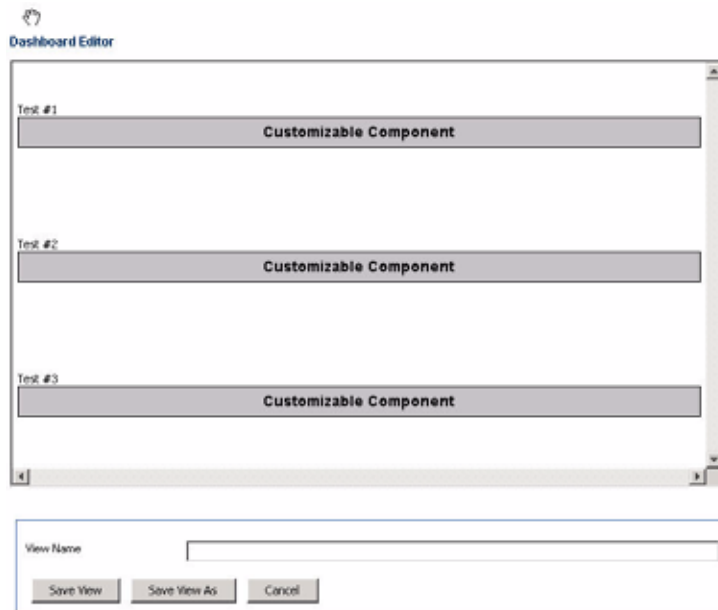


Figure 9-4: Custom Dashboard Template 2

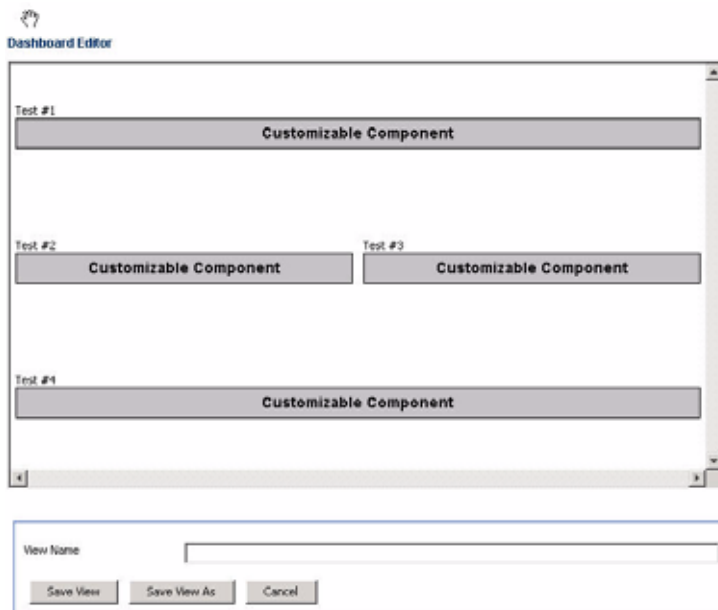


Figure 9-5: Custom Dashboard Template 3

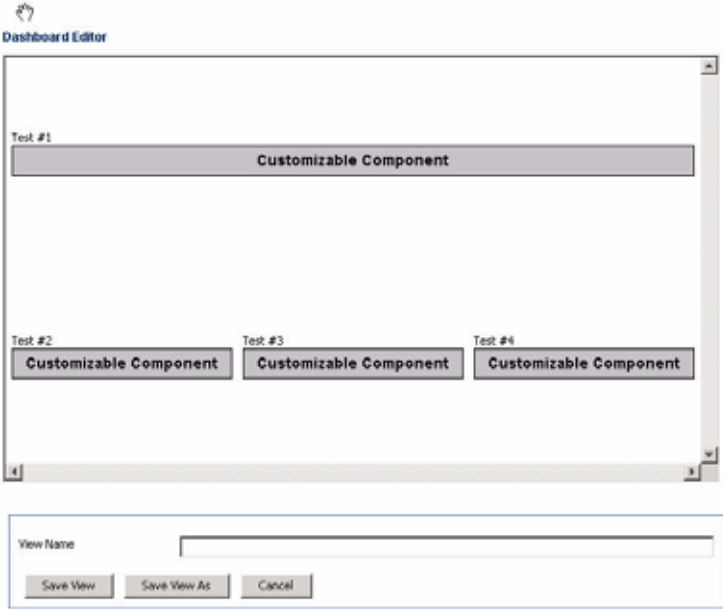


Figure 9-6: Custom Dashboard Template 4

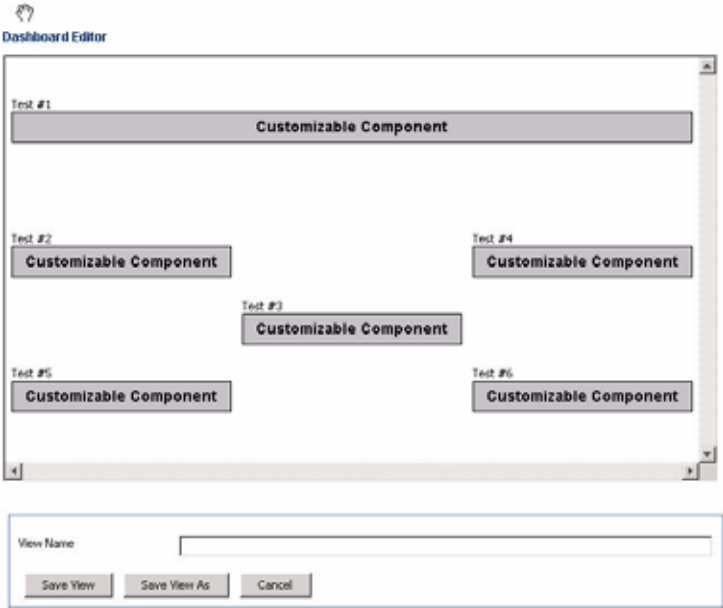
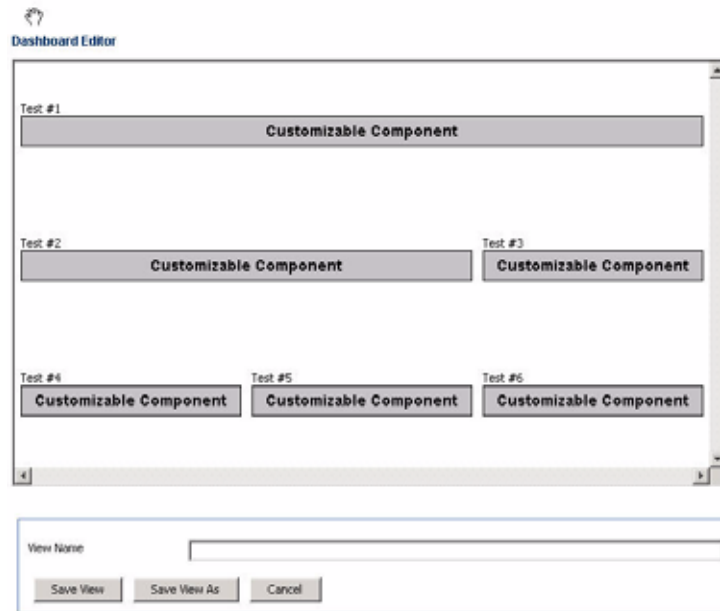


Figure 9-7: Custom Dashboard Template 5



Configuring Custom Components

2. Double-click on a **Customizable Component** area to configure what you would like to include in each component.
3. Select a component from the drop-down list.

Figure 9-8: Custom Dashboard Select Table Component

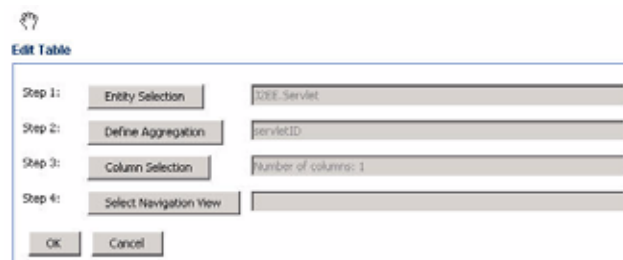


4. In this example, we select the **Table** type.

Custom Table

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

Figure 9-9: Custom Dashboard Edit Table



Select Entity

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

Figure 9-10: Custom Dashboard Entity Selection

The screenshot shows a dialog box titled "Select Entity" with a tree view of system components. The components are organized into several categories, each with a dropdown arrow on the right:

- Entity Type:** J2EE:Servlet
- Resource:** WLP_10.2, oc4j_soa
- Machine:** Frontier/192.168.1.105
- Server:** home.1@192.168.1.105, oc4j_soa.1@192.168.1.105
- Application:** SOADEMO, SOADEMO-CREDITSERVICE-CreditService-WS, SOADEMO-RAPIDSERVICE-RapidService-WS, ascontrol, ccore, coreman, datatags, default, default_SOAOrderBooking_1_0_ApproveOrder, esb-dt
- WebApplication:** soaui
- Servlet:** "Faces Servlet", "resources"

At the bottom of the dialog, there are two buttons: "Select Entity" and "Cancel".

Select Aggregation

9. Click **Define Aggregation**.
10. The key selected must aggregate by the same key metric name.

11. Click **OK**.

Figure 9-11: Custom Dashboard Aggregation Selection



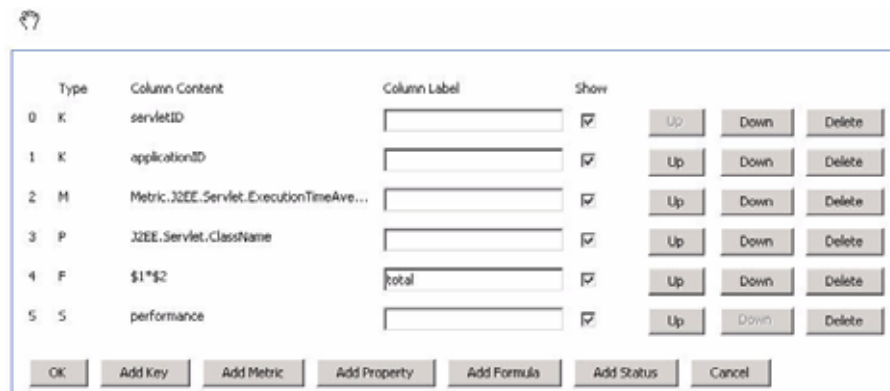
Select Columns

12. Click **Column Selection**.
13. 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.
14. Type in the labels for the columns added. The labels display in the component table header for each column.
15. You can click the **Up**, **Down** or **Delete** buttons to edit the columns.

Note: You need to add at least one Property or Metric in order for 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. See the example in [Figure 9-12](#).

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

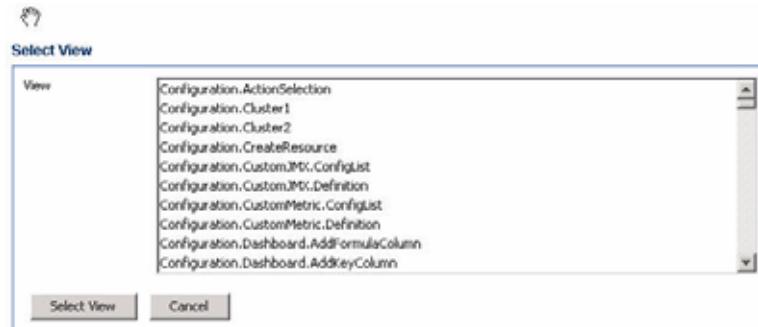
Figure 9-12: Custom Dashboard Add Columns



Select Navigation View

Users 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. It should be noted that only expert users such as Oracle professional services should utilize this feature as it requires a strong knowledge of the mapping capabilities as well as the application itself. Views can be selected from a drop-down list (see [Figure 9-13](#)), but once completed, the user configuring the navigation view must take extra care to ensure that IDs match up appropriately in order to handle the drill-down event.

Figure 9-13: Navigation View Selection



NOTE: Please contact Oracle Professional Services via support@Oracle.com if you are interested in this advanced navigation view feature.

Custom chart

We will now utilize a similar process to add a chart to our custom dashboard. Charts can be used 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. Various graphs can be configured in order to display the metrics as desired.

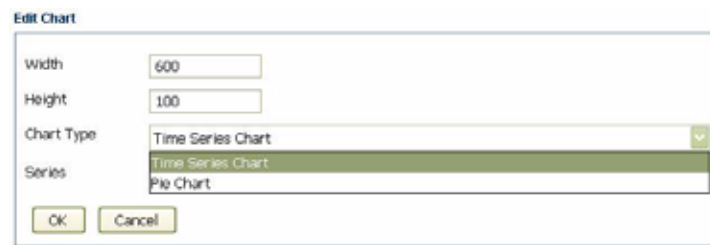
18. Double-click again on a different **Customizable Component** area to configure what you would like to include in each component and select “Chart” from the drop-down list shown in [Figure 9-8](#).

Chart type / dimensions

Users 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.

19. Select Time Series Chart Type as shown in [Figure 9-14](#).

Figure 9-14: Chart Type Selection



20. Click OK

Selecting entity

21. Click **Entity Selection**.
22. Select the entries in the Entity table. The selections have to be logical and have a unique entry for each entity.

23. Click **Select Entity** to continue with the configuration.

Figure 9-15: Custom Dashboard Entity Selection

The 'Select Entity' dialog box is shown with the following selections:

- Entity Type: J2EE.Servlet
- Resource: cc41_soa
- Machine: Frontier/192.168.1.105
- Server: cc41_soa.1@192.168.1.105
- Application: SOA DEMO
- WebApplication: soaui
- Servlet: "Faces Servlet"

Buttons: Select Entity, Cancel

Selecting metric

24. Select the metric that should be displayed in the chart as shown in [Figure 9-16](#).

Figure 9-16: Chart Metrics Selection

The 'Select Metric' dialog box is shown with the following list of metrics:

- Metric: J2EE.Servlet.ExecutionTimeAverage
- Metric: J2EE.Servlet.ExecutionTimeHigh
- Metric: J2EE.Servlet.ExecutionTimeLow
- Metric: J2EE.Servlet.ExecutionTimeTotal
- Metric: J2EE.Servlet.InvocationTotalCount

Buttons: OK, Cancel

Selecting aggregation

25. Select the metric that should be displayed in the chart as shown in [Figure 9-17](#).

Figure 9-17: Chart Metrics Aggregation Selection



Custom label

Next we will 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 in order to make the content instantly recognizable by colleagues utilizing the new dashboard.

26. 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 shown in [Figure 9-8](#).
27. 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 as shown in [Figure 9-18](#).

Figure 9-18: Label Configuration



28. Click OK

Custom image

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

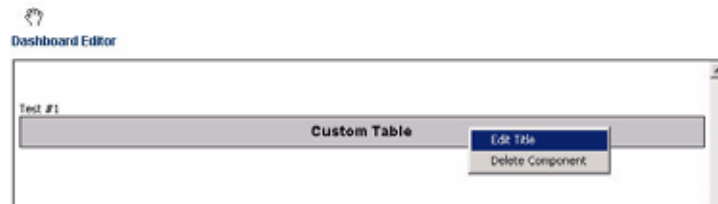
29. Double-click again on a different **Customizable Component** area to configure what you would like to include in each component and select “Image” from the drop-down list shown in [Figure 9-8](#).



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.

Figure 9-19: Custom Dashboard Edit Table



Save

30. When you have completed configuring the components, type in a view name in the Dashboard Editor window.
31. Click **Save View**.
32. The new template shows in the Custom Dashboards list.

Save View As

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

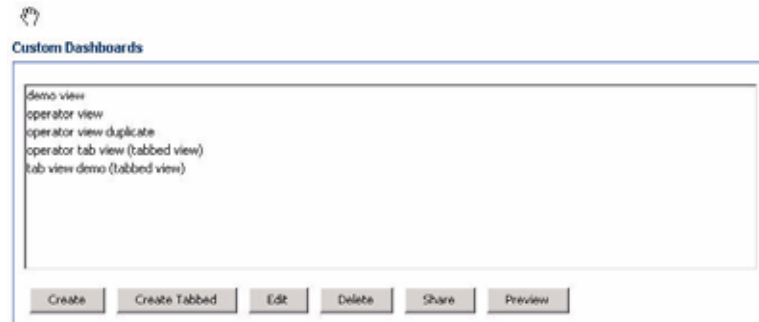
33. To duplicate a template, double-click its name from the Custom Dashboards window.
34. Type in a new name in the View Name field.
35. Click **Save View As**.
36. The new template view shows in the Custom Dashboards list.

Create Tabbed Dashboard

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

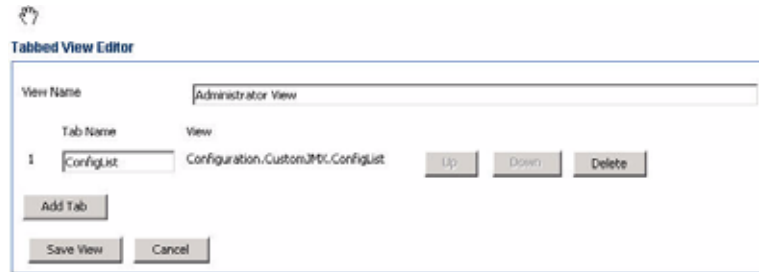
1. Click **Create Tabbed**.

Figure 9-20: Create Tabbed Dashboard



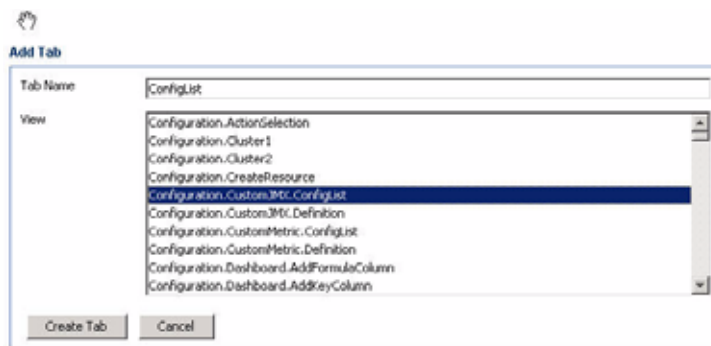
2. Type in the View Name. This name shows in the Custom Dashboards list window.

Figure 9-21: Tabbed View Editor



3. Click **Add Tab**.
4. Select the view for this tab from the displayed list.
5. Click **Create Tab**.

Figure 9-22: Add Tab

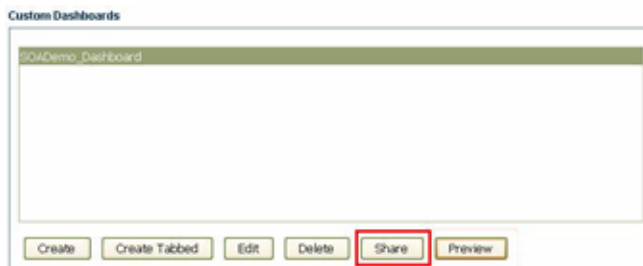


6. Type in the Tab Name in the empty text box.
7. Continue to add tabs as needed.
8. Click **Save View**.

Share Views

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

Figure 9-23: Share Dashboard



Preview dashboards

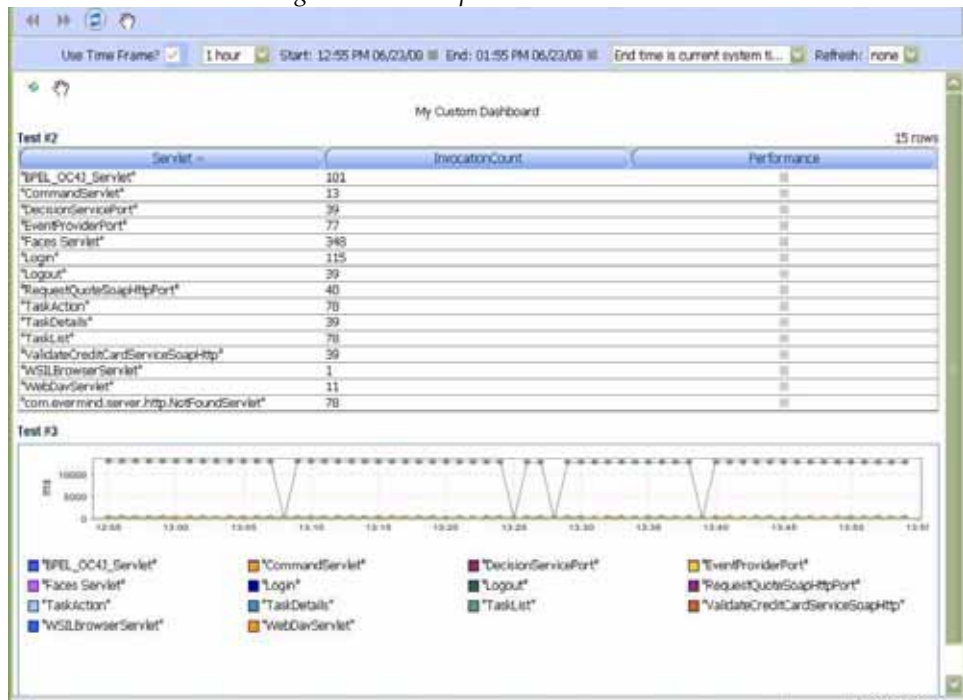
Existing dashboards can easily be created by clicking the "Preview" button as illustrated in [Figure 9-24](#).

Figure 9-24: Preview Dashboard



An example dashboard is shown in [Figure 9-25](#) that contains three Custom Components including a label, table, and a time series chart for active servlets on all managed servers.

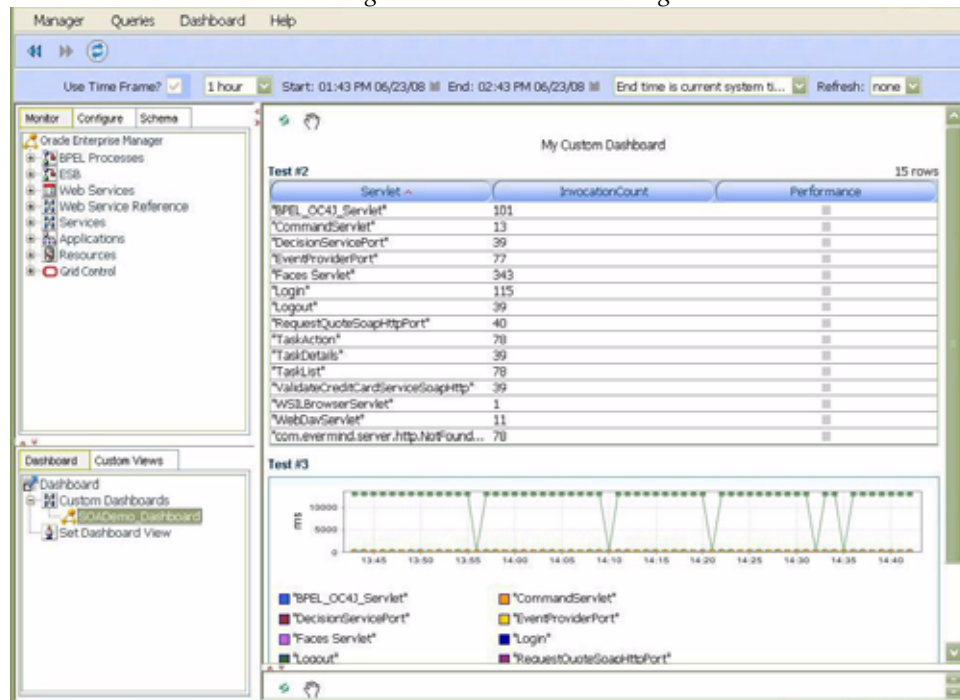
Figure 9-25: Sample Preview Dashboard



Utilizing and Viewing Existing Dashboards

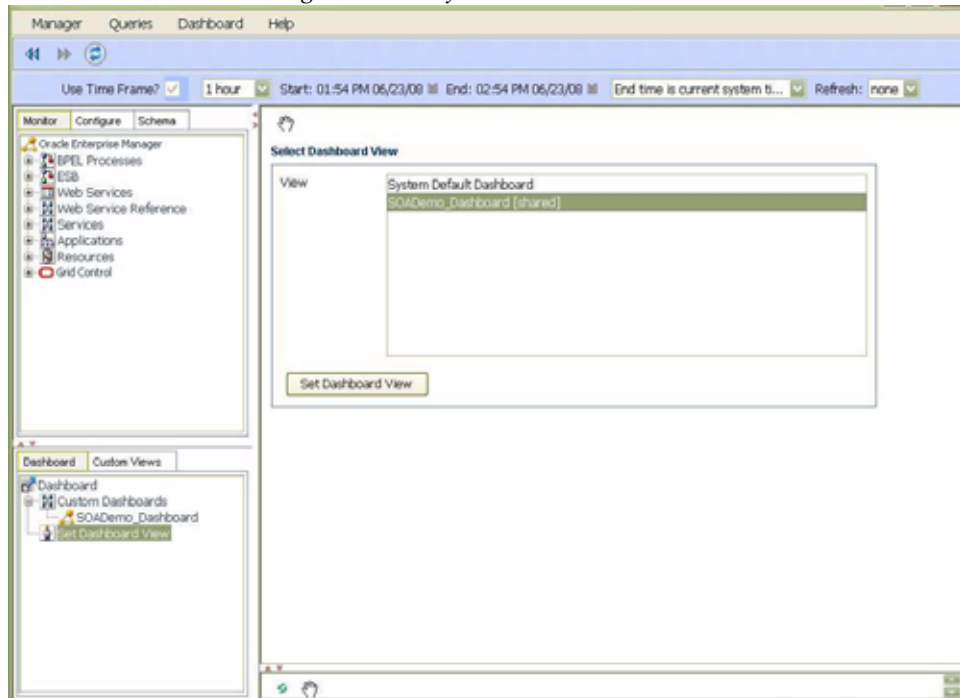
Once a new dashboard is created, it can either be viewed on its own by navigating to it via the navigation tree in the lower left-hand pane of the CAMM™ UI as shown in [Figure 9-26](#).

Figure 9-26: Dashboard Navigation



In addition, users can select their custom dashboards as the main dashboard display by selecting “Set Dashboard View” from the lower left-hand navigation bar and accessing the right-click menu. They would then be presented with a list of existing custom dashboards to select from. Once selected, the new custom dashboard would be displayed when the CAMM™ UI starts up or if the user selects the top-most node. It should also be noted that the default dashboard can be assigned via the top menu bar in addition to the navigation tree.

Figure 9-27: Default Dashboard Selection



10

Oracle CAMM™ Methodology

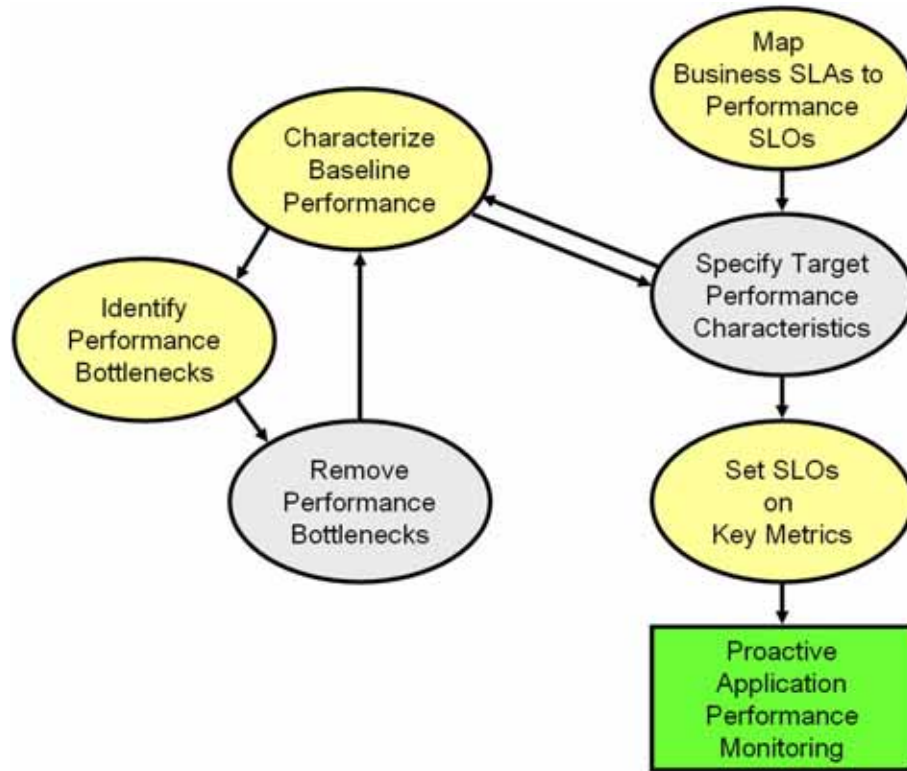
This chapter includes the following topics:

- [Oracle™ Methodology Activities](#)
- [Map Business SLAs to performance SLOs](#)
- [Characterize Baseline Performance](#)
- [Identify Performance Bottlenecks](#)
- [Set SLOs on Key Metrics](#)
- [Conclusion](#)

CAMM™ automatically selects performance metrics and tracks contextual relationships for various applications. The Oracle™ 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 10-1: 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 10-1](#) 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.
These three steps 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, we need to set performance SLOs on key metrics to establish a proactive monitoring environment. This environment provides us with warnings when key performance metrics start to report abnormalities. These warnings enable us to proactively solve potential problems before they begin to impact business.

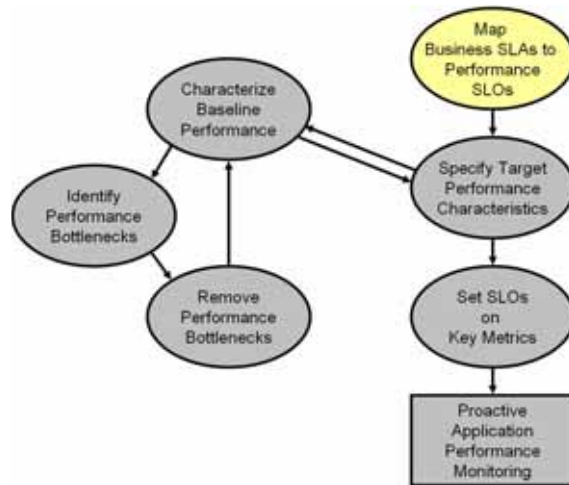
This chapter explains why the Oracle™ Methodology activities are important and covers the following activities in more detail:

- Map business SLAs to performance SLOs
- Measure baseline performance
- Identify performance bottlenecks
- Set SLOs on key system 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.

Oracle™ Methodology Activities

Map Business SLAs to Performance SLOs



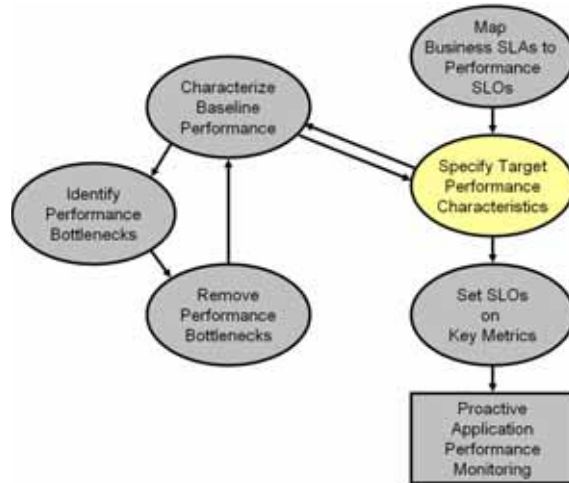
To successfully setup a proactive application performance monitoring environment, the very first step is to map a set of business objectives to a set of performance thresholds for us to monitor. These business objectives are often referred to as business service level agreements (SLAs). These business 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, etc.) 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 very straightforward.

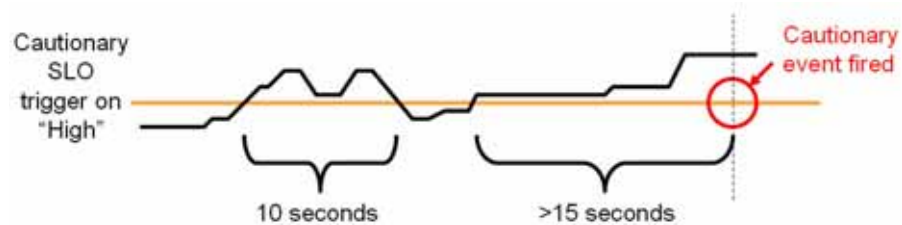
Specify Target Performance Characteristics



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, we 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, we can minimize the number of duplicate alerts generated. [Figure 10-2](#) diagram illustrates this concept.

Figure 10-2: Control Number of Alerts



In [Figure 10-2](#) example, we defined the minimal violation duration to be 15 seconds. So if the cautionary state does not persist for more than 15 seconds, no cautionary alert would be fired.

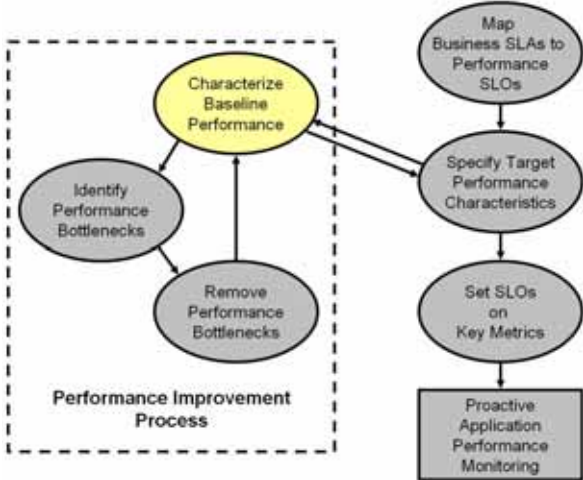
For other applications, it is necessary to define both a high and low performance thresholds. Having both thresholds would effectively define a normal range of operation for these applications. With CAMM™, both high and low triggers can be set for any SLO.

With a set of clearly defined target performance characteristics, you will be 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.

Performance Improvement Process

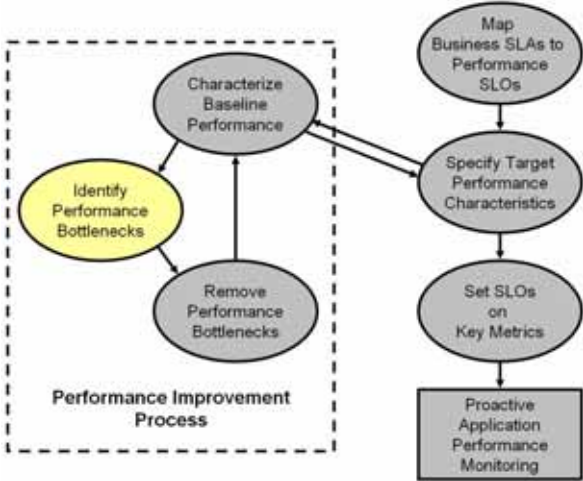
The next three activities should be grouped together as a single performance improvement process. This process would start with characterizing the baseline performance of our application, move on to identifying performance bottlenecks, and finish with removing performance bottlenecks. We would continue to perform these activities in iterations until the performance of our application meets the target characteristics.

Characterize Baseline Performance



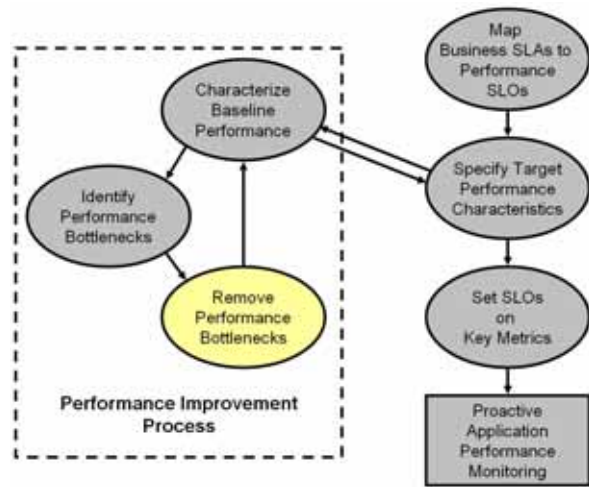
Once the specification of the target performance characteristics is completed, the next activity is to capture the performance baseline for our application. The performance baseline will be compared with the set of target performance characteristics to determine if further performance improvement is needed. If so, you will improve application performance iteratively through the next two steps until the performance meets the target characteristics. With CAMM™, this activity can be done very easily.

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. With CAMM™, this activity can be done very easily.

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 are:

- 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,

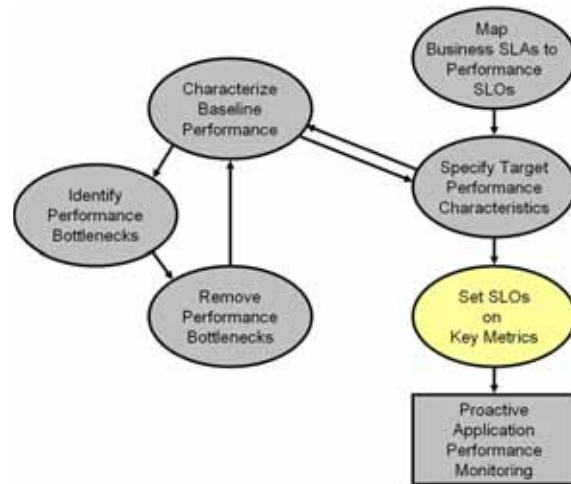
you would seek help from those vendors.

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.

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.

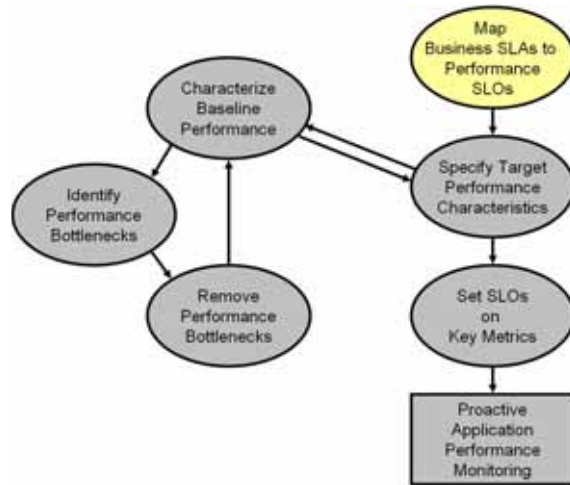
To determine which system metrics to monitor, it is critical to figure out the correlations between overall system performance and specific system metrics. You would use this information to decide which of the system metrics to monitor. Once appropriate system metrics are identified, you will then determine 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.

In order 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 conclusion 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.

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 extremely easy to perform this mapping exercise. In this section, our example shows you how to use CAMM™ to determine the proper performance threshold values for a set of business SLAs.

In our example, we were given the following high-level business SLAs:

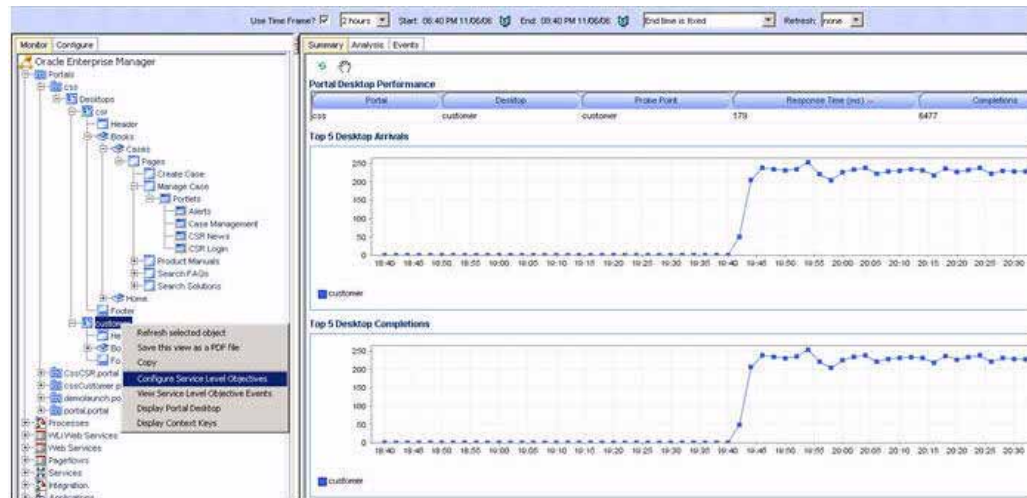
Table 10-1: Example - Guidelines for Business SLAs

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.

In our example, we are going to map three business SLAs to actual performance SLOs that we monitor. The first two SLAs are related to performance of two portals - a customer self-service portal and a customer service representative portal. We want to use this example, because these portals are actually just two different desktops of a single portal application. As such, these desktops share many common components. Since the performance requirements for these common components differ under various scenarios, it is very important to be able to set different performance thresholds based on the context. CAMM™ allows you to do this very easily.

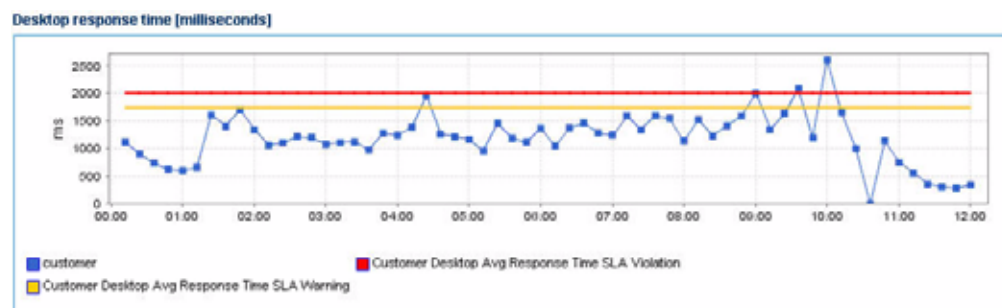
Let's map the first business SLA to a performance SLO. This SLA requirement states that the average response time for customer self-service portal (desktop) should be less than 2 seconds. In CAMM™, we would set a high-level performance SLO at the desktop. Using the hierarchy in the CAMM™ UI, you would select the *customer* desktop and right-click to set the SLO.

Figure 10-3: Configure Service Level Objectives



Mapping the business SLA to performance SLA in CAMM™ is very straight forward. 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. The result would look like the capture in [Figure 10-4](#):

Figure 10-4: Cautionary and Violation SLOs

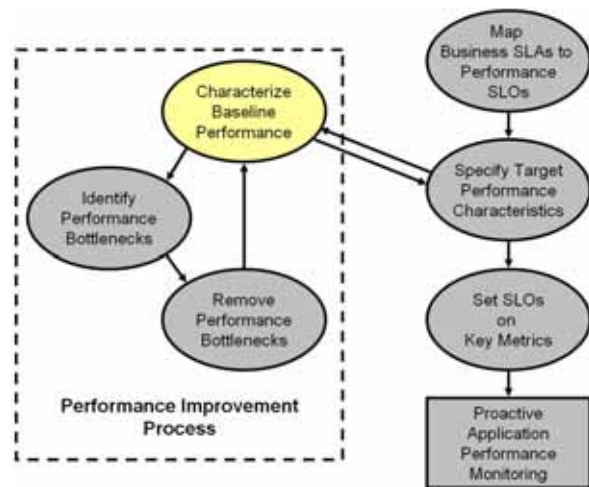


As indicated in [Figure 10-4](#), we see some violations already. We can calculate how often these violations occur to figure out whether or not our current system is able to meet the SLA requirement 99% of the time. With CAMM™, we can visually see whether there are any obvious violations very easily. If there are any violations or close calls, we should confirm by examining actual data. In our example, we have data for at least 24 hours. We would use CAMM™'s export function to prepare raw data for this calculation.

For the other two business SLAs, we would just set performance SLOs on the appropriate metrics. For customer service representative portal SLA, we would set a SLO on the *csr* average desktop response time. For SLA on schedule service call process, we would set a SLO on the average elapse time for **schedule service call** process.

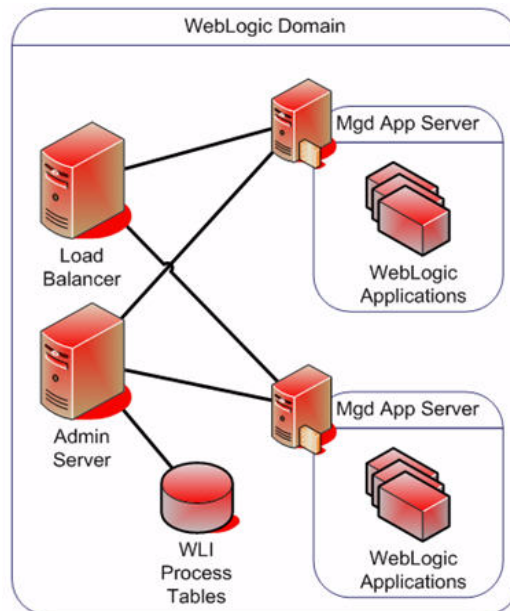
As you can see in the example, setting SLOs that map directly to business SLAs is very easy with CAMM™ since it monitors at both functional and technology levels. The mapping process would be significantly more difficult with tools that only measure at the technology level. With these tools, you would need to make educated guesses and/or be intimately familiar with the application to link performance of low-level metrics to high-level business SLAs.

Characterize Baseline Performance



Also known as performance baselining, characterize baseline performance involves a set of activities to capture the baseline performance of a system under specific level of load. In this example, we will measure the baseline performance of a portal application deployed to a WebLogic® cluster. [Figure 10-5](#) illustrates the configuration of the cluster.

Figure 10-5: WebLogic® Cluster Configuration



In this example, we generated a steady load against the portal application. [Figure 10-6](#) is an example of CAMM™ displaying the performance data during the first four hours of this load test.

Figure 10-6: CAMM™ Displaying Portal Activities

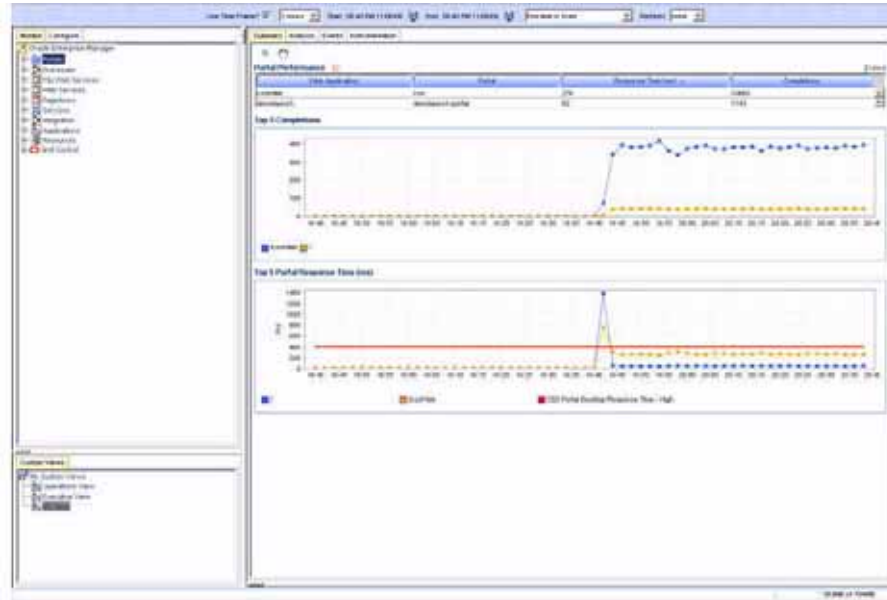
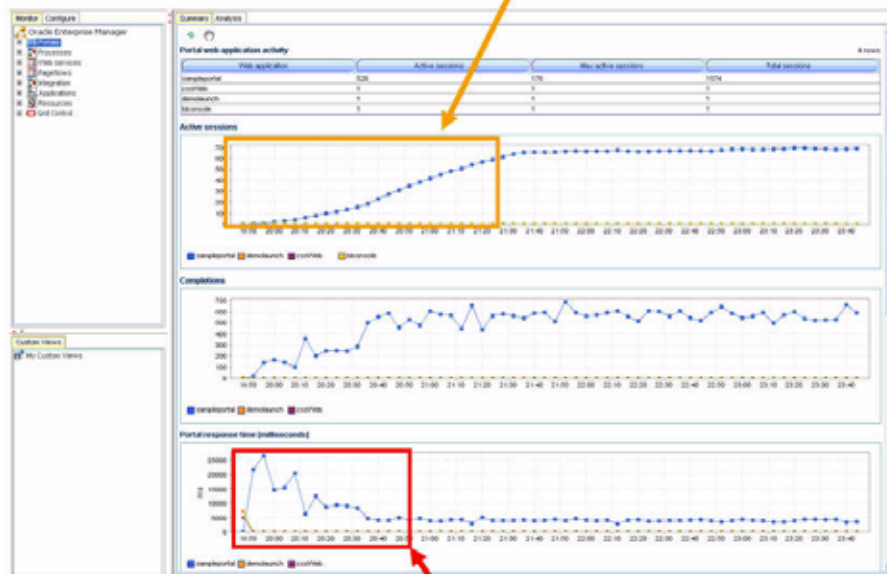


Figure 10-7: Visual Analysis Using CAMM™

of active session continues to grow steady for 90 minutes



Portal response time settles down after 30 minutes

As you can see in [Figure 10-7](#), 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.

The portal performance follows a typical pattern of slow performance initially and gradually reaches 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 minutes period can be characterized as *startup performance during increasing load*. After thirty minutes, performance of the portal application stabilizes. In our example, the performance of the portal application stabilizes at average response time of approximately five seconds with seven hundred active sessions.

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.

In [Figure 10-9](#) and [Figure 10-10](#) the graphs represent the performance of the portal application across the cluster. They offer a comparison of the performance of the portal application on individual Servers 192.168.3.185 and 192.168.3.186.

Figure 10-8: Application Performance on Server 192.168.3.185



Figure 10-9: Application Performance on Server 192.168.3.186



Figure 10-10: Comparing Load Differences Between Servers in the Cluster

Load on Server 192.168.3.185



Load on Server 192.168.3.186



By comparing the *Application invocation counts measurements* for both servers, we can observe the load levels on these servers. Using CAMM™, you can see the load for Server 192.168.3.185 is approximately 2000 invocations per measurement window (fifteen seconds). That same measurement is approximately 1200 invocations per fifteen seconds for Server 192.168.3.186.

Due to the lower load on 192.168.3.186, we would expect faster response time and lower computing resource utilization. You can use CAMM™ to quickly confirm this theory. By comparing application performance measurements after the cluster has reached a steady state, we can see a big difference between these two servers.

Figure 10-11: Comparing Response Times Between Servers in a Cluster

Application Performance on Server 192.168.3.185

Application	Status	Response Time (ms)	Invocation Count	Active Sessions
portalApp	■	3723	29275	687

Application Performance on Server 192.168.3.186

Application	Status	Response Time (ms)	Invocation Count	Active Sessions
portalApp	■	3137	17185	685

As indicated in [Figure 10-11](#), application performance on the less loaded server is 15.7% faster. We can further verify by looking at resource usage for both servers.

Figure 10-12: Compare Resource Usages in a Cluster-Server 192.168.3.185

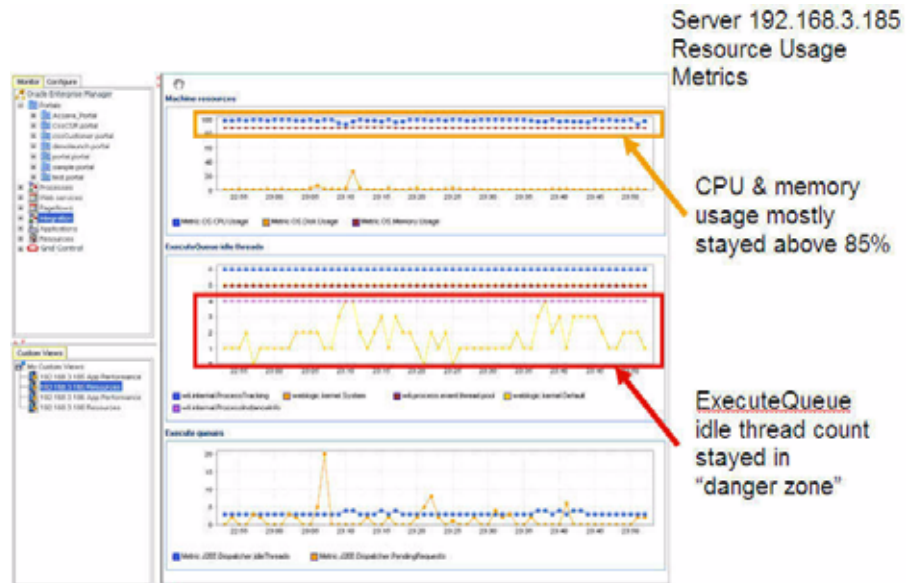
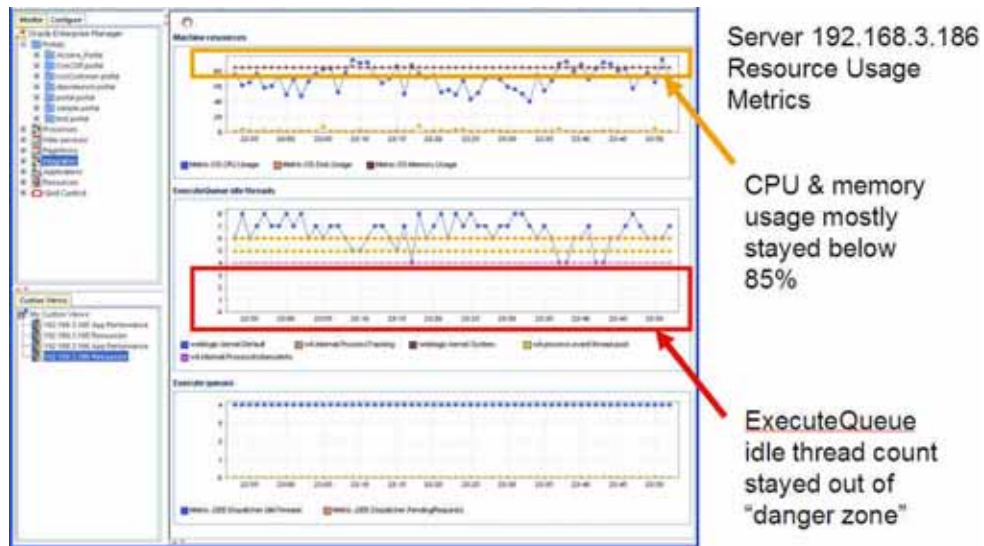


Figure 10-13: Compare Resource Usages in a Cluster-Server 192.168.3.186



By verifying that the less loaded server has lower resource utilization and faster performance, you can draw the following observations about the performance characteristics of this portal application running on this environment:

- Since Server 192.168.3.185'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™.

Application Invocation Count	= 29275
Time frame	= 1 hour
Average Peak Load per Second	= 29275 / 3600 seconds = 8.13 application invocations/sec

For those wanting to squeeze the most out of their system, the maximum load must fall below 8.13 application invocations per second for individual servers with this configuration.

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™.

- In this example, we should examine load balancing algorithm and the configuration of the load balancer. In our example, the cluster's load balancer fronts both WebLogic® servers. The uneven load observed on these servers is an

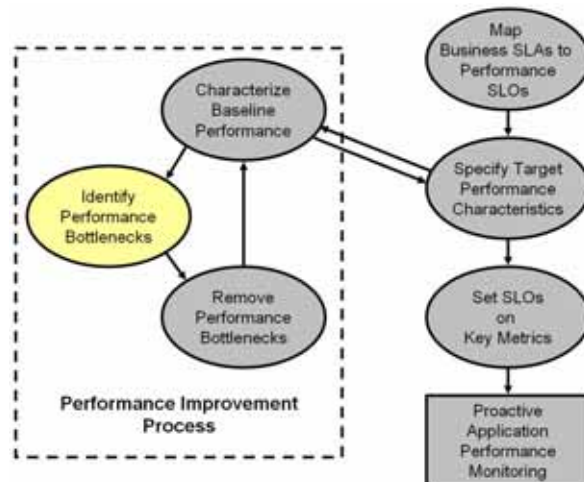
obvious indication of load balancing issues.

As shown in [Figure 10-14](#), comparing load and resource usage of two servers in a cluster confirms resource usage is inversely correlated to the load.

Figure 10-14: Comparing Load and Resource Usage in a Cluster



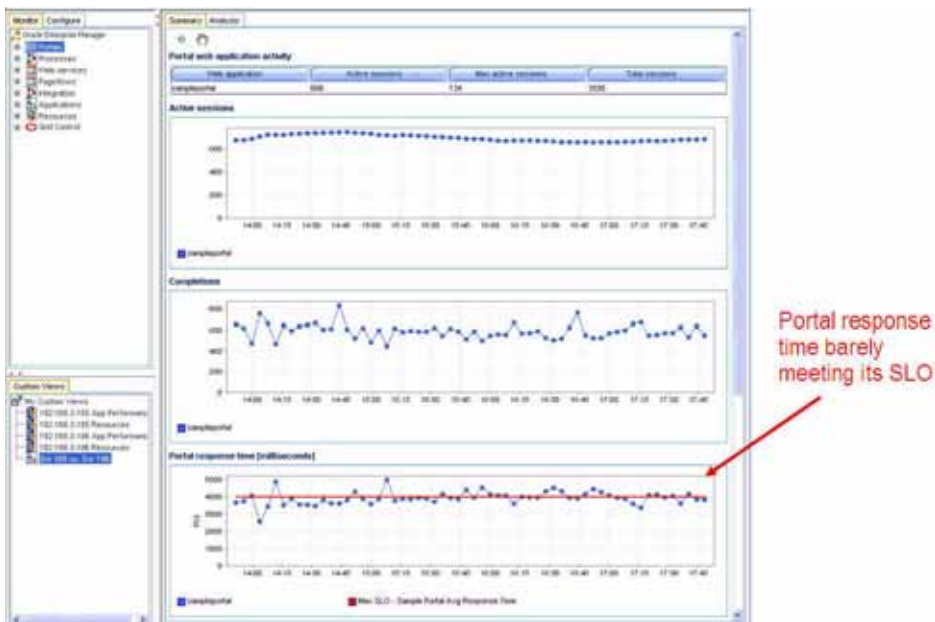
Identify Performance Bottlenecks



CAMM™ can also be used to quickly identify performance bottlenecks in QA, staging, and production settings. In our example, we will use CAMM™'s hierarchical model to identify an application performance bottleneck. Furthermore, we will provide an example of using CAMM™ to track down an application performance problem caused by resource starvation.

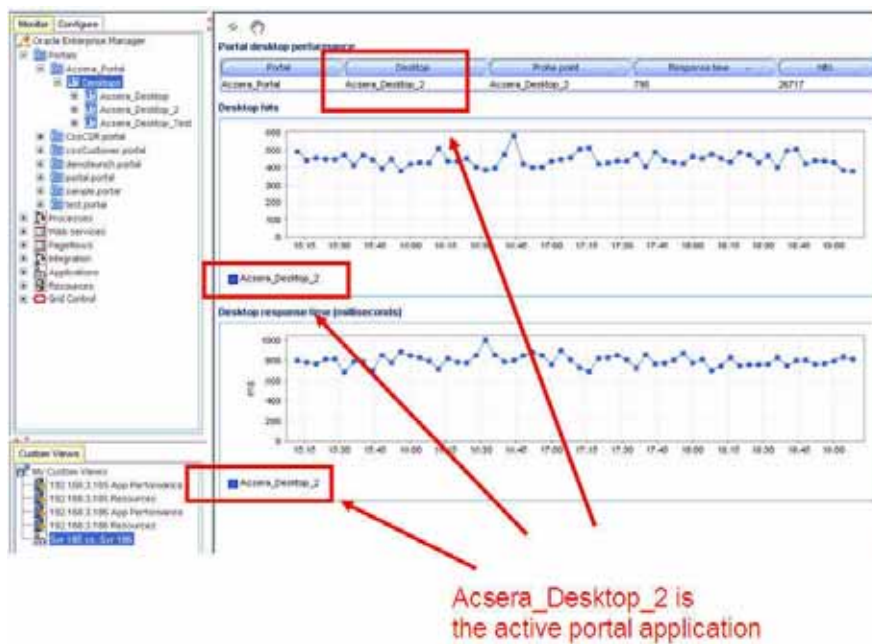
In our first example, we will identify the performance bottleneck of a portal application running on WebLogic®. Since CAMM™ organizes performance metrics into a hierarchy, we would start the investigation at the root of the portal hierarchy.

Figure 10-15: Portal Application Not Able to Meet Its Performance SLO



In our 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.

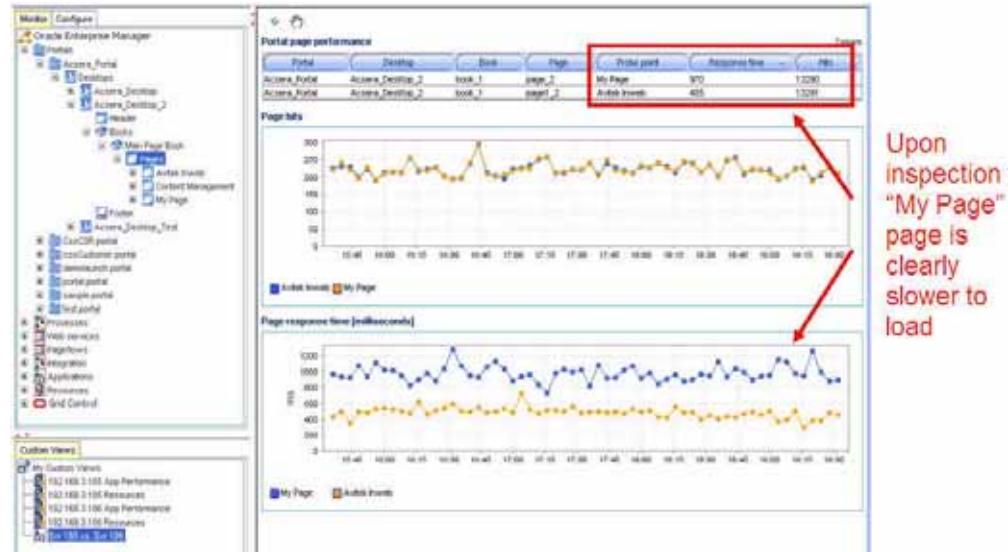
Figure 10-16: Drill Down on the Hierarchy to Identify Active Desktop



Since there is only one active portal desktop, continue to look in the Acsra_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 our case, it is clear, as indicated in [Figure 10-17](#), that the *My Page* is the slowest. Therefore, we will start our investigation by drilling down the hierarchy for My Page.

Figure 10-17: Drill Down on Hierarchy to Find the Slowest Page



Drilling down further on the My Page hierarchy, you can compare performance of all portlets that make up the My Page. In [Figure 10-18](#), CAMM™ clearly identifies the *ThreadedDiscussion* portlet as the worst performing.

Figure 10-18: Drill Down on Hierarchy to Find the Slowest Portlet



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.

In addition to organizing performance metrics by logical hierarchies, CAMM™ organizes performance information by type. This next example will use multiple performance metrics from different hierarchies in CAMM™ to determine the source of the performance bottleneck.

Figure 10-19: Visually Identify SLO Violation

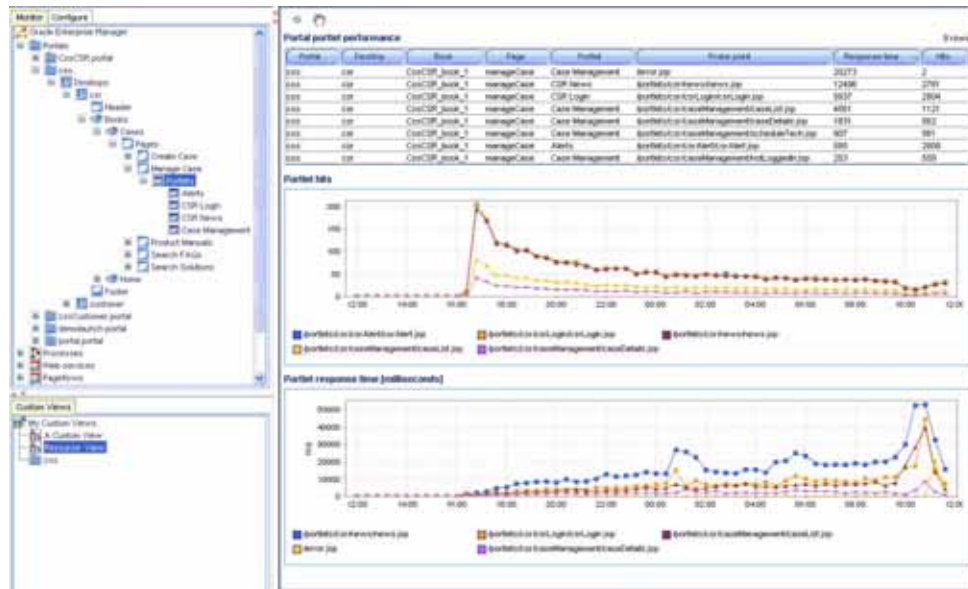


In this example, CAMM™ alerted us that the average response time for *csr* desktop has exceeded both the cautionary and violation SLOs previously defined. Using the CAMM™ to inspect these SLOs violations, we saw significant performance degradation during a ninety minute period. This continuous SLO violation warrants additional investigation to locate the source of this performance slowdown.

By looking at the *Desktop Hits* graph in [Figure 10-19](#), 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. [Figure 10-20](#) shows this drill down to the portlets level.

Figure 10-20: Analysis of Portal Components Revealed Consistent Behavior

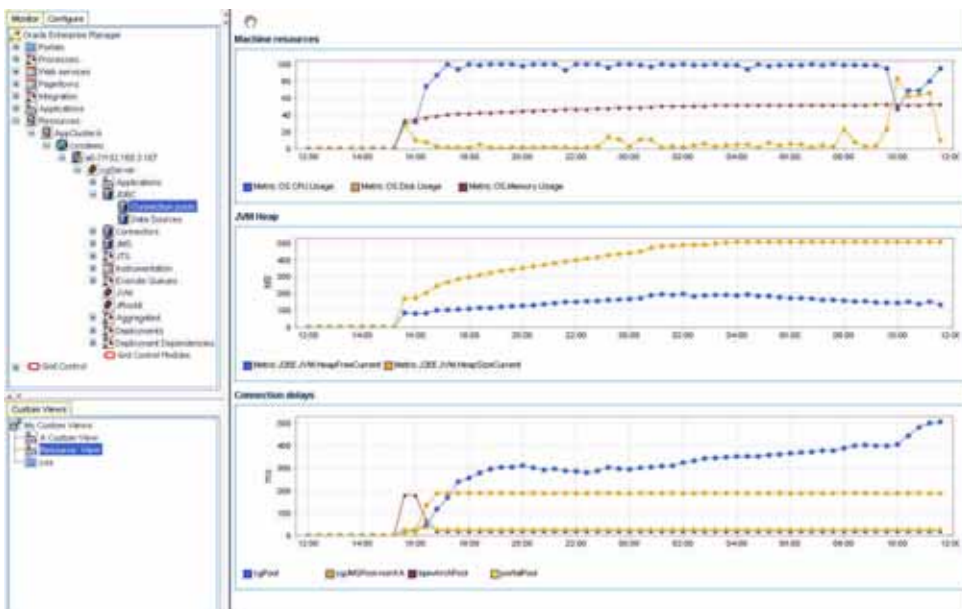


In [Figure 10-20](#), we don't see any specific portlet behave radically during the time period in question. In fact, when the desktop performance begins to degrade, the performance of all portlets also degrades. Based on this information, we can conclude that no specific portlet in the desktop was the source of this performance slowdown.

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.

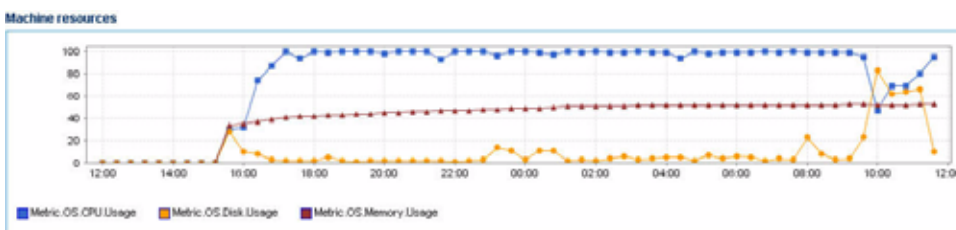
To perform this type of analysis, we would pull together different performance tables and graphs to create custom views in CAMM™. In the screen shot below, our custom view includes graphs from OS Agent, JVM, and JDBC. Using these graphs, we can analyze the correlation between machine resources, JVM resources, and Application Server managed resources (JDBC connection pool).

Figure 10-21: Use a Custom View to Correlate Different Metrics



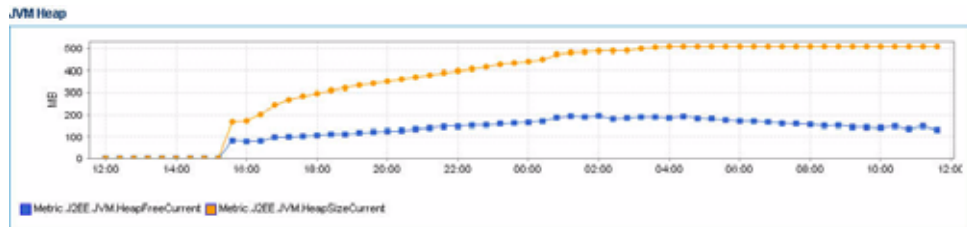
These graphs reveal some interesting patterns. We will analyze them individually. First, we will look at the graph from the OS Agent.

Figure 10-22: Metrics from OS Agent Reveals Abnormalities



In the graph above, we notice a sudden drop in CPU utilization and a sudden increase in disk utilization during the time period in question. This pattern indicates 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.

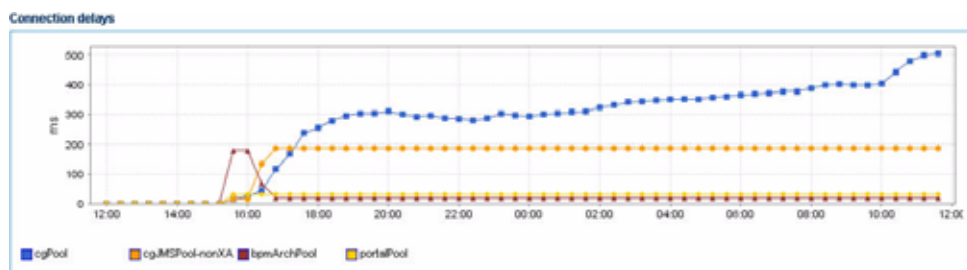
Figure 10-23: JVM Heap Metrics Reveals Abnormal Memory Usage



The graph above shows total JVM heap size and free 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.

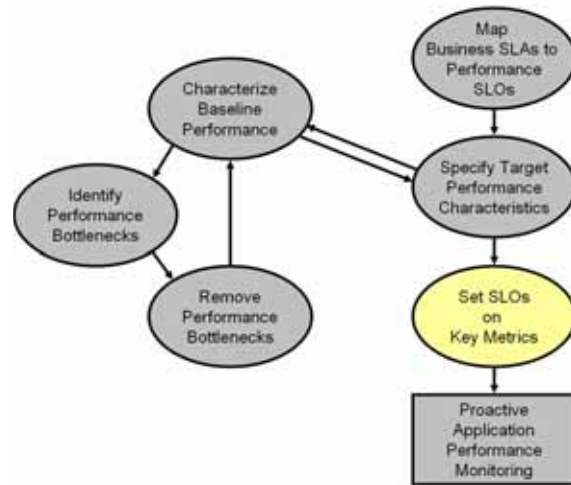
This abnormal consumption of memory caused the total JVM heap to reach its pre-defined maximum. It is also very likely this memory leak caused the increase in virtual memory paging activities and corresponding reduction in CPU utilization. This reduction in CPU utilization impacts the response time for all components running in this machine including JDBC Connections as indicated in [Figure 10-24](#).

Figure 10-24: Connection Response Time Impacted by Paging Activities



In the example above, 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.

Set SLOs on Key Metrics



This step in the Oracle™ Methodology allows us 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.

In our previous examples, we have demonstrated how to use CAMM™ to quickly understand performance characteristics, identify performance bottlenecks, and isolate sources of performance problems. In this

example, we will proactively set thresholds and actions for key WebLogic® system metrics. [Table 10-2](#) is the list of key system metrics for WebLogic® Platform:

Table 10-2: 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: <ol style="list-style-type: none"> 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.

Table 10-2: List of Key System Metrics for WebLogic® (Continued)

Key System Metric	Reason to Monitor
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 our example, we will configure SLOs and actions with CAMM™.

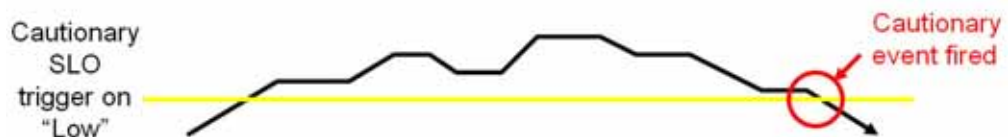
Our first task is to set a cautionary and a violation SLO for ExecuteQueue Idle Thread Count metric so the appropriate person can be alerted when available ExecuteQueue is running low. To configure SLOs, right-click on the Execute Queues metric and select Configure service level objects. In our example, we will create the following SLOs for ExecuteQueue Idle Threads:

Table 10-3: 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. See [Figure 10-25](#).

Figure 10-25: Mechanics of SLO Trigger Set to Low



For our example, we would create the following actions for the SLOs configured above:

Table 10-4: Actions for SLO

SLO Name	Action Name	Action Type
Low ExecuteQueue Idle Threads	Enter Low ExecuteQueue Idle Threads event into server log	Log

Table 10-4: Actions for SLO (Continued)

SLO Name	Action Name	Action Type
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 Oracle's recommendation:

Table 10-5: ExecuteQueue Pending Requests

SLO Name	Metric	Threshold Type	Threshold Value	Trigger on
ExecuteQueue Pending Request Warning	Metric.J2EE.Dispatcher.PendingRequests	Cautionary	5 ~ 10 ^a	High
ExecuteQueue Pending Request Violation	Metric.J2EE.Dispatcher.PendingRequests	Violation	10 ~ 20	High

^a. **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. See [Figure 10-26](#).

Figure 10-26: Mechanics of a SLO Trigger Set to High



Table 10-6: Total JVM Heap Size

SLO Name	Metric	Threshold Type	Threshold Value	Trigger on
JVM Heap Reached Max	Metric.J2EE.JVM.HeapSizeCurrent	Cautionary	512 MB ^a	High
JVM Heap Reached 0	Metric.J2EE.JVM.HeapSizeCurrent	Violation	0 MB	Low

^a**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 10-7: 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 10-8: 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 ^a	Low

^a In our 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 10-9: 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.

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. While the steps involved with the Oracle™ Methodology apply to all application performance management efforts, the examples offered leverage the unique capabilities of CAMM™. 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. Combining CAMM™ and Oracle™ Methodology is the best way to achieve a highly effective application performance monitoring environment.

