

IBM Enterprise Workload Manager V2.1

How to configure operating systems
and enable middleware for EWLM

Sample scenario for building a
real-world Domain Policy

Load balancing and partition
management exploitation



G. Michael Connolly
Samuel Alexander
Paola Bari
Andre Botura
Mark Ecker
Kyungmee Park
Cathrin Perzl
Hong Xia
Miho Yamazaki

Redbooks



International Technical Support Organization

IBM Enterprise Workload Manager V2.1

September 2006

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

Archived

Second Edition (September 2006)

This edition applies to Version 2 Release 1 of IBM Enterprise Workload Manager.

© Copyright International Business Machines Corporation 2004, 2006. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this redbook	xi
Become a published author	xii
Comments welcome	xiii
Chapter 1. IBM Enterprise Workload Manager overview	1
1.1 EWLM strategy	2
1.2 EWLM architectural view	3
1.2.1 Domain manager	5
1.2.2 Managed server	7
1.2.3 Control Center	8
1.2.4 Domain Policy	10
1.2.5 Instrumentation	11
1.3 Interaction with existing workload management	15
1.3.1 Workload management on zSeries	15
1.3.2 Workload management on pSeries	17
Chapter 2. Installing and configuring EWLM	21
2.1 Overview	22
2.2 Packaging	22
2.3 Before installing	22
2.4 Virtualization Engine common components installation	29
2.5 Installing and configuring EWLM domain manager	34
2.5.1 Installing EWLM domain manager code	34
2.5.2 Configuring the domain manager	37
2.6 Installing and configuring EWLM managed server code	42
2.6.1 Installing EWLM managed server code	42
2.6.2 Configuring EWLM managed server	46
2.7 Checking the installation	50
2.7.1 Post-installation steps for Windows	50
2.7.2 Post-installation steps for AIX, Linux, Solaris, HP-UX, and i5/OS	51
2.7.3 Managing Control Center users	54
2.8 Installing and configuring EWLM on z/OS	58
2.8.1 Preventive service planning	59
2.8.2 Installation considerations	59
2.8.3 Configuring domain manager	61
2.8.4 Configuring the managed server	67
2.9 Migrating from EWLM Release 1	72
2.9.1 Migrating from EWLM R1 code	72
2.9.2 Saving a Domain Policy on EWLM R1	72
2.9.3 Importing a Domain Policy	74
2.10 Uninstalling EWLM	75
Chapter 3. Using a firewall and securing EWLM	77
3.1 Firewalls	78
3.1.1 General firewall overview	78

3.1.2	EWLM firewall support	79
3.1.3	ITSO EWLM firewall configurations	87
3.2	Security	89
3.2.1	Browser to Control Center	90
3.2.2	Control Center to domain manager	95
3.2.3	Domain manager to managed servers	96
3.2.4	Security considerations for EWLM processes	101
3.2.5	Security considerations for instrumented applications	102
Chapter 4.	Enabling middleware for EWLM	103
4.1	Overview	104
4.2	Pre-instrumentation tasks	105
4.3	Enabling DB2 Universal Database for ARM	106
4.4	Enabling WebSphere Application Server for ARM	111
4.4.1	Enabling WebSphere Application Server v5.1.1 for ARM	111
4.4.2	Enabling WebSphere Application Server Version 6 for ARM	113
4.4.3	Enabling WebSphere Application Server Version 6 for z/OS for ARM	116
4.5	Enabling IBM HTTP Server for ARM	119
4.5.1	WebSphere HTTP plug-in	119
4.5.2	Independent HTTP plug-ins	122
4.6	Verifying application ARM enablement	124
Chapter 5.	EWLM concepts and setup	127
5.1	EWLM concepts	128
5.1.1	EWLM elements	132
5.2	Building the Domain Policy	139
5.2.1	Assessment	139
5.2.2	Classification	140
5.2.3	Building the Domain Policy	141
5.3	Administering EWLM through the Control Center	150
Chapter 6.	EWLM monitoring and reporting	159
6.1	Overview	160
6.2	First-level reports	160
6.2.1	Exceptions report	160
6.2.2	Service Classes report	161
6.2.3	Transaction Classes report	162
6.2.4	Process Classes report	162
6.2.5	Partition Classes report	163
6.2.6	Managed Servers report	163
6.2.7	Load Balancers report	164
6.2.8	Partitions report	164
6.3	Second-level reports	165
6.3.1	Process class details	165
6.3.2	Service class details	165
6.3.3	Transaction class details	166
6.3.4	Partition class details	168
6.3.5	Managed server details	168
6.3.6	Load Balancers details	170
6.3.7	Partitions details	172
6.3.8	Application and server topology reports	172
6.3.9	Real-time performance monitors	177
6.4	Performance statistics logging	182
6.4.1	Configuring performance statistics logging	182

6.5 Scenarios and examples	185
6.5.1 Capacity planning example	186
Chapter 7. EWLM Management	191
7.1 pSeries and IBM System p5 partition management	192
7.1.1 Terminology and general concept of POWER5 partitions	192
7.1.2 Partition workload group	194
7.1.3 Monitoring the performance of partition workload groups	197
7.2 EWLM and workload balancing overview	200
7.2.1 Configuring an EWLM load balancing setup	200
7.2.2 Interfaces	203
7.2.3 Load balancing algorithm	203
7.2.4 Application-level identification	205
7.2.5 Where to send the work (weight calculations)	206
7.2.6 SSL implementation	209
7.2.7 Monitoring the routing environment	209
Chapter 8. The ITSO EWLM environment	211
8.1 Running Trade6 in the ITSO environment	212
8.2 ITSO Trade6 base environment	212
8.2.1 Assessment of ITSO environment	212
8.2.2 Identifying the EWLM boundaries	213
8.2.3 Identifying the edge server	213
8.2.4 Identifying transactions	215
8.3 ITSO Domain Policy	216
8.3.1 Partition settings for the ITSO environment	217
8.4 Example of more complex Trade6 classification	218
8.4.1 Finding classification values for transactions	218
8.4.2 Defining transaction classes	219
8.4.3 Verification of transactional classification and goal settings	220
Chapter 9. Scenarios	223
9.1 Improve performance of most important workload	224
9.1.1 Objective and expectation	224
9.1.2 Starting point	224
9.1.3 Running the test case	225
9.1.4 Results and considerations	226
9.2 Improve performance of less important workload	227
9.2.1 Objective and expectation	227
9.2.2 Starting point	227
9.2.3 Running the test case	228
9.2.4 Results and considerations	229
9.3 Partition class for non-ARM instrumented application	230
9.3.1 Objective and expectation	230
9.3.2 Starting point	230
9.3.3 Running the test case	232
9.3.4 Results and considerations	235
9.4 Monitoring a multi-tiered workload with EWLM and z/OS	235
9.4.1 Objective and expectation	235
9.4.2 Starting point	237
9.4.3 Running the test case	239
9.4.4 EWLM Control Center reports	240
9.4.5 RMF postprocessor workload activity reports	243
9.4.6 DB2 PE accounting and statistics reports	245

9.4.7 Result and considerations	247
Chapter 10. EWLM traces and logs	249
10.1 EWLM Control Center	250
10.2 Commands	251
10.3 Virtualization Engine and EWLM logs	252
10.3.1 Virtualization Engine and EWLM domain manager installation log files	252
10.3.2 Configuration summary logs	254
10.3.3 Common command logs	255
10.3.4 Configuration Wizard logs	256
10.3.5 WebSphere Application Server logs	257
10.3.6 Error logs	257
10.3.7 Middleware components log files	257
10.4 EWLM traces and snapshots	257
10.4.1 Trace of algorithm	258
10.4.2 Trace and snapshot of domain manager and managed server	260
10.4.3 EWLM Control Center traces	264
10.5 Summary table of log files	265
10.6 AIX syslog facility	267
10.7 ARM Serviceability Adapter	268
Chapter 11. Performance considerations	271
11.1 Domain manager resource description	272
11.1.1 Sizing factors	272
11.2 Performance tests	273
11.2.1 Environment	273
11.2.2 Key measurement variables	273
11.2.3 Domain manager on z/OS	275
11.2.4 z/OS test methodology	277
11.2.5 General sizing recommendations on z/OS	279
11.2.6 Domain manager on Windows	282
11.2.7 Windows test methodology	283
11.2.8 General sizing recommendations on Windows	285
11.2.9 Domain manager on AIX	286
11.2.10 AIX test methodology	287
11.2.11 General sizing recommendations on AIX	290
11.2.12 Domain manager on i5/OS	291
11.2.13 i5/OS test methodology	292
11.2.14 General sizing recommendations on i5/OS	294
11.3 Additional considerations	295
11.4 Resource requirements for the EWLM managed server	295
11.4.1 Considerations for the z/OS managed server	297
11.4.2 Classifying the EWLM managed server to z/OS WLM	302
11.4.3 zAAP eligibility	302
Appendix A. EWLM domain policies	303
The default and sample policies	304
Appendix B. Additional material	307
Locating the Web material	307
Using the Web material	307
How to use the Web material	307
Related publications	309

IBM Redbooks	309
Other publications	309
Online resources	309
How to get IBM Redbooks	310
Help from IBM	310
Index	311

Archived

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ™
eServer™
iSeries™
i5/OS®
pSeries®
xSeries®
z/OS®
zSeries®
AIX 5L™
AIX®
CICS®
DB2 Universal Database™

DB2®
DRDA®
HACMP™
Informix®
IBM®
IMS™
Lotus®
Micro-Partitioning™
MVS™
OS/2®
OS/390®
Parallel Sysplex®

POWER™
POWER4™
POWER5™
Redbooks™
RACF®
RMF™
RS/6000®
System p5™
Tivoli®
Virtualization Engine™
WebSphere®
Workplace™

The following terms are trademarks of other companies:

EJB, Java, JDBC, JRE, JVM, Solaris, Sun, Sun Microsystems, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Excel, Internet Explorer, Microsoft, Windows Server, Windows, Win32, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

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

StoneGate is a trademark of Stonesoft, Inc.

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

Preface

This IBM® Redbook provides an introduction to the Enterprise Workload Manager (EWLM) Version 2 Release 1. In addition to describing the overall product concept and functionality, it presents a detailed discussion of the elements that comprise an EWLM solution.

Step-by-step instructions take you through the installation of EWLM code on multiple platforms, for both the domain manager and managed servers, and also show how to enable the instrumentation of the middleware for a 3-tier Web application. The features for administering EWLM are described, along with the monitoring, managing, and reporting capabilities.

Sample scenarios implemented in the ITSO environment are used to guide you through the process of classifying workload, and defining and deploying your own EWLM policy. These scenarios are then used to demonstrate the business goal-based partition management capabilities of EWLM.

Techniques for securing your EWLM domain as well as the load balancing capabilities of EWLM are described. Troubleshooting hints and tips are provided, along with some basic performance considerations to help you design the optimum EWLM solution.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

G. Michael Connolly is an IT consultant at the International Technical Support Organization, Poughkeepsie Center. He has more than 30 years of IBM software development experience in both distributed systems and the mainframe zSeries®. He holds a BA in Humanities from Villanova University. His areas of expertise include TCP/IP Communications, UNIX® System Services, WebSphere® for z/OS®, and most recently the IBM Virtualization Engine™ EWLM product.

Sam Alexander is a Software Engineer with the Workplace™, Portal, and Collaboration Software division of the IBM Software Group in Westford, Massachusetts. He has spent the last two years helping develop the search functionality in Lotus® Workplace and WebSphere Portal 6. In addition to software development, his background includes quality assurance and performance testing. He has a Master's degree in Computer Science from Boston University's Metropolitan College and a Bachelor's degree in Business Management from the University of North Carolina at Wilmington.

Paola Bari is an Advisory Programmer at the International Technical Support Organization, Poughkeepsie Center. She has 22 years of experience as a systems programmer in OS/390®, z/OS, and Parallel Sysplex®, including several years of experience in WebSphere MQ and WebSphere Application Server.

Andre Botura is a Senior IT Specialist in IBM Brazil. He is an IBM Certified Advanced Technical Expert in AIX® and holds a degree in Robotics Engineering from Universidade Paulista in Sao Paulo. He joined IBM in 1991 as a Customer Engineer right after finishing Technical High School. He works with AIX (since Version 3.2.4). His areas of expertise include AIX, SP, HACMP™, TSM, Shark, and POWER4™ and POWER5™ systems. He works in implementation and support for all AIX customers in Brazil.

Mark Ecker is an Advisory zSeries IT Specialist from the United States. During most of his six years with IBM, he has worked as a Pre-sales Technical Support Specialist for zSeries. His areas of expertise include zSeries hardware and software, capacity analysis, and solution design.

Kyungmee Park is a Consulting IT Specialist at the Technical Sales Support in IBM Korea. She has 14 years of experience as a Software Engineer, developing DOS, OS/2®, and Applications on distributed platforms, including several years of experience as a Technical Specialist in WebSphere Application Server. Her areas of expertise include Java™ 2 Enterprise Edition, DB2®, WebSphere, and Performance Benchmarks. She holds a Bachelors of Engineering degree in Computer Science from Sogang University, Korea.

Cathrin Perzl is an IT Specialist from IBM Munich, Germany. She works as a Technical Consultant in the Partnership Solution Center in Munich as a member of the Systems and Technology Group. Her areas of expertise include pre-sales consultation and implementation of pSeries®, AIX, and workload management tools to support IBM sales, IBM Business Partners, and clients. Cathrin is an IBM eServer Certified Specialist in AIX System Administration, AIX System Support, and p690 Technical Support. She holds a Master's degree in Mathematics and Physics from York University, UK.

Hong Xia is an Advisory IT Specialist with Technical Sales Support in IBM China. She has more than nine years of IBM Informix®, DB2, and AIX experience on distributed platforms. Her areas of expertise include relational database, UNIX systems management, and performance tuning, and a is technical consultant on pSeries. Hong is an IBM Certified Specialist in AIX System Administration and a DB2 Advanced Database Administrator.

Miho Yamazaki is an IT Specialist of Technical Sales Support in IBM Japan. Miho has eight years of experience in AIX, HACMP, RS/6000®, and IBM pSeries, including several years of experience in DB2 and WebSphere MQ. Miho provides pSeries pre-sales technical consultation to IBM sales and IBM Business Partners in Techline. She holds a Bachelor's degree in chemistry from Keio University in Japan.

Thanks to the following people for their contributions to this project:

Richard Conway, Octavian Lascu, Dino Quintero
International Technical Support Organization, Poughkeepsie Center

Nicholas Amon, Patrick Chan, CheKim Chhuor, Roan Dawkins, Yuksel Gunal, David Manners, Kin Ng, Jayesh Patel, Hiren Shah, Helen Tsang, Peter Yocom
IBM Poughkeepsie

Alan Bivens, Pavitra Ghanta, Mathew S. Thoennes
IBM Watson Lab

Tom Schuler
IBM Rochester

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:

ibm.com/redbooks

- ▶ Send your comments in an Internet note to:

redbook@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Archived



IBM Enterprise Workload Manager overview

This chapter provides an introduction to the IBM Enterprise Workload Manager (EWLM). It explains the fundamental concepts on which EWLM is based and describes the functionality of the elements that build this solution.

1.1 EWLM strategy

Enterprise Workload Manager is a product of the IBM Virtualization Engine solution that can dynamically monitor and manage distributed heterogeneous workloads to achieve user-defined business goals.

Today's e-business environments are very complex. Workflow is dynamic and the infrastructures the workflows run in have become increasingly difficult to monitor and manage. Before we can optimize an environment like this, we need to get an understanding of what to optimize. Monitoring of business transactions today can include resources spanning multiple hardware platforms, operating systems, networks, and applications. Performance tuning and workload management is happening vertically while most business transactions in a three-tier architecture (Web, application, data) encompass a horizontal workflow, as shown in Figure 1-1.

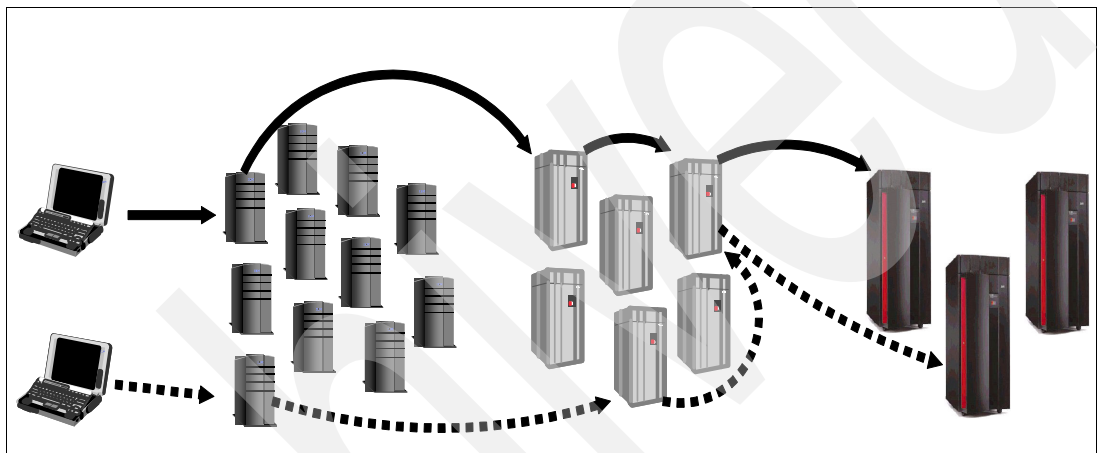


Figure 1-1 e-business environment

The server configuration shown in Figure 1-1, which is very common, is a multi-tiered environment supporting multiple logical business applications. These servers could be physically located in one data center or could be spread across sites in different cities or countries. In many cases, elements of the server and application infrastructure are shared by multiple business applications, or by application components serving different business purposes. Simple views of the configuration become quite complex when, for example, your CRM application suddenly becomes Web-enabled and then integrates with your product inventory application, which is then integrated with your billing system.

Currently, vertical *management silos* are common, where one group controls Windows® servers, another handles UNIX servers, and a third may manage the mainframes. For some large middleware applications such as WebSphere Application Server, an application group manages the application servers and yet another group manages the database system. With this management paradigm, a number of problems frequently occur that are difficult and time consuming to resolve. For example, simple performance problems at one tier can materialize as major resource contention issues on some other tiers. Sorting out the cause from the effect can take hours, days, or sometimes weeks.

While you are investigating and repairing the problem, how does the problem really affect your business? Multiple business applications and processes can be hurt by a given problem, to varying degrees. While you have ten or more people in a room pointing fingers and debating the possibility that the database is the real source of slow response times, your business could be losing clients. You need to find and fix problems faster, and better yet, you need to avoid problems.

EWLM is one element of the transformation of your IT infrastructure into an on demand business. It is an expansion of the concepts introduced by the z/OS Workload Manager over ten years ago, applying proven and mature technology to the multi-tier, heterogeneous environment. First and foremost, it helps your IT infrastructure “think” like you do, like a business. Starting from service-level objectives established for your business applications, it learns the relationships between the servers, the application middleware instances, and your business processes. Reporting information from a business perspective lets you rapidly understand when goals are not being achieved, where the bottlenecks are, and what the business impact is.

Detailed internal performance statistics are maintained by EWLM, explaining where time was spent and why, so you can quickly zoom into the area where investigation is required. It will also help you avoid investigating non-problem situations. The statistics maintained by EWLM form the basis for the introduction of goal-oriented autonomic management techniques, like those from the z/OS platform. So, while you are looking at a potential problem, the environment is adapting itself to mitigate, or even eradicate, the problem.

As is the case with the z/OS Workload Manager, the EWLM product provides management capabilities to monitor workloads and dynamically reallocate available resources to meet business objectives. Adaptive heuristic management is a necessary element of being *on demand*: Avoiding problems, repairing problems, and maximizing server CPU utilization.

One of the strengths of the zSeries platform and the z/OS operating system is the ability to run multiple workloads at the same time within one z/OS image or across multiple images. The function that makes this possible is dynamic workload management, which is implemented in the Workload Manager component of the z/OS operating system.

The idea of z/OS Workload Manager is to make a contract between the installation (end user) and the operating system. The installation classifies the work running on the z/OS operating system in distinct service classes and defines goals for them that express the expectation of how the work should perform. WLM uses these goal definitions to manage the work across all systems of a sysplex environment.

1.2 EWLM architectural view

To get started with EWLM there are some simple concepts that we need to introduce. EWLM is an implementation of policy-based performance management. The scope of management is a set of servers that you logically group into what is called an EWLM *Management Domain*. The set of servers included in the Management Domain has some type of relationship, for example, the set of servers supporting a particular line of business. The line of business may consist of multiple business processes spread across a few servers or a thousand servers.

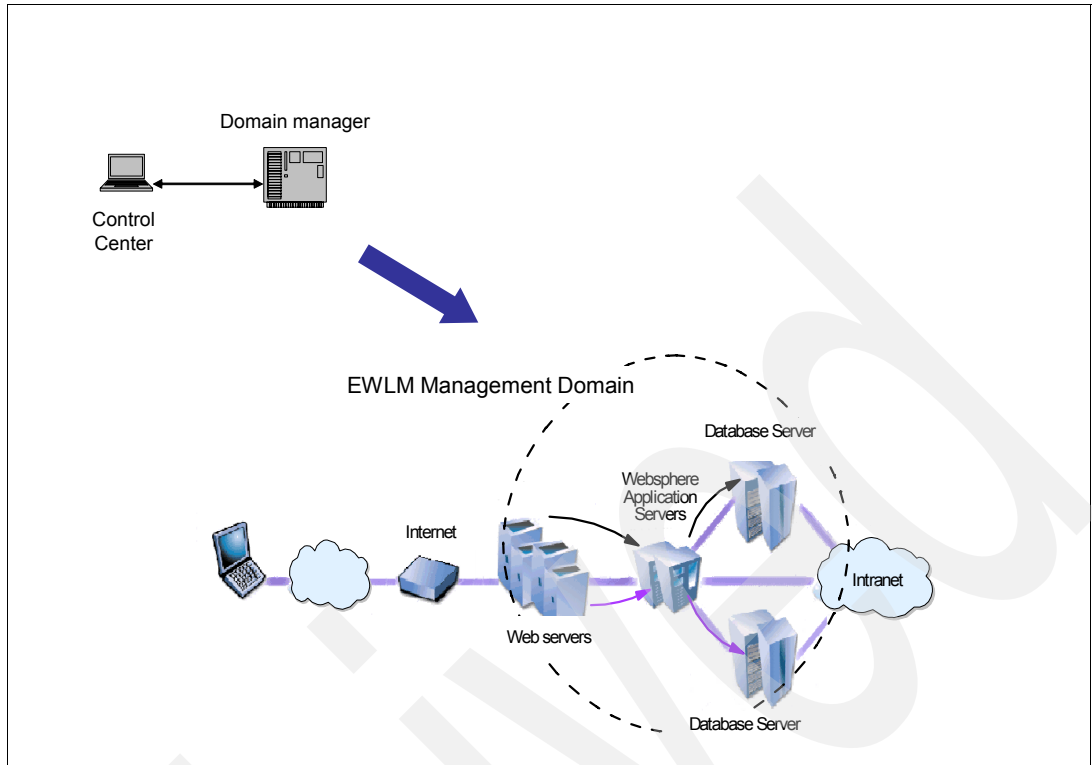


Figure 1-2 EWLM architecture

There is a management focal point for each EWLM Management Domain, called the EWLM *domain manager*. As shown in Figure 1-2, there is a one-to-one relationship: One domain manager instance per Management Domain. The domain manager coordinates policy actions across the servers, tracks the state of those servers, and accumulates performance statistics on behalf of the domain.

On each server (operating system) instance in the Management Domain there is a thin layer of EWLM logic installed called the EWLM *managed server*. From one perspective the managed server layer is positioned between applications and the operating system. The managed server layer understands each of the supported operating systems, gathering resource usage and delay statistics known to the operating system.

A second role of the managed server layer is to gather relevant transaction-related statistics from middleware applications. The application middleware implementations, such as WebSphere Application Server, understand when a piece of work starts and stops, and the middleware understands when a piece of work has been routed to another server for processing, for example, when a Web server routes a servlet request to a WebSphere Application Server.

The managed server layer dynamically constructs a server-level view describing relationships between transaction segments known by the applications with resource consumption data known by the operating system. A summary of this information is periodically sent to the domain manager, where the information is gathered together from all the servers in the Management Domain to form a global view.

An important aspect of the EWLM approach is that all data collection and aggregation activities are driven by a common service level policy, called the EWLM *Domain Policy*. This policy is built by an Administrator to describe the various business processes that the domain supports and the performance objectives for each process. For example, a book store would

probably have a business transaction called *Add to shopping cart* that would have a performance goal of perhaps a 1.5-second response time.

To permit the EWLM domain manager to construct an end-to-end view of each type of business transaction, the processing segments performed by each participating middleware instance must be correlated, meaning pieced together. For example, the *Add to shopping cart* transaction would be received by a Web server. It could flow to a WebSphere Application Server instance that might update a database. The transaction might even check your credit limit by routing a sub-transaction to another server. The tricky part of this correlation is to get all of the applications in the path of each transaction to cooperate, without knowing that they are cooperating.

The final aspect of EWLM to introduce is the EWLM *Control Center*, a browser-based application server tailored to the needs of an EWLM administrator, analyst, and operator. This is where all the EWLM concepts come together — where you can create a service level *Domain Policy* and activate that policy on hundreds of servers with a single click. Reporting data is then available to let you view performance from a business perspective. And if you need the details, you can see the topology of servers and applications supporting each transaction type and understand where the time was spent.

In addition to the monitoring capability, EWLM provides two management capabilities: *Partition management*, where EWLM can dynamically manage CPU resources across multiple partitions within the same physical POWER5 machine; and *load balancing*, where EWLM can monitor and provide recommendations to the Load Balancer on how to distribute work to achieve optimal performance.

The information is organized in such a way that an administrator or analyst can easily drill down to what is relevant.

1.2.1 Domain manager

The domain manager is a software stack that you install and configure for your EWLM environment. It consists of two operating system processes: One supports the Control Center and one coordinates policy across the managed servers and aggregates performance statistics. The capacity of the server that you install the domain manager on is dependent on the number of managed servers and the Domain Policy complexity. See Chapter 11, “Performance considerations” on page 271, for more details.

The domain manager has a complete view of what is going on in the domain. It keeps 24 hours of historical data collected from the Managed Servers. The data can be displayed for an interval of as little as one minute up to an interval of one hour.

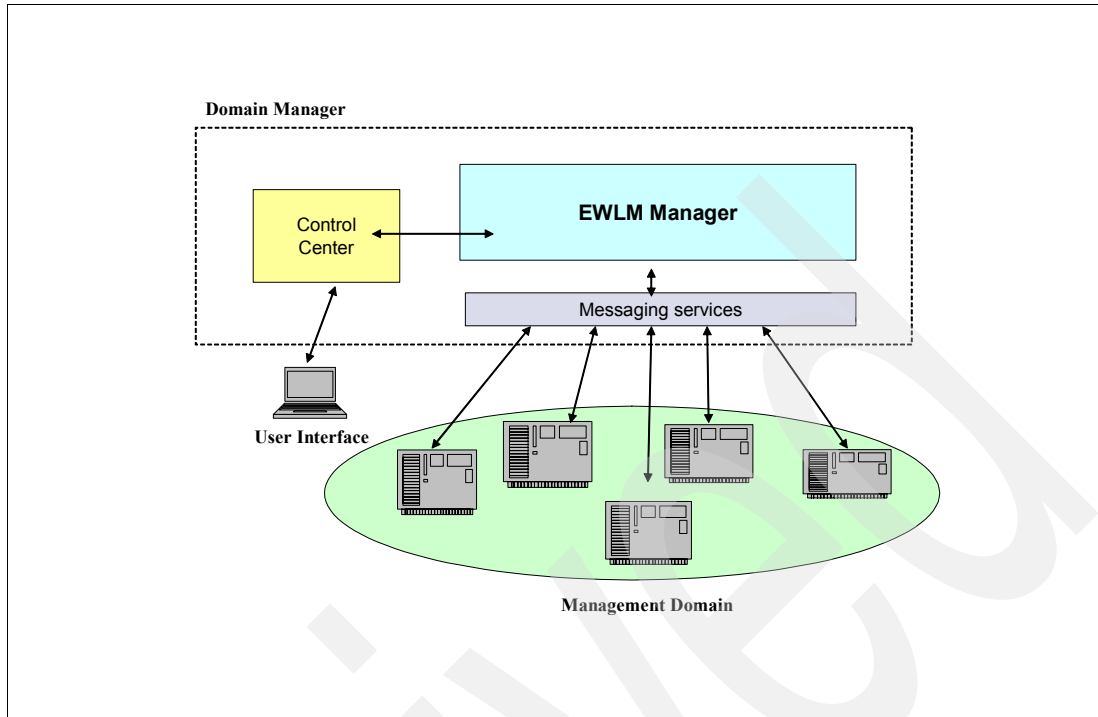


Figure 1-3 Management Domain

The Managed Servers and Domain Manager communicate via a lightweight publish and subscribe messaging protocol. The domain manager does not need to rely on every piece of data being delivered from the Managed Servers; therefore, persistence of the messaging communication is not required. This allows for greater flexibility and scalability in the EWLM architecture. For environment consistency across multiple demilitarized zones (DMZ), this communication can be secured using SSL. Security and firewalls are discussed in detail in Chapter 3, “Using a firewall and securing EWLM” on page 77.

The Domain Manager ensures global coordination of policy actions initiated from the Control Center. When a policy is activated, the Domain Manager sends it to all the Managed Servers in the management domain and immediately begins collecting data from the Managed Servers in the context of the new policy. The Domain Manager aggregates the data that it collects and then stores it in an internal database for historical purposes. This data can be viewed in the Control Center to monitor and manage how well the business goals are being met and to understand the topology of transactions, applications, and servers in the management domain.

Tip: The domain manager provides coordination and activation of policies across the EWLM Management Domain. When activating across hundreds of servers, the policy will be committed if 70 percent are successful. Otherwise, the change will be backed out.

The Control Center process in Figure 1-3 is a WebSphere Application Server instance. The domain manager communicates with the Control Center to get the data that is viewed and updated.

The Domain Manager in the current release of EWLM does not have backup or failover capabilities. If it fails, it must be restarted. As noted previously, the domain manager does not persist messages to and from the Managed Servers. The Managed Servers initiate communication with the domain manager using the listening port and IP address or host

name assigned to the domain manager at configuration. Once a managed server is brought into the domain (for example, the managed server has connected to a specific domain manager), the domain manager keeps the topology data about the managed server for seven days. If the managed server stops communicating to the domain manager, the managed server is removed from the topology after this period of time. We recommend that you use a host name that is part of a Domain Name Server (DNS) rather than using a numeric IP address.

1.2.2 Managed server

The Managed Server component provides the EWLM platform support needed to collect and aggregate data and to share the data with the Domain Manager. The Managed Server is a common component that is installed on each operating system instance. Once installed and started, the Managed Server immediately makes contact with the Domain Manager to indicate it is part of the management domain, using a configured address and port. Communication is initiated from the Managed Server to the Domain Manager most of the time. This connection remains intact until broken. It can be secured with SSL and supports router, proxy, and stateful inspection firewalls.

The operating system extensions provide the ARM implementation interface for ARM instrumented middleware and an interface to collect process and system utilization statistics. These operating system extensions are integrated into each of the supported IBM operating systems; supported non-IBM operating system extensions are shipped with the EWLM product.

Overall, there are three types of data that the Managed Server assimilates, correlates, and summarizes: *Transaction data* flowing through ARM instrumented middleware, *process and partition data* for non-ARM instrumented applications like operating system processes and batch jobs, and *system resource data*.

In order for the Managed Server to work with this data in the context of a Domain Policy, the policy information must be available to the Managed Server. The Domain Manager deploys the policy information to all the Managed Servers. This is initiated when an EWLM administrator deploys a policy from the EWLM Control Center. Each Managed Server receives and stores a copy of this policy information in an internal database.

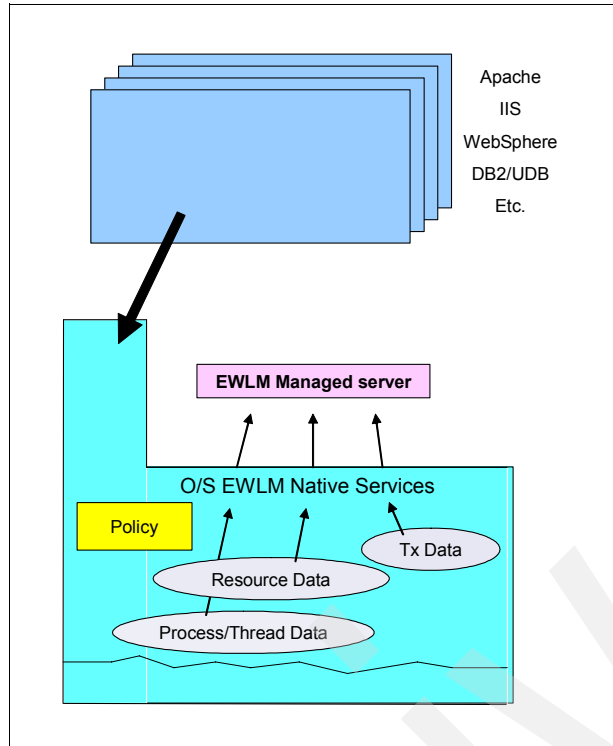


Figure 1-4 EWLM managed server policy and data

Figure 1-4 provides a detailed view of how the instrumented application, operating system and extensions, and Managed Server interact to assimilate, correlate, and aggregate data using the policy, transaction data, resource data, and process data. It is at the Managed Server where the data is captured and classified in the context of a Domain Policy.

The IBM Virtualization Engine concept of system services, system technologies, and operating systems all working together is clearly demonstrated in this component architecture.

1.2.3 Control Center

The EWLM Control Center (from this point forward called the Control Center) is the EWLM Web-enabled user interface that supports three categories of tasks:

- ▶ Administration of EWLM policies
- ▶ Operational actions against the EWLM Management Domain
- ▶ Display of performance statistics

The Control Center only communicates with the Domain Manager. It has no direct interaction with the Managed Servers in the domain. It requires a user to log in and uses role-based authentication. User IDs must be created at the operating system level first and then the user ID can be added to the Control Center using configuration commands that identify the role of Administrator, Monitor, or Operator, as described later in “Adding and deleting Control Center users” on page 55.

The Control Center runs on the WebSphere Application Server instance that is installed with the EWLM Domain Manager. It runs as a separate process from the Domain Manager, but it must run on the same server as the Domain Manager and its scope is limited to the one Management Domain. The flow of data between the browser and the Control Server can be

SSL encrypted. SSL encryption is the highly recommended way of operating. Refer to 3.2.1, “Browser to Control Center” on page 90 for further details.

Administration activities carried out through the Control Center entail creation and maintenance of EWLM domain policies, which include service classes, transaction classes, and process classes. Performance goals can be set at the service policy level for each service class.

The Control Center supports two modes of operation — one is *offline*, where you develop and edit a Domain Policy for future deployment; the other is editing a copy of a *live* Domain Policy that has already been deployed to all Managed Servers in the domain. Editing a copy of a live Domain Policy allows you to make adjustments to the existing policy and deploy to all Managed Servers as needed.

Note: Administrators have access to monitoring, operational, and administrative activities. Operators have access to monitoring and operational activities. Monitors only have view capability to monitoring activities.

Operational activities include controlling and monitoring the current operational state of the management domain. The operator can manage the service policy that is in effect for the domain. This includes:

- ▶ Displaying all service policies available for the current deployed Domain Policy
- ▶ Querying the properties of the service policies for this Domain Policy
- ▶ Activating any of the service policies for the current deployed Domain Policy

The operator also has general operational capabilities, which include:

- ▶ Displaying each server or operating instance in the Management Domain
- ▶ Displaying the overall state of each server or operating instance
- ▶ Initiating actions to disable and enable an individual managed server if problems occur

An important capability the Control Center provides is monitoring and reporting. Data is provided in real time and can be shown in 1-minute to 60-minute intervals for the active service policy. Views range from high-level business-driven reporting to response time and resource data.

Note: Data is provided in real time every 10 seconds from all the Managed Servers in the domain to the Domain Manager. The domain manager continuously merges received data into the “live” view of the Management Domain.

High-level reporting is centered around the service classes and how well the Management Domain is doing in achieving its business goals. The service class view identifies the goal versus actual and a calculated performance index. Analysts with the Monitor role can drill down from here into the underlying statistics to help determine where, why, and how these results were achieved. The following monitors are available for drill down in the EWLM Control Center:

- ▶ Performance Index Monitor
- ▶ Goal Achievement Monitor
- ▶ Transaction Rate Monitor
- ▶ Processor Usage Monitor

In addition to monitor drill down, you can also drill down from the service class, transaction class, or process class to get a server or application topology view. The application topology view provides a diagram of applications that processed transactions for a particular class.

The server topology view provides a diagram of the Managed Servers that processed transactions for the particular class. These views provide a transaction workflow in a heterogeneous environment.

Detailed reporting provides performance data for Domain Policy components, for example, workloads, service classes, transaction classes, process classes, or managed servers. The following detailed reports are available in the EWLM Control Center:

- ▶ Managed server details
- ▶ Service class details
- ▶ Process class details
- ▶ Transaction class details
- ▶ Partition class details
- ▶ Partition workload group details
- ▶ Load Balancer details
- ▶ Hop details

1.2.4 Domain Policy

A Domain Policy is a collection of service policies, applications, transaction classes, service classes, and optionally platforms and process classes for an EWLM Management Domain. As you can see in Figure 1-5, a Domain Policy can have an unlimited number of service policies (in the sample you have weekend and week day), but only one service policy can be active at a given time in the domain. You can have multiple service policies that have performance goals that are specific to a certain time of day or week and these policies can be activated as needed. Only one Domain Policy is needed for a domain because it contains all the service policies for the domain.

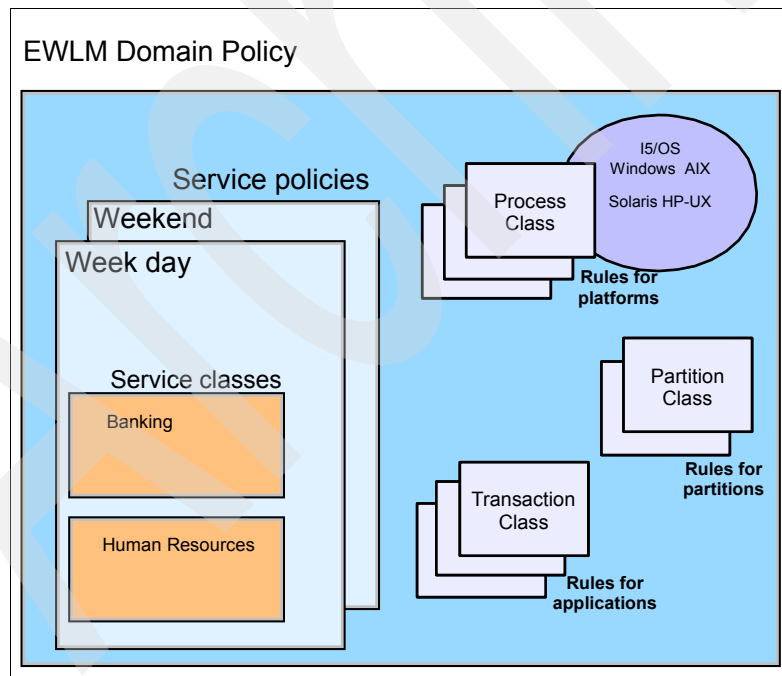


Figure 1-5 EWLM Domain Policy elements

Within the service policy, service classes are defined that specify the performance goals you want to achieve for your transactions and your processes. The service policy also contains transaction classes and process classes that are used to classify your workload and to associate it to service classes and consequently to performance goals. You can customize

service class goals and categorize them in different service policies to achieve different performance objectives. This would make sense when you have different performance objectives for the weekend and week days.

Creating a Domain Policy for your domain requires that you have a certain degree of understanding of the business workflow in your environment. Depending on how many Management Domains are created for your business, it is possible that one Administrator could be responsible for the Domain Policy of the *Banking* domain and another Administrator be responsible for the *Human Resources* domain, each being administered through a separate Domain Manager and Control Center. If all of your business workflow is within one domain, then the Administrator will need to have knowledge of all that business that you want to monitor and manage. This will also require an understanding of all the applications and server instances that participate in the business workflow.

Most organizations have service level agreements (SLAs) in place today that are used as a contract between IT and the business to ensure a certain level of performance for the business workflow in the environment. These SLAs will need to be analyzed in order to convert them to the performance goal terms used in an EWLM Domain Policy. The result can be used to build a Domain Policy for the EWLM Management Domain. If there are no SLAs present in your installation, you can use the EWLM monitoring capability to understand your workload and build a starting model.

You can find more details on the Domain Policy elements and on how to build a Domain Policy for your installation in 5.2.3, “Building the Domain Policy” on page 141.

1.2.5 Instrumentation

Instrumentation provides the data to create the topology of a transaction, that is, what server instances or application instances a transaction flows through. In a typical Web transaction (for example, buy books) the transaction starts at a Web server (edge server), flows to an application server, then to a database server. Without instrumentation, the transaction would look like three separate processes with three separate response times rather than one transaction. This is key to EWLM, providing end-to-end topology views and statistics for business transactions.

In order for EWLM to report on these transactions and potentially manage the domain based on these statistics, Application Response Measurement (ARM) 4.0 has been implemented for application instrumentation. ARM is an Open Group standard composed of a set of APIs that can be used to collect response time information for enterprise applications. At the time of writing there are three types of applications instrumented with ARM 4.0: WebSphere Application Server, DB2 UDB, and Web servers. For an updated list, refer to

<http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp?topic=/veicinfo/>

Let us look at how ARM, EWLM, and instrumented applications work together to provide end-to-end response times and application/transaction topology using correlation.

To monitor work requests in a multi-tiered heterogeneous environment, you should be able to identify and track the performance of those requests across server boundaries. It is possible to collect this data using versions of middleware that have been instrumented with the ARM V4.0 standard. These are a set of interfaces that an application calls and are then used by EWLM to calculate response time and status of work processed by the application.

Figure 1-6 shows a general flow of core ARM function calls used by an application that EWLM supports for capturing transaction data.

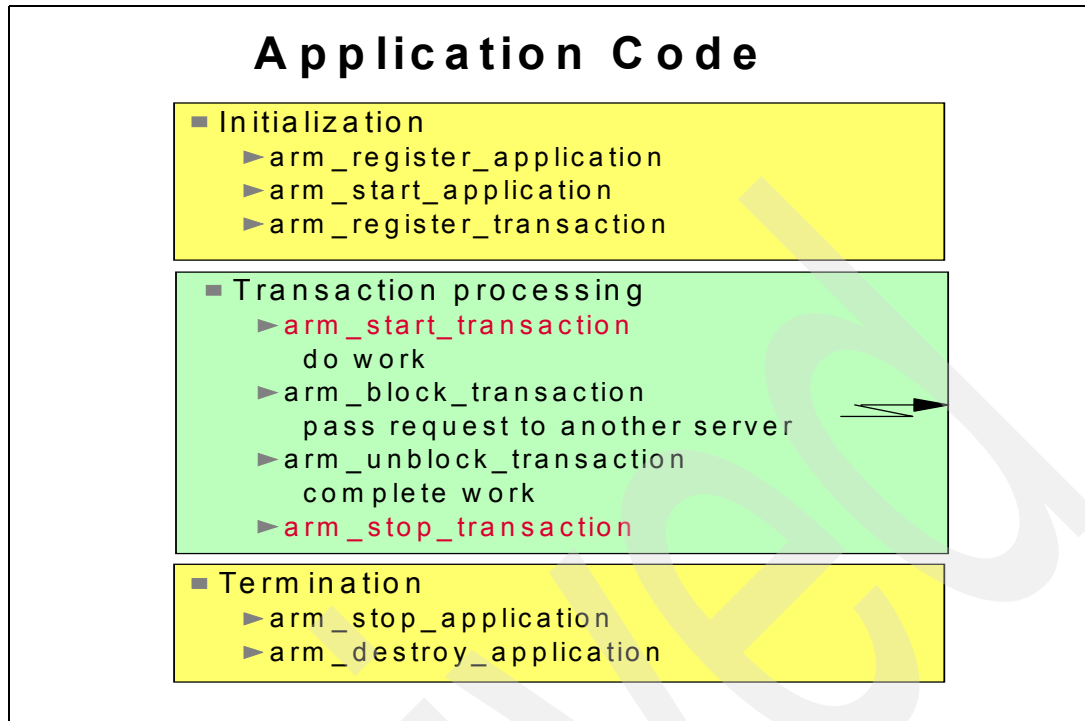


Figure 1-6 API flow

This is a quick look at the function calls:

- ▶ An application issues `arm_register_application` to become registered with EWLM by its application name and `arm_start_application` to indicate that the application is active. Transaction registration via `arm_register_transaction` is similar to application registration, and each transaction is associated with an application.
- ▶ `arm_start_transaction` indicates that a transaction is starting on this application or middleware and EWLM starts response time timing.
- ▶ `arm_block_transaction` indicates that the transaction instance is blocked because it is waiting for some other service to complete (for example, WebSphere waits for a DB2 call to complete).
- ▶ `arm_unblock_transaction` indicates that the transaction is no longer blocked. Therefore, the time between `arm_block_transaction` and `arm_unblock_transaction` can be distinguished from the elapsed time between `arm_start_transaction` and `arm_stop_transaction`.
- ▶ `arm_stop_transaction` signals the end of a transaction; EWLM stops timing the response time.
- ▶ `arm_stop_application` indicates that the application instance is ending. Therefore, no more transactions can be started or stopped. `arm_destroy_application` removes the registration.

Note: A transaction in an EWLM environment is encapsulated between `arm_start_transaction` and `arm_stop_transaction` function calls issued by an application or middleware.

Now let us look at a sample scenario. Figure 1-7 has three EWLM managed servers and a domain manager in an EWLM domain. When a user request is initiated, the transaction flows

through the Apache, WebSphere Application Server, and DB2 middleware. The ARM instrumentation in each application layer is shown as a `arm_start_tran()` and `arm_stop_tran()`.

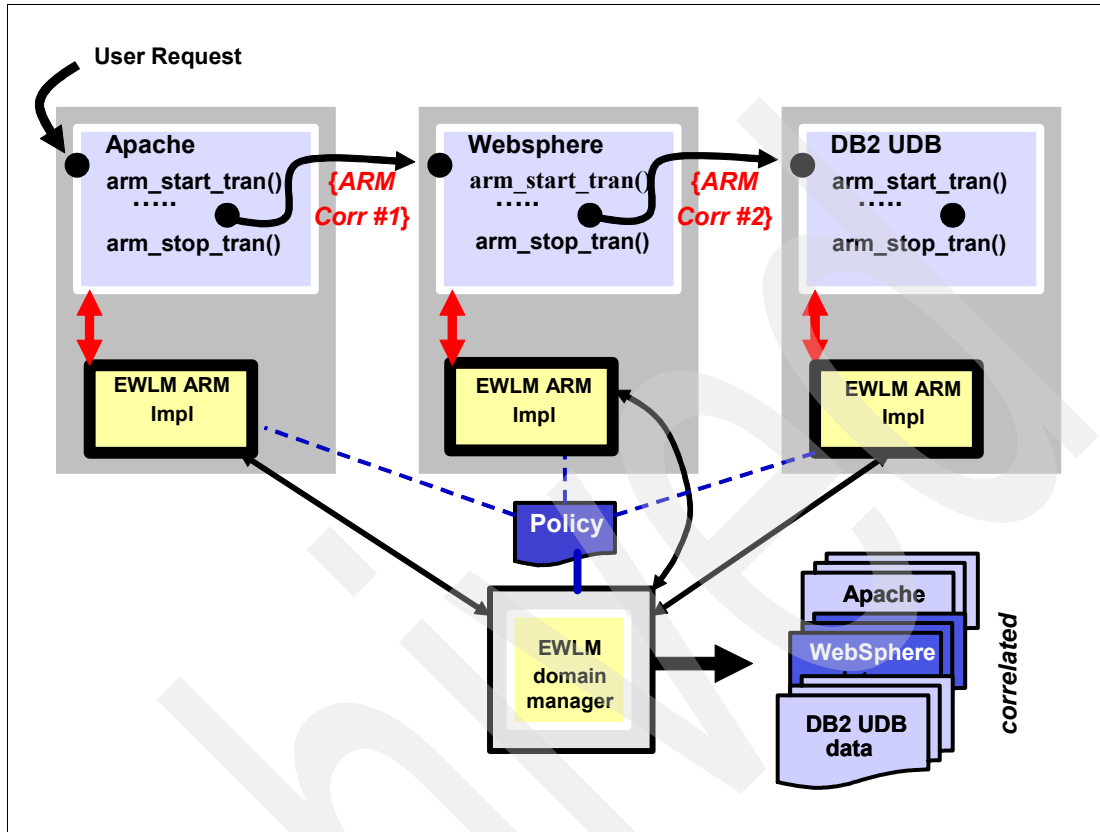


Figure 1-7 Transaction correlation using EWL, ARM, and middleware

The EWL ARM implementation interacts with the EWL managed server logic (not depicted separately in this figure) to capture and send in-progress and completed transaction data to the EWL domain manager.

In this particular scenario in Figure 1-7, work arrives at the Apache server with no input correlator, formally referred to as a parent ARM correlator. When `arm_start_tran()` is called from the Apache server, the EWL ARM implementation notices that there is no parent ARM correlator, and performs classification to determine which transaction and service classes to associate with the transaction instance. The EWL ARM implementation also constructs and returns an `arm_start_tran()` output argument child correlator, formally referred to as a current ARM correlator, which reflects two key pieces of information:

- ▶ Description of the Apache server as the first middleware hop in this transaction instance processing
- ▶ The `arm_start_tran()` processing transaction classification result

It is important to note that the classification action that occurs for the Apache server `arm_start_tran()` call happens because no parent ARM correlator was specified as an input argument. A transaction instance being processed by a middleware hop is considered to be a transaction edge whenever no parent ARM correlator is available, and therefore the transaction is classified by EWL ARM processing.

Note: A *hop* is when a work request flows from one application to another. Hop 0 is the place where an application is first assigned a correlator for the work request. Hop 0 occurs at what is referred to as the *transaction edge*.

The Apache server inserts the current ARM correlator received from its `arm_start_tran()` call into the HTTP request header, and eventually transfers control to the WebSphere Application Server to proceed with transaction instance processing. When the WebSphere Application Server is prepared to process this transaction instance, it extracts the Apache server's current ARM correlator from its input HTTP header and specifies that as the parent ARM correlator input argument on its `arm_start_tran()` call. This time, the EWLM ARM implementation ignores filters specified on the `arm_start_tran()` call because of the parent ARM correlator, and instead constructs an output argument current ARM correlator reflecting the following:

- ▶ Description of the WebSphere Application Server as the second middleware hop in this transaction instance processing
- ▶ The same transaction classification result that was reflected in the parent ARM correlator, which is the result achieved by the Apache server

After the `arm_start_tran()` call, the WebSphere Application Server determines that the transaction instance requires a database query, and performs a JDBC™ call to the DB2 Universal Database server to accomplish this. The WebSphere Application Server current ARM correlator is included in the JDBC control transfer protocol, and when the DB2 server is prepared to process the query it extracts this correlator from JDBC input data and specifies that as the parent ARM correlator input argument on the DB2 `arm_start_tran()` call. Again, the EWLM ARM implementation ignores filters specified on the `arm_start_tran()` call because of the parent ARM correlator, and instead constructs an output argument current ARM correlator reflecting the following:

- ▶ Description of the DB2 server as the third middleware hop in this transaction instance processing
- ▶ The same transaction classification result that was reflected in the parent ARM correlator, which is still the result achieved by the Apache server

If the DB2 server were to call yet another server hop to contribute to this transaction instance's processing, the DB2 server's current ARM correlator would become the parent ARM correlator for that additional hop. In our scenario that is not the case, since the DB2 server represents the deepest hop.

When query processing has completed, the DB2 server calls `arm_stop_tran()` and returns control to the WebSphere Application Server. Eventually, WebSphere Application Server completes its portion of the transaction instance processing, calls `arm_stop_tran()`, and returns control to the Apache server. In turn, the Apache server wraps up its processing and calls `arm_stop_tran()`. Transaction (and sub-transaction) instance response times are computed in `arm_stop_tran()` processing for each hop, and provided to EWLM managed server logic along with transaction instance application and server topology information, all of which is aggregated with other transaction instance data and sent to the EWLM domain manager every 10 seconds. It is important to understand that each transaction is not sent to the domain manager. The managed server EWLM code aggregates the data before sending any to the domain manager. This allows for greater scalability of the EWLM Management Domain.

The domain manager collects the data from the managed servers and aggregates that data across the domain. This is used to report on how well each service class is achieving its goal

and can be provided to Load Balancers to manage workload at the edge servers. Eventually, as EWLM evolves, this data will also be used to manage the entire domain.

1.3 Interaction with existing workload management

The following sections summarize the relationships between EWLM and the existing workload management capabilities available on the zseries and pSeries platform, how they interact, and how and when you can use these functions.

1.3.1 Workload management on zSeries

As shown in Figure 1-8, EWLM and z/OS WLM policy structures are conceptually the same. The main difference is related to the work managers and to the logical components that receive and submit the work requests. As work managers, EWLM supports applications and z/OS WLM platforms, and zWLM supports several subsystems, including CB, CICS®, DDF, and IMS™.

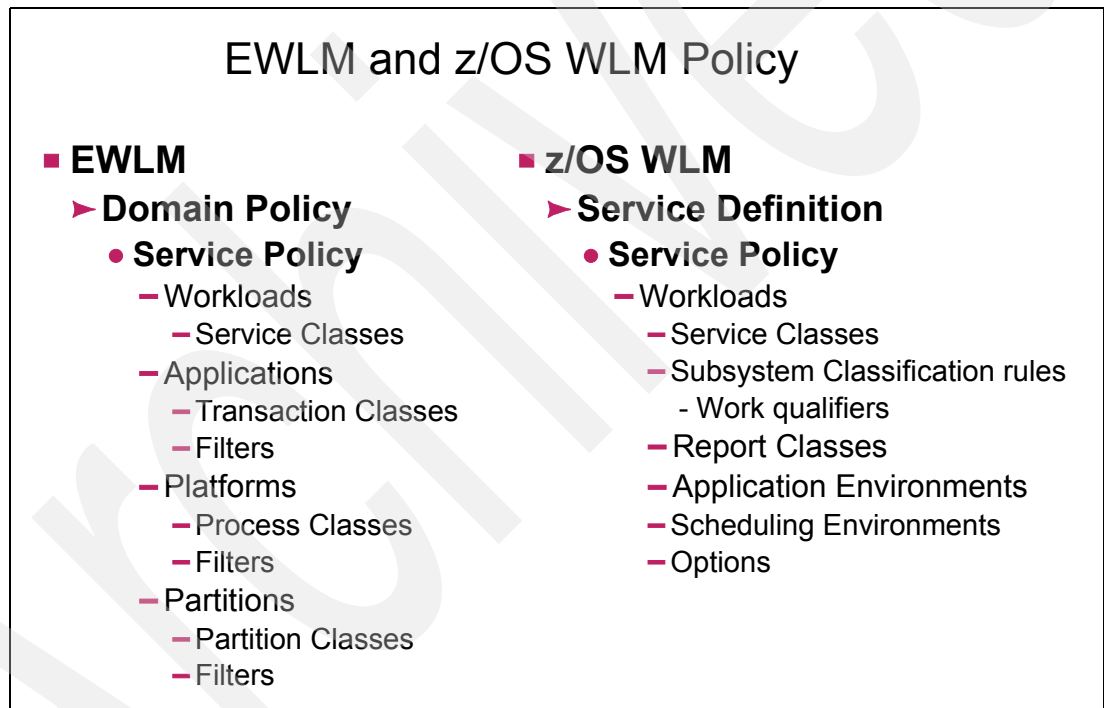


Figure 1-8 EWLM and z/OS WLM comparison

To associate a work request to a service class, a work manager passes some attribute values that EWLM matches against filters and z/OS WLM matches against subsystem work qualifiers. This process is called *classification*.

The attribute values for work requests received by the entry application of an EWLM Management Domain might be different from those passed to z/OS WLM, especially if the classification on the distributed platform happened in an application environment different from the entry point on z/OS. For this reason it may not always be possible to have a one-to-one mapping for EWLM service classes and z/OS WLM service classes.

ARM enablement on z/OS

Workload Manager on z/OS has provided workload management services to middleware such as DB2, WebSphere, CICS, and IMS for a long time. For example, DB2 DIST uses WLM enclave services to schedule and process received work requests. In the current implementation, EWLM on z/OS leverages the existing WLM services and maps them to ARM calls, so that an EWLM managed server can simply pick up the performance data and send it to the domain manager. Figure 1-9 depicts a general flow.

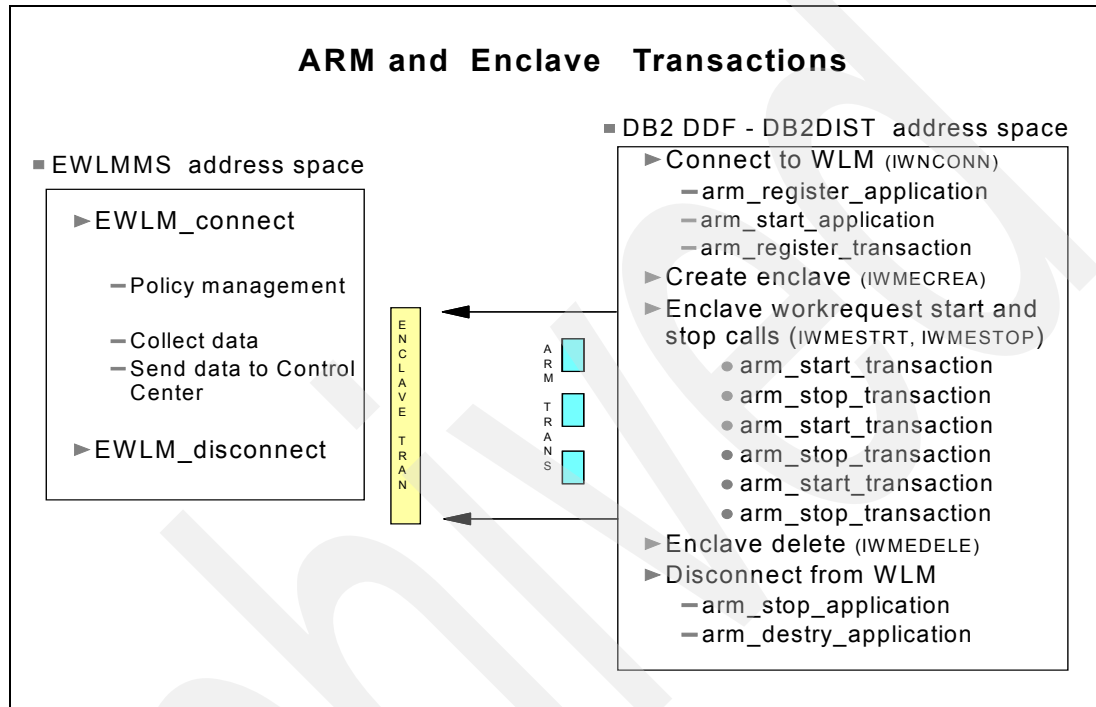


Figure 1-9 ARM and enclave

When the EWLM address space is started it connects to WLM. Its main functions are:

- ▶ Manage the EWLM policy received from the domain manager.
- ▶ Collect the performance data and send it to the domain manager.

When DB2 is started it connects to WLM and specifies that it will participate in EWLM. The DB2 DIST address space is then registered to ARM on z/OS as an EWLM application participant.

ARM data is captured only if a correlator is sent in from an ARM-enabled distributed application such as WebSphere; otherwise, DB2 continues processing distributed requests as it usually does.

When DB2 receives the correlator from a distributed application, DB2 DDF creates an enclave and starts a work request. Attribute values such as PLAN, PACKAGE, and CONNECTION are passed with the correlator to zWLM to classify the work request and associate it to a service class.

When an enclave is created by DDF, an arm_start_transaction is implicitly invoked to mark the beginning of a transaction, and then an arm_stop_transaction gets called at the end of processing a database request message. A pair of start and stop transaction is associated with each database request message from a distributed application or DRDA® requester until a commit is received and the enclave is then deleted.

In other words, in an EWLM environment, a DB2 transaction no longer has the same meaning as one enclave transaction, or a unit-of-work between two DB2 commits. In our case, a transaction becomes an ARM work request, and its elapse time is simply the duration of receiving and processing a database request message to DB2 from the DRDA requestor.

Figure 1-10 illustrates where in monitor reports you can find the number of ARM and enclave transactions.

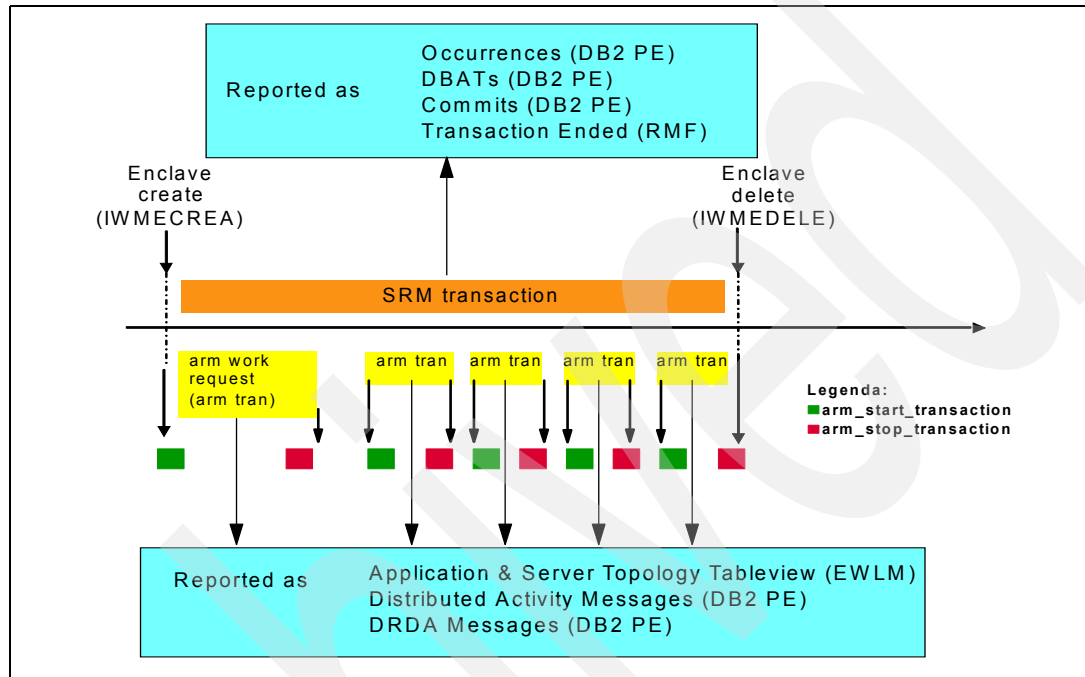


Figure 1-10 ARM and enclave transaction relationships

The EWLM Control Center reports the number of transactions (ARM work requests). For DB2, this number of ARM transactions (ARM work requests) is reported as messages in the Distributed Activity and DRDA REMOTE LOCSDB2 sections of the Performance Expert accounting and statistics reports.

An enclave transaction corresponds to an SRM transaction and its number is reported as occurrences, DBATS, and commits by DB2 Performance Expert, and Transaction Ended by RMF™ in the Workload Activity report in the Service class and Report class.

So, as far as the relationship between EWLM and z/OS WLM goes, at this time they are driven by different service policies, operate independently, and have different scopes. EWLM manages multi-tiered application environments that may run on different hardware platforms and operating systems including z/OS and has an end-to-end scope of control. The z/OS WLM scope is limited to z/OS platforms. The future might see functionality that converges these policies.

1.3.2 Workload management on pSeries

For pSeries installations there are several platform-specific workload management tools available. This includes, for example, AIX Workload Manager (WLM) and Partition Load Manager (PLM). The question is how does the Enterprise Workload Manager (EWLM) fit into this picture? Are these workload management tools competitive products or are they complementary to each other? Can they coexist or do customers need to make a decision

about which of them to use in their environment? We discuss and clarify these issues in the following paragraphs.

AIX Workload Manager is a free-of-charge feature of the AIX operating system and works within one AIX image. It can manage CPU and memory resources as well as I/O bandwidth for an individual server or a partition. The aim is to prioritize specific applications or processes within a single operating system instance. If you have three competing applications running on one server, for example, AIX WLM can make a prioritization according to customer needs and give different a CPU time for each application, leading to different performances for the competing applications. This is shown in Figure 1-11.

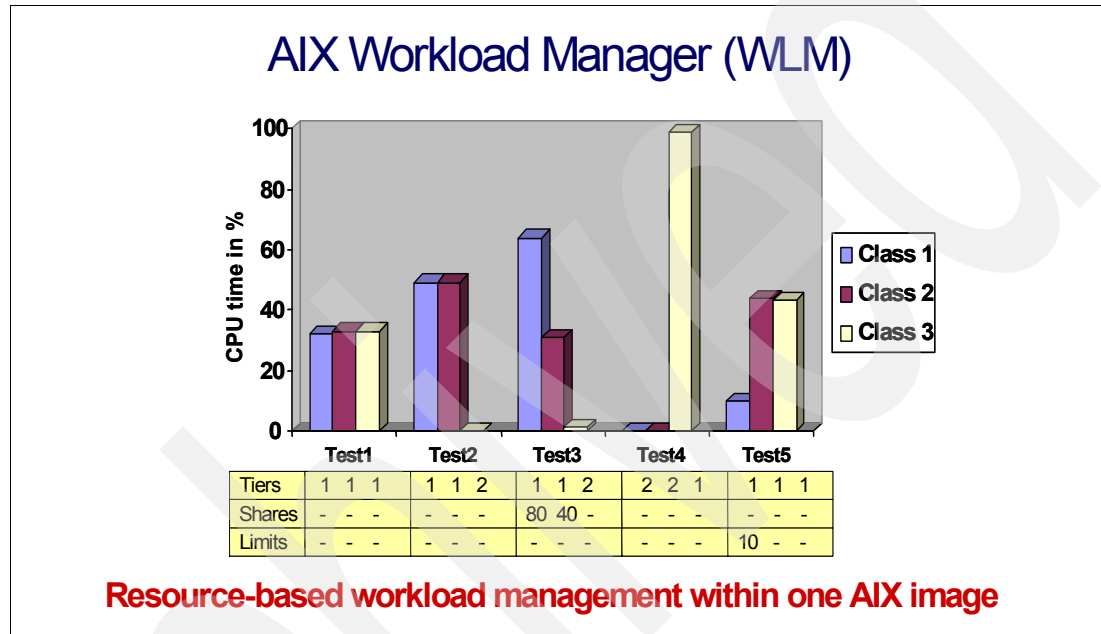


Figure 1-11 AIX Workload Manager (WLM)

AIX Workload Manager is a resource-based workload management tool. This means the customer defines the amount of resources that each application should get relative to other applications in case of resource contention. The actual performance goal is unknown to WLM — it only knows the relative share of resources between applications. Terms like *transactions* and *response times* are unknown and irrelevant to WLM and can therefore not be managed or monitored.

Partition Load Manager (PLM) is a cross-partition workload management tool within a partitionable pSeries server. Instead of having competing applications within one AIX image as with WLM, PLM deals with competing partitions within a p5 server. PLM has no knowledge about the applications running on each partition. It only compares the CPU and memory utilization of the partitions and allocates free resources or shifts resources between partitions according to the configuration, as shown in Figure 1-12. The main advantage is that in case of changing workloads in the partitions, PLM automatically reconfigures resources according to the policy.

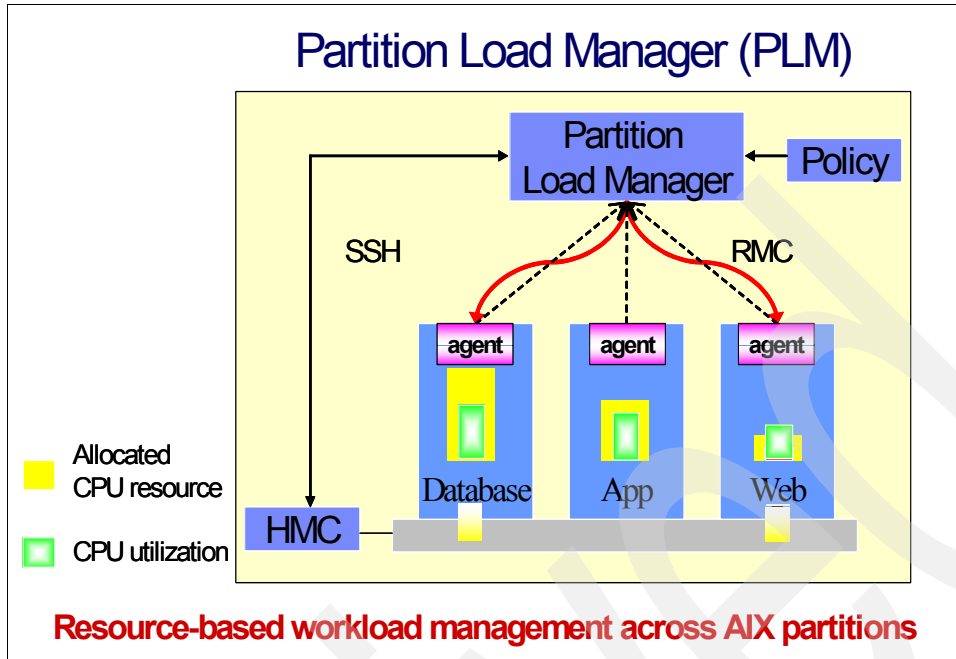


Figure 1-12 Partition Load Manager (PLM)

PLM is also resource-based and has no understanding of transactions and response times. It comes as part of the Advanced POWER™ Virtualization (APV) on IBM System p5™. Like EWLM, it is part of the Virtualization Engine and therefore part of the IBM virtualization strategy.

How does EWLM fit into this picture? We have AIX Workload Manager as a resource-based workload management tool within one AIX image. We have Partition Load Manager as a resource-based workload management tool within partitionable pSeries servers. For EWLM we now leave the exclusive pSeries world and include other IBM and non-IBM hardware and operating systems like Linux®, Solaris™, Windows, and others. We are not interested in the performance of individual servers or applications but in the fulfillment of response time goals for transactions running through a multi-tier architecture in heterogeneous environments. The most important task of EWLM is to deliver and manage end-to-end topology and statistics of business transactions across hardware boundaries.

The second main difference of EWLM is that it is not resource based but goal based. That means that instead of deciding how many resources to give to each application to achieve a specific goal, with EWLM we assign a specific performance goal and let the workload management tool decide itself how to distribute resources best in order to achieve this goal.

From this conceptual comparison it is clear that AIX WLM, PLM, and EWLM are not competing tools. They are all workload management tools but are designed to serve different purposes. Depending on the environment, one or the other product might be more suitable. A co-existence of these workload management tools is also possible.

Archived



Installing and configuring EWLM

In this chapter we describe how to:

- ▶ Install the Virtualization Engine common components.
- ▶ Install and configure the EWLM domain manager and managed servers.

2.1 Overview

This chapter describes the installation and configuration of the EWLM environment through the following steps:

1. Install the IBM Virtualization Engine common components.
2. Install the EWLM domain manager code.
3. Install the EWLM managed servers code.

We also discuss the migration path from the EWLM Release 1 to EWLM V2.1.

Important: If you currently have EWLM R1 installed and you intend to install EWLM V2.1, you should proceed to 2.9, “Migrating from EWLM Release 1” on page 72.

2.2 Packaging

EWLM V2R1 is packaged and delivered with different product numbers based on the platforms supporting the domain manager and the managed servers. The following list summarizes the EWLM product numbers:

- ▶ 5733-EWA IBM Virtualization Engine Enterprise Workload Manager for i5/OS® V2.1
 - i5/OS version only
 - Ships with i5 Enterprise Edition
- ▶ 5733-EWM IBM Virtualization Engine Enterprise Workload Manager for Power V2.1
 - Choice of i5/OS, AIX, or Linux on Power
 - Includes EWLM domain manager, Virtualization Engine console, common components, EWLM agents for AIX, i5/OS, Linux, HP-UX
 - Optional feature code for agents for Windows and Solaris
- ▶ 5724-EWM IBM Virtualization Engine Enterprise Workload Manager V2.1
 - Choice of Windows or Linux on xSeries®
 - Component #1 includes EWLM domain manager, Virtualization Engine console, common components, EWLM agents for AIX, i5/OS, Linux, HP-UX
 - Component #2 includes EWLM domain manager, Virtualization Engine console, common components, EWLM agents for Windows and Solaris
- ▶ 5655-EWM IBM Virtualization Engine Enterprise Workload Manager for z/OS V2.1
 - z/OS version only
 - Includes EWLM domain manager, common components, and z/OS agent
- ▶ 5648-f08 IBM Virtualization Engine for Linux on z/Series (proposed name)
 - Includes IBM Director, Virtualization Engine console and common components
 - Feature code #1 – EWLM domain manager and agents for AIX, i5/OS, Linux, HP-UX
 - Feature code #2 – EWLM domain manager and agents for Windows and Solaris

2.3 Before installing

Prior to installing the Virtualization Engine common components and EWLM check to insure that you have met the minimum hardware and software requirements.

Be sure that you have the correct operating systems for the products you intend to install. See Table 2-1.

Table 2-1 Software requirements and list of supported functions

Operating system	Domain manager and control center	Managed server and firewall broker	Managed server - Load balancing	Partition management ^a
IBM i5/OS V5R3	x	x		x
IBM AIX 5L™ v5.2		x	x	
IBM AIX 5L v5.3	x	x	x	x
RedHat Enterprise Linux AS 4.0	x			
SUSE LINUX Enterprise (SLES) 9 ^b	x	x	x	x
Microsoft Windows ^c	x ^d	x	x	
z/OS V1R6	x	x	x	
HP-UX 11iv1		x	x	
Sun™ Microsystems™ Solaris 9 (SPARC Platform Edition)		x	x	

a. Partition management is only supported on IBM Systems p5 and eServer™ p5 systems.

b. Fixpack 30 is required for Linux managed server support.

c. Support for Windows implies support for Windows 2000 Server, Windows 2000 Advanced Server, Windows Server® 2003 Standard Edition, and Windows Server 2003 Enterprise Edition, unless stated otherwise.

d. Domain manager is supported only on Windows Server 2003 Standard Edition and Windows Server 2003 Enterprise Edition.

Important: Make sure you apply the most recent EWLM fix packages. Go to the IBM Virtualization Engine systems services fixes Web site:

<http://techsupport.services.ibm.com/server/VirtualizationEngine>

Click **By OS** → **Required OS Fixes** and download the necessary fixes for your environment.

Table 2-2 shows the recommended hardware requirements for the domain manager to handle a moderate policy definition with up to 50 managed servers. We also recommend that you monitor the maxHeap size for the WebSphere Application Server. The default value of 256 MB may be not enough for complex environments.

Attention: If you are installing the Virtualization Engine suite, we recommend that you add an extra 4 GB of disk space.

Table 2-2 Domain manager hardware requirements

Environment	Processor speed/capacity	Memory	Disk space ^a
AIX	1.4 GHz	1.5 MB	2 GB
i5/OS	700 CPW	1.5 GB	2 GB
Linux on iSeries ^{TMb}	1.4 Ghz	512 MB	2 GB
Linux on POWER ^b	1.4 Ghz	512 MB	2 GB
Linux on xSeries ^b	2 GHz	512 MB	2 GB
Windows on xSeries ^c	2 Ghz	512 MB	2 GB
z/OS	Any hardware environment that supports z/OS v1R6 or later		

a. Specifies the minimum amount of disk space to install and run EWLM. Depending on the complexity of the EWLM environment, you might need more space.

b. Support for Linux implies support for RedHat Enterprise Linux AS 4.0 and SUSE LINUX Enterprise Server (SLES) 9.

c. Support for Windows implies support for Windows Server 2003 Standard Edition and Windows 2003 Enterprise Edition.

Following are the minimum requirements for the Web browser used to access the Control Center:

- ▶ Internet Explorer® (IE) 6.0 with Service Pack 1.
- ▶ IBM Java Runtime Environment (JRETM) 1.4.2 with Service Release 2 or later.
- ▶ Adobe SVG viewer. You can download the appropriate SVG viewer at the following Adobe Web site:
<http://www.adobe.com/svg/viewer/install/>
- ▶ Pop-up windows enabled.
- ▶ Add the domain manager host name to the Web browser's local intranet settings to avoid security errors.

Table 2-3 shows the necessary users and their rights for installation, configuration, and management of EWLM environment.

Table 2-3 Operating system users

OS name	User role	User requirements
Domain manager requirements		
Linux	EWLM installation and configuration	Root authority.
	WAS administration	Root authority.
	DB2 owner	No special requirements.
	ITDS DB2 instance owner	The root user must be part of the primary group of this user.
	EWLM Control Center roles	Administrator. Operator. Monitor.
AIX	EWLM installation and configuration	Root authority.
	WAS administration	Root authority.
	DB2 owner	No special requirements.
	ITDS DB2 instance owner	The root user must be part of the primary group of this user.
	EWLM Control Center roles	Administrator. Operator. Monitor.
Windows	EWLM installation and configuration	- Member of Administrators group. - Act as part of OS. - Log on as a service.
	WAS administration	- Member of Administrators group. - Act as part of OS. - Log on as a service.
	DB2 owner	No special requirements.
	ITDS DB2 instance owner	No special requirements.
	EWLM Control Center roles	Administrator. Operator. Monitor.
i5/OS	EWLM installation and configuration	Requires *ALLOBJ, *SECADM, *JOBCTL, and *IOSYSCFG special authority.
	WAS administration	No special requirements.
	DB2 owner	No special requirements.
	ITDS DB2 instance owner	No special requirements.
	EWLM Control Center roles	Administrator. Operator. Monitor.

OS name	User role	User requirements
z/OS	EWLM installation and configuration	Superuser authority.
	WAS administration	Superuser authority.
	EWLM Control Center roles	Administrator. Operator. Monitor.
Managed server requirements		
AIX	EWLM installation and configuration	Root authority or with capabilities=CAP_EWLM_AGENT,CAP_PR OPAGATE.
Windows	EWLM installation and configuration	Requires user to belong to Administrators group.
i5/OS	EWLM installation and configuration	Requires Managed Server to run under QWLM2 user profile, which has the necessary access rights.
Solaris	EWLM installation and configuration	Root authority or with effective user identification (EUID) of zero.
HP-UX	EWLM installation and configuration	Root authority.
z/OS	EWLM installation and configuration	Superuser authority or necessary rights seated up by the superuser.

There are also specific requirements for Virtualization Engine common components and EWLM installation on some platforms. Take this inTO consideration when planing the installation. The following list shows the platforms and their requirements:

- AIX** Ensure that the following file sets are installed before you install the netWLM fileset: bos.rte 5.2.0.30 and bos.net.tcp.client 5.2.0.30.
Enable EWLM using the `ewlmcfg -c` command.
Make sure that you have installed the netWLM fileset (bos.net.ewlm.rte). After installing this fileset, remember to refresh the inetd. Use the command `refresh -s inetd`.
- Windows** Reboot after installation.
- Solaris** Reboot after installation.
Install `1sof` if you are going to use load balancing.
- HP-UX** Reboot after installation.
Install `1sof` if you are going to use load balancing.

Finally, we recommend that you to fill out the following tables prior to starting the installation. There are three tables: Table 2-4 is for Virtualization Engine common component values, Table 2-5 is for EWLM domain manager values, and Table 2-6 on page 28 is for EWLM managed server values. The values in the shaded cells under the Default column are the product defaults. Non-shaded cells under the Default column are values we chose for our ITSO environment.

Table 2-4 IBM Virtualization Engine common components installation and configuration parameters

Virtualization Engine common components installation and configuration		
Parameter	Default ^a	Your value
Install IBM Tivoli® Directory Server on this system:		
▶ Server port	389	
▶ Admin daemon port	3589	
▶ Encryption seed key	123456789012	
▶ New environment name	VE Environment 1	
Use an existing IBM Tivoli Directory Server:		
▶ LDAP server IP address or host name	Not used	
▶ LDAP server port	Not used	
▶ LDAP server admin ID	Not used	
▶ LDAP server admin password	Not used	
DB2 instance owner:		
▶ User ID	vedb2ur	
▶ User password	itso	
ITDS DB2 instance owner (must be created in operating system prior to starting the installation):		
▶ User ID	itdsadm	
▶ User password	itso	
ITDS administrator:		
▶ User ID	cn=Admin	
▶ User password	itso	

a. The product default values are in shaded cells. The values in non-shaded cells are from our ITSO environment and can be used as an example.

Table 2-5 EWLM domain manager installation and configuration parameters

EWLM domain manager installation and configuration		
Parameter	Default ^a	Your value
Config ID	DM1	
Domain name	ITSO Domain	
Domain manager IP address or host name	ewlm1	
Domain manger port	5773	

EWLM domain manager installation and configuration		
Parameter	Default^a	Your value
WebSphere starting port	5775	
Control Center to domain manager port	5800	
Security level	None	
SSL certificate source	EWLM default	
Firewall broker to domain manager port	5802	
Firewall broker SOCKS server IP address or host name	Not used	
Firewall broker SOCKS server port	Not used	
WebSphere administrator (must be created in operating system prior to starting the installation):		
▶ User ID	wasadmin	
▶ User password	itso	
Control Center users (must be created in operating system prior to starting the installation):		
▶ Administrator users	ewlmadm	
▶ Operator users	Not used	
▶ Monitor users	Not used	

a. The product default values are in shaded cells. The values in non-shaded cells are from our ITSO environment and can be used as an example.

Table 2-6 EWLM managed server installation and configuration parameters

EWLM managed server installation and configuration		
Parameter	Default^a	Your value
Domain manager IP address or host name	ewlm1	
Domain manager port	5773	
Firewall broker IP address or host name	Not used	
Firewall broker port	5805	
Security level	None	
Proxy server to contact the domain manager	Not used	

a. The product default values are in shaded cells. The values in non-shaded cells are from our ITSO environment and can be used as an example.

With all this information in hand, we are ready to start installing the environment. Please proceed to 2.4, “Virtualization Engine common components installation” on page 29, for more information about installing the code in a distributed platform. For information about installing the code in z/OS, please see 2.8.4, “Configuring the managed server” on page 67.

2.4 Virtualization Engine common components installation

Installation of the Virtualization Engine common components is a prerequisite for the installation of EWLM. The Virtualization Engine common components consist of the Virtualization Engine base components, DB2 UDB Enterprise Server Edition, WebSphere Application Server, and the IBM Tivoli Director Server (ITDS). The installation wizard guides you through the tasks needed to install these components prior to installing the EWLM product.

The EWLM domain manager installation is described in 2.5, “Installing and configuring EWLM domain manager” on page 34.

As an example, we show the steps of a Virtualization Engine common components installation in a Windows 2003 server; however, the wizard is very similar for all platforms except the z/OS platform, which we describe in 2.8.2, “Installation considerations” on page 59.

Follow these steps to install the Virtualization Engine common components:

1. Insert the IBM Virtualization Engine Common Components for Windows DVD into the server DVD drive and wait for the Welcome screen. If it does not appear in a few minutes, open the DVD folder with Windows Explorer and double-click the **autorun.bat** file. Click **Next** to continue.

Tip: For a UNIX environment:

- ▶ Mount the CD-ROM and go to <CD mount point>/FILES and execute the installVE<platform>.bin, where <platform> is an index to identify the install platform, for example, installVEAix.bin for AIX.
- ▶ The wizard needs a graphical environment in order to run. If you are installing the code using a **telnet** session, for example, remember to set the DISPLAY variable and also to enable an X server environment on your workstation.

2. The second screen is the license agreement screen. Click the name of the component to read its license agreement terms and conditions. If you accept all terms, select the **I accept the terms of the license agreements** and click **Next**.

3. The wizard prompts for the destination directory of Virtualization Engine, as you can see in Figure 2-1. It is possible to change the default directory by typing a new value into the Directory Name box or clicking **Browse** and selecting a directory. To continue click **Next**.

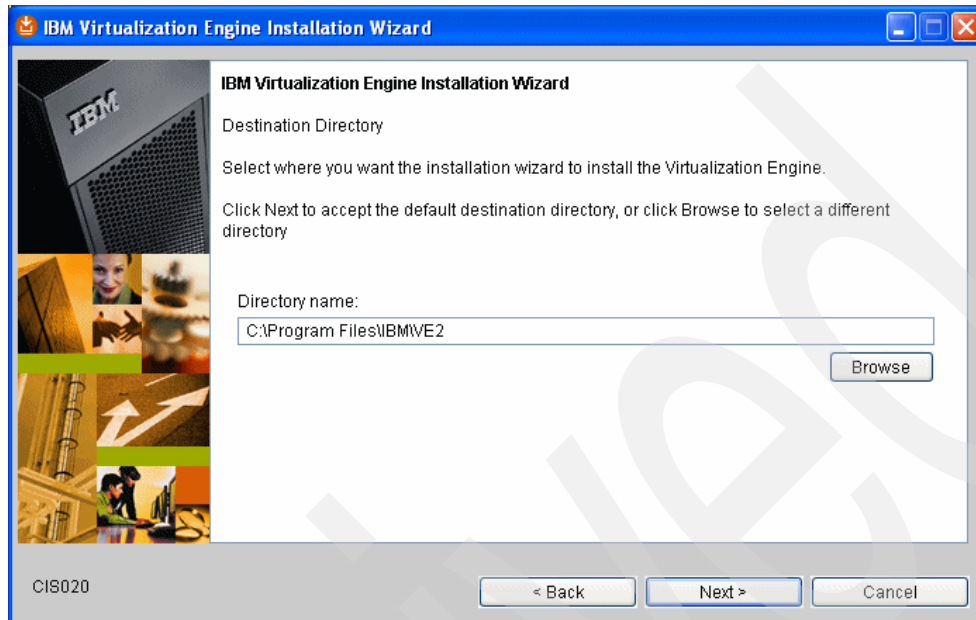


Figure 2-1 Destination Directory screen

4. You may change the default location of the Virtualization Engine log directory by typing in a new value and clicking **Browse** to select a different directory, or simply accept the default by clicking **Next**.
5. The wizard now checks that the minimum system requirements are met and that all components necessary for installation are available. Eventually, it prompts you to choose to either install a new IBM Tivoli Directory Server (ITDS), which is an LDAP server, or if it should use an existing ITDS installation. Make the appropriate choice and click **Next**. If you are not sure about the existence of an ITDS in your environment, select **Install IBM Directory Server on this system** and click **Next**.

Attention: If your system did not meet the requirements, you may be able to continue, but some functions might not work. Refer to 2.3, “Before installing” on page 22, for information about supported platforms.

- The wizard once again checks system requirements and then copies all files necessary for installation into the destination directory. This can take some time. In our case, it took approximately 20 minutes. When the process completes, the screen in Figure 2-2 appears. Click **Next** to continue.

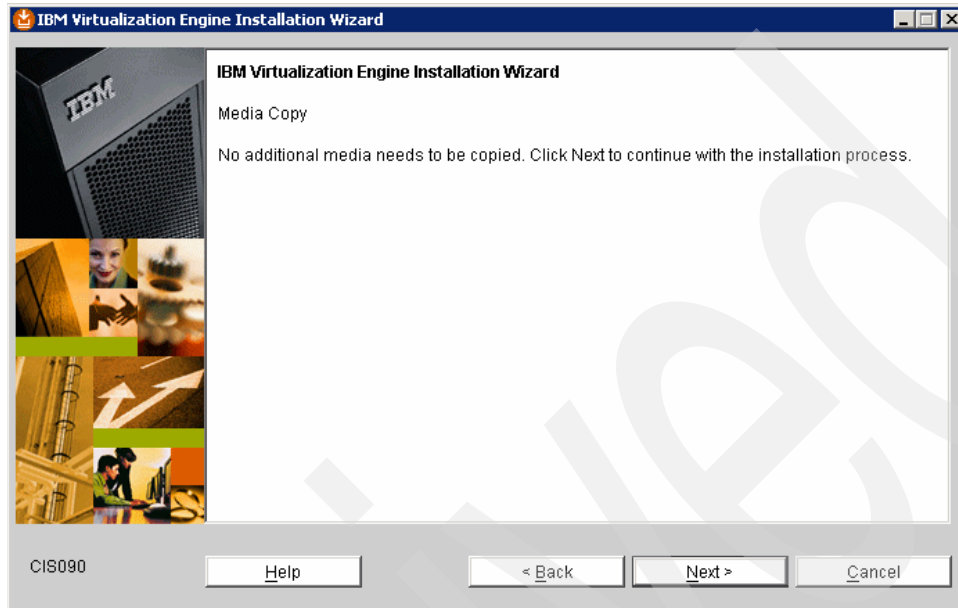


Figure 2-2 Media Copy screen

- Click **Next**, the wizard restarts automatically and unpacks all necessary files. This can also take some time, and when it finishes the wizard prompts you to either create a new operating system user ID for the DB2 database or to use an existing one. If you have already created a user to be the owner of the DB2 instance, choose **Use an existing operating systems user ID for the DB2 instance owner**; otherwise, choose **Create a new operating systems user ID for the DB2 instance owner**. Click **Next** to continue. We chose **Create a new operating systems user ID for the DB2 instance owner**.

Tip: The installation requires some users and passwords. Be sure you know all of the passwords or just make a unique one.

8. The next screen asks for the DB2 user ID and password. Figure 2-3 shows this screen. Enter the values and click **Next**.

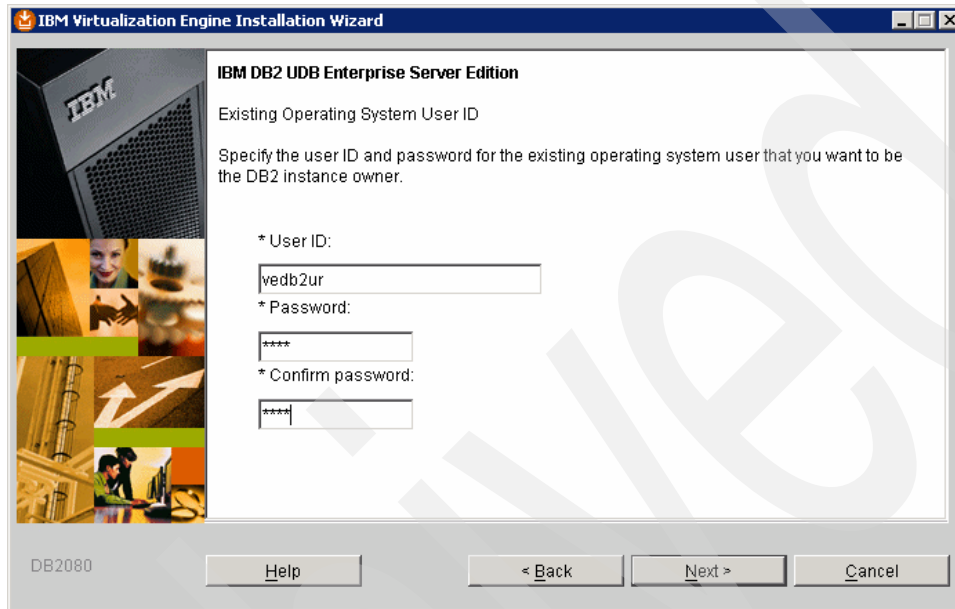


Figure 2-3 DB2 user creation screen

9. Since we previously chose to install a new ITDS, the wizard asks for the ITDS ports. We accept the default values and click **Next** to continue.
10. In order to encrypt the data, ITDS needs a string as a seed for encryption keys. Type anything you want without spaces and click **Next** to continue.
11. Next the wizard asks for the user ID that owns the DB2 instance of ITDS. This user ID must be created prior to the installation and must have administrator authority. Enter the user ID and its password and click **Next** to continue.

Tip: If you did not create the user ID until this point, do it now and click **Back** to go to the seed string screen, then click **Next** again. Now fill in the user ID box and the wizard will validate the new user. Also remember that the user ID must not be longer than eight characters.

12. Enter a password for ITDS administration proposes and click **Next**.

13. Now the wizard displays all the components it will install. Click **Install** to start the installation. In our environment, it took approximately one hour and a half to finish. Figure 2-4 shows the installation progress.

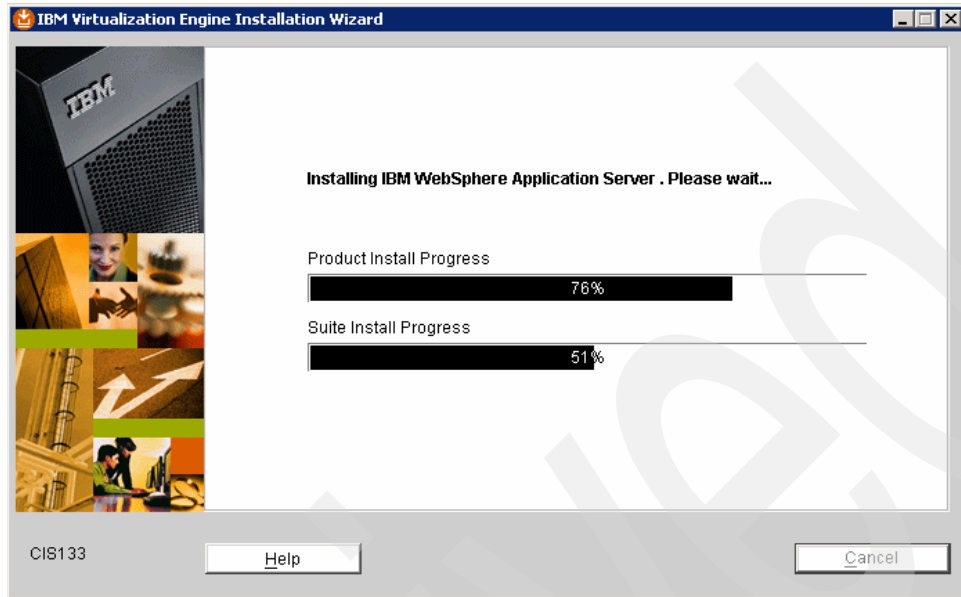


Figure 2-4 Installation progress screen

14. Once the installation finishes, the wizard starts the configuration of DB2 and ITDS. First, it configures the DB2 and then the ITDS. You just have to click **Next** in the following screens until you reach the final installation screen, as shown in Figure 2-5. Depending on the platform, you might be asked to reboot the server.

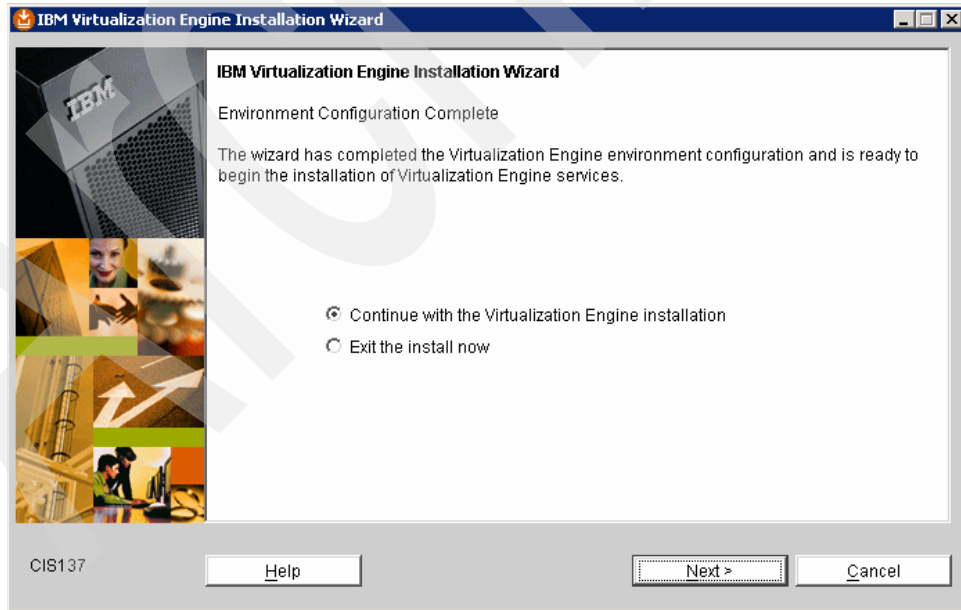


Figure 2-5 Final installation screen

15. The installation of the basic Virtualization Engine components is now complete. Select **Continue with the Virtualization Engine installation** and click **Next** to continue.

16. You now receive the Welcome screen again. Click **Next** and the wizard asks for ITDS information. The password is the one created in step 12. Fill out all the information and click **Next** to proceed.

17. The next screen asks for a name of a new environment for the ITDS. You can accept the default or change to something more meaningful to you. Click **Next** to continue.

The wizard is now ready to install the EWLM domain manager code. Please proceed to 2.5, “Installing and configuring EWLM domain manager” on page 34, for more information.

2.5 Installing and configuring EWLM domain manager

With the Virtualization Engine Common Components installation completed, we can install and configure the EWLM domain manager code.

2.5.1 Installing EWLM domain manager code

Prior to installing the EWLM domain manager, you must install the Virtualization Engine Common Components as described in 2.4, “Virtualization Engine common components installation” on page 29. If you have completed the installation steps from 2.4, “Virtualization Engine common components installation” on page 29, you should be stopped in the screen shown in Figure 2-6.

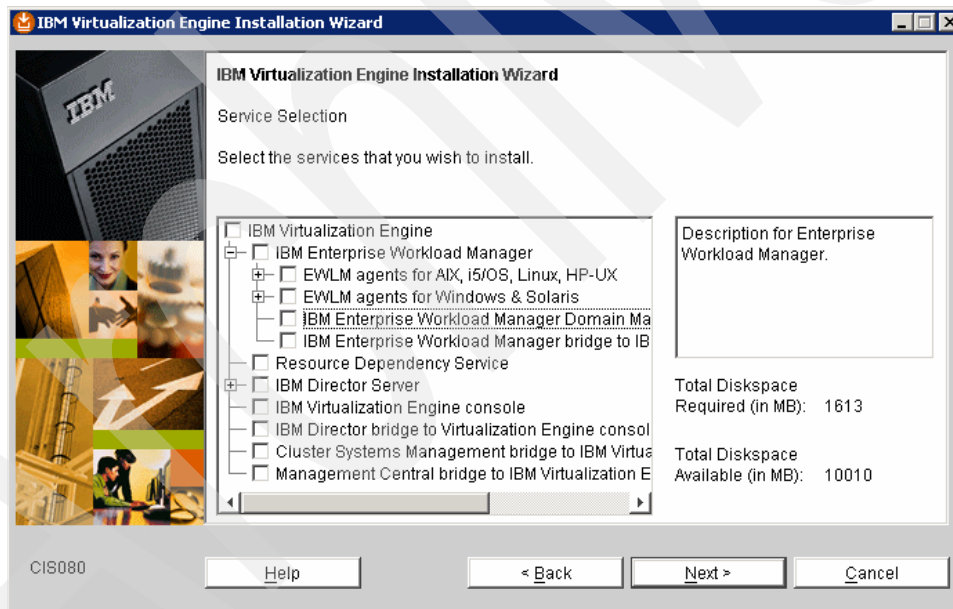


Figure 2-6 EWLM install screen

If you have closed the installation wizard but have not finished the Virtualization Engine Common Components installation, double-click the Virtualization Engine installation wizard icon (shown in Figure 2-7) and follow the instructions until you reach the EWLM installation screen shown in Figure 2-6 on page 34. For UNIX platforms, the installation wizard can be started using the `installVE<platform>.bin` command, where <platform> is a string to identify the platform, for example, AIX, under the /<Virtualization Engine install directory>/Install/FILES/ directory.



Figure 2-7 Virtualization Engine wizard icon

Follow the steps to install EWLM domain manager code:

1. In the screen shown in Figure 2-6 on page 34, Expand the Enterprise Workload Manager tree by clicking in the plus sign (+), select **IBM Enterprise Workload Manager Domain Manager**, and click **Next** to continue.
2. The next screen shows the necessary components for the EWLM domain manager installation. Click **Next** to continue.
3. Since the domain manager components are not shipped on the same media as the Virtualization Engine common components, the wizard now prompts for the necessary media. The screen shown in Figure 2-8 now appears asking that the EWLM domain manager code CD be inserted into the CD-ROM drive. Click **Copy Media** to continue.

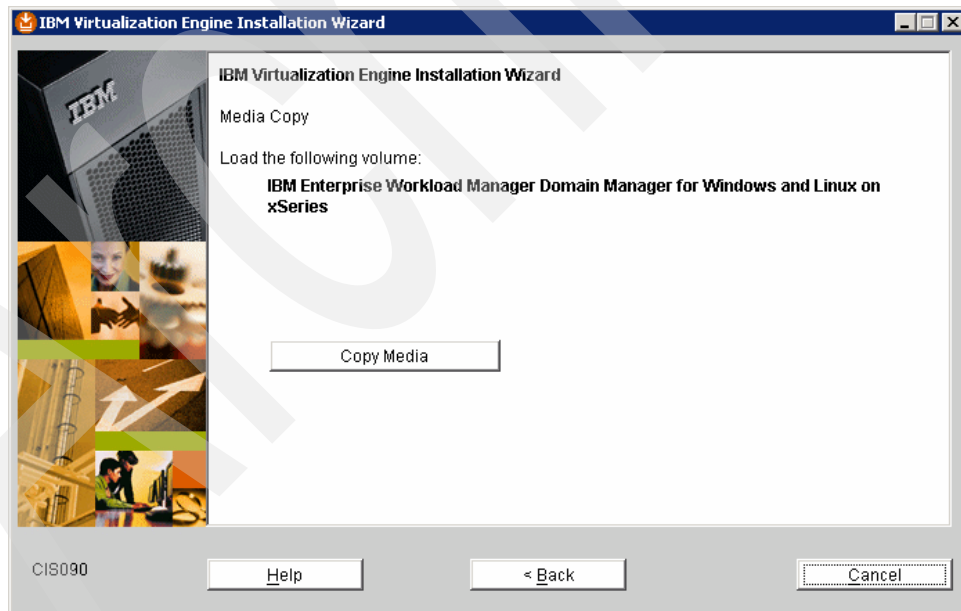


Figure 2-8 Copy media screen

4. Now that the necessary media are copied, click **Next** to continue. The wizard restarts and continues with the installation process.
5. The wizard now asks for a new LDAP repository. Click **Next**.
6. Select **Create a new identity mapping repository** and click **Next**.

7. Now select **Use recommended default values to configure the server for identity mapping** and click **Next**. The next screens shows the configuration of the LDAP repository. Click **Next** to configure the LDAP.
8. The following screen is the installation directory screen, as you can see in Figure 2-9. You can change the information directory by typing a new path or using the Browse button. This is the directory where EWLM stores log information.

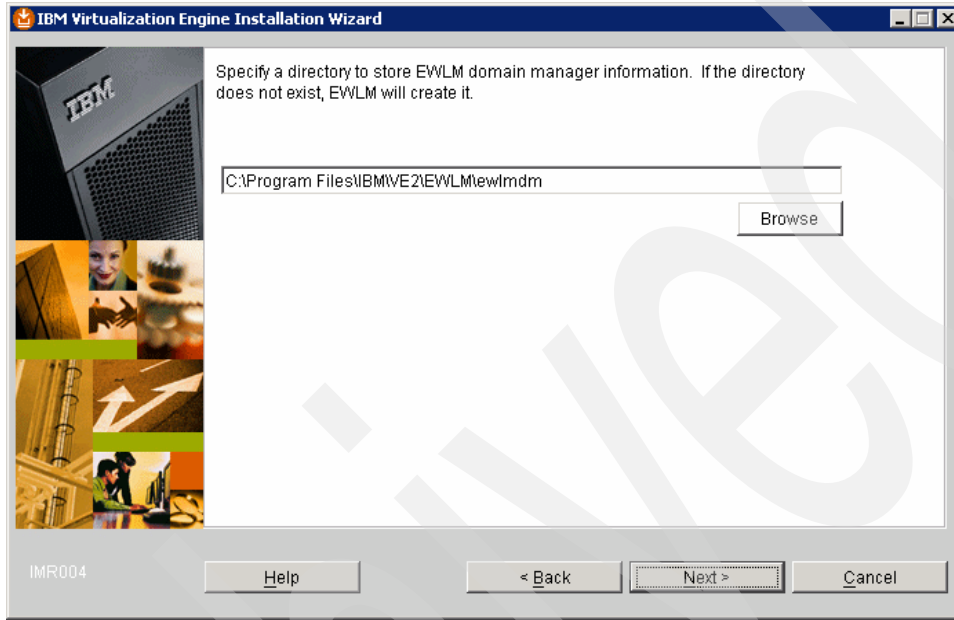


Figure 2-9 Information directory screen

9. The next screen shows a summary of the packages that will be installed. Click **Install** to begin the installation. At the end of the installation, the wizards asks you to click **Next** to finish some steps.

10. When the wizard finishes restarting itself, the screen shown in Figure 2-10 appears. Choose **Now** to begin the configuration of the domain manager. The installation wizard performs some final installation steps and then calls the configuration wizard. Please proceed to 2.5.2, “Configuring the domain manager” on page 37, to continue the configuration.

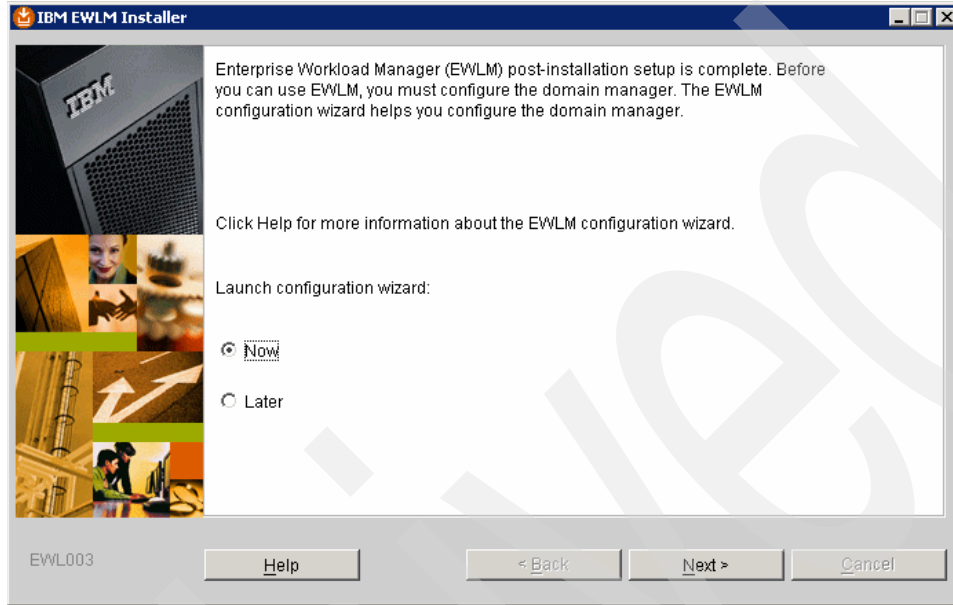


Figure 2-10 Final domain manager installation screen

2.5.2 Configuring the domain manager

After finishing the EWLM domain manager code installation, it is necessary to configure the domain manager. For this, we used the configuration wizard. Table 2-7 shows which platforms are currently supporting the Wizard.

Table 2-7 Platforms that support domain manager configuration Wizard

Platform	configWizardDM	Default installation path
AIX	X	/opt/IBM/VE2/EWLM/bin
i5/OS	Not supported	
Linux	X	/opt/IBM/VE2/EWLM/bin
Windows	X	C:\Program Files\IBM\VE2\EWLM\bin
z/OS	Not supported	

If you just finished the domain manager installation and selected **Now** on the last installation screen, you will see the screen shown in Figure 2-11. Select **Create domain manager and Control Center** and click **Next**. It is started by the `configWizardDM` command and the default path to this command is shown in Table 2-11 on page 56. If you installed the EWLM in a different path, then go to the bin directory under your installation path.

Tip: If you have left the installation wizard without selecting **Now** to start the configuration wizard, go to `<EWLM domain manager installation directory>/bin` and run `configWizardDM`. This should be the `.bat` file in Windows or a `.sh` file in the UNIX platform.

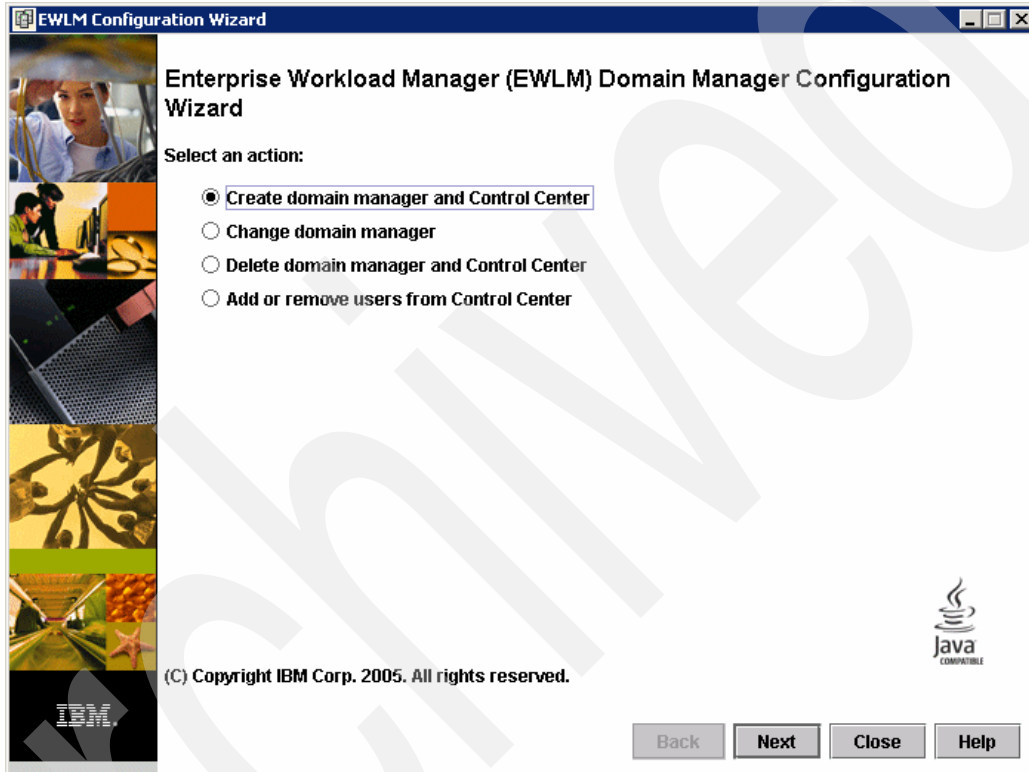
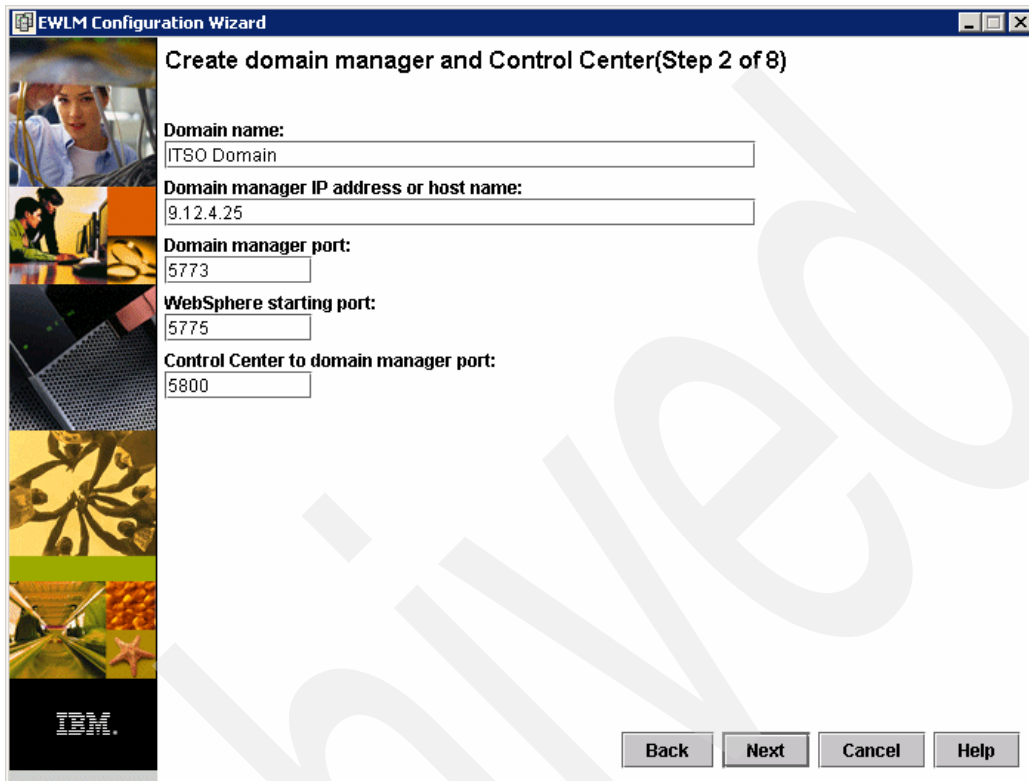


Figure 2-11 Configuration wizard

The steps to install and configure the EWLM domain manager are:

1. You are now prompted for a configuration ID. This configuration ID is use to identify this particular domain manager configuration so it is recommended that you use a name that reflects this configuration.
2. Next the wizard asks for the domain name, domain manager IP address or host name, and domain manager port. The domain name can be anything meaningful for your environment. The *Domain Manager port* is the port that is used by the managed server to contact the domain manager. The *WebSphere starting port* is the first of fifteen consecutive ports that are used by WebSphere. These are the ports used to access and manage the domain manager. The *Control Center to domain manager port* is the port used by the control center to communicate with the domain manager instance in

WebSphere. Make sure that the needed ports are not in use by any other application in your environment. In this example we accept the default values. Figure 2-12 shows this screen. Click **Next** to continue.



The screenshot shows a window titled "EWLM Configuration Wizard" with the subtitle "Create domain manager and Control Center(Step 2 of 8)". On the left side, there is a vertical strip of images: a woman looking at a screen, a person working at a computer, a close-up of a keyboard, a close-up of a mouse, a close-up of a network switch, and the IBM logo. The main area contains five labeled input fields:

- Domain name: ITS0 Domain
- Domain manager IP address or host name: 9.12.4.25
- Domain manager port: 5773
- WebSphere starting port: 5775
- Control Center to domain manager port: 5800

At the bottom right, there are four buttons: "Back", "Next", "Cancel", and "Help".

Figure 2-12 IP address and ports for domain manager and control center

3. Now you can choose the security level of the communication between the domain manager and the managed server. Choose the appropriate level for your environment and click **Next**.
4. The wizard now asks for the source of the certifications for SSL communication. If you selected SSL security in the previous screen, choose to use either the EWLM default source or an administrator-defined source and click **Next** to continue. If you previously chose None for security, just click **Next** to continue.
5. Now you configure the Firewall Broker if you previously chose to install it. Select **Enable firewall broker(s) for communication between the domain manager and manager server** and supply all necessary information. If you do not have a Firewall Broker in your environment, just click **Next**.
6. In the next screen, we need to provide a user ID for WebSphere administration. This user ID must first be created on the operating system. Enter the user ID and its password twice then click **Next** to continue.
7. You now enter the users and groups that can access the EWLM domain manager. Just type the user ID in the appropriate role in the left boxes and click **Add**. The user appears in the box on the right. After finishing adding all users, click **Next** to continue. For information about users and rules, please refer to 2.7.3, "Managing Control Center users" on page 54.
8. In the final screen, check the box to start the Control Center. Click **Next** to continue.

9. The wizard configures the domain manager using the information you provided. The screen in Figure 2-13 show the progress of the configuration.

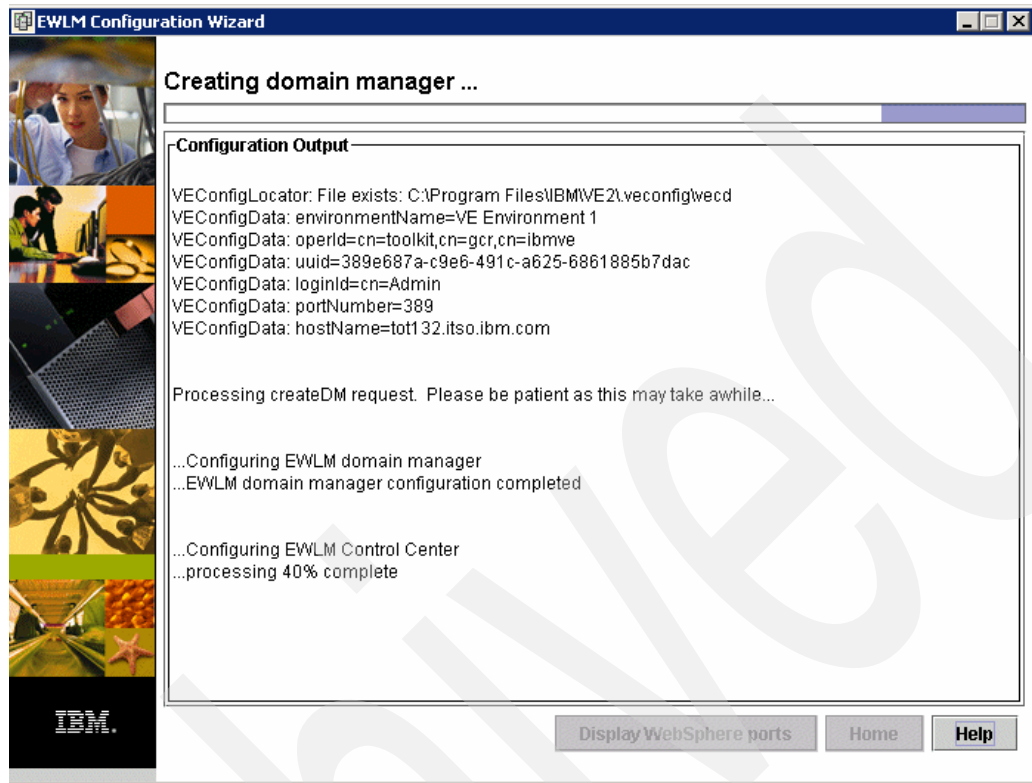


Figure 2-13 Configuring the domain manager

10. A window confirming the success of the configuration appears when it finishes. Click **OK** to close it, click **Home** to go back to the initial domain manager configuration screen, and then click **Close** to exit the configuration wizard.

We have now completed the steps for the EWLM domain manager installation and configuration. We can now proceed to install and configure the managed servers. See 2.6, “Installing and configuring EWLM managed server code” on page 42.

If you are migrating from EWLM R1, please go to 2.9.1, “Migrating from EWLM R1 code” on page 72 for more information.

Directory structure of domain manager

In this section we show the main directories within the EWLM domain manager directory structure.

First let us take a look in the structure of the domain manager in the UNIX platform in Figure 2-14.

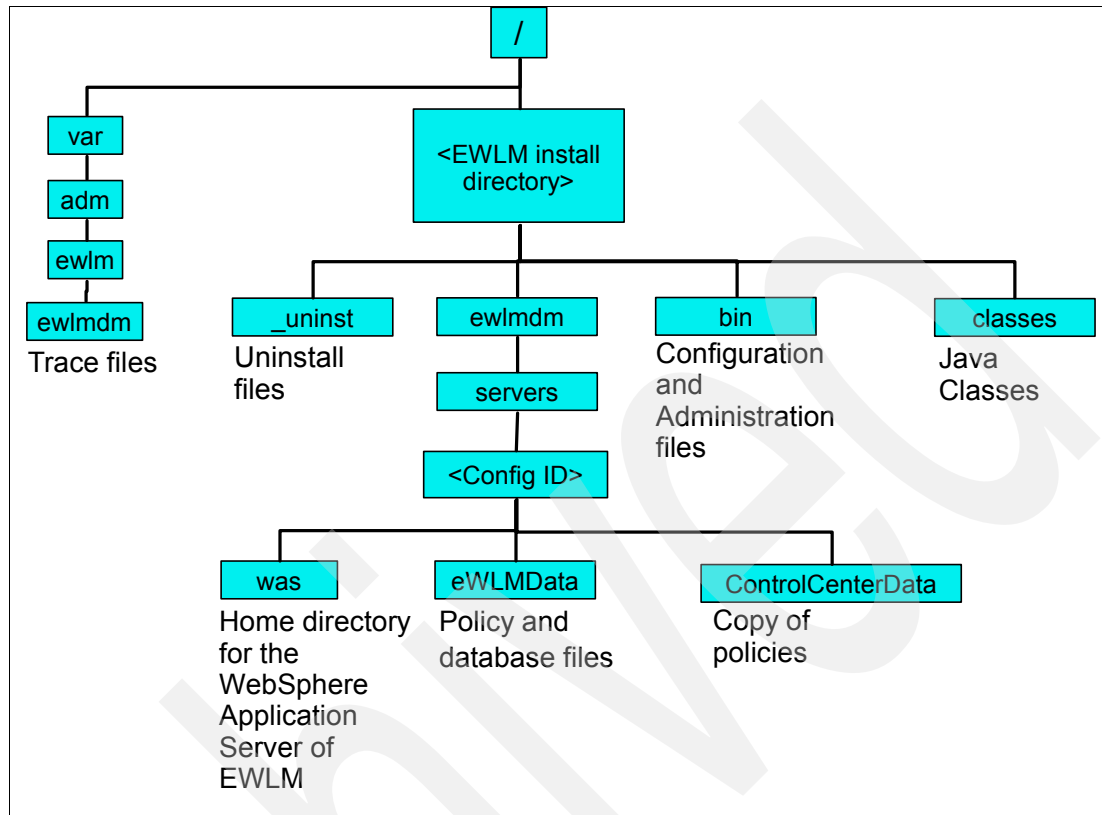


Figure 2-14 Directory structure of domain manager in a UNIX platform

In Windows the structure is as shown in Figure 2-15.

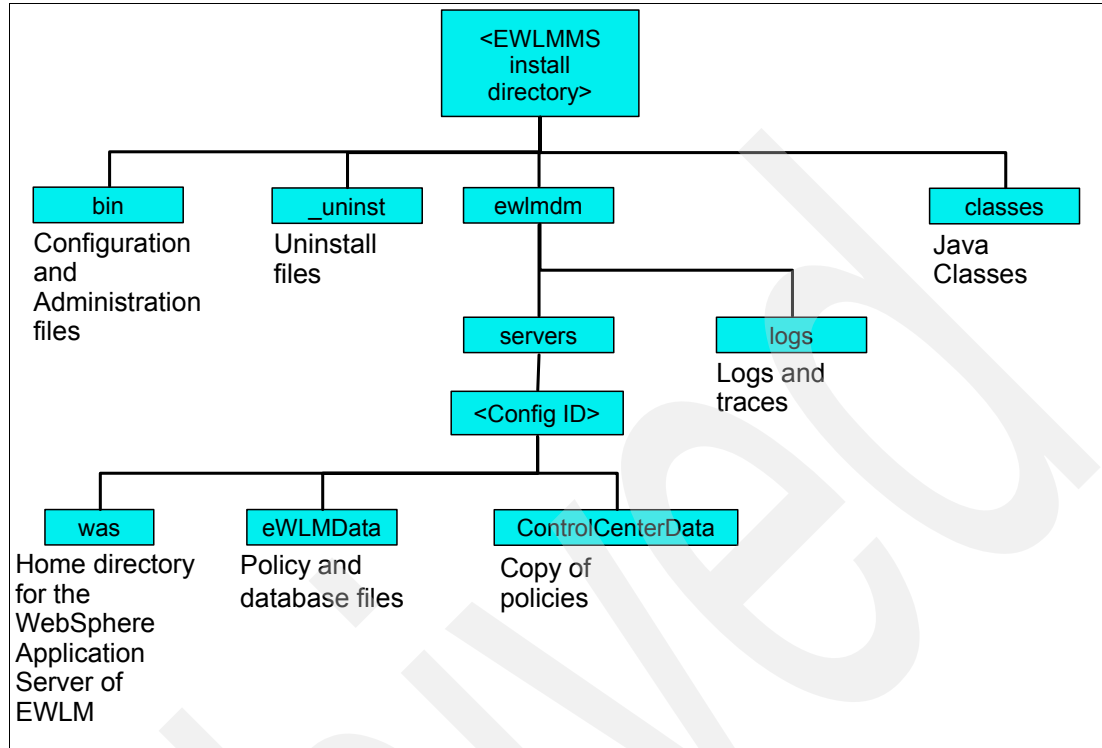


Figure 2-15 Directory structure for EWLMM domain manager in Windows

2.6 Installing and configuring EWLMM managed server code

Now that the domain manager is installed and configured, we can install and configure EWLMM managed server code.

2.6.1 Installing EWLMM managed server code

Prior to the installation of the EWLMM managed server code, we copied the compressed files from the installation CD onto our domain manager platform, uncompressed the files, and then transferred the files to the managed server platforms. We recommend that you also do this in order to have a single point of distribution. If you prefer you can insert the installation CD directly into a CD-ROM drive on the managed server system. Table 2-8 shows the files for each platform.

Table 2-8 Managed servers installation files

IBM Virtualization Engine Enterprise Workload Manager V2.1			
Agents for IBM operating systems		Agents for non-IBM operating systems	
Platform	File ^a	Platform	File ^a
AIX	EWLMMSAixRPD	Linux for xSeries	EWLMMSLinXRPD
i5/OS	EWLMMSi5OSRPD	Linux for POWER	EWLMMSLinPPCRPD

IBM Virtualization Engine Enterprise Workload Manager V2.1		
Agents for IBM operating systems	Agents for non-IBM operating systems	
	Linux for zSeries	EWLMMSLinZRPD
	HP-UX	EWLMMShpuxRPD
	Solaris	EWLMMSSolarisRPD
	Windows	EWLMMSSWinRPD

a. All files are located under the <CDROM mount point>/FILES/EWLM directory (for example, d:\FILES\EWLM).

Each compressed file has the structure shown in Figure 2-16.

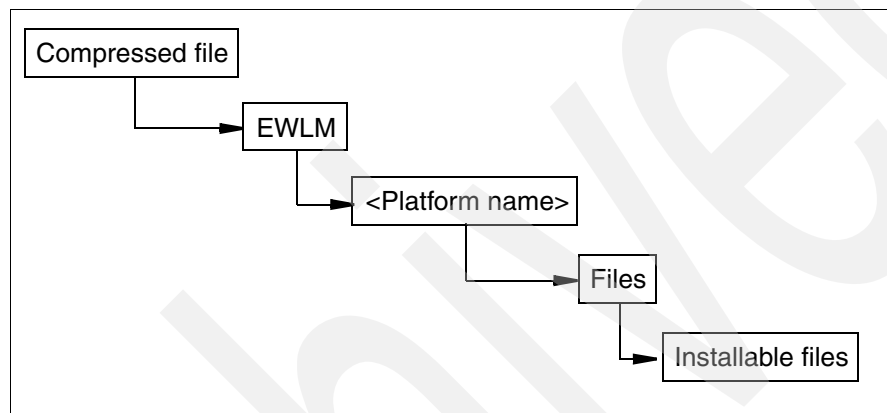


Figure 2-16 Compressed file structure

It is a good idea to uncompress the file in the domain manager before transferring it to the managed servers. In the end, you will have three or four files to transfer to your managed server systems, one of them being the installation wizard. Transfer the files from the domain manager into a temporary directory on the managed server system using, for example, the **ftp** command. Now we are ready to install the managed server code.

Important note for managed server for Linux platforms: The following steps install only a stub because the EWLM V2.1 release did not include Linux support. Linux support is included in Fixpack 30, which we soon cover. Fixpack 30 requires the stub, so be sure to do the following steps before applying Fixpack 30.

1. Go to the temporary directory where the installable files were transferred and execute the **EWLMMS<platform>**, where <platform> is a string to identify the platform, for example, AIX. This file can be a .bin or .exe file depending on your platform. In this example we install managed server code on a Windows platform, so we execute the EWLMMSSWin.exe. The Welcome screen appears. Click **Next** to continue.

Tip: The wizard needs a graphical environment in order to run. If you are installing the code in a UNIX platform using a **telnet** session, for example, remember to set the DISPLAY variable and also to enable an X server environment on your workstation. Installation using console commands is also supported.

2. Now you select the destination directory for the installation. You can either accept the default values, change the directory by typing a new path, or click **Browse** and select another destination directory. Click **Next** to continue.
3. The wizard asks for the managed server information directory. In this directory the managed server information and logs are stored. You can either accept the defaults value, change the directory by typing a new path, or click **Browse** and select another destination directory. Click **Next** to continue.
4. Now select the components to install. Check the Managed Server box and if you will use Firewall Broker to communicate with the domain manager, also check the Firewall Broker box. Figure 2-17 shows the features selection screen. Click **Next** to continue.

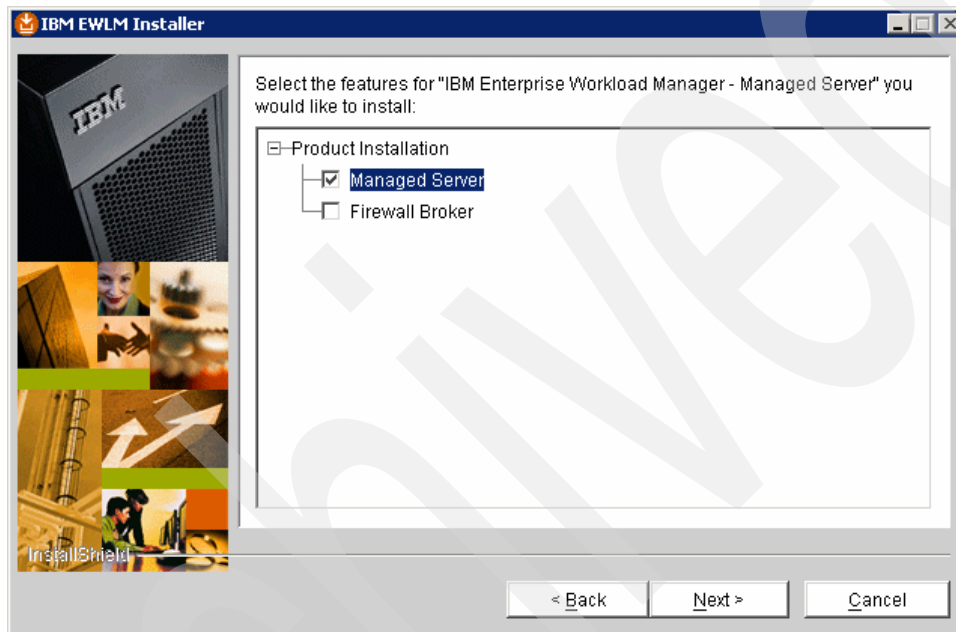


Figure 2-17 Features selection screen

5. The wizard now gives you a summary of the installation. Click **Next** to continue.
6. The installation begins. When it finishes, the wizard asks you to click **Next**. There will be a wait while it performs some final steps.

7. Scroll down the final screen so you see the selection to start the configuration wizard, as in Figure 2-18.

Note: If you are installing a Managed Server on a Linux platform you must select **Later** because we need to first apply Fixpack 30 before configuring the managed server. This is described in the next section.

For all other platforms you should select **Now** to start the managed server configuration.

Click **Next** to continue.

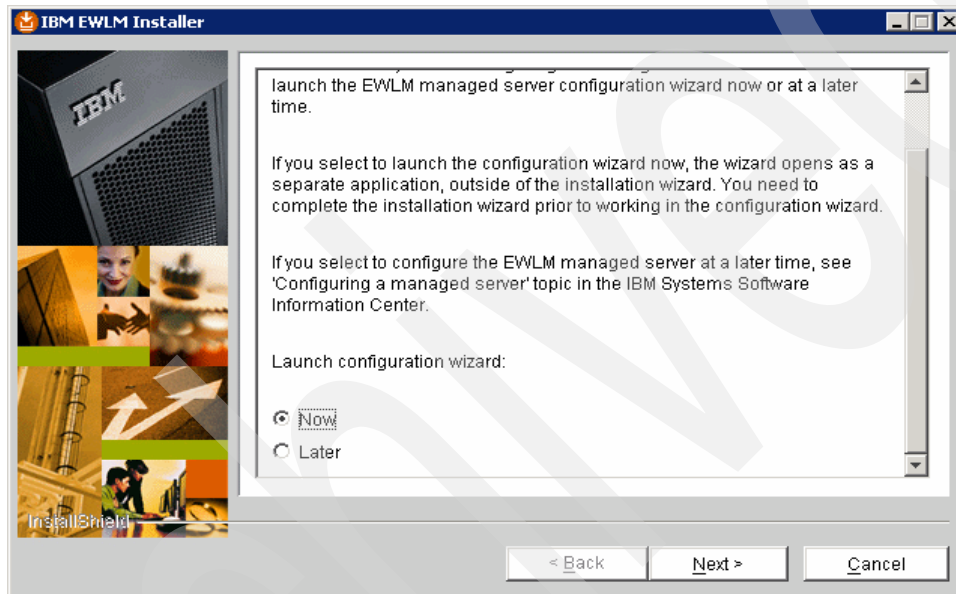


Figure 2-18 Final managed server installation screen

For all platforms except Linux, please proceed to 2.6.2, “Configuring EWLM managed server” on page 46, for instructions on managed server configuration.

Installing Fixpack 30 for EWLM managed server for Linux

Support for managed servers on the Linux platform is included in Fixpack 30. The fixpack requires that you have first performed the common installation steps described in 2.6.1, “Installing EWLM managed server code” on page 42, which installs a stub.

The following is a list of tasks we performed to install Fixpack 30:

1. If you have not already done so, perform the steps in 2.6.1, “Installing EWLM managed server code” on page 42. This provides the required stub for Linux managed servers.
2. Unjar UK12315.jar into a temporary directory.
3. Run the Fixpack 30 executable, FILES/EWLMMSLinPPC.bin. You may need to make this file executable with the command:

```
chmod u+x EWLMMSLinPPC.bin)
```
4. Click **Next** on the Welcome screen.
5. Click **Next** on the screen prompting you to ensure that all active EWLM processes are ended.

- The wizard will display the installation location of the existing managed server (the stub), as shown in Figure 2-19. Click **Next**. The fixpack will now overwrite the stub.

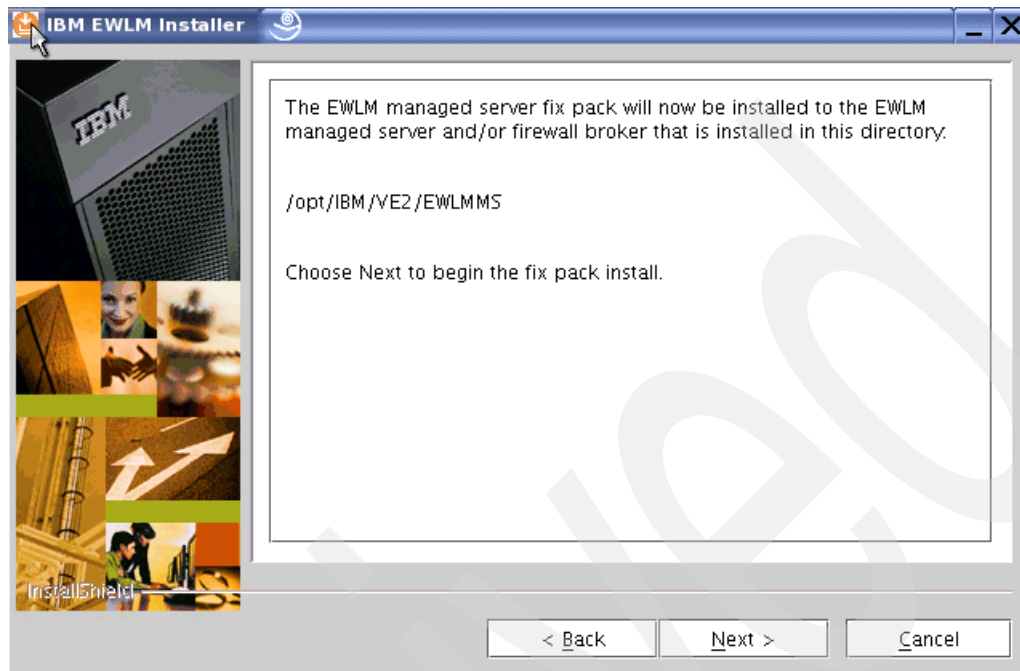


Figure 2-19 Fixpack installation dialog

- Click **Next** when you are prompted to do the EWLM post-install system setup.
- Click **Finish** on the final screen.

Now that Fixpack 30 is installed you can follow the steps in 2.6.2, “Configuring EWLM managed server” on page 46, to configure your managed server on the Linux platform.

2.6.2 Configuring EWLM managed server

Note: If you are configuring the EWLM managed server on the Linux platform, ensure that you have installed Fixpack 30 as described in the previous section.

With the managed server code installed, we can configure it to access the domain manager. If you just finished the managed server code installation and selected Now, you have the screen shown in Figure 2-20 on page 47 to start the configuration. If you did not select Now, go to <EWLM managed server install directory>/bin and execute the **configWizardMS** command. This is a .bat file for Windows and a .sh file for UNIX platforms.

Table 2-9 shows a list of platforms that support the configuration wizard along with the default installation path for the command.

Table 2-9 Platforms for which the managed server configuration Wizard is supported

Platform	configWizardMS	Default installation path
AIX	X	DM - /opt/IBM/VE2/EWLM/bin MS - /opt/IBM/VE2/EWLMMS/bin
HP-UX	X	MS - /opt/IBM/VE2/EWLMMS/bin

Platform	configWizardMS	Default installation path
i5/OS	Not supported	
Linux	X	DM - /opt/IBM/VE2/EWLM/bin MS - /opt/IBM/VE2/EWLMMS/bin
Solaris	X	MS - /opt/IBM/VE2/EWLMMS/bin
Windows	X	DM - C:\Program Files\IBM\VE2\EWLM\bin MS - C:\Program Files\IBM\VE2\EWLMMS\bin
z/OS	Not supported	

Tip: The wizard needs a graphical environment in order to run. If you are configuring a UNIX platform using a **telnet** session, for example, remember to set the DISPLAY variable and also to enable an X server environment on your workstation. Configuration using console commands is also supported.

Follow these steps to configure the managed server:

1. As shown in Figure 2-20, select **Create managed server** and click **Next** to continue.



Figure 2-20 Configuration wizard for managed servers

2. You provide the IP address and port for communication between the managed server and the domain manager. This port is the same port you specified as the Domain manager port in step 3 in 2.5.2, “Configuring the domain manager” on page 37. If there is a Firewall Broker in your environment, check the Firewall Broker box and enter the necessary information.
3. Select the appropriate level of security for your environment and click **Next**.

4. If there is a proxy server between the managed server and the domain manager, check the box "Use proxy server to connect to domain manager," enter the appropriate information, and click **Finish**. Otherwise, just click **Finish** to continue. Figure 2-21 shows the final installation screen. Click **OK** and then click **Home** to go back to the initial configuration screen, and click **Close** to exit the wizard.

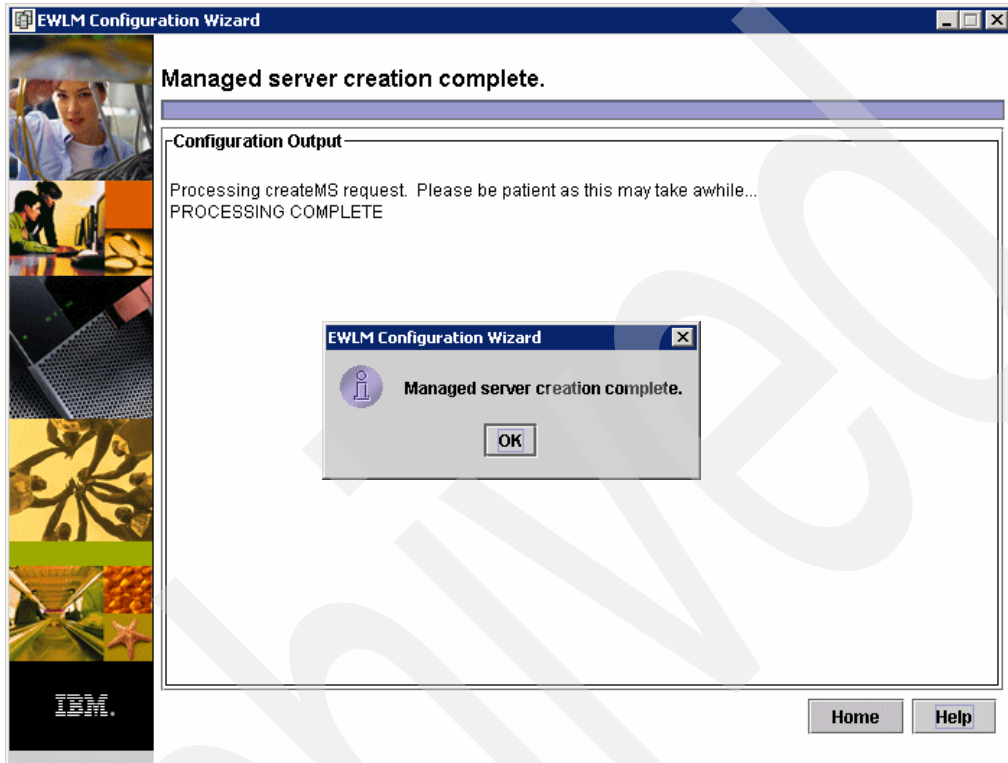


Figure 2-21 Final configuration screen of the managed server

Directory structure for managed server

In this section we present the major directories within the managed server installation. This structure is different for Windows than for the other platforms. First let us take a look at the Windows structure in Figure 2-22.

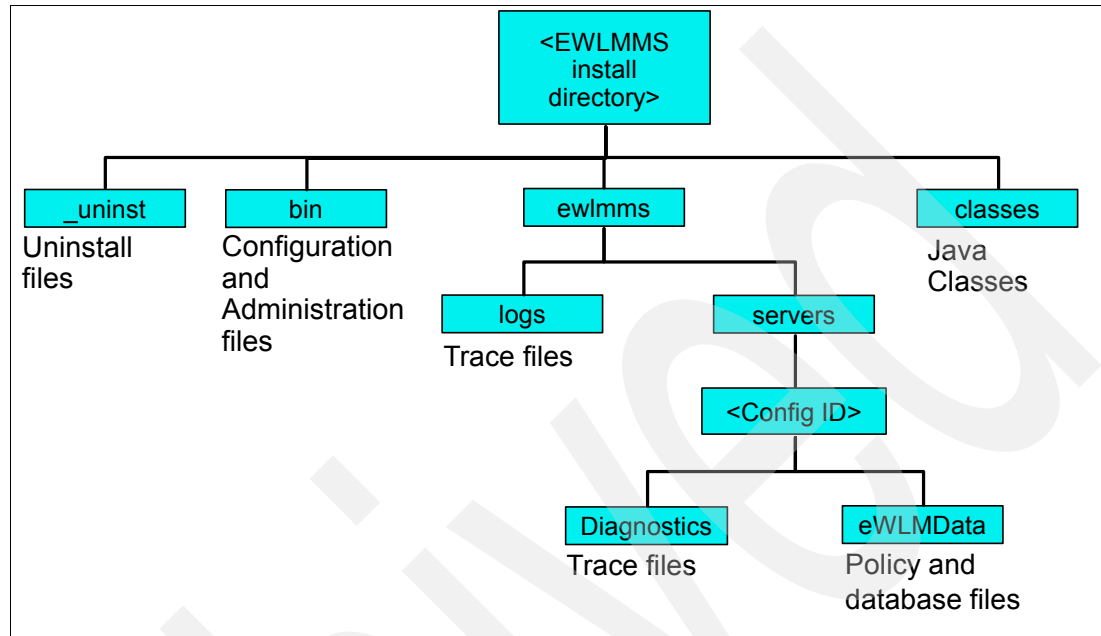


Figure 2-22 Managed server directory structure in a Windows platform

The directory structure for a UNIX platform like AIX and Linux is shown in Figure 2-23.

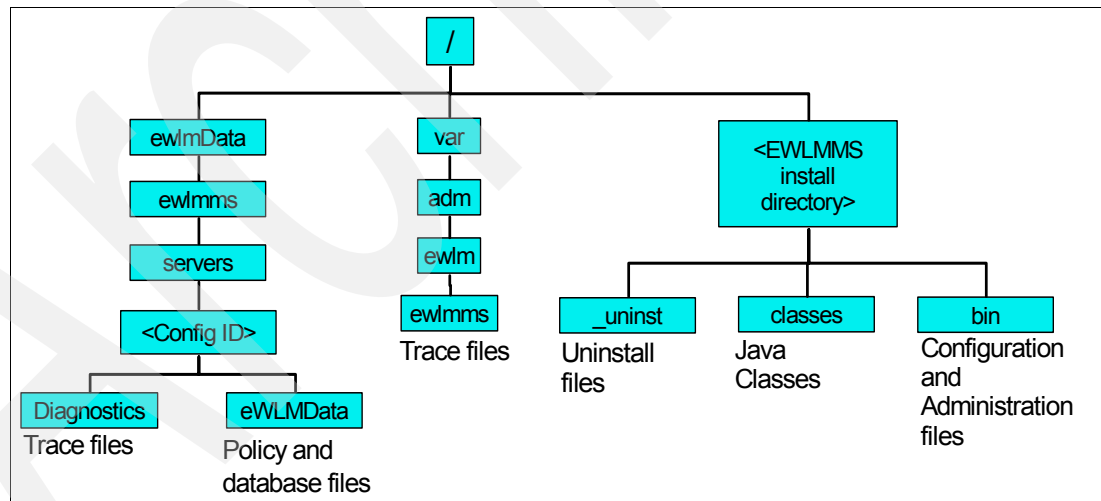


Figure 2-23 Managed server directory structure in a UNIX platform

Now all major components of the EWLM environment are installed and configured. Please proceed to 2.7, “Checking the installation” on page 50, for more information about post-installation issues.

2.7 Checking the installation

We now have all the EWLM components installed; however, depending on the platform, there are some post-installation procedures you might want to perform. Basically, we check if all services for the EWLM domain manager and managed servers are up and running.

If you are migrating from EWLM R1, please refer to 2.9.3, “Importing a Domain Policy” on page 74 for instructions on how to import the previous saved Domain Policy.

For UNIX platforms, the services do not start automatically when the system is rebooted, and so to start it automatically, we suggest that you create a shell script and configure the init process to start it at boot time.

Let us take a look at these post-installation procedures for each platform.

2.7.1 Post-installation steps for Windows

The Windows environment requires a reboot after it finishes the EWLM domain manager or managed server installation. After the reboot, click **Start** → **Administrative Tools** → **Services** and you should see the screen shown in Figure 2-24.

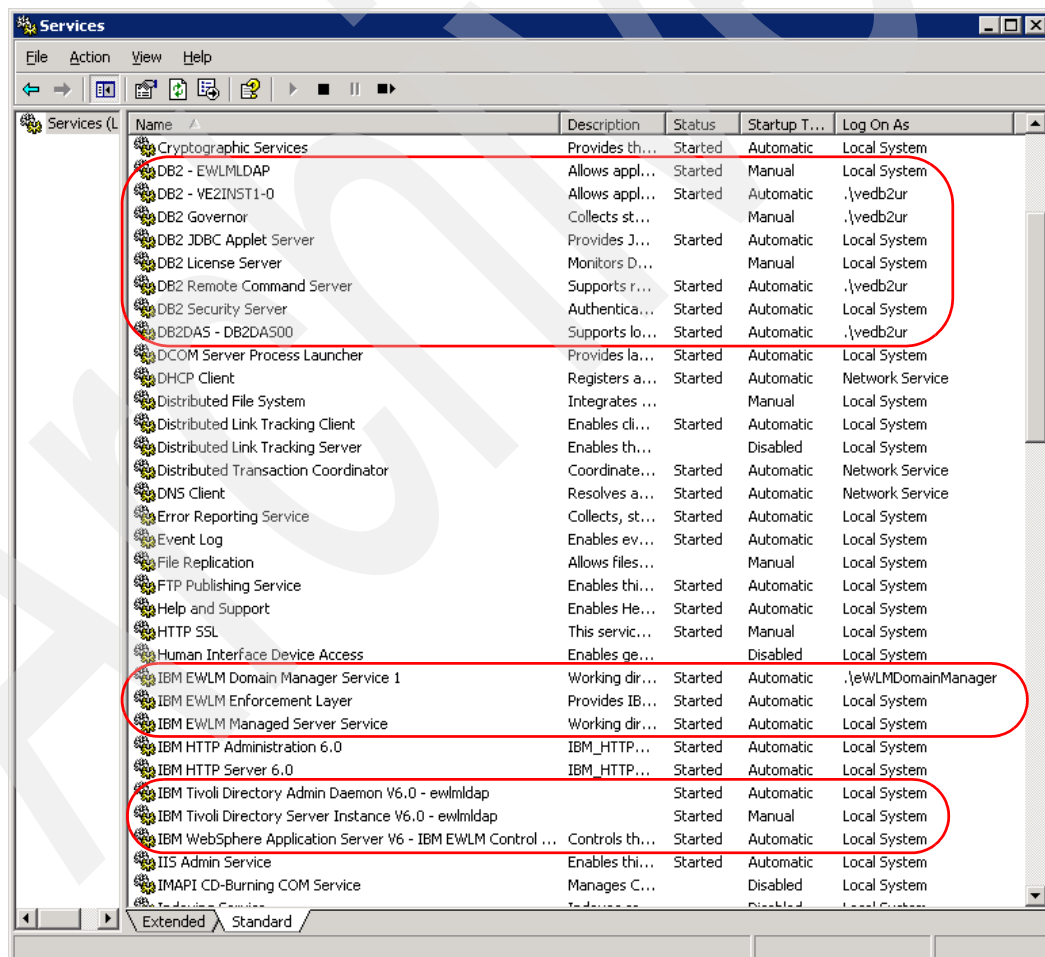


Figure 2-24 EWLM services for Windows

As you can see, inside the rounded rectangles in Figure 2-24 there are multiple services in Windows that are related to EWLM and Virtualization Engine. The DB2 and ITDS are

services related to Virtualization Engine and must be running during the installation of the EWLM domain manager code. Once the EWLM domain manager code is installed, and if you are not using the Virtualization Engine for any other purposes, they need not be running.

For the EWLM domain manager, the services that must be running are:

- ▶ IBM EWLM Domain Manager Service 1
- ▶ IBM EWLM Enforcement Layer
- ▶ IBM WebSphere Application Server V6 - IBM EWLM Control Center - <Config ID>

For the EWLM managed server, the service that must be running is the IBM EWLM Managed Server Service.

If one of the services for EWLM is not running, you can try to start by right-clicking the service and selecting **Start**, or you can use the EWLM commands.

For the domain manager, the commands are under the <EWLM install directory>\bin directory. Within this directory are the start and stop commands for both WebSphere and the domain manager. For example, to start the domain manager open a command prompt by clicking **Start** → **All Programs** → **Accessories** → **Command Prompt** and type:

```
C:\Documents and Settings\ewlm1> cd c:\program files\ibm\ve2\ewlm\bin
C:\program files\ibm\ve2\ewlm\bin> startDM DM1
    where DM1 is the ConfigID
```

To start WebSphere and the Control Center open a command prompt by clicking **Start** → **All Programs** → **Accessories** → **Command Prompt** and type:

```
C:\Documents and Settings\ewlm1> cd c:\program files\ibm\ve2\ewlm\bin
C:\program files\ibm\ve2\ewlm\bin> startWAS DM1 -adminUser wasadmin -adminPW itso
    where DM1 is the ConfigID
    adminUser is the owner of the WebSphere
    adminPW is the password for the owner of the WebSphere
```

If you experience problems starting any of the services, please see Chapter 10, “EWLM traces and logs” on page 249.

2.7.2 Post-installation steps for AIX, Linux, Solaris, HP-UX, and i5/OS

For AIX, Linux, Solaris, HP-UX, and i5/OS operating systems, the EWLM managed server service does not start automatically when the system reboots. Here are two ways we configured our AIX and Linux machines to automatically start and stop the managed server process.

Configuring AIX to automatically start and stop the managed server

For our managed server running AIX we created the shell script in Example 2-1.

Example 2-1 Start script for the managed server

```
#!/bin/sh
#####
# Script to start the managed server
#####

ID=AIXMS          ### This is the ConfigID parameter
DIR=/opt/IBM/VE2/EWLMS  ### This is the install directory
EXE_DIR=$DIR/bin

#####

$EXE_DIR/startMS.sh $ID
```

Change the ID variable to reflect your ConfigID and also the DIR variable to point to your EWLM managed server code installation directory.

Remember to give the script execute permission as well as to add the script in your operating system specific initialization control file, such as the /etc/inittab for AIX and the CL program for i5/OS.

Configuring Linux to automatically start and stop the managed server

Although you can directly edit the /etc/inittab to specify services that should be started for a particular run level, Linux follows a different convention. This is how we configured the managed servers running SuSe Linux in the ITSO environment.

Each file in /etc/rc.d is a shell script for managing a service that accepts three parameters: start, stop, and restart. These scripts are used by the init process (or the administrator) to start, stop, and restart a service. When a runlevel is booted, these scripts will be called by init via a special symbolic link, which we will explain shortly. Following this convention, we wrote the script shown in Example 2-2. You will need to update the CONFIG_ID variable to the one you used. We saved the file in /etc/rc.d with file name ewlm_man_server.

Example 2-2 Shell script with file name /etc/rc.d/ewlm_man_server

```
#!/bin/sh

CONFIG_ID=elinux2          ### ConfigID of managed server
EWLM_HOME=/opt/IBM/VE2/EWLMS  ### Install dir
EWLM_BIN=$EWLM_HOME/bin    ### Location of startMS.sh & stopMS.sh

start()
{
    $EWLM_BIN/startMS.sh $CONFIG_ID
}

stop()
{
    $EWLM_BIN/stopMS.sh $CONFIG_ID
}

case "$1" in
    start)
        start
```

```

        ;;
stop)
    stop
    ;;
restart|reload)
    stop
    start
    ;;
*)
    /bin/echo $"Usage: $0 {start|stop|restart}"
    exit 1
esac

```

Linux contains a special directory for each runlevel. For example, the directory for runlevel 5 is `/etc/rc.d/rc5.d`. If a service should be started in a runlevel, then two symbolic links pointing to a script in `/etc/rc.d` are created: one for starting the service and the other for stopping the service. When a runlevel is booted, Init runs all symbolic links that start with S (start) and passes the start parameter. Similarly, links that start with K (kill) are called with the parameter stop when the runlevel is exited, such as during a shutdown.

You will also notice a number after S and K. The number specifies the order in which the symbolic links will be run. Lower numbers are run first. This allows services that depend on other services to be started after the dependant services.

The ITS0 Linux servers are configured for runlevel 5; therefore, in `/etc/rc.d/rc5.d`, we created two symbolic links with the commands shown in Example 2-3. Example 2-4 shows a directory listing of the symbolic links.

Important: The number in this S symbolic link should be greater than the number used in the S symbolic link for `ewlm`. This is because `ewlm` is required to be running before the managed server is started. If this is not the case, you will receive errors on the console, and the managed server will fail to start. For example:

```

S16ewlm -> ../ewlm
S99ewlm_man_server -> ../ewlm_man_server

```

Similarly, the managed server must be stopped before `ewlm`, or you will get errors on the console. For example:

```

K05ewlm_man_server -> ../ewlm_man_server
K06ewlm -> ../ewlm

```

Example 2-3 Commands used to create start and stop symbolic links in `/etc/rc.d/rc5.d`

```

ln -s ../ewlm_man_server ./S99ewlm_man_server
ln -s ../ewlm_man_server ./K05ewlm_man_server

```

Example 2-4 Directory listing of our symbolic links in `/etc/rc.d/rc5.d` pointing to our `ewlm_man_server` script

```

elinux2:/etc/rc.d/rc5.d # ls -l *ewlm_man_server*
lrwxrwxrwx 1 root root 18 May 25 15:41 K05ewlm_man_server -> ../ewlm_man_server
lrwxrwxrwx 1 root root 18 May 25 15:38 S99ewlm_man_server -> ../ewlm_man_server

```

Now when the system is stopped, the managed server will be brought down gracefully. When the system is restarted and runlevel 5 is entered, the managed server will be automatically started.

If you experience problems starting the managed server, please see Chapter 10, “EWLM traces and logs” on page 249.

Configuring the domain manager

For AIX, Linux, and i5/OS operating systems, the EWLM domain manager and Control Center do not start automatically when the system reboots. For our ITSO environment, we created a shell script to start both services automatically at system reboot. See Example 2-5.

Example 2-5 Start script for the domain manager and Control Center

```
#!/bin/sh
#####
# Script to start the domain manager
#####

ID=AIXDM          ### This is the ConfigID parameter
DIR=/opt/IBM/VE2/EWLM  ### This is the install directory
EXE_DIR=$DIR/bin

#####

$EXE_DIR/startWAS.sh $ID
$EXE_DIR/startDM.sh $ID
```

Remember to give the script execute permission as well as add the script in your operating system specific initialization control file, like the /etc/inittab for AIX and the CL program for i5/OS.

If you experience problems starting any of the services, see Chapter 10, “EWLM traces and logs” on page 249.

2.7.3 Managing Control Center users

In this section we explain the roles for the control center users and how to create and delete them. As part of the installation, you must add an Administrator user for the Control Center.

All control center user IDs must first be created and have a valid password at the operating system level of the domain manager. There are no special requirements needed for these user IDs.

Before we start creating the user IDs, let us take a look at the roles for the control center users.

User roles

The EWLM Control Center is accessed using one of three roles. These roles are Administrator, Operator, and Monitor. Table 2-10 shows the roles and their privileges.

Table 2-10 User interface authority guidelines

Authority	To whom	General guideline
Administrator	System administrators Performance analysts	Grant this authority to individuals directly responsible for the performance of the environment and who make changes that affect performance. Everyone else should be granted lesser authority. Although it can be, it is recommended that this not be root.
Operator	Operations staff	Grant this authority to individuals who are responsible for managing all managed servers in a domain and who will be deploying and activating policies.
Monitor	Application development Project lead	Grant this authority to individuals who only have a need to monitor. This may include but is not limited to those listed here.

Administrator

The administrator of EWLM creates all the components inclusive of the domain and service policies for the domain represented by the domain manager. This individual is responsible for configuring all the components of EWLM to attain the service goals of the enterprise. Administrators have the authority to deploy the Domain Policy and activate the service policy. The authority for both Operator and Monitor roles is included within Administrator.

Operator

The Operator function within EWLM has the capability to view and activate service policies and to view and control managed servers from the EWLM perspective. An enterprise may employ several service policies to manage different service goals during off peak or non-prime hours. The operator would use the activate service policy function to alternate between those different service policies. The operator has monitor capabilities as well.

Monitor

The Monitor function within EWLM has read-only capability and can display most of the components within the domain represented by the domain manager.

Adding and deleting Control Center users

Now that we know the roles and their privileges, we can create the user IDs.

Attention: Remember to create the user and give a valid password in the operating system level before proceeding with the creation of the user in the control center.

Users can be added and deleted from the control center by either the command line, using the `changeCC` command, or by the configuration wizard.

To add a user using the command line, use the **changeCC** command as in Example 2-6, where we add a user called *ewlmoper* into the *Operator* role. This command is in the <EWLM install directory>/bin and is a .bat file for Windows and a .sh file for UNIX platforms.

Example 2-6 Adding a user with changeCC command

```
# cd /opt/IBM/VE2/EWLM/bin
# ./changeCC.sh -addUser AIXDM -adminUser wasadmin -adminPW itso -roleUser ewlmoper -role Operator
Processing changeCC -addUser request. Please be patient as this may take awhile...
...processing 33% complete
...processing 66% complete
PROCESSING COMPLETE
```

Example 2-7 shows the **changeCC** command to delete a user.

Example 2-7 Removing a user with changeCC command

```
# cd /opt/IBM/VE2/EWLM/bin
# ./changeCC.sh -removeUser AIXDM -adminUser wasadmin -adminPW itso -roleUser ewlmoper -role Operator>
Processing changeCC -removeUser request. Please be patient as this may take awhile...
...processing 33% complete
...processing 66% complete
PROCESSING COMPLETE
```

Optionally, the wizard can be used to add or delete users. It is started by the **configWizardDM** command and the default path of this command is shown in Table 2-11. If you installed the EWLM in a different path, then locate the bin directory under your installation path.

Table 2-11 Platforms on which configuration Wizard is supported

Platform	configWizardDM	Default installation path
AIX	X	/opt/IBM/VE2/EWLM/bin
i5/OS	Not supported	
Linux	X	/opt/IBM/VE2/EWLM/bin
Windows	X	C:\Program Files\IBM\VE2\EWLM\bin
z/OS	Not supported	

To start the wizard go to <EWLM installation directory>/bin and run the `configWizardDM` command. It is a .bat file for Windows and a .sh file for UNIX platforms. Follow the steps to add or delete a user with the wizard.

Tip: The wizard needs a graphical environment in order to run. If you are running it in a UNIX platform using a `telnet` session, for example, remember to set the `DISPLAY` variable and also to enable an X server environment on your workstation.

1. In the first screen, select **Add or remove users from Control Center** and click **Next**, as you can see in Figure 2-25.

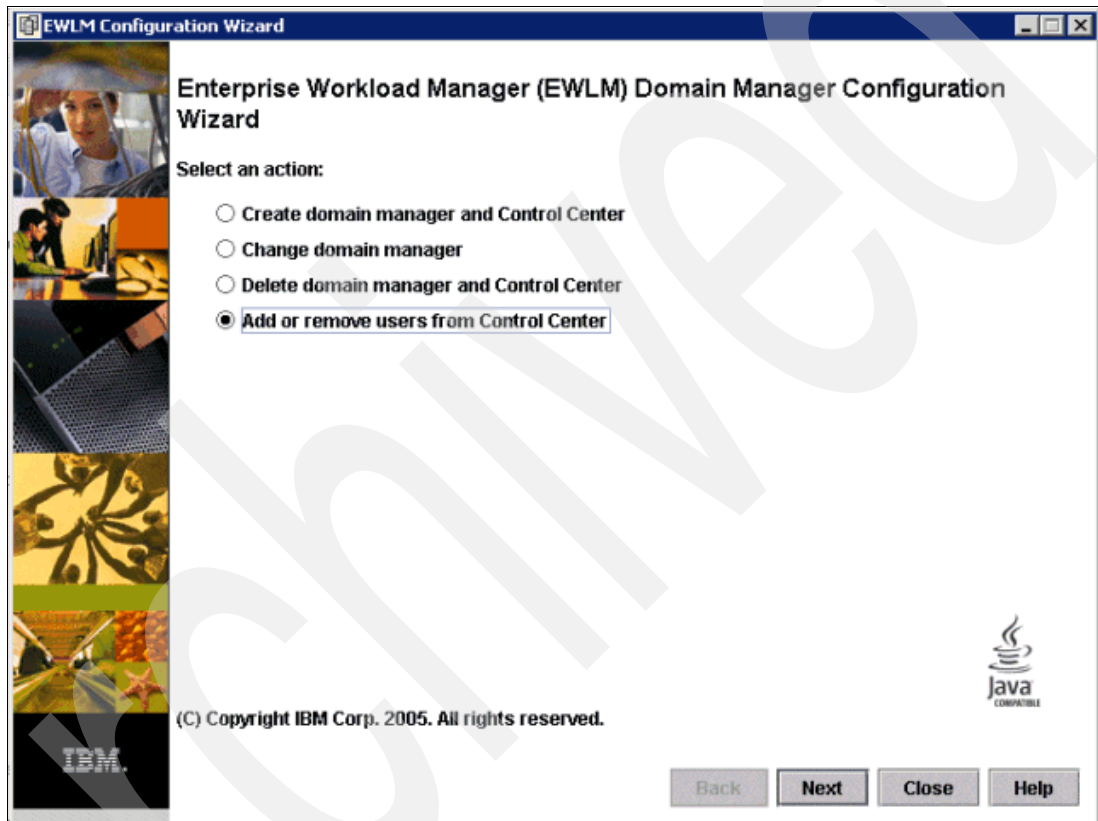


Figure 2-25 Create users with EWLM Configuration Wizard

2. Select the correct ConfigID for your environment and click **Next**.
3. Enter the WebSphere administrator user ID and its password twice and click **Next**.
4. The wizard prompts you to retrieve the information about existing control center users. If you are removing an existing user, then select **Yes**; otherwise, if you are only adding a new user you may select **No**. Figure 2-26 shows this screen.

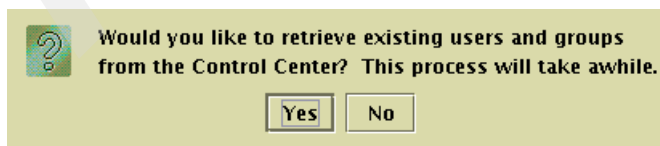


Figure 2-26 Retrieve users screen

5. We chose **Yes**, and after a wait while the wizard performed the retrieval task the wizard presented the screen in Figure 2-27.

To add a user, enter the user ID in the appropriate role in the left box of the screen and click **Add**. The user ID appears on the box in the right. Repeat it until you add all users.

To delete a user, select the user ID and click **Remove**.

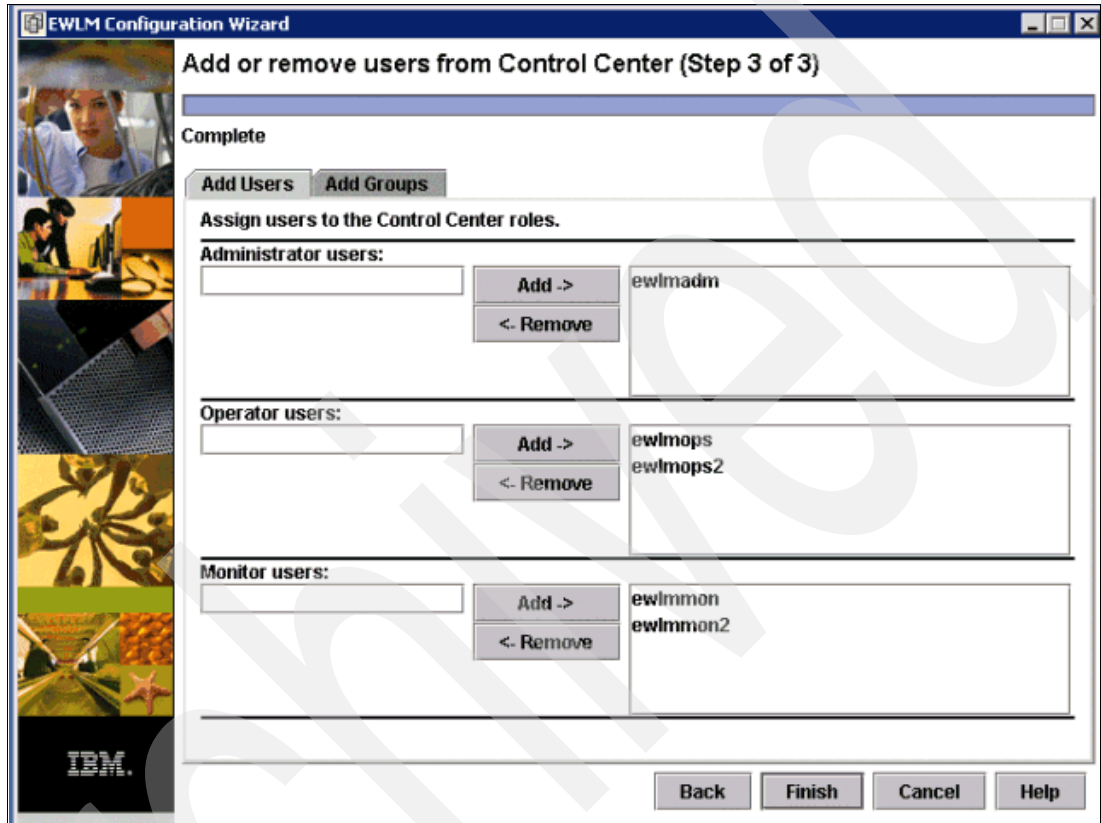


Figure 2-27 Add or remove user and group types with the EWLM configuration wizard

Click **OK** at the success screen, then click **Home** to get back to the initial wizard screen, and click **Finish** to exit.

2.8 Installing and configuring EWLM on z/OS

EWLM for z/OS is distributed as a z/OS priced feature at no additional cost. EWLM for z/OS consists of the following FMIDs:

- ▶ HVE1210, IBM Virtualization Engine Enterprise Workload Manager for z/OS
- ▶ HJVA140, IBM SDK for z/OS, Java 2 Technology Edition, V1.4
- ▶ HDM1210, IBM Virtualization Engine Domain Manager for z/OS
- ▶ HVW1210, IBM Virtualization Engine WebSphere for z/OS
- ▶ HCJ1210, IBM Virtualization Engine Runtime Support for ICU4J/iLog Jviews
- ▶ HGC1210, IBM Virtualization Engine Runtime Support for GCR Toolkit/CMR

The EWLM product can be ordered as CBPDO or ServerPac.

2.8.1 Preventive service planning

Before installing EWLM for z/OS, you should review the current Preventive Service Planning (PSP) information. If you obtained EWLM for z/OS as part of a CBPDO, there is HOLDDATA and PSP information included on the CBPDO.

Table 2-12 lists the UPGRADE and SUBSET values to be used to locate the EWLM for z/OS PSP.

Table 2-12 PSP UPGRADE and SUBSET ID

UPGRADE	SUBSET	Description
VEV2R1	EWLMMMS	EWLM for z/OS
VEV2R1	EWLMDM	EWLM Domain Manager for z/OS
VEV2R1	EWLMWAS	EWLM WebSphere for z/OS
VEV2R1	EWLMRT1	EWLM Runtime for ICU4J and iLOG Jviews
VEV2R1	EWLMRT2	EWLM Runtime for GCR Toolkit and CMR
JAVAOS390	HJVA140	IBM SDK for z/OS V1.4

2.8.2 Installation considerations

The following sections identify the system requirements for installing and activating EWLM for z/OS. The target environment can run on any hardware that supports the software listed in Table 2-13. This table describes the software requirements for running the EWLM code.

Table 2-13 Software prerequisites

Program number	Product name and minimum service level
5694-A01	z/OS V1.6 or later
5655-G52	z/OS.e V1.6 or later
5655-I56	IBM SDK for z/OS, Java 2 Technology Edition, V1.4 (included in this product) at PTF level UQ90449 or later
5694-A01	z/OS V1.6 or later: APAR OA12005 and UA15456 (USS)
5655-G52	z/OS V1.6 or later: APAR OA12005 and UA15456 (USS)

SMP/E is required to install the EWLM for z/OS product. Please refer to the product program directory to verify the required DASD space for the distribution and target libraries and for the SMP/E installation jobs.

We recommend that you install EWLM for z/OS in the same SMP/E zones as your z/OS or z/OS.e (V1.6 or later) system, including SMPCSI, target, and distribution data sets (except for the HFS data sets).

As a result of the SMP/E installation, the following libraries will be updated or allocated if not existing: PROCLIB, SAJVSMP1, SBBWCOMB, SAGCEXEC, SAIJEXEC, SBBWEXEC, SBDMEEXEC, SHVEXEC, SAGCSAMP, SAIJSAMP, SBBWSAMP, SBDMSAMP, and SHVESAMP.

Table 2-14 lists the HFS files with EWLM executable code and their mount points.

Table 2-14 HFS file and mount points

HFS file	Mount point
SAGCHFS	/usr/lpp/VE_R2/VE_LIB
SAJVJ14	/usr/lpp/java/J1.4/IBM
SBBWCHFS	/usr/lpp/VE_R2/VEWAS
SBDMCHFS	/usr/lpp/VE_R2/EWLM
SHVECHFS	/usr/lpp/VE_R2/EWLMMS

Note: Refer to the EWLM product program directory for instructions on the SMP/E installation steps.

You might want to consider the following during your installation:

- ▶ The IBM Software Development Kit for z/OS Java 2 Technology Edition Version 1.4 (program number 5655-I56), which is included as part of EWLM for z/OS, is a functional successor to various Java products from IBM. If you have previously installed this product, you should take the following considerations into account when installing EWLM for z/OS.
- ▶ If you already have a copy of IBM SDK for z/OS, Java 2 Technology Edition, V1.4 with PTF UQ90449 or later installed, you can continue to use the existing version or install the IBM SDK for z/OS, Java 2 Technology Edition, V1.4 shipped with EWLM for z/OS and replace your existing IBM SDK for z/OS, Java 2 Technology Edition, V1.4 with it. The SDK version provided with EWLM incorporates all the required maintenance levels.
- ▶ Verify or update the BPXPRMxx member in PARMLIB as in the following example:


```
FILESYSTYPE TYPE(UDS) ENTRYPPOINT(BPXТУINT) NETWORK DOMAINNAME(AF_UNIX) DOMAINNUMBER(1)
MAXSOCKETS(10000) TYPE(UDS)
```
- ▶ The SBBWCOMB library requires being APF authorized. You can dynamically update your APF list with the SET PROG command and update the PROGxx Parmlib member to preserve the change across IPLs. Be sure you also put the SBBWCOMB library in LNKLIST.
- ▶ To perform the EWLM code installation, you have to execute all of the installation steps from a user ID that is defined to z/OS UNIX System Services, and that has the following attributes:
 - UID(0) or READ access or higher to BPX.SUPERUSER in the FACILITY class
 - READ access or higher to BPX.FILEATTR.PROGCTL, BPX.FILEATTR.APF, and BPX.FILEATTR.SHARELIB in the FACILITY class
- ▶ You should also plan for the following user IDs that will be used to install the EWLM domain manager and managed server:
 - User IDs and GROUP for the WebSphere Application Server administrator. This user ID requires a TSO and OMVS segment.
 - User ID to start the WebSphere Application Server - The customization script does not explicitly require this user ID. When not specified, the WebSphere Application Server administrator user ID will be used, although we do not recommend sharing the same IDs. This user ID requires an OMVS segment.

- User ID to start the WebSphere Application Server servants - The customization script does not explicitly require this user ID. When not specified, the WebSphere Application Server administrator user ID will be used, although we do not recommend sharing the same IDs. This user ID requires an OMVS segment.
- User IDs to start the managed server and domain manager - We recommend that these user IDs have the same group ID (GID) as the WebSphere Application Server IDs. These user IDs require an OMVS segment.
- Control Center user IDs - These IDs should only require the OMVS segment.

2.8.3 Configuring domain manager

The following is the list of tasks that you need to perform to configure the domain manager.

Update your configuration file

Copy <EWLM_ROOT>/samples/ewlmdm_environment.conf to /etc, where <EWLM_ROOT> is the absolute path to where the EWLM code is installed. In our case, this is /usr/lpp/VE_R2/EWLM, for example, cp ewlmdm_environment.conf /etc.

Verify and update the following parameters in the copied version in /etc:

- ▶ Verify that WEBSPHERE_ROOT=/usr/lpp/VE_R2/VEWAS is consistent with your installation. This should point to the absolute path where the EWLM WebSphere is installed.
- ▶ Verify that EWLM_ROOT=/usr/lpp/VE_R2/EWLM is consistent with your installation. This should point to the absolute path where the EWLM code is installed.
- ▶ Update EWLM_DATA_ROOT= with the absolute path of where you would like EWLM to create the working directories.
- ▶ Update JREBIN_ROOT= with the absolute path of the bin directory of your JRE. In our case this is /usr/lpp/java14/J1.4/bin/.
- ▶ Verify that LOGFILE_ROOT=/var/adm/ewlm/ewlmdm is consistent with your installation. This should point to the absolute path where the EWLM code should create the log files.

Example 2-8 ITSO domain manager configuration file

```
WEBSPHERE_ROOT=/usr/lpp/VE_R2/VEWAS/VEWAS_code/
EWLM_ROOT=/usr/lpp/VE_R2/EWLM
EWLM_DATA_ROOT=/u/ew1m2u
JREBIN_ROOT=/usr/lpp/java/J1.4/bin
LOGFILE_ROOT=/u/ew1m2u/logs
```

configWAS.sh

This script is used to create the list of parameters describing the domain manager configuration that will be used as input in the following createWAS step.

In /usr/lpp/VE_R2/VEWAS/scripts, you can launch the script with the command **./configWAS.sh <config.cfg>**, where config.cfg is the output file created by the script with all the requested information. Below are the parameters you will be prompted with:

```
WAS_VE_CODE_HFS → HFS dsname with the VEWAS code
WAS_VE_CODE_DIRECTORY → mount point for the HFS VEWAS code
WAS_VE_CONFIG_HFS → HFS dsname with the WebSphere Application Server customization data
WAS_VE_CONFIG_DIRECTORY → mount point for the HFS customization data
WAS_VE_USERID → WebSphere Administration user ID
WAS_VE_UID → WebSphere Administration UID
```

WAS_VE_GROUP → WebSphere Administration group name
 WAS_VE_GID → WebSphere Administration GID
 WAS_VE_SYSTEM_NAME → System name where WebSphere Application Server will run
 WAS_VE_SYSPLEX_NAME → Sysplex Name
 WAS_VE_SYSTEM_IPNAME → Hostname corresponding to the System name
 WAS_VE_PORT_BASE → Prefix for the starting number for the set of port used by EWL
 WAS_VE_CELL_NAME → WebSphere Application Server cell name
 WAS_VE_PDSE_PREFIX → Full qualifier for the SBBWCOMB data set
 WAS_VE_PROCLIB → Proclib data set name
 WAS_VE_PROC_PREFIX → Prefix for EWL started tasks
 WAS_VE_VOLSER → Volser for the config data set
 WAS_VE_UNIT → Unit for the config data set

Consider the following:

- ▶ The port parameter defines the prefix for a block of 10 free TCP ports that will be used; for example, 19990 would use 19990–19999. Remember that port numbers are only valid up to 32 K.
- ▶ You cannot use directories that are accessed through symbolic links. You need to specify the absolute path for the directories to avoid having the createWAS script fail. In our configuration, the EWL WAS code is installed in /usr/lpp/VE_R2/VEWAS, but /usr/lpp is a symbolic link to /SC69/VEWAS2. In our file, we need to enter /SC69/VEWAS2.
- ▶ You should specify a user PROCLIB data set. Once the procs are created, they will require customization to your installation and then can be copied to your production PROCLIB. Besides this, most installations have the production PROCLIB data sets protected against updates.
- ▶ If your installation storage is SMS managed, you can avoid supplying a specific volser, unless you want to allocate the config HFS on a particular volume. In case you want SMS to direct your allocation, you still need to specify a meaningless value for this parameter, for example, XXXXXX. The same goes for the unit—if the data set is SMS managed, the unit will be ignored.

Example 2-9 shows a sample of the config file we used in our installation.

Example 2-9 Sample of config.cfg file

_WAS_VE_CODE_HFS_	EWL2.VEWAS.HFS
_WAS_VE_CODE_DIRECTORY_	/SC69/VE_R2/VEWAS2
_WAS_VE_CONFIG_HFS_	EWL2U.WAS.HFS
_WAS_VE_CONFIG_DIRECTORY_	/u/ewl2u/vewas
_WAS_VE_USERID_	EWLWAS
_WAS_VE_UID_	0
_WAS_VE_GROUP_	SYS1
_WAS_VE_GID_	0000000000
_WAS_VE_SYSTEM_NAME_	SC69
_WAS_VE_SYSPLEX_NAME_	WTSCPLX1
_WAS_VE_SYSTEM_IPNAME_	wtsc69.itso.ibm.com
_WAS_VE_PORT_BASE_	3220
_WAS_VE_CELL_NAME_	VEWAS
_WAS_VE_PDSE_PREFIX_	EWL2
_WAS_VE_PROCLIB_	BARI.PROCLIB
_WAS_VE_PROC_PREFIX_	EWLW
_WAS_VE_VOLSER_	TST024
_WAS_VE_UNIT_	3390
_WAS_VE_CONFIG_FORMAT_	1.2

installEWLM.sh

Important: This step is optional but highly recommended. Participation in the GCR and IMR is vital to future Virtualization Engine components on z/OS. If you choose not to run installEWLM, any domain managers created without the GCR will need to be taken offline, deleted, then recreated before they can use the GCR and IMR.

You need to run the following script to add this system to the Virtualization Engine Global Configuration Repository (GCR) and Identity Management Repository (IMR). To save registration entries for this system in the GCR and IMR, the domain manager uses an IBM Tivoli Directory Server (ITDS) LDAP server. ITDS does not run on z/OS, but may be installed on other systems. It can be installed with the Virtualization Engine installation wizard. The installEWLM script also creates a Virtualization Engine configuration file that the domain manager uses.

To include GCR, you need to specify the following in the `ewlmdm_environment.conf`:

`VE_BASE` → `VE_LIB` HFS path, i.e. `usr/lpp/VE_R2/VE_LIB`
`VE_LIB` → `VE_LIB` classes HFS path, i.e. `usr/lpp/VE_R2/VE_LIB/classes`

You need to run the `installEWLM` script. In `/usr/lpp/VE_R2/VE_LIB/scripts`, you can launch the script with the command:

```
installEWLM -GCRUser adminuser -GCRPW adminpw -GCRHost gcrLdapHostname -GCRPort gcrLdapPort  
-GCREnv gcrEnvironmentname -IMRUser adminuser -IMRPW adminpw -IMRHost gcrLdapHostname  
-IMRPort imrLdapPort -IMRDN imrDomainname
```

Where:

- ▶ `-GCRUser` is the Global Configuration Repository (GCR) LDAP server administrator ID.
- ▶ `-GCRPW` is the GCR LDAP server administrator password.
- ▶ `-GCRHost` is the GCR LDAP server host name.
- ▶ `-GCRPort` is the GCR LDAP server port number.
- ▶ `-GCREnv` is the GCR environment name to assign to this Virtualization Engine configuration. The name you specify here must match the GCR environment name specified for the other systems that are to share the GCR.
- ▶ `-IMRUser` is the Identity Management Repository (IMR) LDAP server administrator ID. If omitted, this parameter defaults to the value specified for `GCRUser`.
- ▶ `-IMRPW` is the IMR LDAP server administrator password. If omitted, this parameter defaults to the value specified for `GCRPW`.
- ▶ `-IMRHost` is the IMR LDAP server host name. If omitted, this parameter defaults to the value specified for `GCRHost`.
- ▶ `-IMRPort` is the IMR LDAP server port number. If omitted, this parameter defaults to the value specified for `GCRPort`.
- ▶ `-IMRDN` is the IMR EIM domain name. If omitted, this parameter defaults to IBM identity mapping repository. The name you specify here must match the EIM domain name specified for the other systems that are to share the IMR.

The `installEWLM` script creates configuration file `vecd` in directory `vebase/.veconfig`, where `vebase` is a path specified with the `VE_BASE` variable in the `ewlmdm_environment.conf` configuration file located in the `/etc` directory.

You can run this script more than once, but you must first delete the Virtualization Engine configuration file, `vecd`, before rerunning the script.

The log for the installEWLM script is in directory
ve_base/VE_DIAGNOSTICS/RegisterVE.date-time-stamp.

createWAS.sh

You must run the createWAS script to create the Virtualization Engine WebSphere Application Server profile. In /usr/lpp/VE_R2/VEWAS/scripts, you can launch the script with the command:

```
createWAS -cfg filename.cfg -sec path
```

Where:

- ▶ -cfg is the configuration file created by the configWAS script.
- ▶ -sec identifies the path of the generated shell script, commands.sh, that contains commands to define security. The default path is /misc/RACFMSTR under the directory specified through the script as the HFS root.

The possible values for -sec are None to specify that the shell script file should be created but not run or a <path> to specify to use an existing file indicated by path. If -sec is omitted, the commands.sh file is created and run.

For example, in our configuration, we issue:

```
./createWAS.sh -cfg config.cfg -sec None
```

Consider the following:

- ▶ We recommend that you run the script creating the file with the security definition. We do not recommend running such an exec without careful examination. In many installations, the RACF® commands would have to be entered separately by a RACF administrator.
- ▶ As suggested earlier, we recommend that you use two WebSphere user IDs: One, a WebSphere Application Server user ID to be used for WebSphere administration; the other, a user ID to start the WebSphere Application Server, which will be used for the WebSphere controller address space. The configWAS.sh script handles only one user ID and the exec creates only one user ID for WebSphere with a password that does not expire. We recommend creating an administrative user ID (the adminUser ID) with a password that expires according to local installation policies. We further recommend creating two additional WebSphere user IDs, one for the controller and one for the servant, that are protected (that is, nopassword and nooidcard). These user IDs need to be given in the STDATA parameters for the RACF STARTED profiles defined for the controller and servant address spaces. In order to allow WebSphere to function properly, these user IDs need to be connected to the RACF group to which the administrative user ID belongs.
- ▶ createWAS.sh allocates a configuration HFS and mounts it to the mount point specified in configWAS.sh. Remember to add this file and its mount point to the Parmlib BPXPRMxx member to preserve the mount through IPLs.
- ▶ As a result of this script execution, the configuration file addressed by the `_WAS_VE_CONFIG_DIRECTORY_` parameter will be populated by the WebSphere instance profile.
- ▶ Make sure you have RACF authorization to update the Proclib data set and that there are no outstanding ENQs against the Proclib while you are running the script. If ENQs are present, the script would fail with the following error messages:

```
createWAS: Inside copy_procs ...
cp: FSUM6258 cannot open file "///SYS1.PROCLIB(EWLMW)'"': EDC5061I An error occurred when
attempting to define a file to the system.
cp: FSUM6258 cannot open file "///SYS1.PROCLIB(EWLMWA)'"': EDC5061I An error occurred
when attempting to define a file to the system.
```



```

cp: FSUM6258 cannot open file "'SYS1.PROCLIB(EWLMWAZ)'" : EDC5061I An error occurred
when attempting to define a file to the system.
cp: FSUM6258 cannot open file "'SYS1.PROCLIB(EWLMWD)'" : EDC5061I An error occurred
when attempting to define a file to the system.
cp: FSUM6258 cannot open file "'SYS1.PROCLIB(EWLMWDZ)'" : EDC5061I An error occurred
when attempting to define a file to the system.
cp: FSUM6258 cannot open file "'SYS1.PROCLIB(EWLMW0)'" : EDC5061I An error occurred
when attempting to define a file to the system.
cp: FSUM6258 cannot open file "'SYS1.PROCLIB(EWLMWS)'" : EDC5061I An error occurred
when attempting to define a file to the system.
cp: FSUM6258 cannot open file "'SYS1.PROCLIB(EWLMWSZ)'" : EDC5061I An error occurred
when attempting to define a file to the system.
cp: FSUM6258 cannot open file "'SYS1.PROCLIB(EWLMWZ)'" : EDC5061I An error occurred
when attempting to define a file to the system.
createWAS: ./createWAS.sh: failed

```

createDM.sh

This script creates the domain manager. In /usr/lpp/VE_R2/EWLM/bin you can launch the script with the command:

```

createDM configID -adminUser userid -adminPW password -wasProfile profilePath -jp port -ma
address -mp port -dn domainName -auth [None | ServerSSL | ClientServerSSL] -sslks path
-sslpw password

```

Where:

- ▶ -adminUser is the WebSphere Application Server administrator user ID.
- ▶ -adminPW is the password associated with the -adminUser parameter.
- ▶ -wasProfile is the directory of the WebSphere Application Server profile created by the createWAS script, to be used by the EWLM domain manager.
- ▶ -jp is the port used between the Control Center and the domain manager.
- ▶ -ma is the domain manager host name or IP address.
- ▶ -mp is the port where the domain manager is listening for managed servers' connections.
- ▶ -dn is the Management Domain name.
- ▶ -auth is the level of authorization used between the domain manager and managed servers.
- ▶ -sslks only has meaning when -auth is not None and represents the path of the certificate keystore that contains the SSL keys that are used to secure the following communications:
 - Between the domain manager and the managed servers when -auth is ServerSSL or ClientServerSSL
 - Between the domain manager and firewall brokers, if they exist in the EWLM domain, when -auth is ServerSSL or ClientServerSSL
 - Between the domain manager and Load Balancers if you provide a port value for the -lbs parameter
- ▶ -sslpw only has meaning when -auth is not None and is the password used to access the keystore you specified on the -sslks parameter, if applicable.

In our configuration, we issue the following command:

```

./createDM.sh zosDM -adminUser EWLMWAS -adminPW EWLMWAS -wasProfile
/u/ewlm2u/vewas/AppServer/profiles/default -jp 9001 -ma wtsc60.itso.ibm.com -mp 5333 -dn
ITSOEWLM -auth None

```

Consider the following:

- ▶ Since the command with all its options is too long to fit into an OMVS or telnet command line, we suggest that you create a shell script to invoke createWAS.sh. We called the shell script dm.sh, whose content follows:

```
./createDM.sh zosDM -adminUser EWMWAS -adminPW EWMWAS -ma wtsc60.itso.ibm.com -mp 5333 -dn ITS0EWM -auth None -wasProfile /u/ewlm2u/vewas/AppServer/profiles/default -jp 5333
```

- ▶ The output can easily scroll off the screen, especially if there are errors. To address this, we suggest that you redirected stdout and stderr to a file. The invocation of the createWAS.sh script looks like the following:

```
dm.sh > dm.out 2>&1
```

Note: You need to run the **createDM.sh** command from either the WebSphere administration user ID or from a user ID that has the CA certificate in its RACF keyring. Otherwise, the createDM.sh script will not be able to execute the WebSphere administrative commands needed to complete the WebSphere configuration.

When the **createDM** command completes, you should have the HFS configuration described in Figure 2-28.

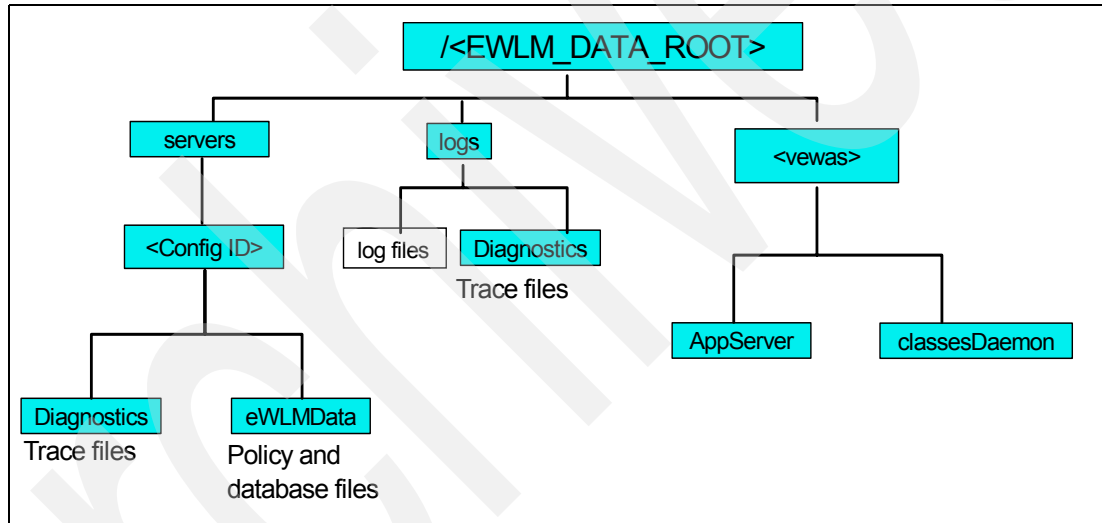


Figure 2-28 Domain manager HFS file configuration

Adding Control Center user IDs

In our configuration, we defined only one user ID to log on to the Control Center with the scope of administrator.

To define the user ID, you need to run the changeCC.sh script in /usr/lpp/VE_R2/EWLM/bin:

```
./changeCC -addUser configID -adminUser userid -adminPW password -roleUser user -role [Administrator | Monitor | Operator]
```

In our configuration, we issue:

```
./changeCC -addUser zosDM -adminUser VEWAS -adminPW VEWAS -roleUser ewlmadm -role Administrator
```

Copying the EWLM procedures

If you install the procedure in an installation proclib, you need to now copy the procedures into your production PROCLIB. Here is the list of the procedures:

- ▶ EWLMDM
- ▶ EWLMEW
- ▶ EWLMSW
- ▶ ELMW
- ▶ EWLMDWA
- ▶ EWLMDWAZ
- ▶ EWLMDWD
- ▶ EWLMDWZ
- ▶ EWLMDWO
- ▶ EWLMDWS
- ▶ EWLMDWSZ
- ▶ EWLMDWZ

You might want to verify that the STDOUT and STDERR DD cards in the procedures are pointing to an output file consistent with your installation.

Starting EWLM

We suggest starting the EWLM domain manager using MVS™ start commands. To start the domain manager, issue the following command:

```
START EWLMDM,CONFIGID='configID'
```

In our case, we issue:

```
S EWLMDM,CONFIGID='zosDM'
```

As an alternative, you can start the domain manager using the startDM script.

Starting WebSphere Application Server

We recommend starting the Virtualization Engine WebSphere using the MVS start command.

To start the Virtualization Engine WebSphere, issue the following command:

```
START EWLMSW,CONFIGID='configID'
```

In our case, we issue:

```
S EWLMSW,CONFIGID='zosDM'
```

As an alternative, you can start the WebSphere Application Server using the startWAS script.

Logging on to the control center

After the domain manager and WebSphere are started, you can log in to the Control Center by pointing your browser to the URL `https://<hostname:port>/webui`. You can discover the port by looking at the createDM log where you can see the port assignments. You can also determine the WAS port assignment by issuing the `displayCC` command.

In our configuration, we use:

```
https://wtsc69.itso.ibm.com:32209/webui
```

2.8.4 Configuring the managed server

Below is the list of tasks that you need to perform to configure the managed server.

Update your configuration file

To do this:

1. Copy <EWLM_ROOT>/samples/ewlmms_environment.conf to /etc, where <EWLM_ROOT> is the absolute path to where the EWLM code is installed. In our case, this is /usr/lpp/VE_R2/EWLMMS. For example:

```
cp ewlmms_environment.conf /etc
```

2. Verify and update the following parameters in the copied version in /etc:
 - Verify that EWLM_ROOT=/usr/lpp/VE_R2/EWLMMS is consistent with your installation. This should point to the absolute path where the EWLM code is installed.
 - Update EWLM_DATA_ROOT= with the absolute path of where you would like EWLM to create the working directories.
 - Update JREBIN_ROOT= with the absolute path of the bin directory of your JRE. In our case this is /usr/lpp/java/J1.4/bin.
 - Verify that LOGFILE_ROOT=/var/adm/ewlm/ewlmdm is consistent with your installation. This should point to the absolute path where the EWLM managed server code should create the log files.

Example 2-10 Sample for the managed server configuration file

```
EWLM_ROOT=/usr/lpp/VE_R2/EWLMMS
EWLM_DATA_ROOT=/u/ewlm2u
JREBIN_ROOT=/usr/lpp/java/J1.4/bin
LOGFILE_ROOT=/u/ewlm2u/logs
```

createMS.sh

This script, located in the <install>/VE_R2/EWLMMS/bin directory, is used to configure the managed server on z/OS. The following command syntax creates the managed server on z/OS:

```
createMS configID -ma address -mp port -auth [None | ServerSSL | ClientServerSSL] -sslks
path -sslpw password
```

Where:

configID Defines the configuration for this managed server instance. This configuration should not already exist; **createMS** creates it for you.

Note: If you use a configuration ID name with lowercase characters, remember when starting the managed server with the START command that you must preserve the case (for example, by using quotations around the directory name).

-ma The IP address or host name of the domain manager. This value must match the -ma value that you specify when you configure the domain manager. This enables the managed server to properly identify and communicate with its domain manager.

-mp The port that the managed server uses to communicate with the domain manager. This value must match the -mp value that you specify when you configure the domain manager.

-auth The authority level used to secure communications between the domain manager and the managed servers. Possible values are None, ServerSSL,

and ClientServerSSL. This level of authorization must match what was specified for the domain manager.

-sslks The path to the certificate keystore that contains the SSL keys that are used for securing the communications between the managed servers and the domain manager or firewall broker, if applicable, when `-auth` is ServerSSL or ClientServerSSL.

-sslpw Password used to access the keystore specified on the `-sslks` parameter.

We used the following command to create the managed server on SC69.

```
cd /usr/lpp/VE_R2/EWLMMS/bin
./createMS.sh zosMS -ma wtsc69.itso.ibm.com -mp 5333 -auth None
```

As a result of executing `createMS`, the configuration ID `zosMS` is created in the `/u/ewlm2u/servers/` directory with the subdirectories shown in Figure 2-29.

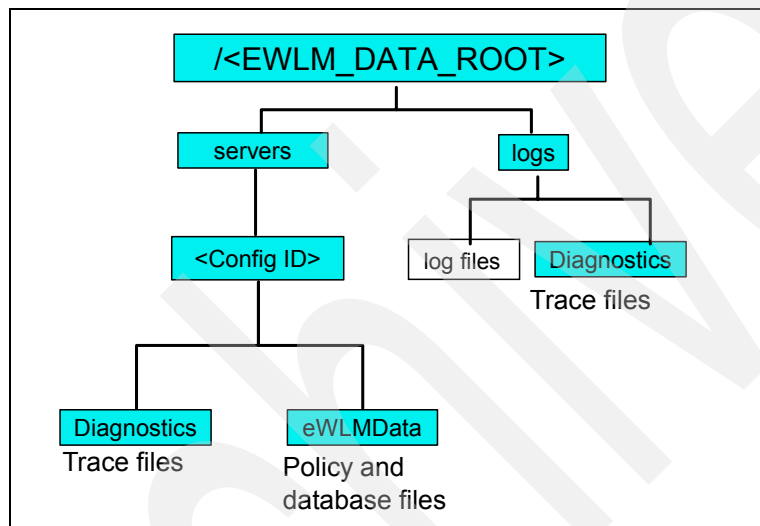


Figure 2-29 Managed server HFS configuration

Authorizing z/OS users to start the EWLM managed server

In your installation, you might want to start the z/OS managed with a user ID that does not have superuser authority but can access the necessary files and directories.

The following tasks are required to define a user that will start the managed server and be associated with the managed server process:

1. Define a group and user that will be used to start the managed server. The managed server process will run using the user ID defined here. Use the user ID and group name that you specify in this step throughout the rest of these instructions. You can use the following RACF commands to create a group and add a user to the group; you must provide similar definitions if your installation is using a different security product:

```
ADDGROUP groupname SUPGROUP(SYS1) OWNER(SYS1) OMVS(GID(group-identifier))
```

```
ADDUSER userid DFLTGRP(groupname) NOPASSWORD OMVS(UID(user-identifier) PROGRAM(/bin/sh) HOME(/usr/lpp/VE_R1/EWLM))
```

In this syntax:

groupname The name of the group that contains EWLM user IDs
userid The user ID that the managed server process will run as

group-identifier The numeric group identifier

user-identifier The numeric user identifier

Example 2-11 is the set of commands we used on SC69 to define our EWLM user as EWLMMS belonging to the EWLMGRP group.

Example 2-11 Defining EWLM user ID and group to the security product

```
ADDGROUP EWLMGRP SUPGROUP(SYS1) OWNER(SYS1) OMVS(GID(6899))
ADDUSER EWLMMS DFLTGRP(EWLMGRP) NOPASSWORD OMVS(UID(6824) PROGRAM(/BIN/SH)
LU EWLMMS

USER=EWLMMS NAME=UNKNOWN OWNER=BARI      CREATED=05.070
DEFAULT-GROUP=EWLMGRP      PASSDATE=N/A      PASS-INTERVAL=N/A
ATTRIBUTES=PROTECTED
REVOKE DATE=NONE      RESUME DATE=NONE
LAST-ACCESS=05.073/09:42:06
CLASS AUTHORIZATIONS=NONE
NO-INSTALLATION-DATA
NO-MODEL-NAME
LOGON ALLOWED      (DAYS)      (TIME)
-----
ANYDAY      ANYTIME
GROUP=EWLMGRP      AUTH=USE      CONNECT-OWNER=BARI      CONNECT-DATE=05.070
CONNECTS=      04      UACC=NONE      LAST-CONNECT=05.073/09:42:06
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE      RESUME DATE=NONE
SECURITY-LEVEL=NONE SPECIFIED
CATEGORY-AUTHORIZATION
NONE SPECIFIED
SECURITY-LABEL=NONE SPECIFIED
***
```

2. Grant the user ID access to call z/OS platform services. To do this, issue the following RACF commands:

```
RDEFINE FACILITY MVSADMIN.EWLM.AGENT UACC(NONE)
PERMIT MVSADMIN.EWLM.AGENT CLASS(FACILITY) ID(userid) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

In this syntax, *userid* is the user ID that the managed server process will use, as defined in the previous step.

Example 2-12 is the set of RACF commands we used on SC69 to enable our user ID, EWLMMS, to use ARM services.

Example 2-12 Granting the usage of ARM services to EWLMMS

```
RDEFINE FACILITY MVSADMIN.EWLM.AGENT UACC(NONE)
PERMIT MVSADMIN.EWLM.AGENT CLASS(FACILITY) ID(EWLMMS) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

3. Provide the user with the ability to start the EWLM managed server by issuing the following RACF commands:

```
RDEFINE STARTED EWLMS.** STDATA(USER(userid) GROUP(groupname) TRUSTED)
SETROPTS RACLIST(STARTED) REFRESH
```

In this syntax:

userid The user ID, created earlier, that is associated with the EWLM managed server process

groupname The group name defined in the previous step

Example 2-13 is the set of commands we used on SC69 to associate the EWLMS user ID to the managed server process.

Example 2-13 Providing a RACF user ID to the EWLM managed server process

```
RDEFINE STARTED EWLMS.** STDATA (USER(EWLMS) GROUP(EWLMGRP) TRUSTED)
SETROPTS RACLIST(STARTED) REFRESH
```

4. Switch to a user with superuser authority to perform the following tasks.
5. Grant the group that you created in the previous step access to the EWLM directories by issuing the following commands:

```
chgrp -R groupname <ewlm_data_root>/servers/<config_id>
chgrp groupname /etc/ewlms_environment.conf
chgrp -R groupname /var/adm/ewlm
```

In this syntax:

groupname The group name representing the EWLM user ID, created in the earlier step

config_id The EWLM configuration ID that was created at the time of the **createMS** command

/var/adm/ewlm The directory where the scripts are allocating their output files

Example 2-14 is the set of commands we used on SC69 to associate the EWLMS user ID to the managed server process.

Example 2-14 Commands to grant access to directories and file to EWLMS user ID

```
chgrp -R EWLMGRP /u/ewlm2u/servers/zosMS
chgrp EWLMGRP /etc/ewlms_environment.conf
chgrp -R EWLMGRP /var/adm/ewlm
```

6. Grant the group access to the configuration file and remove access for all other users by issuing the following commands:

```
chmod g+w /etc/ewlms_environment.conf
chmod o-rx /etc/ewlms_environment.conf
```

Copying the EWLM procedures

If you install the procedure in an installation proclib, you now need to copy the procedures into your production PROCLIB. Here is the list of the procedures:

- ▶ EWLMSFB
- ▶ EWLMS

You might want to verify that the STDOUT and STDERR DD cards in the procedures are pointing to an output file consistent with your installation.

Start EWLM on the z/OS managed server

In the <install>/VE_R2/EWLMMS/bin directory, you can find the createMS.sh script that can be used to start the EWLM managed server.

The recommended way to start the EWLM is to use the procedure provided by the installation in your PROCLIB. At start time, the EWLM process will be associated with the newly created user ID and defined in the RACF STARTED class.

Issue the following command:

```
START EWLMMS,CONFIGID='config_id'
```

In this syntax, CONFIGID is the configuration ID that is created by the createMS configuration script.

Remember to use quotation marks if the configID name includes lowercase characters to preserve the case of parameters.

This is the start of the EWLM managed server on SC69 using the procedure:

```
START EWLMMS,CONFIGID='zosMS'
```

Example 2-15 shows checking the output log to verify that the process started successfully.

Example 2-15 Verifying a successful start by checking the output log

```
cd /var/adm/ewlm/ewlmms/  
obrowse startMS.log  
  
*****  
* BEGIN *  
startMS command started at: Mon Mar 14 09:42:06 EST 2005 Platform = OS/390  
STEP 1: Access EWLM working directory  
EWLM working directory: /ewlm/ewlmms
```

There is no way to bring down EWLMMS gracefully. If you need to stop the managed server, you must cancel the started task.

2.9 Migrating from EWLM Release 1

In this section we give you an approach to migrating from EWLM R1 code.

2.9.1 Migrating from EWLM R1 code

There is no special migration facility from EWLM R1 to EWLM V2.1, so our approach was to first save the policies from our EWLM R1 environment, uninstall EWLM R1, install the EWLM V2.1 code, and restore the policies saved from EWLM R1.

2.9.2 Saving a Domain Policy on EWLM R1

To save the domain policies from EWLM R1 do the following:

1. Log in to the EWLM Control Center and select **Domain policies** under the Set up option. A list of the domain policies appears in the right-hand pane.

2. Select the Domain Policy that you want to save.
3. Select the **Export** function from the pull-down list, and click **Go**, as shown in Figure 2-30.

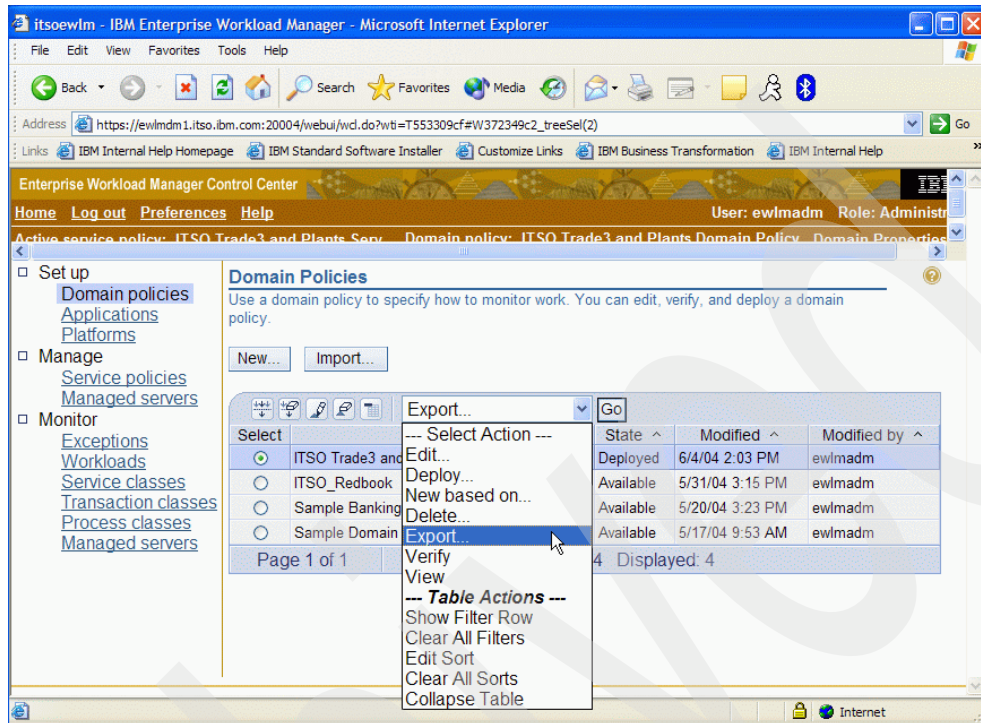


Figure 2-30 Export Domain Policy from EWLM R1

4. Click **Save** and select one destination directory for the file.

Now that we have saved the policies, we can uninstall the EWLM R1 code. To uninstall the domain manager code, we first delete the domain manager and then uninstall the domain manager code and the Virtualization Engine code. Follow these steps to uninstall:

1. Go to the bin directory under the EWLM domain manager install directory and run the `deleteDM` command. See Example 2-16.

Example 2-16 Deleting the domain manager instance

```
# cd /opt/IBM/VE/EWLM/bin
# ./deleteDM.sh /opt/DM1 -adminUser ewlmdm -adminPW 111111 -noPrompt
```

Wait for the command to finish.

2. Go to Virtualization Engine installation directory and run the `uninstallVE<platform>` command, where `<platform>` refers to the operating system where the code is installed, for example, AIX or Win. This command opens a wizard that guides you through the uninstallation of the EWLM domain manager and Virtualization Engine code.

Tip: The wizard needs a graphical environment in order to run. If you are uninstalling a UNIX platform using a `telnet` session, for example, remember to set the `DISPLAY` variable and also to enable an X server environment on your workstation.

The EWLM domain manager code is now uninstalled. We proceed to uninstall the EWLM managed server code by first deleting the managed server.

1. Go to the bin directory of the EWLM managed server installation directory and run the `deleteMS` command, as in Example 2-17.

Example 2-17 Deleting the managed server

```
# cd /opt/IBM/VE/EWLMMS/bin
# ./deleteMS.sh /opt/AIXMS
```

2. Next, to uninstall the managed server go to the `_uninst` directory under the EWLM managed server installation directory and run the `uninstall` command. This opens a wizard that guides you through the uninstallation process.

Tip: The wizard needs a graphical environment in order to run. If you are uninstalling a UNIX platform using a `telnet` session, for example, remember to set the `DISPLAY` variable and also to enable an X server environment on your workstation.

Now all of the EWLM R1 and Virtualization Engine R1 products are uninstalled. You can proceed to 2.4, “Virtualization Engine common components installation” on page 29, for installation instructions. Once you finish the installation, go to 2.9.3, “Importing a Domain Policy” on page 74.

2.9.3 Importing a Domain Policy

This section shows how to import a Domain Policy. It can be used if you are migrating from EWLM R1 and have saved your policy or if you have your own `.xml` policy file.

Log in to the EWLM Control Center and go to **Setup** → **Domain Policies** → **Import**. Figure 2-31 shows this screen.

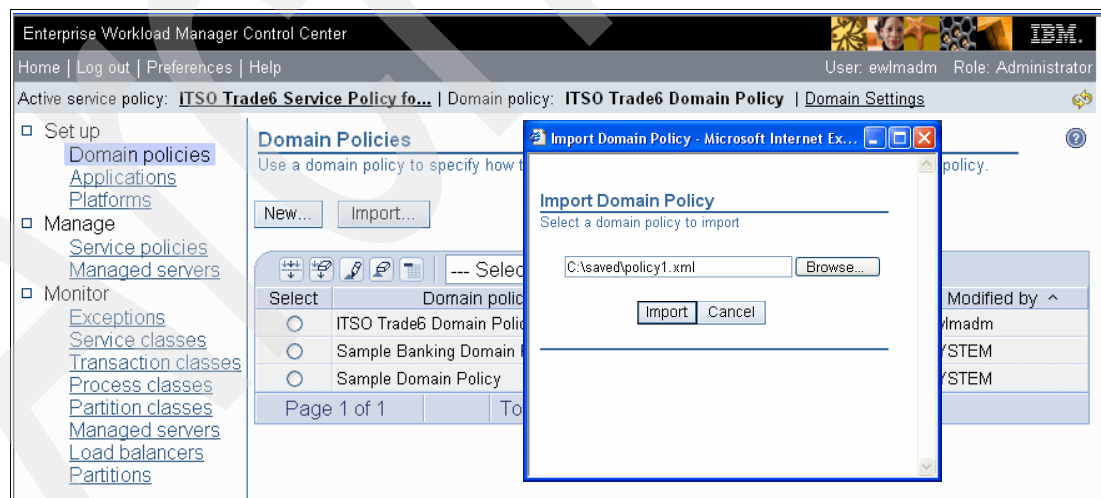


Figure 2-31 Importing a Domain Policy into EWLM V2

In the pop-up window, click **Browse** to find the saved file or type in the full path of the file, then click **Import**.

2.10 Uninstalling EWLM

In this section we describe how to uninstall the EWLM and Virtualization Engine products.

Uninstalling EWLM and Virtualization Engine V2

To uninstall the domain manager code, we first delete the domain manager and then uninstall the EWLM managed server and the Virtualization Engine code. Follow these steps to uninstall:

1. Go to the bin directory under the EWLM domain manager install directory and run the `deleteDM` command. See Example 2-18.

Example 2-18 Deleting the domain manager instance on EWLM V2

```
# cd /opt/IBM/VE/EWLM/bin
# ./deleteDM.sh DM1 -adminUser wasadmin -adminPW itso -noPrompt
```

Wait for the command to finish.

2. Go to the Virtualization Engine installation directory and run the `uninstallVE<platform>` command, where `<platform>` refers to the operating system where the code is installed, for example, AIX or Win. This command opens a wizard that guides you through the uninstallation of the EWLM domain manager and Virtualization Engine code.

Tip: The wizard needs a graphical environment in order to run. If you are uninstalling a UNIX platform using a `telnet` session, for example, remember to set the `DISPLAY` variable and also to enable an X server environment on your workstation.

The EWLM domain manager and Virtualization Engine code are now uninstalled. We now proceed to uninstall the EWLM managed server code by first deleting the managed server.

1. Go to the bin directory of the EWLM managed server installation directory and run the `deleteMS` command, as in Example 2-19.

Example 2-19 Deleting the managed server on EWLM R2

```
# cd /opt/IBM/VE2/EWLMMS/bin
# ./deleteMS.sh AIXMS
You requested to delete configuration ID AIXMS
Are you sure (Y or N) -
Y
Processing deleteMS request. Please be patient as this may take awhile...
PROCESSING COMPLETE
```

2. Go to the `_uninst` directory under the EWLM managed server installation directory and run the `uninstall` command. This opens a wizard that guides you through the uninstallation process.

Tip: The wizard needs a graphical environment in order to run. If you are uninstalling a UNIX platform using a `telnet` session, for example, remember to set the `DISPLAY` variable and also to enable an X server environment on your workstation.

Once you finish all of the uninstallations, you can remove the Virtualization Engine install directory.

Archived



Using a firewall and securing EWLM

This chapter provides a description of the following infrastructure elements you can enable during Enterprise Workload Manager installation and customization:

- ▶ Integration with existing firewall structure
- ▶ Enablement of security

3.1 Firewalls

In today's e-business environment companies need to be pro-active in defending against internal and external attacks to their corporate networks. e-business infrastructures may span multiple security zones, which can be protected with a mix of routers, firewalls, and virtual private networks. The most common security zones for Internet-initiated transactions are Web, application, data and enterprise, and systems management. While our ITSO configurations did not include all of these security zones, the configurations provided should be general enough to apply to different security architectures.

This section provides a general overview of firewall technology, EWLM support of firewalls, and finally describes the firewall configurations we used, stateful inspection, and proxy.

3.1.1 General firewall overview

This section briefly describes different firewall technologies and introduces the StoneGate firewall, which was used in our configuration.

Packet filter firewall

Packet filter firewalls are the oldest kind of firewall. Modern routers can support this kind of firewall, which is based on network-level access lists that describe what kind of traffic can traverse the firewall. A packet filter firewall has the following advantages:

- ▶ Application independence: It operates at IP-address level and does not care about the application protocol.
- ▶ It is transparent to the application for the same reason. You can compare this type of firewall to a router with an access list.
- ▶ High performance: The firewall just checks network layer properties and does not have to check application-layer traffic.

Disadvantages are:

- ▶ Low security: There are no application-layer checks.
- ▶ It is difficult to maintain and the firewall rule base will grow fast if you want the rules to be very granular. The number of rules affects performance because the firewall has to go through the whole rulebase before it can discard a packet.

Proxy firewall

Proxy firewalls are the next generation of firewall technology. In the early days, firewalls and proxies used to be separate entities. Then they were merged together to make a proxy firewall. It has the following advantage:

- ▶ High security: The proxy firewall can examine the application-layer data.

Disadvantages are:

- ▶ Low performance: The proxy firewall will examine the application-layer data, which is CPU-intensive work.
- ▶ Limited application support: Each application has to have a proxy for it. Firewalls have support for most common applications and protocols like HTTP (www) and SMTP (e-mail), but not for more exotic or less frequently used protocols.
- ▶ Lack of high availability: In order to make a firewall highly available one should be able to maintain the state of the proxy, but that is a complex task and it is normally not well done.

Stateful inspection firewall

A *stateful inspection firewall* is the most recent firewall technology. It is a combination of the packet inspection firewall and proxy firewall along with state tables. *State tables* are used to make the firewall aware of related connections. For example, a simple ftp connection has two parts: The command connection and the data connection. Packet filter firewalls had problems with ftp because they had to handle those two connections as separate connections. Stateful inspection firewalls use state tables to keep track of open connections and their relationship with other connections. A stateful inspection firewall would know whether it has an open ftp command connection; then it can allow the data connection to come back from the server without any additional rules in the firewall.

A stateful inspection firewall has the following advantages:

- ▶ Transparency: The application does not have to care about the existence of the firewall.
- ▶ High security: State tables add connection awareness for the firewall.
- ▶ Performance: It is closer to the packet filter firewall.

A stateful inspection firewall has the following disadvantage:

- ▶ Limited application-layer awareness: It does not have proxy-level capabilities.

The StoneGate firewall is a stateful inspection type of firewall that has some application-layer awareness. One of the StoneGate features is the Protocol Agent, which enables it to see the application-layer data stream and perform actions based on that. The Protocol Agent can check that the traffic actually is HTTP traffic and not something else—like a hacker trying to pipe some other traffic to that port. In such a case, the Protocol Agent can disconnect the hacker traffic.

We chose StoneGate as our example firewall because it does not require any additional configuration from the EWLM side. Statistics also show that most modern firewalls are stateful inspection firewalls. Since there are still many existing proxy firewalls, we also explain how to configure EWLM to work with a proxy firewall.

StoneGate is also the only commercial firewall available that can be used across IBM xSeries, iSeries, and zSeries machines. If the WebSphere environment or other parts of the environment are consolidated inside an iSeries or zSeries machine, you can use StoneGate firewall on those machines. You do not need an external firewall, because the StoneGate firewall operates as a virtual firewall inside iSeries or zSeries machine.

3.1.2 EWLM firewall support

EWLM supports stateful inspection firewalls, HTTP Proxy, and SOCKS. HTTP Tunneling is not supported. For stateful inspection firewalls, no additional configuration is required at the Domain Manager or Managed Server. HTTP Proxy requires a configuration change to the Managed Server. A SOCKS server requires a configuration change to the managed server, and an additional EWLM component called the *firewall broker* needs to be installed and configured.

We focus here on stateful inspection and HTTP Proxy since these are the most modern firewalls, and we discuss the SOCKS server since it requires modifications to the EWLM configuration. In all cases, though, it is important to understand what you need to think about when placing EWLM into an existing or new firewall environment. In 2.5, “Installing and

configuring EWLM domain manager” on page 34 we described how to configure the Domain Manager. There are three sets of parameters that are important to know when setting up the firewall for EWLM traffic:

- ▶ Domain manager address: -ma
- ▶ Domain manager port: -mp
- ▶ WebSphere Application Server range of ports: -wasProfile

For the EWLM environment these are the required communication parameters that support traffic between the Managed Server and Domain Manager, traffic between a browser and EWLM Control Center, and traffic between a browser and WebSphere Administration Console. Since the port specified in WasProfile is a starting port for up to ten/fifteen ports that WebSphere Application Server uses, you need to execute the `displayCC` command at the Domain Manager, as shown in Example 3-1, after you created the domain manager. This provides you with the ports assigned to the Administration Console and Control Center traffic.

Example 3-1 displayCC to get assigned ports

```
C:\Program Files\ibm\VE2\EWLM\bin>displayCC -ports DM1 -adminUser wasadmin -adminPW itso

Processing displayCC request. Please be patient as this may take awhile...
...processing 33% complete

...Ports assigned to EWLM Control Center:
  - HTTP 5775
  - HTTPS 5777
...Use -changeCC -controlCenterPorts to change these ports if desired.

...Ports assigned to WebSphere Application Server Administration Console:
  - HTTP 5776
  - HTTPS 5778
...Use -changeCC -adminConsolePorts to change these ports if desired.

...Port assigned to WebSphere Application Server Administration: 5780
...Use -changeCC -adminPort to change this port if desired.

...processing 66% complete
PROCESSING COMPLETE
```

If you need to check the Domain Manager port, the `displayDM` command will show this in the `mp` parameter, as shown in Example 3-2.

Example 3-2 displayDM command to get assigned ports

```
Cd \Program Files\ibm\VE2\EWLM\bin
displayDM DM1

auth(None)
sslks(null)
sslpw(**password suppressed**)
ma(ewlm1.itso.ibm.com)
mp(5773)
jp(5800)
dn(ITS0 Domain)
fp(null)
fb(null)
sa(null)
sp(null)
```



```
lbp(Off)
lbs(Off)
nt(250)
ct(250)
et(250)
rt(250)
ml(250)
lbt(250)
tcomm(0)
tl(Min)
tlog(Off)
dpa(30)
dpr(25)
fl(50)
```

PROCESSING COMPLETE

If using a stateful inspection firewall, rules may need to be added to the firewall for this additional http traffic. When using a proxy, only the proxy rules needs to be added. Figure 3-1 shows an EWLM Management Domain that resides across four security zones. The highlighted traffic with the Domain Manager is the only traffic that we need to be concerned with when considering impacts to firewall setup from an EWLM perspective.

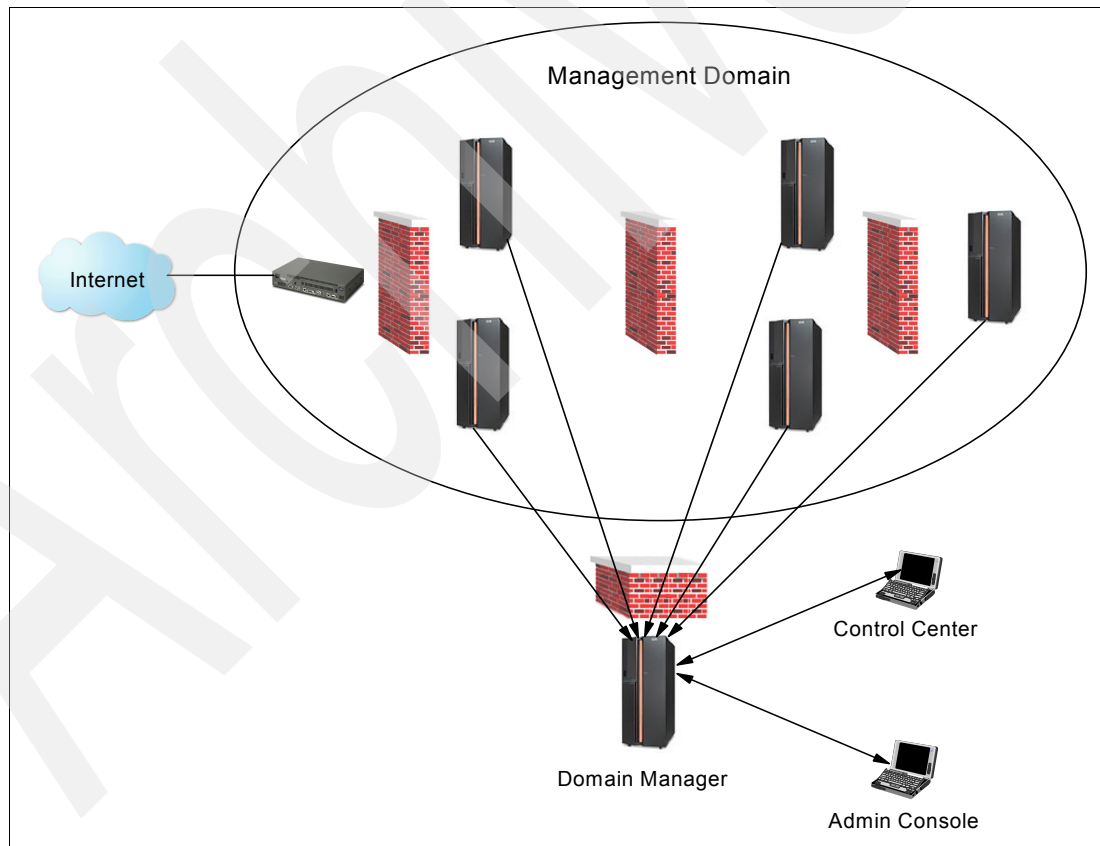


Figure 3-1 EWLM network traffic

Stateful inspection and EWLM

In Figure 3-1, if we assume first that the Domain Manager is in a System Management security zone protected by a firewall, then from an EWLM perspective, no changes need to be

made to the EWLM configuration. Rules may need to be added to the firewall to allow the following traffic; we use the ports and addresses discovered from the `displayCC` and `displayDM` commands in Example 3-1 on page 80 and Example 3-2 on page 80, respectively:

- ▶ Managed server(s) to domain manager at address:port, ewlm1.itso.ibm.com:5773
- ▶ Browser to Control Center at address:port(s), ewlm1.itso.ibm.com:5775,5777
- ▶ Browser to WebSphere Administration Console at address:port(s), ewlm1.itso.ibm.com:5776,5778

Proxy and EWLM

Now if we assume that the domain manager is in a System Management security zone protected by a proxy firewall, EWLM configuration changes must be made for any managed server that must traverse the proxy before reaching the domain manager. In addition, rules may need to be added to the proxy for the Control Center and Admin Console as described in the stateful inspections firewall set up.

The configuration changes that must be made to the Managed Server are shown in Figure 3-2. The Managed Server must be configured with two additional parameters, `-va` and `-vp`, using the `changeMS` command.

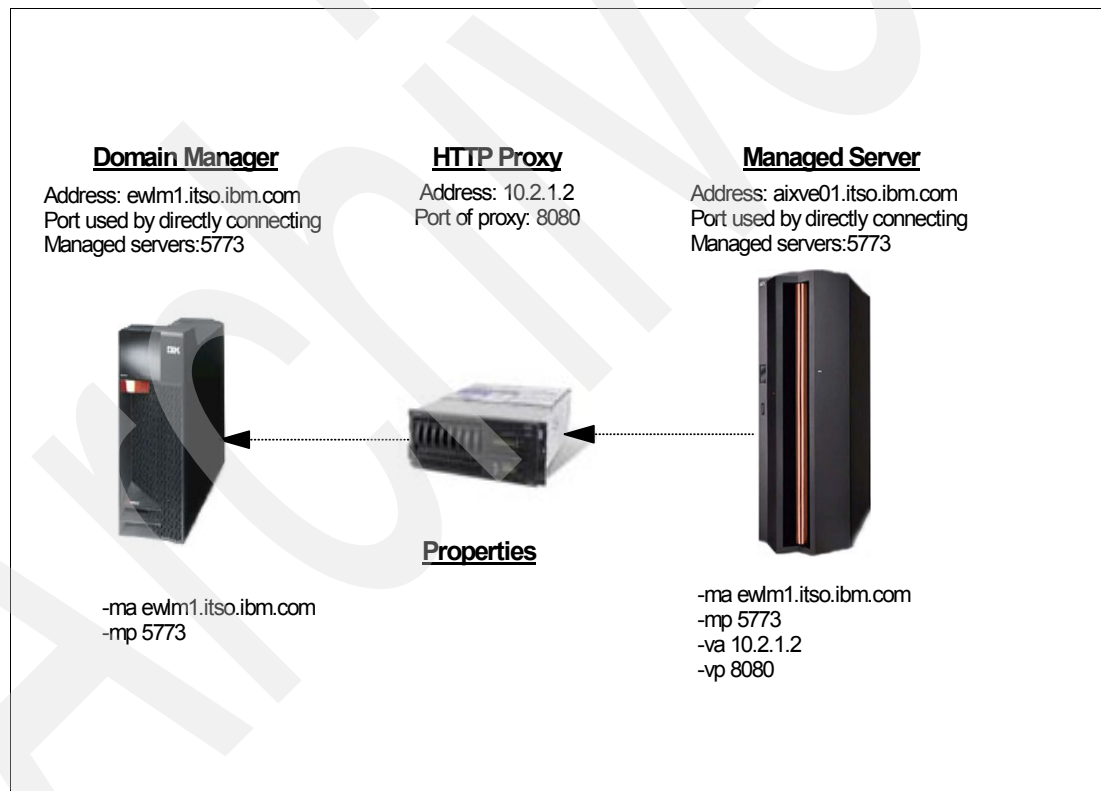


Figure 3-2 HTTP Proxy example

Figure 3-2 on page 82 shows the Managed Server communicating with the Proxy server, then the Proxy communicating with the Domain Manager on the Managed Server's behalf. In order for this communication to occur, the command shown in Example 3-3 needs to be executed at the Managed Server if you add the Proxy server after you configure the managed server for the first time.

Example 3-3 changeMS command

```
changeMS configID [-auth [None : ServerSSL : ClientServerSSL]]
                  [-sslks [file] -sslpw [password]]
                  [-ma address] [-mp port]
                  [-va [address]] [-vp [port]]
                  [-jp [port]]
                  [-nt [value]] [-ct [value]]
                  [-et [value]] [-jt [value]]
                  [-ml [value]] [-tl [value]]
                  [-tlog [class]]
                  [-dpa [value]] [-dpm [value]]
                  [-fl [value]]
```

In order to add the Proxy address and port, Example 3-4 shows the custom **changeMS** command that would be used based on the addresses and ports in Figure 3-2 on page 82. To alter the configuration you need to stop the managed server, issue the **changeMS** command, and then restart the managed server with the **startMS** command.

Example 3-4 changeMS for Proxy support, Windows example

```
changeMS ewlms -va 10.2.1.2 -vp 8080
```

After restarting the managed server, you should be able to log into the Control Center and check the managed servers in the Monitor section. The managed server that you changed should be in *active* state. You will need to execute this procedure at each managed server that needs to communicate with the domain manager via a proxy.

SOCKS and EWLM

If your installation is using a SOCKS server for firewall protection, you must also configure a firewall broker to allow the managed servers to communicate with the domain manager through the SOCKS server. The firewall broker does not have to run on a server that is acting as a managed server. However, it does have to be in the same trusted zone as the managed servers. There are several configuration parameters that must match when you create and configure the domain manager, the managed servers, and the firewall broker. If any of the required parameters are incorrect, the entire communication stream will fail. The firewall broker image is installed on the managed server as part of the EWLM code in the `<installation_path>/IBM/VE2/EWLMMS` directory. You can then plan to use the firewall broker on the Managed Server itself or distribute it to the appropriate platform.

Table 3-1 is a summary of the different configurations with a SOCKS server and how they affect the EWLM code, followed by two detailed examples.

Table 3-1 SOCKS server configurations and EWLM set up

Target installation configuration	EWLM configuration
Protecting the connections from the managed servers to the domain manager using the SOCKS server	<ul style="list-style-type: none"> - Need firewall broker. - Domain manager must be configured to identify firewall broker using the changeDM command with the -fp and -fb parameters. - Firewall broker must be configured to use SOCKS server using changeFB command with the -sa and -sp parameters.

Target installation configuration	EWLM configuration
Protecting the connections from the domain manager to the managed servers using the SOCKS server	EWLM is not affected. It always connects from the managed servers to the domain manager. EWLM is unaware of and unaffected by the existence of the SOCKS server.
Protecting the connections from the managed servers to the domain manager and the connections from the domain manager to the managed server (that is, through a DMZ) using the SOCKS server	<ul style="list-style-type: none"> - Need firewall broker. - Firewall broker must be configured to use SOCKS server using the changeFB command with the -sa and -sp parameters. - Domain manager must be configured to use SOCKS server using the changeDM command with the -sa and -sp parameters. - The domain manager must be configured to identify the firewall broker using the changeDM command with the -fp and -fb parameters. - The domain manager -fb list must use the SOCKS tag for this firewall broker.

In the following discussion, two sample SOCKS server configurations are shown as examples of security zone solutions.

Figure 3-3 shows the managed servers accessing the domain manager through a through a firewall broker and a SOCKS server. In this configuration, the SOCKS server is protecting the zone where the domain manager is located.

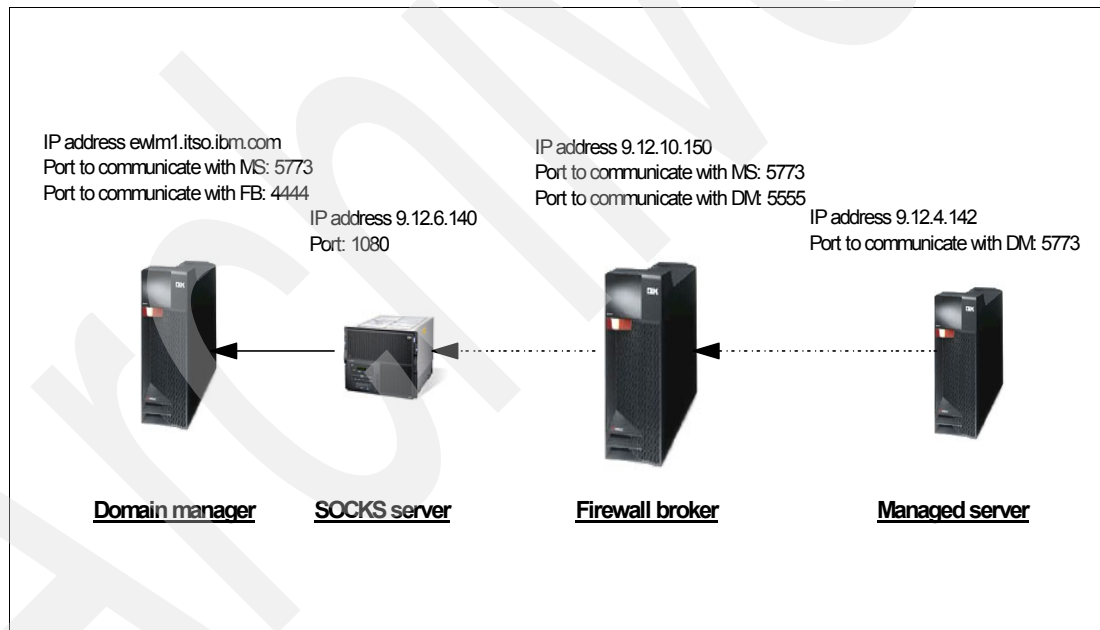


Figure 3-3 SOCKS Server protecting the domain manager zone

To define this configuration the following changes are required (sample commands are in Example 3-5 on page 85):

- ▶ On the Managed Servers specify the following parameters on the **createMS** command:
 - **-ma** and **-mp** must point to the IP address or host name and port of the firewall broker.
- ▶ On the firewall broker, run the **createFB** script with the following parameters:
 - **-ma**, the IP address or host name of the firewall broker. This parameter must match the **-ma** parameter you set on the **createMS** script.

- `-fp`, the port used exclusively for domain manager to firewall broker communications. This parameter maps to the port value passed on the `-fb` parameter of the `changeDM` script.
- `-da`, the IP address of the domain manager. This parameter maps to the `-ma` parameter on the `changeDM` script.
- `-dp`, the port used by the firewall broker to connect to the domain manager. This parameter maps to the `-fp` parameter on the `changeDM` script.
- ▶ On the firewall broker, run the **changeFB** script with the following parameters:
 - `-sa`, the IP address of the SOCKS server. Which server you specify in this parameter depends on how the SOCKS server is set up. If the SOCKS server is protecting the zone where the firewall broker and managed servers are located, then you set the `-sa` parameter on the firewall broker, as in this example. If the SOCKS server is protecting the zone where the domain manager is located, then you set the `-sa` parameter on the domain manager. In a scenario where there are SOCKS servers protecting both zones, as in a DMZ, you would set the `-sa` parameter on both the firewall broker and the domain manager. An example of this setup is described in the next sample configuration.
 - `-sp`, the port of the SOCKS server. As with the `-sa` parameter, where you set `-sp` depends on how the firewall protection is set up.
- ▶ On the Domain Manager specify the following parameters on the **changeDM** command:
 - `-fb`, which identifies a list of firewall brokers with which the domain manager will communicate. Each firewall broker is identified by:
 - IP address, which maps to the `-ma` parameter on the `createFB` script
 - Port, which maps to the `-fp` parameter on the `createFB` script
 - A tag, SOCKS, that indicates that a SOCKS server is required to connect to the firewall broker
 - `-fp`, which is the port the firewall broker uses to communicate with the domain manager. This parameter maps to the `-dp` parameter on the `createFB` script.

Example 3-5 Sample commands to define the SOCKS server environment

on the Domain manager:

```
./changeDM.sh ewlmdm -ma 9.12.6.141 -mp 5773 -fp 4444 -fb 9.12.10.150:5555
```

on the Firewall broker:

```
./createFB.sh ewlFB -ma 9.12.10.150 -mp 5773 -da 9.12.6.142 -dp 4444 -fp 5555
```

```
-auth None
```

```
./changeFB.sh ewlFB -sa 9.12.6.140 -sp 1080
```

on the Managed server:

```
./createMS.sh ewlMS -ma 9.12.10.150 -mp 5773 -auth None
```

In Figure 3-4 a SOCKS server protects the zone where the firewall broker and managed servers are located and another SOCKS server protects the zone where the domain manager is located.

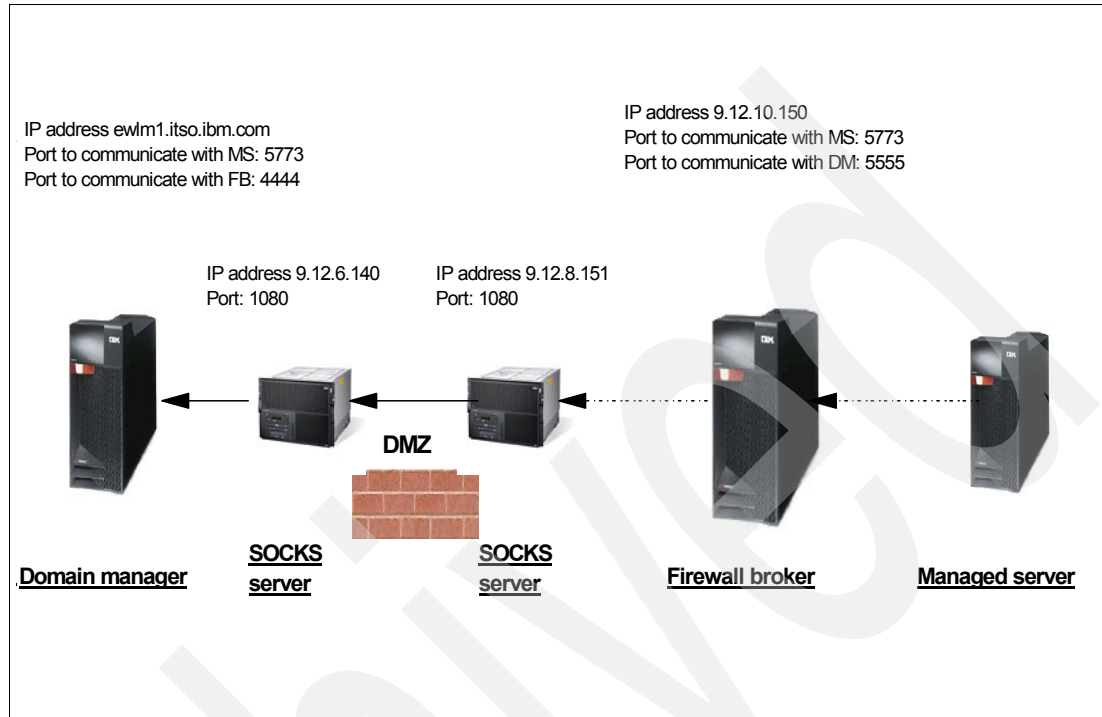


Figure 3-4 SOCKS server with DMZ

This configuration is similar to the one described in Figure 3-3 on page 84, with the difference that you must also set the `-sa` and `-sp` parameters on the `changeDM` script to identify the IP address and port of the SOCKS server that is protecting the zone where the domain manager is located.

The sample definition is shown in Example 3-6.

Example 3-6 Sample commands to define SOCKS server configuration in DMZ

```

on the Domain manager:
./changeDM.sh ewlDM -ma 9.12.6.141 -mp 5773 -fp 4444 -fb 9.12.10.150:5555:SOCKS
    -sa 9.12.4.140 -sp 1080

on the Firewall broker:
./createFB.sh ewlFB -ma 9.12.10.150 -mp 5773 -da 9.12.6.141 -dp 4444 -fp 5555
    -auth None
./changeFB.sh ewlFB -sa 9.12.8.151 -sp 1080

on the Managed server:
./createMS.sh ewlMS -ma 9.12.10.150 -mp 5773 -auth None

```

Handling the firewall broker

As you have seen briefly in the sample, you need to run the `createFB` script to create and configure a firewall broker and the `changeFB` script to change the firewall broker configuration. You can create the firewall broker on the managed server or some other server in the same trusted zone as the managed servers. This script and all of its parameters are supported on all the platforms.

For the command syntax, refer to the InfoCenter at:

<http://public.boulder.ibm.com/eserver/>

3.1.3 ITSO EWLM firewall configurations

This section shows the two firewall configurations we used in our ITSO environment. The StoneGate firewall was set up between the HTTP Plug-in managed server and the rest of the managed servers and domain manager. In a production environment, there would probably be more firewalls in place, but this will demonstrate the managed server and the domain manager communication across a firewall. Therefore, in Figure 3-5, the only managed server to domain manager communication across the firewall is from the HTTP Plug-in managed server to the domain manager.

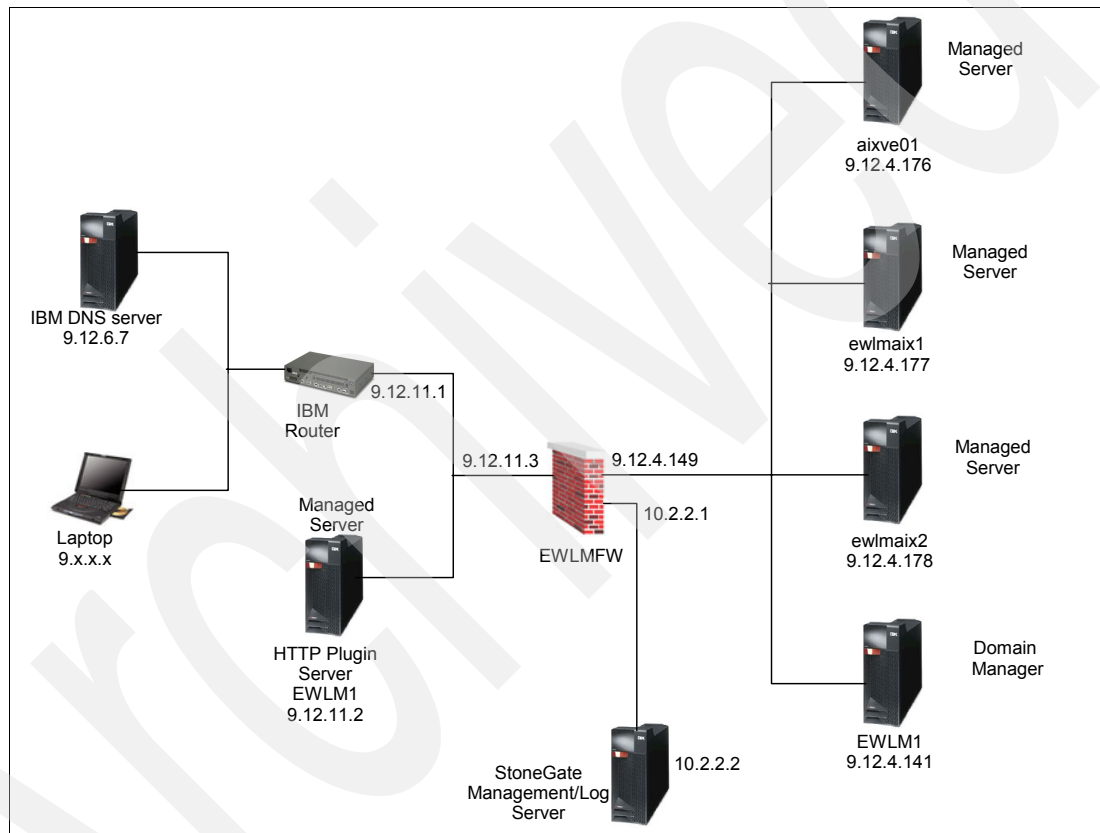


Figure 3-5 Stateful inspection firewall lab configuration

Since there is no Proxy, no additional EWLM configuration needs to take place, but as highlighted in 3.1.2, “EWLM firewall support” on page 79, the stateful inspection firewall rules may need to be updated to include this traffic.

Two rules were added to the StoneGate firewall:

- ▶ To allow traffic to flow through the firewall from the HTTP Plug-in managed server on EWLM1 to the domain manager using port 5773.
- ▶ Since we are connecting a browser from Laptop 9.x.x.x to the Control Center and also to the WebSphere Application Server Administration Console, we added this traffic to the Access rule specifying ports 5775, 5777, 5776, and 5778.

Once the StoneGate firewall was in place and we were sure the Management Domain was communicating as expected, we introduced an open source Proxy server to our configuration in addition to the StoneGate firewall, as shown in Figure 3-6.

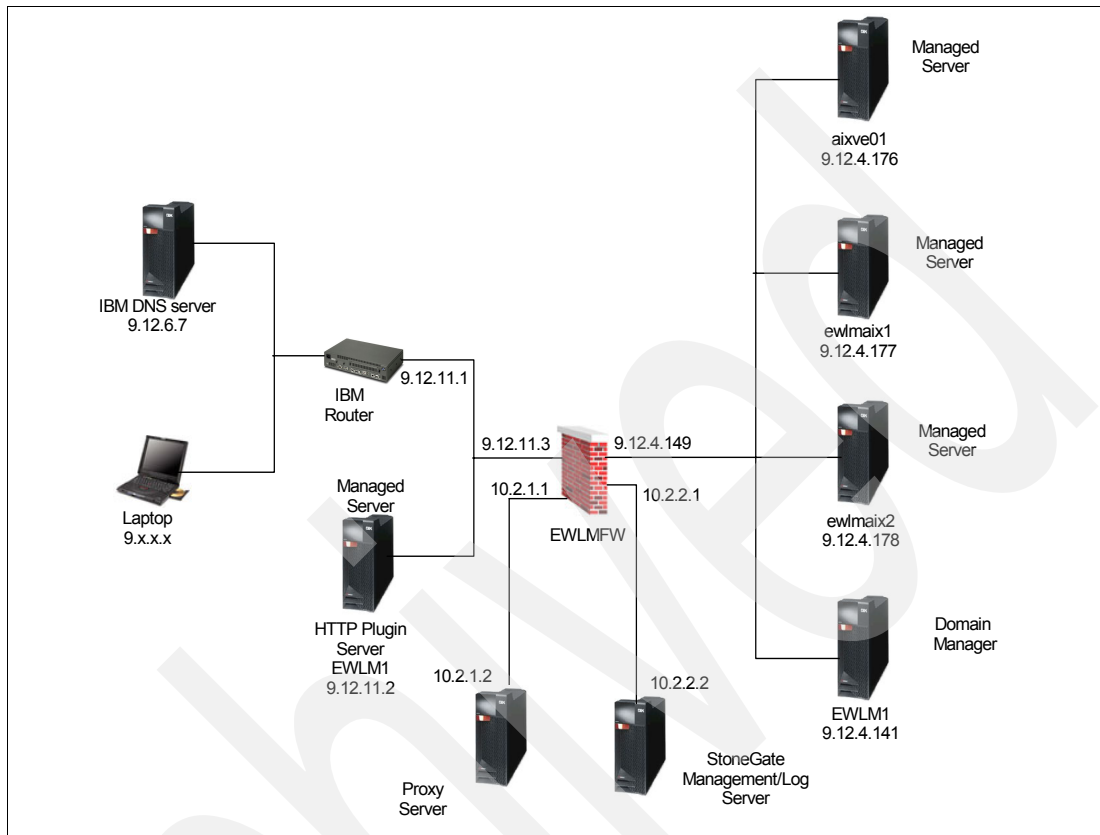


Figure 3-6 Proxy firewall ITSO configuration

Including the Proxy firewall in this configuration required changes to the HTTP Plug-in managed server. The HTTP Plug-in managed server now needed to communicate with the Proxy and then the Proxy communicated with the domain manager on its behalf. The other managed servers did not need to be changed because they are in the same network as the domain manager. In a production environment, this probably would not be the case. In fact, the domain manager might be in its own system management security zone protected by a firewall.

In order to set up communications for the Proxy, first we had to alter the StoneGate firewall rules to allow traffic through the Proxy from the HTTP Plug-in managed server.

Now the HTTP Plug-in managed server must change to communicate with the Proxy instead of directly to the domain manager. The **changeMS** command is executed at the managed server. The address and port of the Proxy server are required.

Example 3-7 Managed server configuration for Proxy server

```
C:\Program Files\IBM\VE\EWLMMS\bin>changeMS ewlMMS -va 10.2.1.2 -vp 8080
```

```
Processing changeMS request, Please be patient as this may take a while...
```

```
PROCESSING COMPLETE
```

You can now execute a `displayMS` command that shows the proxy address and proxy port that the managed server is communicating (ViaProxyPort and ViaProxyHost parameters).

In order to check whether the managed server and domain manager are communicating as expected, sign into the Control Center with a user ID that has Monitor access and select **Managed servers** from the Monitor menu in the left pane. This will give you a list of all the managed servers in the domain and their states.

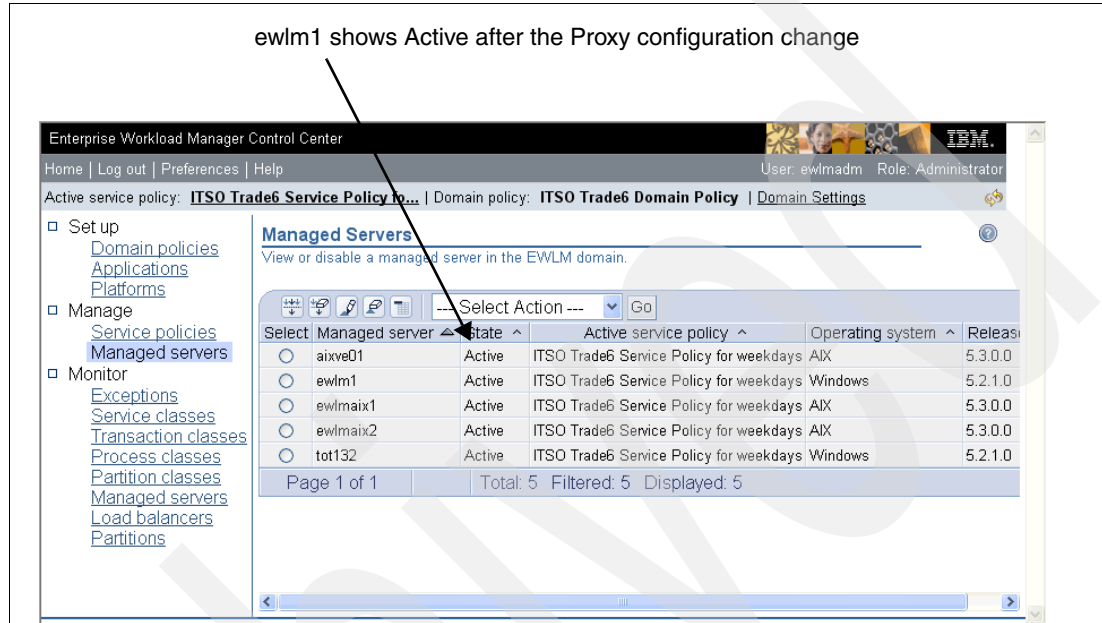


Figure 3-7 Managed servers monitor view

3.2 Security

In our sample scenario, we had the Trade6 workload running through three tiers of servers: IBM HTTP server, WebSphere Application Server, and Database server. When we added the Enterprise Workload Manager code to the installation, we added the capability to monitor and report the end-to-end transaction behavior. This means there is a flow of EWLM information that goes from the managed server to the domain manager and vice versa, and from the domain manager to the user interface. This information about transactions, as shown in Figure 3-8 on page 90, crosses security environments: Browser, domain manager, and managed server. In such an environment there is sensitive information that you need to consider protecting — mainly the Domain Policy definitions and the connections between the domain manager and the managed servers — to avoid intruders. The efforts here, from a security point of view, are not to authenticate the partner but to encrypt the data travelling between the different end points.

Different levels of encryption can also be implemented, especially between the domain manager and the managed servers, where you can implement Server SSL authentication or Client/Server SSL authentication.

Between the Web browser and domain manager you can implement only Server SSL authentication; between the domain manager and the managed server you can implement different security options depending on your installation: None, ServerSSL, and ClientServer SSL.

There are two keystores that the EWLM Administrator needs to consider:

- ▶ The first one protects the connection between the Web browser and the EWLM Control Center.
- ▶ The second one is used for two purposes:
 - To protect communications between the domain manager and managed servers
 - To protect communications between the domain manager and a Load Balancer

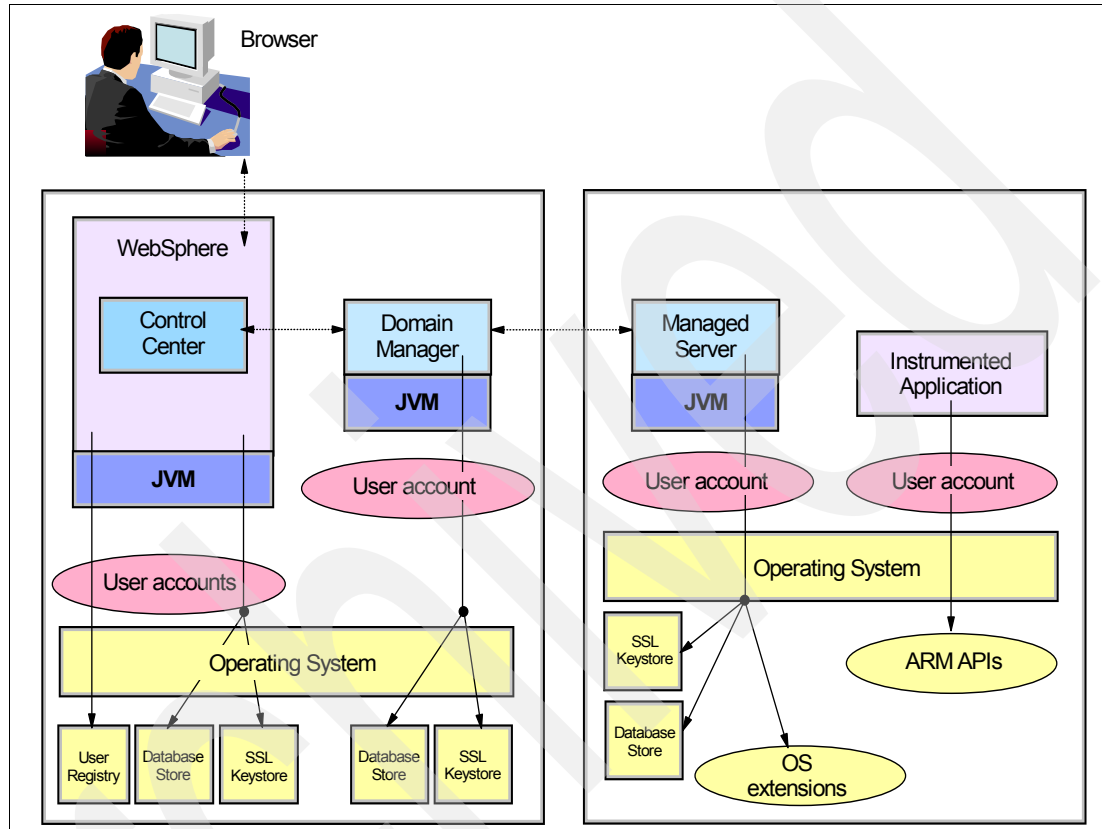


Figure 3-8 Security context and boundaries

The following sections describe which actions are already in place when you install EWLM and what steps are required to further protect the EWLM data travelling across the different domain boundaries:

- ▶ Browser to Control Center
- ▶ Control Center to domain manager
- ▶ Domain manager to managed servers
- ▶ Domain manager and Load Balancer

Note: Once you establish your certificates, remember to set up file system protections to limit access to the keystore file. After the configuration is established, access should be limited to the user ID under which the Control Center's server instance runs.

3.2.1 Browser to Control Center

Figure 3-9 on page 91 shows an EWLM user logging into the Control Center through the browser via a secure login.

In this portion of the data exchange, EWLM performs SSL encryption of the conversation to protect the content of the data between the browser and the Control Center, and the Control Center application performs authentication of the user by verifying that the user ID and password logging in are defined and valid to the application.

For the authentication of the users, and before this login can happen, someone — typically the system administrator responsible for installing and configuring EWLM and for managing its users — must establish the user in the user registry and add the user to a Control Center role. Your Control Center site has already been created using the `createDM` command, which creates the Control Center along with the domain manager. The command establishes the user ID passed in the `adminUser` parameter as an administrator for the WebSphere Application Server instance established for the site.

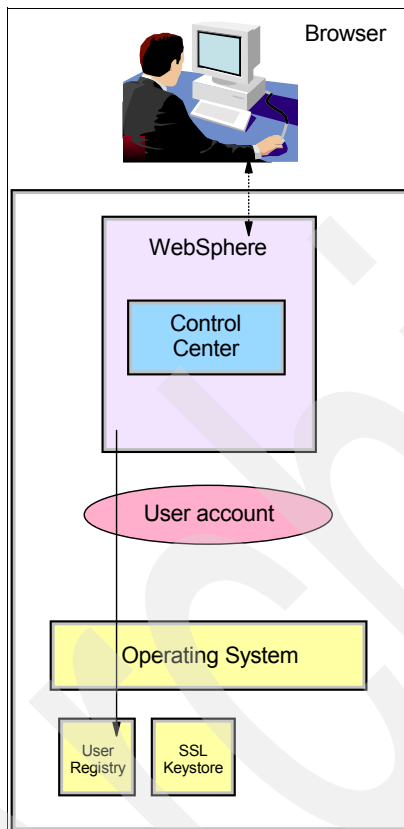


Figure 3-9 Communication between the browser and the Control Center

If we look at the SSL connection, EWLM establishes an SSL connection between the browser and the Control Center, initially using a default pair of public/private keys.

When you log in to the Control Center (for example, in our configuration using the initial port 5775), you notice that EWLM immediately routes your connection on the SSL port 5777 and the conversation becomes SSL-encrypted from there on.

Also, when you perform the initial log in from your browser, you receive a security alert during the sign-on process asking you to accept the default certificate, as shown in Figure 3-10.



Figure 3-10 Security Alert at login

For your installation, we suggest that the EWLM administrator configure EWLM for Server SSL and use a certificate authority to create and install the certificate on the browser. In our configuration, after we created a default login, we acquired a secure (non-default) public/private keypair and used the `changeCC` command to configure EWLM to use the keypair for Server SSL between the Control Center and the browser. Your installation can use the same procedure we used in our environment, or you can use a certificate authority to create certificates instead of using the public/private keypair.

Here are the tasks required to change from a default security setting:

- ▶ Create a keystore.
- ▶ Install the keystore in WebSphere.
- ▶ Trust the new certificate.

Create a keystore

We used the Java SDK tool to generate a certificate with our new key. As we mentioned previously, this step is probably not going to take place in your installation. Most likely this will be replaced by simply acquiring a key/certificate from a certificate authority.

As shown in Example 3-8, in our configuration we generated the keystore `ks` and stored it in the `/home/ibmewlm` directory. This directory has limited access to ensure security of the keystore.

Example 3-8 Keystore creation

```
[ibmewlm@ewlmdm1 ibmewlm]$ pwd
/home/ibmewlm
[ibmewlm@ewlmdm1 ibmewlm]$ keytool -genkey -alias ewlmdm1 -keyalg RSA -keystore ks
-validity 365
Enter keystore password: 111111
What is your first and last name?
  [Unknown]: ewlmdm1
What is the name of your organizational unit?
```

```
[Unknown]: itso
What is the name of your organization?
[Unknown]: IBM
What is the name of your City or Locality?
[Unknown]: Poughkeepsie
What is the name of your State or Province?
[Unknown]: NY
What is the two-letter country code for this unit?
[Unknown]: US
Is <CN=ewlmdm1, OU=itso, O=IBM, L=Poughkeepsie, ST=NY, C=US> correct?
[no]: yes

Enter key password for <ewlmdm1>
(RETURN if same as keystore password): 111111
```

As a result of this command, the keystore *ks* has been successfully created. The next step is to switch the EWLM from using the default key to the key provided in the certificate just created.

Install the keystore in WebSphere

When you issue the **changeCC** command, you need to specify the level of security, in our case `adminDefined` instead of the default `ewlmDefined`, and the keystore name, location, and password. With this command, the keystore will be installed in the Control Center WebSphere instance.

Example 3-9 ChangeCC command to enable security

```
[ibmewlm@ewlmdm1 bin]$ ./changeCC.sh -sslSecurity EWLMDM/ -adminUser wasadmin -adminPW itso
-level adminDefined -keystore /home/ibmewlm/ks -keystorePW 111111
```

```
Processing changeCC -sslSecurity request. Please be patient as this may take
a while...
```

```
...processing 33% complete:
...processing 66% complete
PROCESSING COMPLETE
```

If the WebSphere instance was active at the time, the **changeCC** command restarts the WebSphere instance after the keystore information has been updated. If the WebSphere instance was not active when the **changeCC** command was issued, then the WebSphere instance is updated such that the next time it is started, it will use the new keystore information.

Trust the new certificate

This section is not needed if the certificate is from a certificate authority that the Web browser already trusts. The only time you need to trust the certificate is when it is not currently in your Web browser's truststore.

The first time you access the Control Center from a browser, you still receive the Security Alert warning shown in Figure 3-11, but this time only the first item should have a warning symbol because now the certificate should match the name of your Control Center site.



Figure 3-11 Securing warning after certificate installation

By clicking **View Certificate**, you can see the certificate details and also launch the wizard to install the certificate on your workstation.

The **Install Certificate** option starts the wizard, as shown in Figure 3-12.

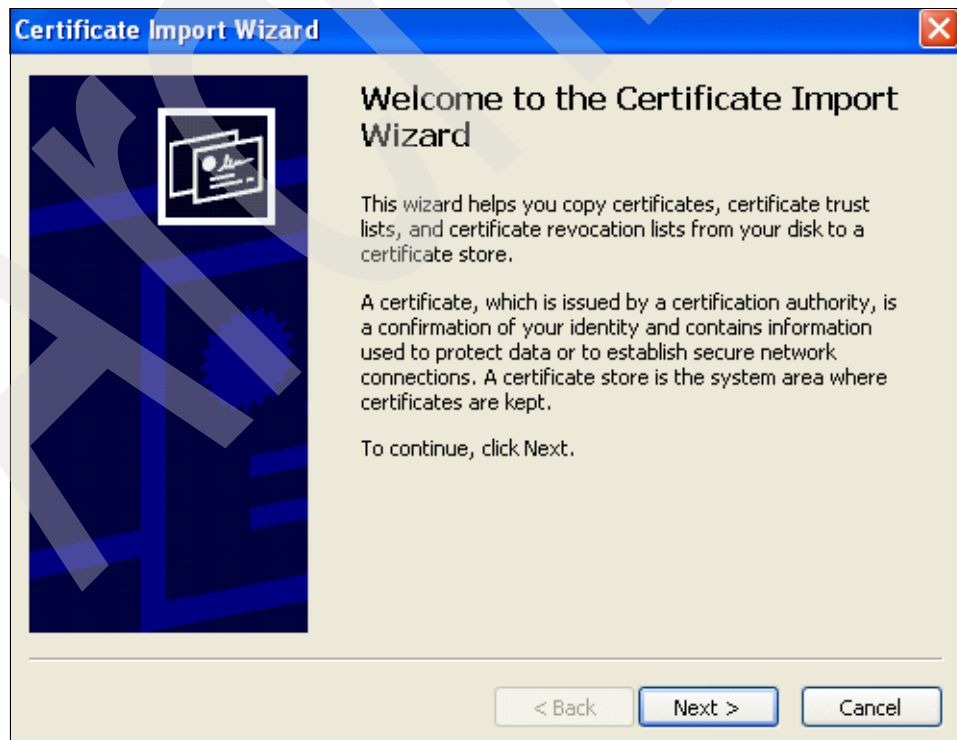


Figure 3-12 Certificate Import Wizard

We take the defaults through the wizard by clicking **Next** on each panel. On the last panel, we click **Finish**, which brings up a confirmation screen like the one shown in Figure 3-13.

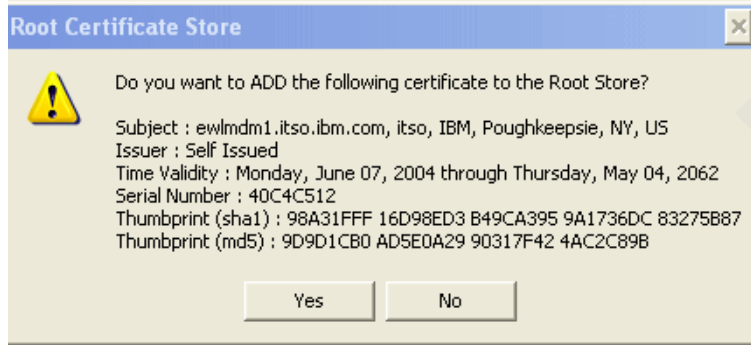


Figure 3-13 Certificate import screen

After replying **Yes**, the certificate is imported on your workstation, a message of successful completion is generated on your screen, and the next time you connect to the Control Center you should not receive the Alert Warning pop-up any longer. This is because the browser is now capable of starting an SSL conversation with the Control Server after it has exchanged the public key provided with the installed certificate.

3.2.2 Control Center to domain manager

As indicated in Figure 3-14, the Control Center and domain manager communicate through a port. The port is defined via the `-jp` parameter on the `createDM`.

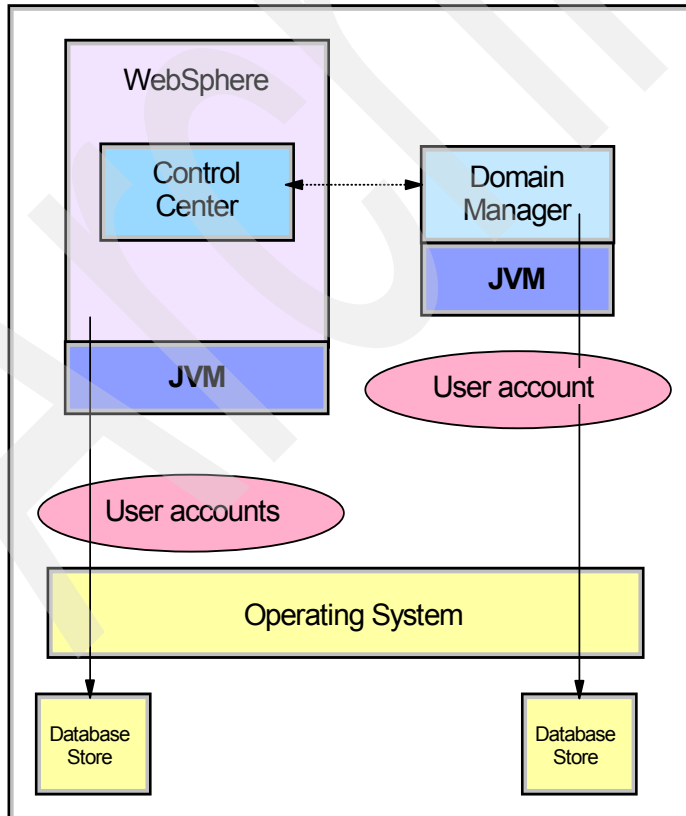


Figure 3-14 Communication between Control Center and domain manager

The communications between the EWLM Control Center and the domain manager are protected by secure interprocess communications (IPC).

Note: On all systems, external access to the `-jp` port should be blocked. On systems that provide means to restrict internal access to ports, the `-jp` port should be limited to the user ID running the Control Center server instance process and to the user ID running the domain manager process.

3.2.3 Domain manager to managed servers

Figure 3-15 shows the domain manager and managed servers communicating through addresses and ports. SSL encryption provides confidentiality of the data that flows between them.

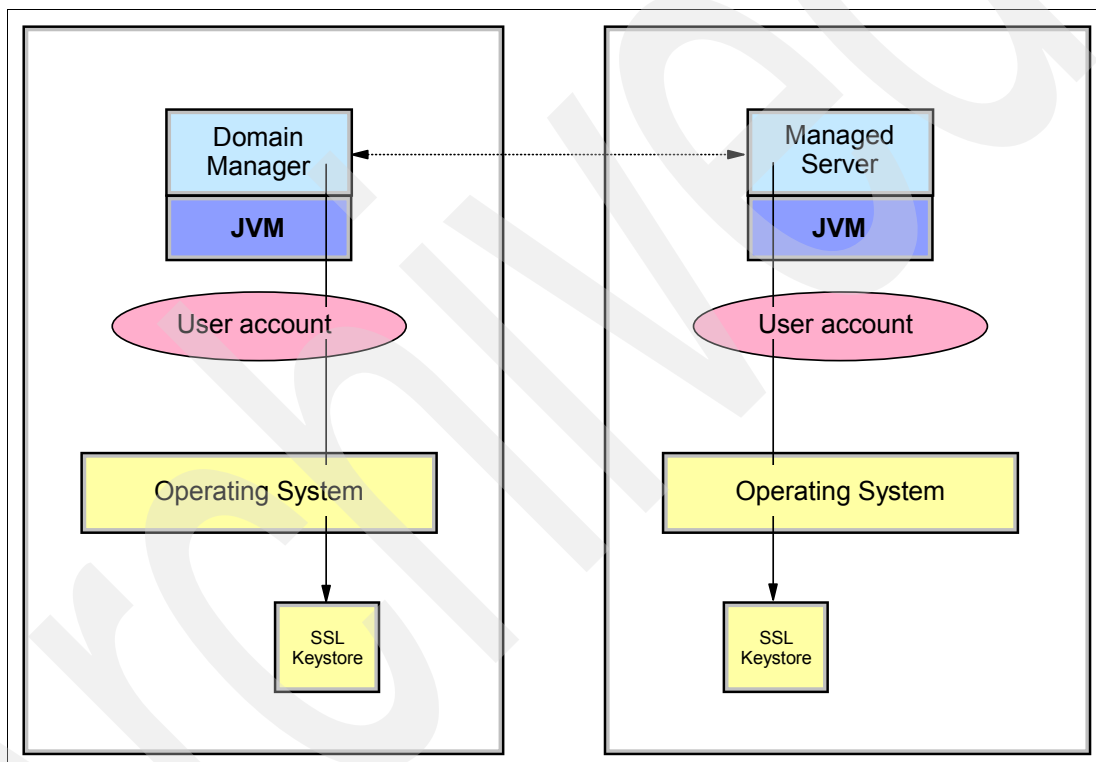


Figure 3-15 Communication between domain manager and Managed Server

SSL, when configured to do so, also provides authentication of the servers to each other. This SSL authentication uses public/private keypairs in keystore files and keystore passwords passed in with the **Change** and **Create** commands.

Depending on your installation configuration and your security requirements, there are two main ways that you can implement security in this part of the configuration of the EWLM domain: Server SSL security and Client/Server SSL security. The following sections explain these implementations.

Note: You need to set up file system protections on the domain manager and managed servers to limit access to the keystore files. After the configuration is established, access should be limited to the user IDs each of the servers run under.

Server SSL security

This configuration is suggested for environments where authentication is not a strong requirement. In fact, the domain manager is the element that owns the trusted certificate. In this configuration, the acquired certificate needs to be imported on the managed server so the public key can be stored in the keystore file. This is because the managed server is a passive user (meaning that there is no browser that can accept the certificate on behalf of the managed server).

We performed the following tasks to enable this security environment in our configuration:

1. Create the keystore for the domain manager on the domain manager system. We called this file `ks_dm` and stored it in the current directory, as shown in Example 3-10.

Example 3-10 Creating a keystore on domain manager

```
[ibmewlm@ewlmdm1 ibmewlm]$ keytool -genkey -alias dm -keyalg RSA -keystore ks_dm -validity 365
```

```
Enter keystore password: 111111
What is your first and last name?
  [Unknown]: ewlmdm1.itso.ibm.com
What is the name of your organizational unit?
  [Unknown]: itso
What is the name of your organization?
  [Unknown]: IBM
What is the name of your City or Locality?
  [Unknown]: Poughkeepsie
What is the name of your State or Province?
  [Unknown]: NY
What is the two-letter country code for this unit?
  [Unknown]: US
Is <CN=ewlmdm1.itso.ibm.com, OU=itso, O=IBM, L=Poughkeepsie, ST=NY, C=US> correct?
  [no]: yes

Enter key password for <dm>
  (RETURN if same as keystore password): 111111
```

2. After the keystore has been created, we exported the public key `ks_dm` into the `ks.cerr` certificate.

Example 3-11 Exporting the keystore

```
[ibmewlm@ewlmdm1 ibmewlm]$ keytool -export -alias dm -keystore ks_dm -rfc -file ks.cert
Enter keystore password: 111111
Certificate stored in file <ks.cert>
```

3. Next, we imported the public key back into the domain manager keystore so the domain manager can perform internal communication.

Example 3-12 Importing the keystore

```
[ibmewlm@ewlmdm1 ibmewlm]$ keytool -import -alias dmSELF -file ks.cert -keystore ks_dm
Enter keystore password: 111111
Certificate already exists in keystore under alias <dm>
Do you still want to add it? [no]: yes
Certificate was added to keystore
```

4. At this point, we are ready to send (in binary) the certificate `ks.cert` to all the managed nodes in our configuration.
5. On each managed node, we then import the certificate `ks.cert` into the managed node keystore. Example 3-13 shows the import step on the `ewlm4` server.

Example 3-13 Import of certificate on managed server

```
root@ewlm4:/opt/IBM/VE/EWLMMS/bin> keytool -import -alias dm -file ks.cert -keystore ks_ms

Enter keystore password: 111111
Owner: CN=ewlmdm1.itso.ibm.com, OU=itso, O=IBM, L=Poughkeepsie, ST=NY, C=US
Issuer: CN=ewlmdm1.itso.ibm.com, OU=itso, O=IBM, L=Poughkeepsie, ST=NY, C=US
Serial number: 40c5c68a
Valid from: 6/8/04 10:00 AM until: 5/5/62 5:37 PM
Certificate fingerprints:
    MD5: 0E:86:DF:CB:80:83:54:61:C2:27:7A:D8:0D:8E:FA:0B
    SHA1: 4C:0E:0F:99:50:7A:A8:FC:D8:1A:8A:9F:DO:84:DD:BD:75:E6:C8:E8
Trust this certificate? [no]: yes
Certificate was added to keystore
```

As a result of this command, we were asked if we trusted the certificate, and then the public key was saved in the managed server keystore.

At this point both domain manager and managed server have the key to be able to start using Server SSL security; we just need to tell the EWLM configuration to switch from using no security to Server SSL.

On the domain manager (see Example 3-14):

1. Stop the domain manager.
2. Issue the **changeDM** command to alter the security environment.
3. Issue the **displayDM** command to verify the changes are in place.
4. Start the domain manager.

Example 3-14 Enabling domain manager security

```
[root@ewlmdm1 bin]# ./changeDM.sh EWLMMDM -auth ServerSSL -sslks /home/ibmewlm/ks_dm -sslpw
111111

Processing changeDM request. Please be patient as this may take a while...
PROCESSING COMPLETE

[root@ewlmdm1 bin]# ./startDM.sh EWLMMDM
```

At this point all the connected managed servers will need to be stopped if active in order to make the changes. The next step is to alter their security configuration.

On each managed server (see Example 3-15):

1. Issue the **changeMS** command to alter the security environment.
2. Issue the **displayMS** command to verify the changes are in place.
3. Start the managed node.
4. Verify that the connection status is Active on the EWLM Control Center under the Monitoring → Managed Servers task.

Example 3-15 EWLM4 managed server

```
root@ewlm4:/opt/IBM/VE/EWLMMS/bin>./changeMS.sh ewlmMS -auth ServerSSL -sslks
/usr/java14/bin/ks_ms -sslpw 111111 <
```

Processing changeMS request. Please be patient as this may take a while...

PROCESSING COMPLETE

Client/Server SSL security

This configuration focuses on both authentication and encryption of data. In this configuration, each managed server owns its own private key.

The first steps for setting up Client/Server SSL are identical to those for setting up Server SSL described in the previous section. Once we have the Domain Manager and all the managed servers set up for Server SSL, we perform the following tasks on each managed server in order to create the public/private key:

1. We create a public/private keypair for the managed server and place it in the existing `ks_ms` keystore used to hold the public key of the Domain Manager. Make sure that the keypair alias is unique across all managed servers.

Example 3-16 Acquiring public/private key on managed server

```
root@ewlm4:/opt/IBM/VE/EWLMMS/bin> keytool -genkey -alias ms1 -keyalg RSA -keystore ks_ms
-validity 365
```

```
Enter keystore password: 111111
What is your first and last name?
  [Unknown]: ewlm4
What is the name of your organizational unit?
  [Unknown]: itso
What is the name of your organization?
  [Unknown]: IBM
What is the name of your City or Locality?
  [Unknown]: Poughkeepsie
What is the name of your State or Province?
  [Unknown]: NY
What is the two-letter country code for this unit?
  [Unknown]: US
Is <CN=ewlm4, OU=itso, O=IBM, L=Poughkeepsie, ST=NY, C=US> correct?
  [no]: yes
```

```
Enter key password for <ms1>
(RETURN if same as keystore password): 111111
```

2. We then export the public key into a certificate (Example 3-17).

Example 3-17 Exporting the key

```
root@ewlm4:/opt/IBM/VE/EWLMMS/bin> keytool -export -alias msl-keystore ks_ms -rfc -file
ks_ms1.cert
Enter keystore password: 111111
Certificate stored in file <ks_ms1.cert>
```

3. We send the certificate (ks_ms1.cert) to the Domain Manager as a binary file.

4. We import the certificate (ks_ms1.cert) into the Domain Manager's keystore. In our configuration the keystore is called ks_dm.

Example 3-18 Import the certificate on the Domain Manager

```
[ibmewlm@ewlmdm1 ibmewlm]$ keytool -import -alias msl -file ks_ms1.cert -keystore ks_dm

Enter keystore password: 111111
Owner: CN=ewlm4, OU=itso, O=IBM, L=Poughkeepsie, ST=NY, C=US
Issuer: CN=ewlm4.itso.ibm.com, OU=itso, O=IBM, L=Poughkeepsie, ST=NY, C=US
Serial number: 40c5d68b
Valid from: 6/9/04 10:00 AM until: 6/9/05 5:37 PM
Certificate fingerprints:
    MD5: 0E:86:DF:CB:80:83:54:61:C2:27:7A:D8:0D:8E:FA:0B
    SHA1: 4C:0E:0F:99:50:7A:A8:FC:D8:1A:8A:9F:D0:84:DD:BD:75:E6:C8:E8
Trust this certificate? [no]: yes
Certificate was added to keystore
```

5. We repeat this for each managed node we have in our ITSO configuration.

At this point we need to switch from ServerSSL security to ClientServerSSL security.

On the domain manager (see Example 3-19):

1. Stop the domain manager.
2. Issue the **changeDM** command to alter the security environment.
3. Issue the **displayDM** command to verify the changes are in place.
4. Start the domain manager.

Example 3-19 Enabling Client/Server SSL on Domain Manager

```
[root@ewlmdm1 bin]# ./changeDM.sh EWLMMDM -auth ClientServerSSL -sslks /home/ibmewlm/ks_dm
-sslpw 111111

Processing changeDM request. Please be patient as this may take a while...
PROCESSING COMPLETE

[root@ewlmdm1 bin]# ./startDM.sh EWLMMDM
```

On each managed server (see Example 3-20):

1. Issue the **changeMS** command to alter the security environment.
2. Issue the **displayMS** command to verify the changes are in place.
3. Start the managed server.
4. Verify that the connection status is Active on the UI under Monitoring → Managed Servers task.

Example 3-20 Enabling Client/Server SSL on managed server

```
root@ewlm4:/opt/IBM/VE/EWLMMS/bin>./changeMS.sh ewlmMS -auth ClientServerSSL -sslks
/usr/java14/bin/ks_ms -sslpw 111111 <
```

```
Processing changeMS request. Please be patient as this may take a while...
```

```
PROCESSING COMPLETE
```

```
root@ewlm4:/opt/IBM/VE/EWLMMS/bin>./startMS.sh ewlmMS
```

3.2.4 Security considerations for EWLM processes

The EWLM server processes read and write persistent information maintained in the internal database and the diagnostic folders and files.

The folders and files are created at configuration time in the working directories associated with specific domain manager and managed server instances. In general, public access to these directories should be avoided. Where necessary, ownership is assigned or appropriate access is granted to the user IDs and accounts these processes are configured to run under by default on the various system types.

You might choose to enable other specific user IDs and accounts to read the diagnostic information, or possibly to run the processes by granting appropriate access to these directories and files.

The EWLM managed server process also requires access to use specific application interfaces. This access depends on the capabilities of the user ID or account it is running under, which in turn depends on the operating system:

- ▶ On AIX: The managed server must either run as root or as a non-root user that has the CAP_EWLM_AGENT capability.
- ▶ On i5/OS: The STRWLM CL command runs the managed server under the QWLM2 profile, with the necessary access rights.
- ▶ On Solaris: The managed server must be run by either root or a user that has an effective UID (EUID) of 0.
- ▶ On Windows: The managed server must run either as an application (started with the **StartMS.bat** command under an account in the Administrators group), or as a service (started with the **StartMSService.bat** command, which also must be run under an account in the Administrator's group).

3.2.5 Security considerations for instrumented applications

In order to be able to participate in an end-to-end transaction monitoring schema, all the middleware applications need to be able to use ARM application interfaces. Their ability to successfully use the ARM APIs is usually determined by the capabilities of the user ID or account running them. This depends on the operating system hosting the infrastructure:

- ▶ On AIX: The application must either run as root or as a user that has the CAP_ARM_APPLICATION capability.
- ▶ On i5/OS: The application must run under a user ID that has been granted authority to use the QSYS2/LIBARM4 service program, or the application must be owned by a user ID that has this authority and it must adopt the owner's authority.
- ▶ On Solaris: The application must be run by either root or a user listed in the application_auth list in the file /etc/EWLM/auth.
- ▶ On Windows: The application must run in the Local System Account or in an account that belongs to the Administrators group or the EWLMArm4Users group.

Enabling middleware for EWLM

In this chapter we describe procedures to ARM enable the supported IBM middleware, thereby providing an end-to-end view of transaction flow at the EWLM Control Center. The end-to-end view of transaction flow can be possible using Application Response Measurement (ARM) instrumentation. Some middlewares provide ARM instrumentation support on the products.

The specific middleware products covered in this chapter are:

- ▶ DB2 Universal Database V8.2
- ▶ WebSphere Application Server V5.1.1 and V6
- ▶ IBM HTTP Server V1.3.28, V2.0.47, and V6

4.1 Overview

To construct data that can be displayed by the EWLM Domain Manager, each middleware application that you want EWLM to monitor must be instrumented according to the ARM 4.0 standards. An instrumented middleware application cooperates with the EWLM layer on their local operating system so it can understand transaction segment elapsed time. This is part of the data sent to the EWLM domain manager that collects elapsed time. The domain manager collects the data and displays it to the user through the EWLM Control Center. Once the middleware that is hosting your workload is instrumented, you can monitor your application's performance statistics using EWLM Control Center without modifying the application.

This chapter describes how to instrument the following software:

- ▶ DB2 Universal Database V8.2
- ▶ WebSphere Application Server V5.1.1 and V6
- ▶ IBM HTTP Server V1.3.28, V2.0.47, and V6

The middleware products we describe here have been enhanced by IBM to call the ARM instrumentation APIs, so all that is needed is to instruct these products to turn on this capability.

In the following sections we discuss how to enable this capability in these products. Table 4-1 provides the middleware application environment variables that we will be using along with their platform-specific default values.

Table 4-1 Environment variables - Default values

Variable	Component	Platform	Default value
<WAS_HOME>	WebSphere Application Server	AIX	/usr/IBM/WebSphere/AppServer
		i5/OS	/QIBM/ProdData/webas51
		z/OS	- For WebSphere Application Server V6.0.1 /usr/lpp/zWebSphere/V6R0 - For WebSphere Application Server V5.1.1 /usr/lpp/zWebSphere/VrR1M0
		Windows	C:\Program Files\WebSphere\AppServer
		Solaris	/opt/WebSphere/AppServer
		HP-UX	/opt/IBM/WebSphere/AppServer
<EWLMMS_HOME>	EWLM Managed Server	AIX	/opt/IBM/VE2/EWLMMS
		i5/OS	/QIBM/ProdData/VE2/EWLMMS
		z/OS	/usr/lpp/VE_R2/EWLMMS
		Windows	C:\Program Files\IBM\VE2\EWLMMS
		Solaris	/opt/IBM/VE2/EWLMMS
		HP-UX	/opt/IBM/VE2/EWLMMS

Variable	Component	Platform	Default value
<EWLMMS_LIBPATH>	EWLM Managed Server ARM implementation libraries	AIX	/usr/lpp/ewlm/java/lib
		i5/OS	/QSYS.LIB/QWLM2.LIB
		z/OS	/usr/lpp/VE_R2/EWLMMS/classes
		Windows	<EWLMMS_HOME>\classes\ms
		Solaris	usr/lib
		HP-UX	usr/lib
<IHS13_HOME>	IBM HTTP Server V1.3.28	AIX	/usr/IBMHttpServer
		i5/OS	/QIBM/ProdData/HTTPA
		Windows	C:\Program Files\IBMHttpServer
		Solaris	/opt/IBMHttpServer
		HP-UX	/opt/IBMHttpServer
<IHS20_HOME>	IBM HTTP Server V2.0.47	AIX	/usr/IBMIHS20
		i5/OS	/QIBM/ProdData/HTTPA
		Windows	C:\Program Files\IBM HTTP Server 2.0
		Solaris	/opt/IBMIHS
		HP-UX	/opt/IBMIHS
<IHS6_HOME>	IBM HTTP Server V6	AIX	/usr/IBMIHS
		i5/OS	/QIBM/ProdData/HTTPA
		Windows	C:\Program Files\IBM HTTP Server
		Solaris	/opt/IBMIHS
		HP-UX	/opt/IBMIHS

4.2 Pre-instrumentation tasks

Before enabling any middleware, an EWLM Managed Server needs to be installed on each server operating system. In addition, a Domain Manager needs to be installed on one server in the EWLM domain that will serve as the EWLM control point.

Table 4-2 on page 106 is a checklist of the tasks you should complete before starting instrumentation. We do not describe how to install IBM HTTP Server, WebSphere Application Server, and DB2 Universal Database in this book, but explain how to enable the instrumentation provided by these middleware products.

For information about installing IBM WebSphere Application Server Network Deployment V5.1 or V6 and IBM HTTP Server, refer to the WebSphere Application Server Version 5.1 or Version 6 Information Center at:

<http://publib.boulder.ibm.com/infocenter/ws51help/>
<http://publib.boulder.ibm.com/infocenter/ws60help/>

For information about installing DB2 Universal Database, refer to the DB2 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2help/>

Table 4-2 Pre-instrumentation tasks

Role	Domain manager	Web server	Application server	Database server
Install and configure EWLM Domain Manager.	X			
Configure EWLM Control Center.	X			
Create users and assign roles.	X			
Install and configure IBM HTTP Server.		X		
Install and configure Web server plug-in.		X		
Install and configure WebSphere Application Server V5.1.1 or V6.			X	
Install and configure DB2 Universal Database V8.2 or later.				X
Install EWLM Managed Server.		X	X	X
Create new EWLM Managed Server.		X	X	X
Instrument operating system.		X	X	X
Verify operating system instrumentation.		X	X	X
Verify communication between Domain Manager and Managed Server.	X	X	X	X

4.3 Enabling DB2 Universal Database for ARM

Enabling the ARM instrumentation for the DB2 Universal Database is performed differently for each operating system platform; therefore, we specifically describe the steps for each platform.

AIX

If you have DB2 Version 8.2 FixPak9 or later, then ARM instrumentation is enabled by default. You can check the level of DB2 using the `db21eve1` command with the db2 instance, as shown in Example 4-1.

Example 4-1 Check DB2 version and fix level

```

$ db21eve1
DB21085I Instance "db2inst1" uses "32" bits and DB2 code release "SQL08022"
with level identifier "03030106".
Informational tokens are "DB2 v8.1.1.89", "0D_14086", "U800789_14086", and
FixPak "9".
Product is installed at "/usr/opt/db2_08_01".

```

However, you should also check whether the ARM instrumentation has been enabled on the system using the `lsarm` command, as shown in Example 4-2.

Example 4-2 Check if ARM is enabled and it returns not enabled

```
root@aixve01: /> lsarm -a
There are no ARM applications defined.
```

If the results of the `lsarm` command indicate there are no ARM applications defined, then you can enable the instrumentation of the DB2 Universal Database on AIX by following these steps:

1. Set AIX authorization attributes.
2. Set `DB2LIBPATH` variable.

Set AIX authorization attributes

On AIX, the ARM platform support uses the native OS security model. In order for an application to ARM register itself, you have to add two capabilities, `CAP_ARM_APPLICATION` and `CAP_PROPAGATE`, to the user profile of the instance owner who runs the application. The root user automatically has all of the capabilities but not the DB2 instance owner, for example, `db2inst1`. To provide the capabilities to the DB2 instance owner:

1. Start a terminal session.
2. Log in as root user.
3. Add the capabilities to the DB2 instance owner ID `db2inst1` using `chuser` and verify it using `lsuser`. Follow the sequence shown in Example 4-3.

Example 4-3 Add ARM-related capabilities to user db2inst1

```
root@aixve01: /> chuser "capabilities = CAP_ARM_APPLICATION,CAP_PROPAGATE" db2inst1
root@aixve01: /> lsuser db2inst1
db2inst1 id=109 pgrp=db2grp1 groups=db2grp1,staff,dasadm1 home=/home/db2inst1
shell=/usr/bin/ksh login=true su=true rlogin=true daemon=true admin=false sgroups=ALL
admgroups= tpath=nosak ttys=ALL expires=0 auth1=SYSTEM auth2=NONE umask=22 registry=files
SYSTEM=compat logintimes= loginretries=0 pldwarranty=0 account_locked=false minage=0
maxage=0 maxexpired=-1 minalpha=0 minother=0 mindiff=0 maxrepeats=8 minlen=0 histexpire=0
histsize=0 pwdchecks= dictionlist= capabilities=CAP_ARM_APPLICATION,CAP_PROPAGATE fsize=-1
cpu=-1 data=491519 stack=32767 core=-1 rss=-1 nofiles=2000 time_last_login=1128640848
tty_last_login= host_last_login=aixve01 unsuccessful_login_count=0 roles=
```

4. Verify the user's capabilities at the `/etc/security/user` file, as shown in Example 4-4.

Example 4-4 Verify user's capabilities

```
root@aixve01: /> grep -p db2inst1 /etc/security/user
db2inst1:
    admin = false
    umask = 22
    capabilities = CAP_ARM_APPLICATION,CAP_PROPAGATE
```

Set DB2LIBPATH variable

In order for a DB2 instance to call the ARM instrumentation APIs, the `DB2LIBPATH` variable must have the correct library path. To verify the `DB2LIBPATH` variable perform the following steps:

1. Log in as the DB2 instance owner `db2inst1`:

```
root@aixve01: /> su - db2inst1
```

2. Verify the DB2LIBPATH variable using the **db2set** command, as shown in Example 4-5.

Example 4-5 Verify the DB2LIBPATH variable on AIX

```
$ db2set
DB2LIBPATH=/usr/lib
DB2COMM=tcPIP
DB2AUTOSTART=YES
```

3. If it has the different value, then you can change it by using the **db2set** command, as shown in Example 4-6. Please note that this will overwrite the previous value.

Example 4-6 Set the DB2LIBPATH variable on AIX

```
$ db2set DB2LIBPATH=/usr/lib
```

4. Restart the DB2 instance as shown in Example 4-7.

Example 4-7 Restart DB2 instance on AIX

```
$ db2 force application all
DB20000I The FORCE APPLICATION command completed successfully.
DB21024I This command is asynchronous and may not be effective immediately.
$ db2 terminate
DB20000I The TERMINATE command completed successfully.
$ db2stop
10/07/2005 08:52:38 0 0 SQL1064N DB2STOP processing was successful.
SQL1064N DB2STOP processing was successful.
$ db2start
10/07/2005 08:52:42 0 0 SQL1063N DB2START processing was successful.
SQL1063N DB2START processing was successful.
```

i5/OS

ARM is enabled on DB2 UDB by default. No other configuration of DB2 is required.

z/OS

The following maintenance is required to provide ARM enablement on DB2 Universal Database Version 8.2 on z/OS. After you apply the following maintenance, DB2 is ARM-enabled by default and no additional steps are required.

- ▶ APAR PQ91509 EWLM correlator support to allow performance measurement using ARM instrumentation.
- ▶ APAR PQ99707 with the connection reset optimization code for JCC clients.
- ▶ APAR PK11801 for performance improvement.
- ▶ The JCC driver should be at level 2.6.16 or later.
- ▶ Verify that the DT zparm value is Inactive, specially if you are migrating from DB2 V7 where the default value was A.

Transactions initiated through DDF are accounted as separate transactions. DB2 transactions initiated through DDF are accounted as separate transactions, but DB2 transactions initiated by a local WebSphere Application Server executed in binding mode are accounted to the WebSphere thread. In this release, DDF cannot be hop 0 for a transaction to be measured by EWLM. It will not create a correlator for the transactions but will only handle the correlator coming from a distributed environment.

Windows

ARM is enabled on DB2 UDB by default, so the DB2 instance should be able to register to ARM. The only required step is to verify that the DB2 instance user is a member of the proper group.

Verify the DB2 instance user group

The user that is running the DB2 instance process should be a member of the eWLMArm4Users or Administrators group. The DB2 instance user is added to the Administrators group by the DB2 installer automatically, but you should check it using the following sequence of steps:

1. Click **Start** → **Control Panel** → **Administrative Tools** → **Computer Management**.
2. In the Computer Management window, click **System Tools** → **Local Users and Groups** and click **Users**, and then select the **db2admin** DB2 instance user, as shown in Figure 4-1.

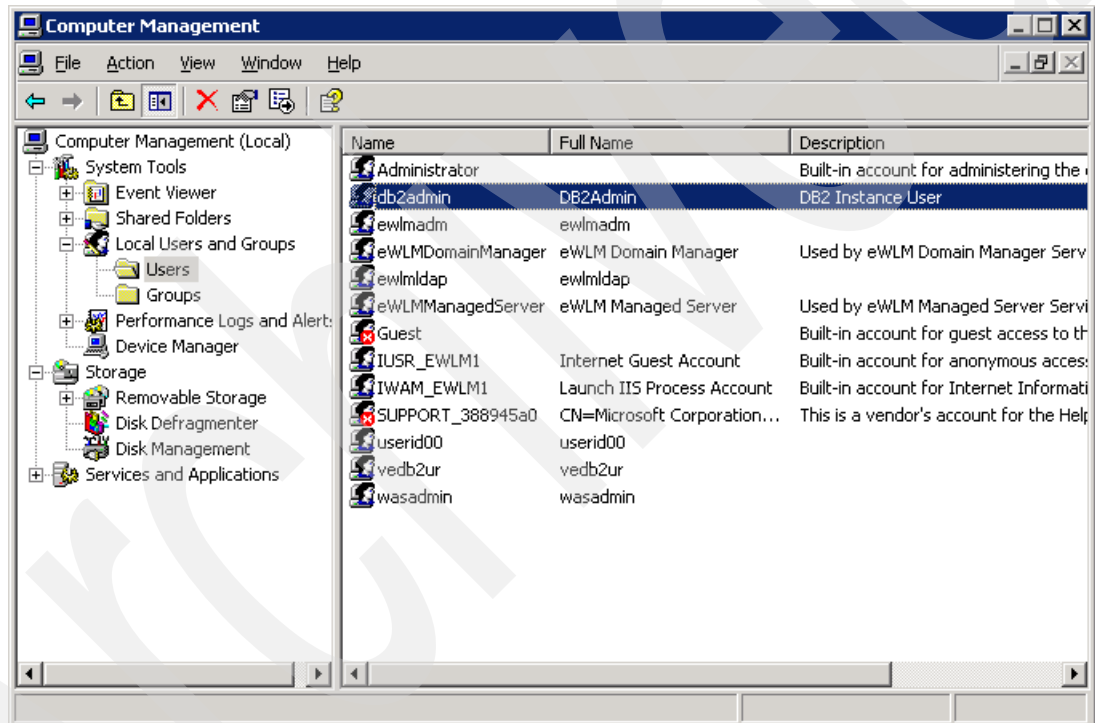


Figure 4-1 Administering local users and groups

3. Verify that db2admin belongs to the Administrators group, as shown in Figure 4-2.

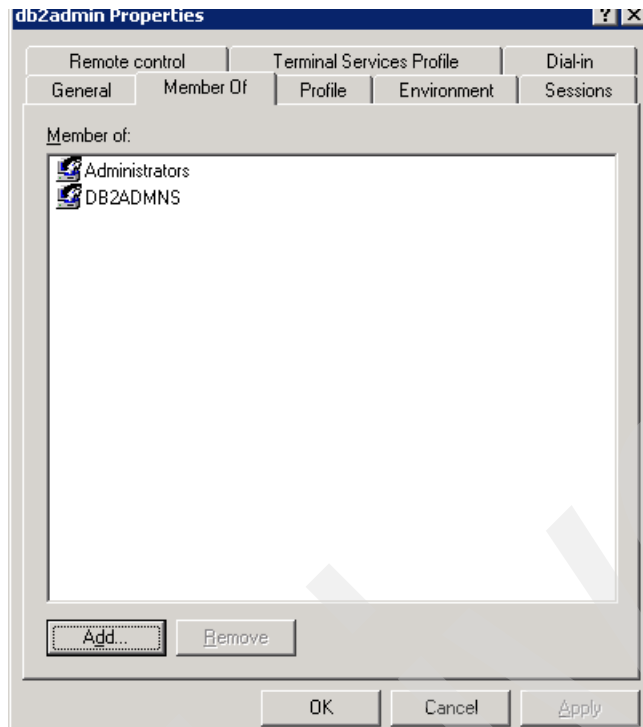


Figure 4-2 db2admin group membership

4. If you change the group membership, then you need to restart the DB2 instance using Services snap-in, as illustrated in Figure 4-3.

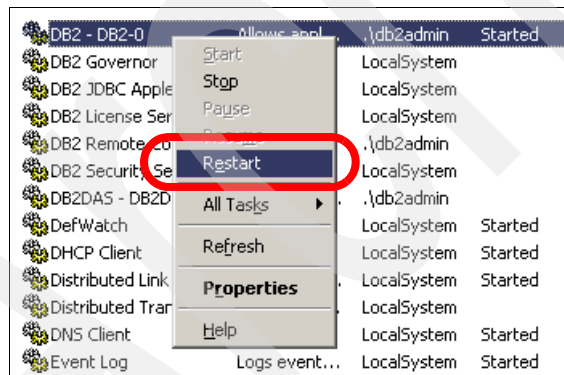


Figure 4-3 Restart DB2 instance on Windows

Solaris

To instrument the DB2 Universal Database on Solaris, you need to perform the following steps:

1. Add the instance owner to the auth file.

For a DB2 instance, you just need to add the UDB user name in your installation /etc/EWLM/auth file:

```
# ... your list(s) of authorized application user names here ...
nobody,db2inst1,root
```

2. Log in as the instance owner and set the following variable:

```
db2set DB2LIBPATH=/usr/lib
```

3. Restart the DB2 instance.

HP-UX and Linux

ARM is not enabled on the current available version of the DB2 Universal Database. This support will be available in the near future.

4.4 Enabling WebSphere Application Server for ARM

This section describes how to instrument the following versions of WebSphere Application Server:

- ▶ WebSphere Application Server Version 5.1.1
- ▶ WebSphere Application Server Version 6

Select from below the version of WebSphere Application Server you are running.

4.4.1 Enabling WebSphere Application Server v5.1.1 for ARM

Use the following steps to instrument WebSphere Application Server V5.1.1:

1. Enable PMI Request Metrics.
2. Create Java Virtual Machine custom properties.

Enable PMI Request Metrics

When Performance Monitoring Infrastructure (PMI) Request Metrics is enabled, WebSphere Application Server calls the ARM interface to pass the data to EWLM. This information can be viewed through the EWLM Control Center and system log file. The correlation information is generated and logged for the Web server plug-in and the JDBC drivers.

To enable PMI Request Metrics:

1. Log in to the WebSphere Administrative Console.
2. Click **Troubleshooting** → **PMI Request Metrics**.

3. Enable Request Metrics and Application Response Measurement (ARM) by checking both properties, as shown in Figure 4-4. Leave the Trace Level set to HOPS.

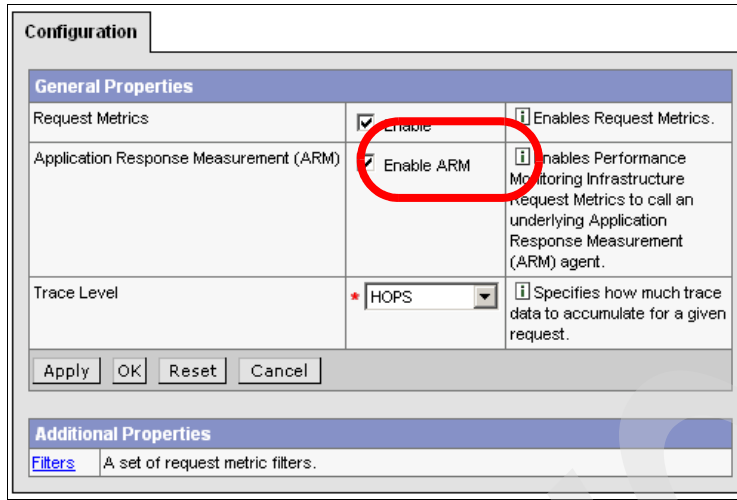


Figure 4-4 Enable Request Metrics and Application Response Management

4. Click **OK**.

Create Java Virtual Machine custom properties

Once you enable PMI Request Metrics, you need to set the proper JVM™ custom properties for the type of ARM agent and the ARM transaction factory. You have to create the custom properties for all application servers you want EWLM to monitor. Also, these settings affect the WebSphere HTTP plug-in configuration.

Create Java Virtual Machine custom properties using the following steps:

1. Log in to the WebSphere Administrative Console.
2. Click **Servers** → **Application Servers** → *server name* → **Process Definition** → **Java Virtual Machine** → **Custom Properties**.
3. If the properties shown in Table 4-3 are not present in the list, create them as shown in Figure 4-5 on page 113.

Table 4-3 Java Virtual Machine custom properties

Name	Value
ArmTransactionFactory	com.ibm.wlm.arm40SDK.transaction.Arm40TransactionFactory
com.ibm.websphere.pmi.reqmetrics.ARMIMPL	arm4
com.ibm.websphere.pmi.reqmetrics.PassCorrelatorToDB	true
com.ibm.websphere.pmi.reqmetrics.loggingEnabled	true
java.library.path	<EWLMMMS_LIBPATH>
ws.ext.dirs	<EWLMMMS_HOME>/classes/ARM

4. The example of the ITSO environment for JVM custom properties is shown in Figure 4-5.

Select	Name	Value
<input type="checkbox"/>	ArmTransactionFactory	com.ibm.wlm.arm40SDK.transaction.Arm40TransactionFactory
<input type="checkbox"/>	com.ibm.websphere.pmi.reqmetrics.ARMIMPL	arm4
<input type="checkbox"/>	com.ibm.websphere.pmi.reqmetrics.PassCorrelatorToDB	true
<input type="checkbox"/>	com.ibm.websphere.pmi.reqmetrics.loggingEnabled	true
<input type="checkbox"/>	java.library.path	c:\Program Files\IBM\VE\EWLMMS\classes\ms
<input type="checkbox"/>	ws.ext.dirs	c:\Program Files\IBM\VE\EWLMMS\classes\ARM

Figure 4-5 Java Virtual Machine custom properties

The value of `com.ibm.websphere.pmi.reqmetrics.loggingEnabled` is an optional property. If you set this parameter to true and the PMI Request Metrics trace level is set to HOPS, PERF_DEBUG, or DEBUG, you will see the correlator information in the application server's SystemOut.log, as shown in Example 4-8. Note that if this property is not created, the correlator information is not logged.

Example 4-8 PMI request metrics log at SystemOut.log

```
[6/1/04 11:34:17:572 EDT] 328673d8 PmiRmArmWrapp I PMRM0003I: parent:ver=1,ip=9.12.11.2,time=1085613001328,pid=1624,reqid=71255,event=1 - current:ver=1,ip=9.12.11.2,time=1085613001328,pid=1624,reqid=71255,event=1 type=URI detail=/trade/app elapsed=1
[6/2/04 9:23:09:709 EDT] 329a73d8 PmiRmArmWrapp I PMRM0003I: parent:ver=1,ip=9.12.4.139,time=1084905294782,pid=311300,reqid=65818624,event=1 - current:ver=1,ip=9.12.4.139,time=1084905294782,pid=311300,reqid=65818625,event=1 type=JDBC detail=select q1."ADDRESS", q1."PASSWORD", q1."USERID", q1."EMAIL", q1."CREDITCARD", q1."FULLNAME" from ACCOUNTPROFILEEJB q1 where ( q1."USERID" = ?) for update of "ADDRESS" elapsed=6
```

5. Click **Save**.

For the configuration changes to take effect, the changes must be saved to the master configuration and synchronized to the nodes.

6. Check **Synchronize changes with nodes** and click **Save** to save all your changes.

7. Restart the application server.

4.4.2 Enabling WebSphere Application Server Version 6 for ARM

Use the following steps to instrument WebSphere Application Server V6:

1. Enable PMI Request Metrics.
2. Add Java Virtual Machine custom properties.

Enable PMI Request Metrics

When Performance Monitoring Infrastructure (PMI) Request Metrics is enabled, WebSphere Application Server calls the ARM interface to pass the data to EWLM. This information can be viewed through the EWLM Control Center and system log file. The correlation information is generated and logged for the Web server plug-in and the JDBC drivers.

To enable PMI Request Metrics:

1. Log in to the WebSphere Administrative Console.
2. Click **Monitoring and Tuning** → **Request Metrics**.

3. In the Configuration table of the Request Metrics page, select **Enable Request Metrics**.
4. Choose **Components to be instrumented**. You can select **ALL** or a specific component in the WebSphere Application Server, such as Servlet, EJB™, JDBC, WebServices, JMS, or AsyncBeans.
5. The recommended WebSphere Application Server Request Metrics trace level is Hops for use with EWLM.
6. Select **Application Response Measurement(ARM) Agent**.
7. Selecting **Standard Logs** is optional. If you select this option and the PMI Request Metrics trace level is set to HOPS, PERF_DEBUG, or DEBUG, you will see the correlator information in the application server's SystemOut.log, as shown in Example 4-9. Note that if this option is not selected, the correlator information is not logged.

Example 4-9 PMI request metrics log at SystemOut.log

```
[10/7/05 17:53:57:218 CDT] 00000056 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=10.1.1.177,time=1128725506635,pid=360660,reqid=4098,event=1 -
current:ver=1,ip=10.1.1.177,time=1128725506635,pid=360660,reqid=4099,event=1 type=JDBC
detail=executeQuery elapsed=45
[10/7/05 17:53:57:240 CDT] 00000056 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=10.1.1.177,time=1128725506635,pid=360660,reqid=4098,event=1 -
current:ver=1,ip=10.1.1.177,time=1128725506635,pid=360660,reqid=4098,event=1 type=URI
detail=/trade/servlet/PingJDBCRead elapsed=232
```

8. Choose **Agent Type** as ARM4.
9. Specify `com.ibm.wlm.arm40SDK.transaction.Arm40TransactionFactory` for the ARM transaction factory implementation class name.
10. Settings should look like the screen shown in Figure 4-6.

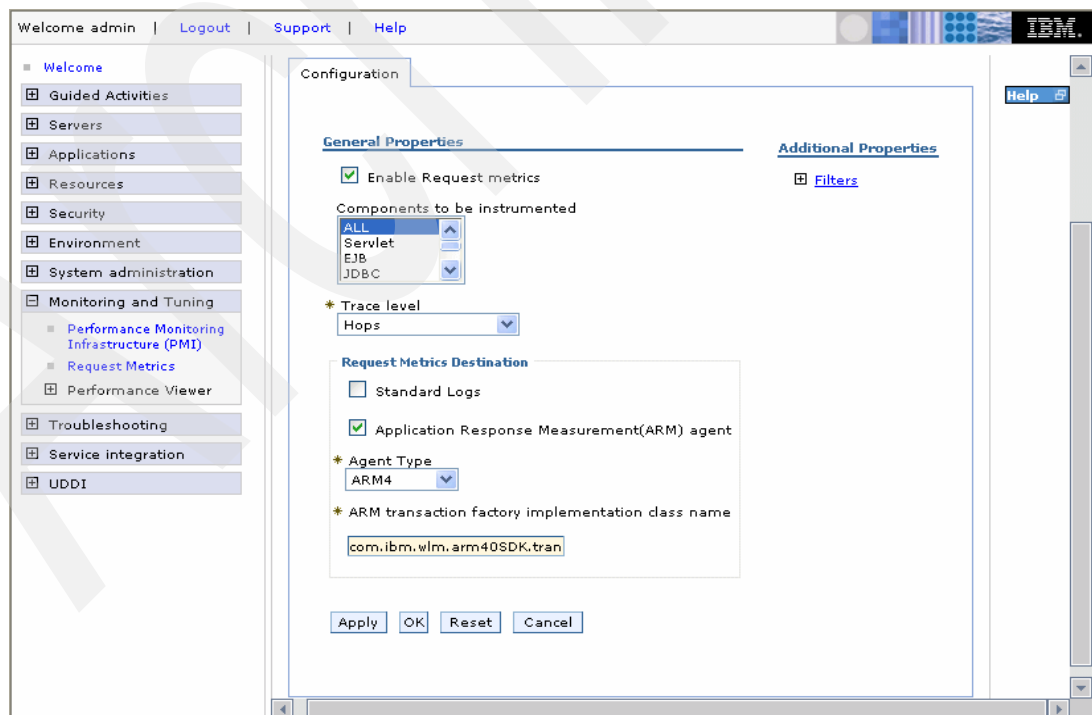


Figure 4-6 Enable Request Metrics and Application Response Measurement

11. Click **Apply**.

Add Java Virtual Machine custom properties

Once you enable PMI Request Metrics, you need to set the proper JVM custom properties. You have to create the custom properties for all application servers you want EWLM to monitor. Also, these settings affect the WebSphere HTTP plug-in configuration.

Add new Java Virtual Machine custom properties using the following steps:

1. Log in to the WebSphere Administrative Console.
2. Click **Servers** → **Application Servers** → *server name* → **Server Infrastructure** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine** → **Custom Properties**.
3. The properties required are shown in Table 4-4.

Table 4-4 Java Virtual Machine custom properties

Name	Value
com.ibm.websphere.pmi.reqmetrics.PassCorrelatorToDB	true
java.library.path	<EWLMMS_LIBPATH>
ws.ext.dirs	<EWLMMS_HOME>/classes/ARM

4. If you do not have these properties, add them using **New** button in the administration console, as shown in Figure 4-7.

Application servers

Application servers > TradeServer1 > Process Definition > Java Virtual Machine > Custom Properties > New

Specifies arbitrary name and value pairs of data. The value is a string that can set internal system configuration properties.

Configuration

General Properties

* Name
qmetrics.PassCorrelatorToDB

* Value
true

Description

Apply OK Reset Cancel

Figure 4-7 Add new Java Virtual Machine custom property

5. The example of newly added JVM custom properties is shown in Figure 4-8.

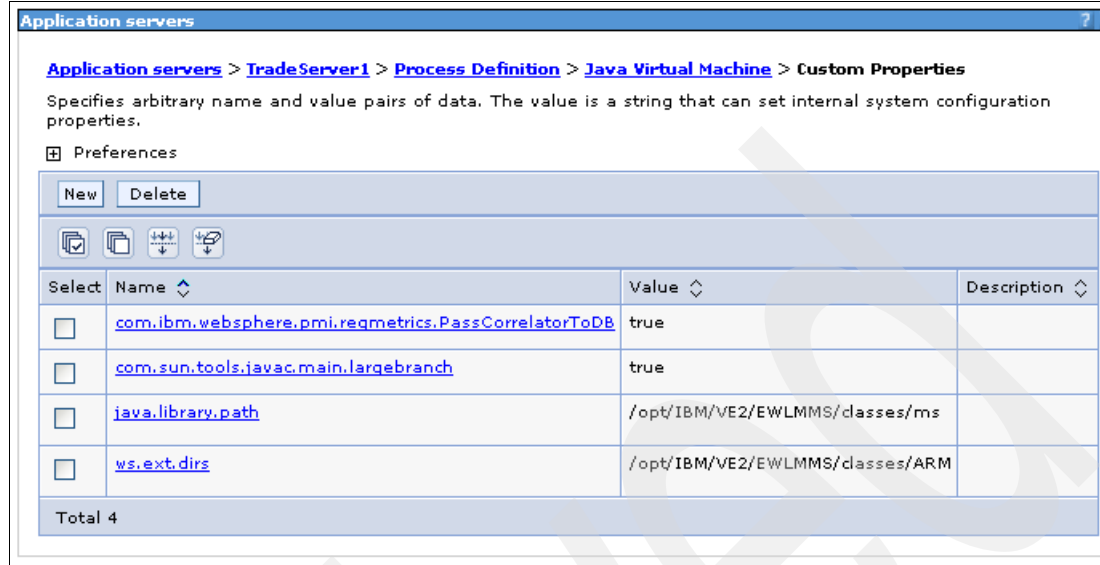


Figure 4-8 Java Virtual Machine customer properties

6. After confirming the changes, click **Save**. Be sure to select the **Synchronize changes with nodes** option when prompted. For the configuration changes to take effect, the changes must be saved to the master configuration and synchronized to the nodes.
7. Restart the application server.

4.4.3 Enabling WebSphere Application Server Version 6 for z/OS for ARM

To enable ARM instrumentation on the WebSphere Application Server, enable ARM and the Request Metrics. After you complete the following customization steps, restart the WebSphere Application Server.

Enable PMI Request Metrics

The following steps are required to enable PMI metrics:

1. Log on to the administrative console of WebSphere Application Server V6 at `http://<hostname>:9060/ibm/console`, where 9060 is the default HTTP port for a Network Deployment Cell.

2. Select **Monitoring and Tuning** → **Request metrics** in the administrative console navigation tree.
 - a. On the Configuration tab of the Request Metrics panel, check the **Enable Request metrics** box. Figure 4-9 on page 117 shows a sample of this panel.
 - b. Under Components to be instrumented, select **ALL**.
 - c. Choose the trace level you want to use from the Trace level list. We choose **Hops**.
 - d. Under Request Metrics Destination:
 - i. Standard Logs is optional.
 - ii. Check **Application Response Measurement (ARM) agent**.
 - iii. Specify **ARM4** from the Agent Type pull-down list.
 - iv. Type the class name in the ARM transaction factory implementation class name text field: `com.ibm.wlm.arm40SDK.transaction.Arm40TransactionFactory`.
3. Click **Apply** and **OK**.
4. Click **Save** and **Save to Master Configuration**.

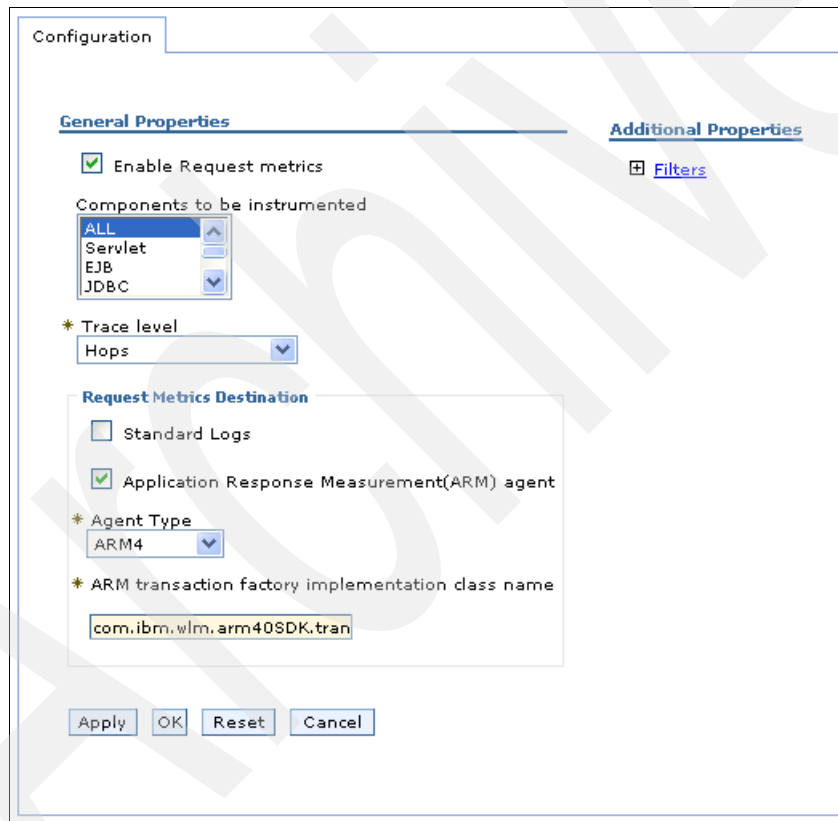


Figure 4-9 PMI metrics Configuration panel

Add custom properties to the JVM

The following steps are required to update the JVM custom properties:

1. Copy the `arm4.jar` from the default location `/usr/lpp/VE_R2/EWLM/classes` into a directory of your choice. The `.jar` file must reside in the `/ARM` subdirectory of the directory you choose. In our example, we choose `/u/ewlmadm/ARM`. Remember to give permission so that the WebSphere Application Server for z/OS has read access to the directory. After

you copy the directory, log on to the WebSphere Administrative Console to complete the steps.

2. Go to **Servers** → **Application Servers** → <server name>. In the Java and Process Management section, select **Process Definition**.
3. Select **Servant**.
4. In the Additional Properties section, expand **Java Virtual Machine** → **Custom Properties**.
5. Add these custom properties, as shown in Figure 4-10:
 - com.ibm.websphere.pmi.reqmetrics.PassCorrelatorToDB = true.
 - ws.ext.dirs = <user_directory>/ARM where the <user_directory> is the directory specified in step 1 on page 117.

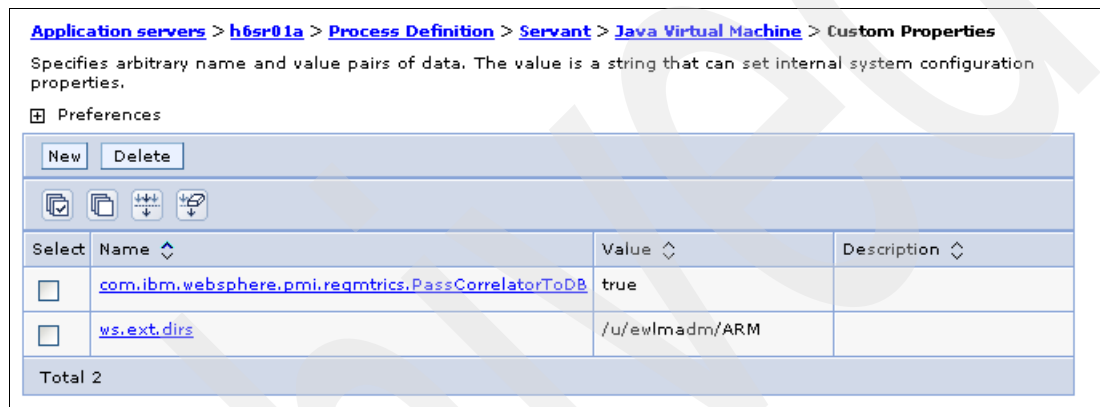


Figure 4-10 JVM Custom Properties panel

6. Set the WAS SERVER ONLY server region libpath variable.
 - a. Expand **Environment** → **WebSphere variables**.
 - b. Append the WAS SERVER ONLY server region libpath variable to include `:/usr/lpp/VE_R2/EWLM/lib:/usr/lib`, as shown in Figure 4-11.

<input type="checkbox"/>	WAS_SERVER_ONLY_ICU_DATA	\${WAS_INSTALL_ROOT}/bin/
<input type="checkbox"/>	WAS_SERVER_ONLY_control_region_classpath	\${USER_INSTALL_ROOT}/properties:\${WAS_INSTALL_ROOT}/properties:\${WAS_INSTALL_ROOT}/lib/bootstrap.jar:\${WAS_INSTALL_ROOT}/lib/
<input type="checkbox"/>	WAS_SERVER_ONLY_control_region_libpath	\${JAVA_HOME}/bin:\${JAVA_HOME}/bin/classic:\${WAS_INSTALL_ROOT}/lib
<input type="checkbox"/>	WAS_SERVER_ONLY_default_jvm_options	-Djava.security.policy=\${USER_INSTALL_ROOT}/properties/server.policy -Dfile.encoding=ISO8859-1 -Dsmpe.install.root=\${WAS_INSTALL_ROOT} -Dcom.ibm.itp.location=\${WAS_INSTALL_ROOT}/bin -Dwas.history.dir=/tmp -Djava.awt.headless=true -Djava.util.logging.config.class=com.ibm.ws.logging.impl.DefaultLoggerFactory -Dserver.root=\${USER_INSTALL_ROOT} -Dwas.install.root=\${WAS_INSTALL_ROOT} -Dwas.repository.root=\${USER_INSTALL_ROOT} -Dws.ext.dirs=\${JAVA_HOME}/lib:\${WAS_INSTALL_ROOT}/classes:\${WAS_INSTALL_ROOT}/lib:\${WAS_INSTALL_ROOT}/lib
<input type="checkbox"/>	WAS_SERVER_ONLY_node_name	h6dmnode
<input type="checkbox"/>	WAS_SERVER_ONLY_node_short_name	H6DM
<input type="checkbox"/>	WAS_SERVER_ONLY_private_bboo_jvm_in_ctl	1
<input type="checkbox"/>	WAS_SERVER_ONLY_server_region_classpath	\${USER_INSTALL_ROOT}/properties:\${WAS_INSTALL_ROOT}/properties:\${WAS_INSTALL_ROOT}/lib/bootstrap.jar:\${WAS_INSTALL_ROOT}/lib/
<input type="checkbox"/>	WAS_SERVER_ONLY_server_region_libpath	\${JAVA_HOME}/bin:\${JAVA_HOME}/bin/classic:\${WAS_INSTALL_ROOT}/lib:/usr/lpp/VE_R2/EWLM/lib:/usr/lib
<input type="checkbox"/>	WAS_TEMP_DIR	\${USER_INSTALL_ROOT}/temp
<input type="checkbox"/>	ras_log_logstreamName	WAS.ERROR.LOG

Figure 4-11 WebSphere variables panel

Considerations for Java2 Security

If Java2 Security is enabled: In order to use WebSphere Application Server V6 with EWLM, you must update the WebSphere Application Server `server.policy` files in the `WAS6_Profile/properties` directory, adding the following lines in that file (change `EWLMMS_HOME` according to your EWLMMS installation path):

```
grant codeBase "file:/EWLMMS_HOME/classes/ARM/arm4.jar" {  
    permission java.security.AllPermission;  
};
```

This avoids a Java 2 security exception for violating the Java 2 security permission.

4.5 Enabling IBM HTTP Server for ARM

There are two types of instrumented plug-ins for the IBM HTTP Server:

- ▶ The *WebSphere HTTP plug-in* that is provided by WebSphere installation
- ▶ The *independent plug-in* that is provided for non-WebSphere Application Server application usage

Also, there are three versions of the IBM HTTP Server currently supported by EWLM:

- ▶ IBM HTTP Server 1.3.28
Only the WebSphere HTTP plug-in is available for this version.
- ▶ IBM HTTP Server 2.0.47
Both the WebSphere HTTP plug-in and the independent plug-in are available.
- ▶ IBM HTTP Server 6.0
Both the WebSphere HTTP plug-in and the independent plug-in are available.

Instrumenting the IBM HTTP Server depends on which version of IBM HTTP Server you are currently running and what type of plug-in you will use.

4.5.1 WebSphere HTTP plug-in

The Web-serving plug-ins that ship with WebSphere Application Server can be installed using the WebSphere installation wizard.

WebSphere V5 HTTP plug-in

The WebSphere HTTP plug-in first receives a user's HTTP request, and by referencing the plug-in configuration file, `plugin-cfg.xml`, determines whether the request should be handled by the Web server or an application server. If the plug-in determines that the request is for an application server, the plug-in forwards the request, along with the ARM correlator information, to the appropriate Web application.

The WebSphere HTTP plug-in that ships with the WebSphere Application Server is already instrumented, so all you have to do is install the plug-in to the IBM HTTP Server and update the plug-in configuration file after the WebSphere Application Server instrumentation is successfully completed. Perform the following steps to enable ARM for the WebSphere HTTP plug-in:

1. Install the WebSphere HTTP plug-in to the IBM HTTP Server using the WebSphere Application Server V5 installation wizard. This installation package is corresponding to the operating system on which the IBM HTTP Server runs.

2. Apply WebSphere Application Server fix pack 1, if needed.
3. Verify the IBM HTTP Server configuration file. (This is updated by the WebSphere Application Server installation program during the installation of plug-in.)

For the IBM HTTP Server 1.3.28, verify the following line at <IHS13_HOME>/conf/httpd.conf:

```
LoadModule ibm_app_server_http_module <WAS_HOME>/bin/mod_ibm_app_server_http.so
WebSpherePluginConfig <WAS_HOME>/config/cells/plugin-cfg.xml
```

For the IBM HTTP Server 2.0.47, verify the following line at <IHS20_HOME>/conf/httpd.conf:

```
LoadModule was_ap20_module <WAS_HOME>/bin/mod_ibm_app_server_http.so
WebSpherePluginConfig <WAS_HOME>/config/cells/plugin-cfg.xml
```

4. Update the plug-in configuration file after WebSphere Application Server ARM instrumentation is successfully completed.

- a. Log in to the WebSphere Administrative Console.
- b. Click **Environment** → **Update Web Server Plug-in**.
- c. Click **OK** to regenerate the Web server plug-in configuration file.

- d. To verify ARM enablement for plug-in, open the plug-in configuration file <WAS_HOME>/config/cells/plugin-cfg.xml and search for this line:

```
<RequestMetrics armEnabled="true" loggingEnabled="false" rmEnabled="true"
traceLevel="HOPS">
```

Ensure that loggingEnabled and traceLevel attributes match the values you specified at PMI Request Metrics, described previously in “Enable PMI Request Metrics” on page 111.

- e. The plug-in configuration file must reside in the location specified in the IBM HTTP Server configuration file. Open the IBM HTTP Server configuration file stored in <IHS13_HOME>/config/httpd.conf or <IHS20_HOME>/config/httpd.conf and search for the line:

```
WebSpherePluginConfig "<WAS_HOME>/config/cells/plugin-cfg.xml"
```

Transfer the plug-in file to this path on the IBM HTTP Server machine.

5. Restart the IBM HTTP Server.

WebSphere V6 HTTP plug-in

The WebSphere HTTP plug-in first receives a user’s HTTP request and by referencing the plug-in configuration file, plugin-cfg.xml, determines whether the request should be handled by the Web server or an application server. If the plug-in determines that the request is for an application server, the plug-in forwards the request, along with the ARM correlator information, to the appropriate Web application.

The WebSphere HTTP plug-in that ships with WebSphere Application Server is already instrumented, so all you have to do is install the plug-in to the IBM HTTP Server and update the plug-in configuration file after the WebSphere Application Server instrumentation is successfully completed. Perform the following steps to enable ARM for the WebSphere HTTP plug-in:

1. Install the WebSphere HTTP plug-in to the IBM HTTP Server using the WebSphere Application Server installation wizard. Use the installation package corresponding to the operating system on which the IBM HTTP Server runs.
2. Apply the appropriate WebSphere Application Server fix pack.

3. Verify the IBM HTTP Server configuration file. For our ITSO IBM HTTP Server 6.0 on Windows, the following two lines appear in the <IHS6_HOME>/conf/httpd.conf file:


```
LoadModule was_ap20_module "C:\Program Files\IBM\WebSphere\Plugins\bin\mod_was_ap20_http.dll"
WebSpherePluginConfig "C:\Program Files\IBM\WebSphere\Plugins\config\webserver1\plugin-cfg.xml"
```
4. Update the plug-in configuration file after WebSphere Application Server instrumentation is successfully completed.
 - a. Log in to the WebSphere Administrative Console.
 - b. Click **Servers** → **Web servers**.
 - c. Select the **webserver1** check box.
 - d. Click the **Generate Plug-in** to regenerate the Web server plug-in configuration file, as shown in Figure 4-12.

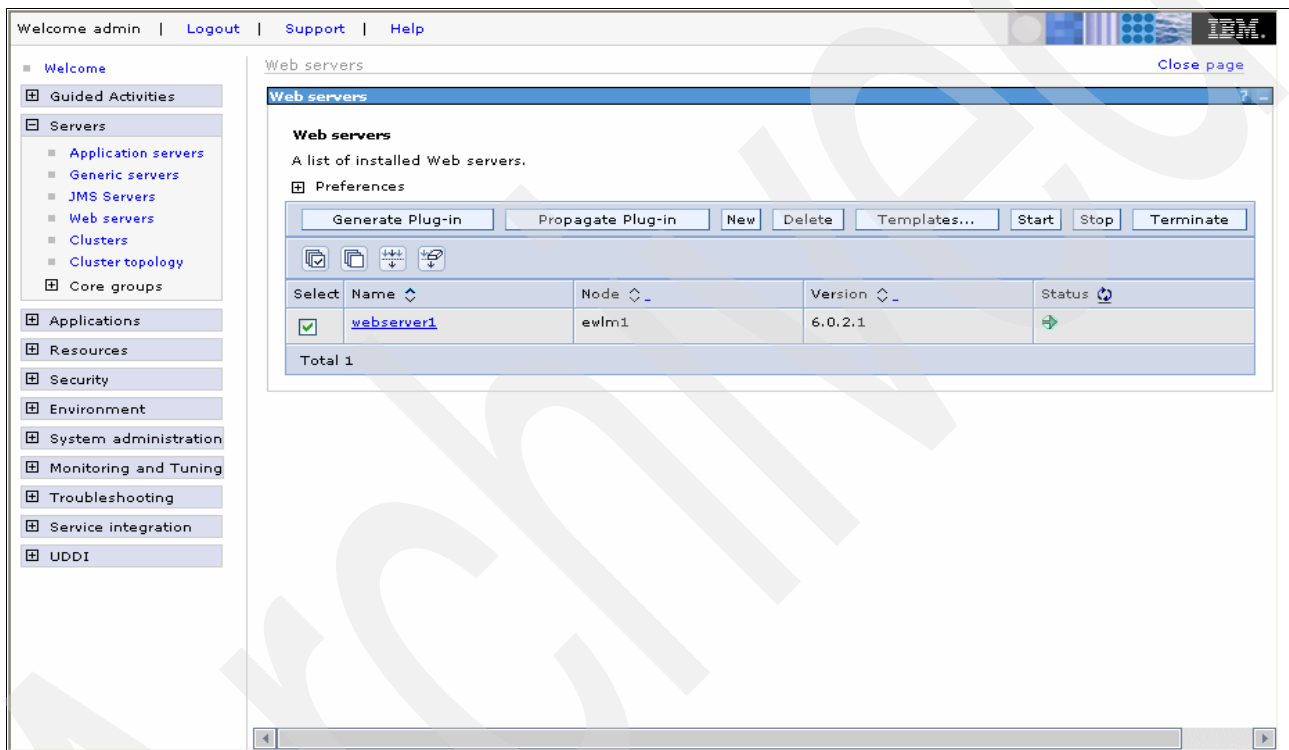


Figure 4-12 Regenerate Web server plug-in

- e. You now see the message shown in Figure 4-13, which provides the location of the newly generated plug-in configuration file. This is an example of WebSphere Application Server V6 Network Deployment.

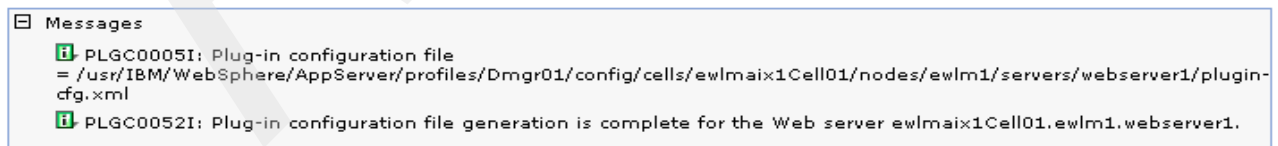


Figure 4-13 Generated plug-in file

- f. To verify ARM enablement for the plug-in, open the plugin-cfg.xml file and search for this line:


```
<RequestMetrics armEnabled="true" loggingEnabled="false" rmEnabled="true"
traceLevel="HOPS">
```

Ensure that loggingEnabled and traceLevel attributes match the values you specified at PMI Request Metrics described previously in “Enable PMI Request Metrics” on page 113.

- g. The plug-in configuration file must reside in the location specified in the IBM HTTP Server configuration file. On the IBM HTTP Server platform open the IBM HTTP Server configuration file stored in <IHS6_HOME>/config/httpd.conf and search for the line:

```
WebSpherePluginConfig "<WAS_HOME>/Plugins/config/webserver1/plugin-cfg.xml"
```

Transfer the plugin-cfg.xml file to this path on the IBM HTTP Server machine.

5. Restart the IBM HTTP Server.

4.5.2 Independent HTTP plug-ins

Independent plug-ins are intended for non-WebSphere Application Servers.

IBM HTTP Server plug-in

The independent plug-in is available for IBM HTTP Server V2.0.47 and later, and is a plug-in to capture requests that do not go through the WebSphere Application Server HTTP plug-in. If you have Tomcat, JBoss, or CGI, you can use the independent plug-in to monitor the transaction downstream hops. The WebSphere plug-in provides instrumentation for requests that do not flow to WebSphere (like static pages) so there is no need for the independent plug-in when the WebSphere plug-in is being used. The independent plug-in should only be used when there is no WebSphere Application Server in your application transaction flow.

We do not describe how to use the independent plug-in in this book.

Perform the following steps to instrument the independent IBM HTTP Server plug-in:

1. Copy the plug-in module mod_arm4_ap20.so to the <IHS20_HOME>/modules directory.
2. Add these lines at the end of <IHS20_HOME>/conf/httpd.conf:

```
LoadModule arm4_module modules/mod_arm4_ap20.so
<IfModule mod_arm4_ap20.c>
  ArmApplicationName "Apache HTTP Server v2.0.42"
  ArmTransactionName "HTTP Request"
  ArmInstrumentHandler on
</IfModule>
```

3. Restart the IBM HTTP Server.

IIS plug-in

The independent plug-in for IIS is shipped with EWLM. The IIS plug-in for EWLM on a Windows operating system does not require a specific hardware configuration. Microsoft Windows Server 2000 and 2003 are both supported. You can find additional information about enabling IIS instrumentation at the InfoCenter at:

<http://www.ibm.com/servers/library/infocenter>

The current IIS does not make any arm_block_transaction / arm_unlock_transaction calls to the ARM service provider.

Although we did not use the independent plug-in in the ITSO configuration, we would like to provide some information about how to enable ARM instrumentation for the plug-in. Both Microsoft Internet Information Services (IIS 5.0 and IIS 6.0) servers should be supported. However, tests were done only on Microsoft IIS 5.0, with only minimal testing on IIS 6.0. The IIS plug-in is delivered as a dll file: EWLMIIISFilter.dll.

The EWLM IIS plug-in is actually an ISAPI filter (a program that responds when the application server receives an HTTP request). You can either install filters for all of the sites (global filter) or you can install filters for individual Web sites. You must be a member of the Administrators group on the local server to perform the following procedure, or you must have been delegated the appropriate authority.

To install the EWLM Plug-in (ISAPI filter) for use with a Microsoft Internet Information Server, follow these steps:

1. Copy the EWLM filter DLL to an appropriate folder, such as the `SCRIPTS` or `CGI-BIN` subdirectory. (You can also have the file in your own directory, `C:\IISpluginTST`.)
2. Open the IIS Manager by selecting **Start** → **Run** and typing:

```
mmc %systemroot%\system32\inetsrv\iis.msc
```
3. Expand the local computer and right-click the Web server or Web site to which you want to add a filter. To use the ISAPI filter with all Web sites, select the Web server or Server Name icon. To use the ISAPI filter with a specific Web site, select the icon for that Web site (for example, the default Web site).
4. Right-click the icon, select **Properties**, click the **ISAPI Filters** tab, and click **Add**.
5. Type a name for the ISAPI filter. If you have your own directory, click **Browse** and select the **EWLM IIS filter**. Click **OK**.
6. Stop the IISADMIN service. To do this, either type `net stop iisadmin /y` at a command prompt or use Administrative Tools/Services.
7. Restart the World Wide Web Publishing Service (by typing `net start w3svc` at a command prompt or using Administrative Tools/Services).
8. Browse back to the ISAPI Filters tab and verify that the filter is loaded properly. You should see a green arrow pointing up in the Status column.

Note: If you are adding filters to a Web site, you will not see any global filters inherited from the Web server's master properties. You will see only the filters installed for the Web site, even though both sets of filters are run. The ISAPI Filters tab specifies a load order, with the filter at the top of the list loading first. Normally, `Sspifilt.dll`, the ISAPI filter for SSL, is at the top of the list to allow any other filters to access data before IIS encrypts and transmits or decrypts and receives HTTPS traffic.

If you are running IIS 6.0, the following are additional steps to allow authentication:

1. Create a new application pool from the IIS Manager by right-clicking the **Application Pool** folder. Select **New** → **Application Pool**. Specify the name and click **OK**.
2. Right-click the newly created **Application Pool** → **Properties** option. Select the **Identity** tab.
3. Select **Configurable** and enter the user name and password you desire. A good choice is to enter the built-in account for Internet Information Services (`IWAM_%COMPUTER_NAME%`) and the corresponding password. Click **OK**.
4. Select the Web sites from the directory tree to specify access control by right-clicking the EWLM Plug-in IIS Web site. Click **Properties** and select the **Directory Security** tab. Click **Edit** for Authentication and access control.
5. Enable *anonymous* access and enter the built-in account for anonymous access: (`IUSR_%COMPUTER_NAME%`). Click **OK**.
6. Select the **Home Directory** tab on the Properties sheet. At the bottom, select the Application pool that you just created, and click **OK**.

7. Select **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Computer Management** → **Local Users and Groups** → **Groups** → **eWLMArm4Users** → **Add** → **Advanced** → **Find Now** → **SERVICE** → **OK**.
8. Restart the IIS plug-in after the updates have been made.

4.6 Verifying application ARM enablement

Each platform has its own ARM verification command.

AIX, Solaris, HP-UX, and Linux

You can verify that instrumentation has been enabled using the `lsarm` command on AIX, Solaris, HP-UX, and Linux:

```
$ lsarm -a
```

The command outputs the ARM registered applications:

```
APPL: IBM DB2 Universal Database
APPL: WebSphere
APPL: IBM Webserving Plugin
APPL: Apache HTTP Server
```

i5/OS

You can verify that instrumentation has been enabled by looking at the application middleware in question. If it has ARM calls, it will have the QPMWARM4 activation group active.

z/OS

You can use the following command to verify the status of ARM services on z/OS:

```
DISPLAY WLM,AM
```

Example 4-10 is the way the command for verification on z/OS system looks.

Example 4-10 Display WLM,AM command sample

```
D WLM,AM

RESPONSE=SC69
IWM075I 15.07.05 WLM DISPLAY 500
EWLM ARM SERVICES ARE ENABLED
EWLM MANAGED SERVER JOBNAME=EWLMS2 ASID=0049
EWLM POLICY NAME=ITSO TRADE3 AND PLANTS SERVICE POLICY
NUMBER OF REGISTERED PROCESSES=1, APPLICATIONS=1
```

Windows

The `ewlmWinAdTool` command allows you to verify the applications registered to the ARM interface. Click **Start** → **Run...**, type `cmd`, click **OK**, and follow the sequence shown in Example 4-11.

Example 4-11 ewlmWinAdTool output

```
C:\Program Files\ibm\VE2\EWLMS\classes\ms>ewlmwinadtool ARM4PsInfo

Windows Platform Administrative and Diagnostic Tool for EWM
Ver. 2.1.00021.5270 *
```

EWLM V2.1 (C) Copyright IBM Corporation, 2004-2005.

>>> ARM4PsInfo: 2 process(es) is(are) doing ARM instrumentation...

```
pid +|-          gupid(hex)          #ARs #TRs #AIs #TIs #TBs
-----
508 [+] 08AFFAC2.01C5D19C.000001FC : 1/64 1/64 1/256 0/16384 0/16384
AR[00]: E232DBCA00000000h (IBM Webserving Plugin)
TR[00]: E232DC2600000000h (WebRequest)
.APP: (IBM Webserving Plugin) E232DBCA00000000h
AI[00]: 8AD1E2EC00000000h (aaid=8AD1E2ECh, aaid=F7236B3E00000000h)
.GRP: (IBM_HTTP_Server/6.0.2 Apache/2.0.47 (Win32))
.INS: (EWLM1/PID=0000000508)
.APP: (IBM Webserving Plugin) E232DBCA00000000h

pid +|-          gupid(hex)          #ARs #TRs #AIs #TIs #TBs
-----
3156 [+] 10F522F2.01C5D19C.00000C54 : 3/64 3/64 3/256 0/16384 0/16384
AR[00]: E10E52F700000000h (IBM DB2 Universal Database)
AR[01]: E10E52F800000001h (IBM DB2 Universal Database)
AR[02]: E10E52F900000002h (IBM DB2 Universal Database)
TR[00]: E10E535300000000h (SQL)
.APP: (IBM DB2 Universal Database) E10E52F700000000h
TR[01]: E10E535400000001h (SQL)
.APP: (IBM DB2 Universal Database) E10E52F800000001h
TR[02]: E10E535500000002h (SQL)
.APP: (IBM DB2 Universal Database) E10E52F900000002h
AI[00]: 8AD1E2ED00000000h (aaid=8AD1E2EDh, aaid=F7236B3F00000001h)
.GRP: (VE2INST1)
.INS: (5)
.APP: (IBM DB2 Universal Database) E10E52F700000000h
AI[01]: 8AD1E2EE00000001h (aaid=8AD1E2EEh, aaid=F7236B3F00000001h)
.GRP: (VE2INST1)
.INS: (6)
.APP: (IBM DB2 Universal Database) E10E52F800000001h
AI[02]: 8AD1E2EF00000002h (aaid=8AD1E2EFh, aaid=F7236B3F00000001h)
.GRP: (VE2INST1)
.INS: (7)
.APP: (IBM DB2 Universal Database) E10E52F900000002h
```

Alternative way to verify for DB2

For the DB2 environment, another way to verify the instrumentation (at a lower level) is to turn on tracing with DB2 and make sure that the ARM libraries are loaded by entering the sequence of commands shown in Example 4-12.

Example 4-12 Verify ARM instrumentation for DB2

```
db2stop
db2trc on -f /tmp/DB2-trace.trc
db2start
db2trc off
db2trc format /tmp/DB2-trace.trc /tmp/DB2-trace.txt

grep -i arm /tmp/DB2-trace.txt
```

This sequence should return the trace output similar to Example 4-13.

Example 4-13 Instrumentation sample

6C696261	726D342E	736F		libarm4.so
FFBFFE59	6C696261	726D342E	736F0000	...Ylibarm4.so..
61726D5F	72656769	73746572	5F617070	arm_register_app
61726D5F	73746172	745F6170	706C6963	arm_start_applic
61726D5F	72656769	73746572	5F747261	arm_register_tra
61726D5F	73746172	745F7472	616E7361	arm_start_transa
61726D5F	62696E64	5F746872	656164	arm_bind_thread
61726D5F	626C6F63	6B5F7472	616E7361	arm_block_transa
61726D5F	756E626C	6F636B5F	7472616E	arm_unblock_tran
61726D5F	756E6269	6E645F74	68726561	arm_unbind_threa
61726D5F	73746F70	5F747261	6E736163	arm_stop_transac
61726D5F	64657374	726F795F	6170706C	arm_destroy_appl
61726D5F	6765745F	6572726F	725F6D65	arm_get_error_me
61726D5F	73746F70	5F617070	6C696361	arm_stop_applica
61726D5F	6765745F	636F7272	656C6174	arm_get_correlat

What the example shows is that DB2 is now ARM instrumented.

Verification for the WebSphere HTTP Plug-in

For the WebSphere HTTP plug-in, you can verify that ARM instrumentation is enable by using the trace option. Open the plug-in file on the IBM HTTP Server system and search for this line:

```
<RequestMetrics armEnabled="true" loggingEnabled="false" rmEnabled="true"
traceLevel="HOPS">
```

If you change the loggingEnabled parameter to true and traceLevel is set to other than NONE, you will see the correlator information in the HTTP plug-in log file, as shown in Example 4-14.

Example 4-14 The correlator information at http_plugin.log

```
[Mon Oct 10 14:24:40 2005] 000011ec 000002a4 - PLUGIN:
parent:ver=1,ip=9.12.4.141,time=1128968628390,pid=4588,reqid=0,event=1 -
current:ver=1,ip=9.12.4.141,time=1128968628390,pid=4588,reqid=0,event=1 type=HTTP
detail=/trade elapsed=31 bytesIn=0 bytesOut=0
[Mon Oct 10 14:24:40 2005] 000011ec 000002a0 - PLUGIN:
parent:ver=1,ip=9.12.4.141,time=1128968628390,pid=4588,reqid=11,event=1 -
current:ver=1,ip=9.12.4.141,time=1128968628390,pid=4588,reqid=11,event=1 type=HTTP
detail=/trade/images/tradeTopology.gif elapsed=16 bytesIn=0 bytesOut=40289
[Mon Oct 10 14:25:00 2005] 000011ec 000002a0 - PLUGIN:
parent:ver=1,ip=9.12.4.141,time=1128968628390,pid=4588,reqid=12,event=1 -
current:ver=1,ip=9.12.4.141,time=1128968628390,pid=4588,reqid=12,event=1 type=HTTP
detail=/trade/web_prmtv.html elapsed=47 bytesIn=0 bytesOut=10996
[Mon Oct 10 14:25:07 2005] 000011ec 000002a0 - PLUGIN:
parent:ver=1,ip=9.12.4.141,time=1128968628390,pid=4588,reqid=13,event=1 -
current:ver=1,ip=9.12.4.141,time=1128968628390,pid=4588,reqid=13,event=1 type=HTTP
detail=/trade/servlet/PingJDBCRead elapsed=93 bytesIn=0 bytesOut=525
```



EWLM concepts and setup

This chapter provides the concept of workload management and how it relates to the EWLM Domain Policy and its components. It also provides a description of the Control Center function and how to use it to administer EWLM through the definition and deployment of a Domain Policy.

5.1 EWLM concepts

In order to understand how EWLM works, we could take a look at the traffic in a city, as shown in Figure 5-1. Usually, when we drive with a car between two locations, the time we need is determined by the constant speed we are allowed to go, the amount of traffic on the streets, and the traffic regulations at crossings and intersections. Based on this, we can already identify the time as the basic measure to go from a start to a destination point; we can also easily understand that the amount of traffic is the determining factor of how fast we can go between these two points. While the driving speed is usually regulated and very much constant, the amount of cars on the street determine how long we have to wait at intersections and crossings before we can pass them. As a result of this, we can identify the crossings, traffic lights, and intersections as the points where we encounter delays on our way through the city; a traffic management system can now use these considerations to manage the running traffic; for example, dedicated lanes for buses and taxis allow them to pass the waiting traffic during rush hours and therefore to travel faster than usual passenger cars. This is a common model of prioritizing a certain type of vehicles against others in the traffic systems; so far we saw that it is possible to use the time as a measure between two points for driving in a city, we identified the points where contention may occur, and found a simple but efficient method of prioritizing and managing the traffic.

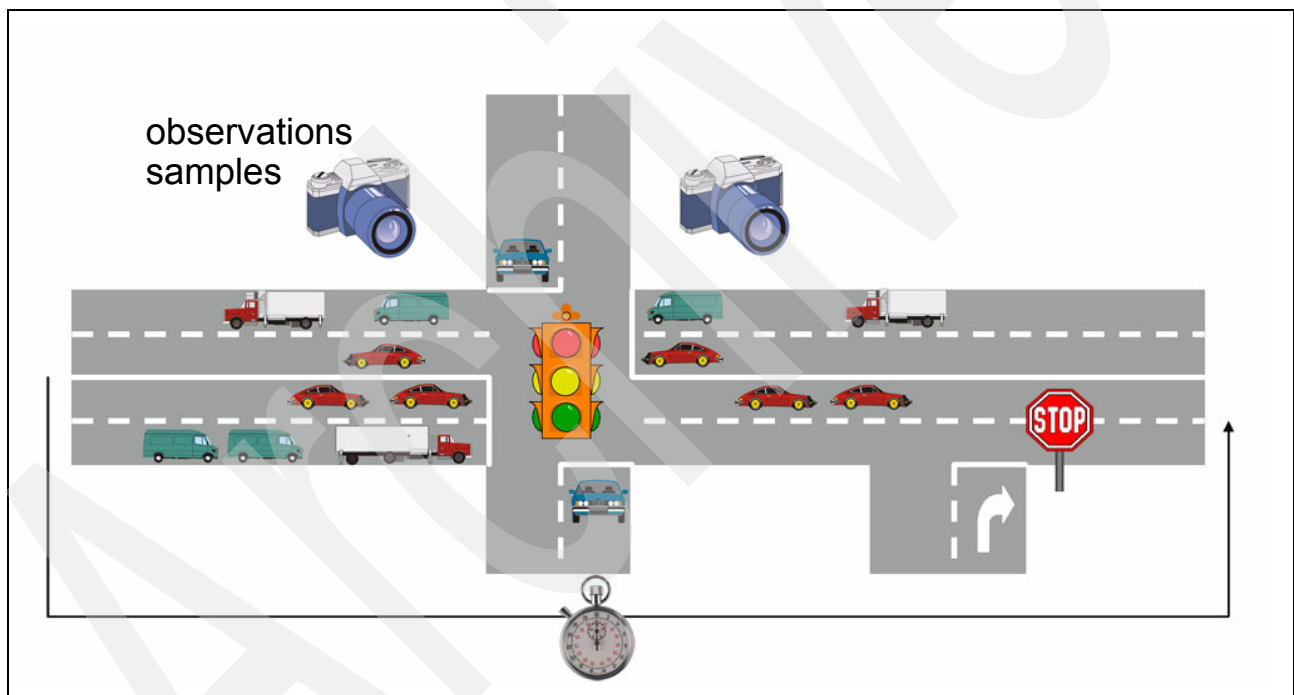


Figure 5-1 Traffic management concepts

These concepts can be easily ported to a system; all we need is a technique to:

- ▶ Identify the work running in the system.
- ▶ Measure the time it runs.
- ▶ Find out the contention points.

The identification of work requests is supported by the middlewares and the operating system; they tell EWLM when a new unit of work enters the system and when it leaves the system. EWLM provides rules to separate the work into distinct classes. This is named *classification* and allows an installation to deal with different types of work running through the enterprise.

Contentions occur when the work wants to use the system resources. These are the CPUs, the I/O devices, and storage, but also software constructs like processes or networking. EWLM monitors these resources to be able to understand how many resources each application would require or will wait for.

The classification and EWLM observation of how the work uses the resources provide the base to manage the enterprise. This management is done based on the goals the installation defines for the work. After classifying the work into distinct classes, the installation associates a goal with each class; the goal determines how much service the work in the class is able to receive. These classes of work are named *service classes*.

Besides the goal, it is also necessary to define which work is more important than other work. In the case where several service classes do not achieve their target goals, the *importance* will help decide which service class should be helped in getting resources.

Figure 5-2 describes the EWLM constructs used for the performance management. In the service policy we have some workloads. For each one we have some service classes indicating the goal and the importance of the associated work, and, in addition, transaction classes, process classes, and partition classes. In 5.1.1, “EWLM elements” on page 132, we analyze each component in detail.

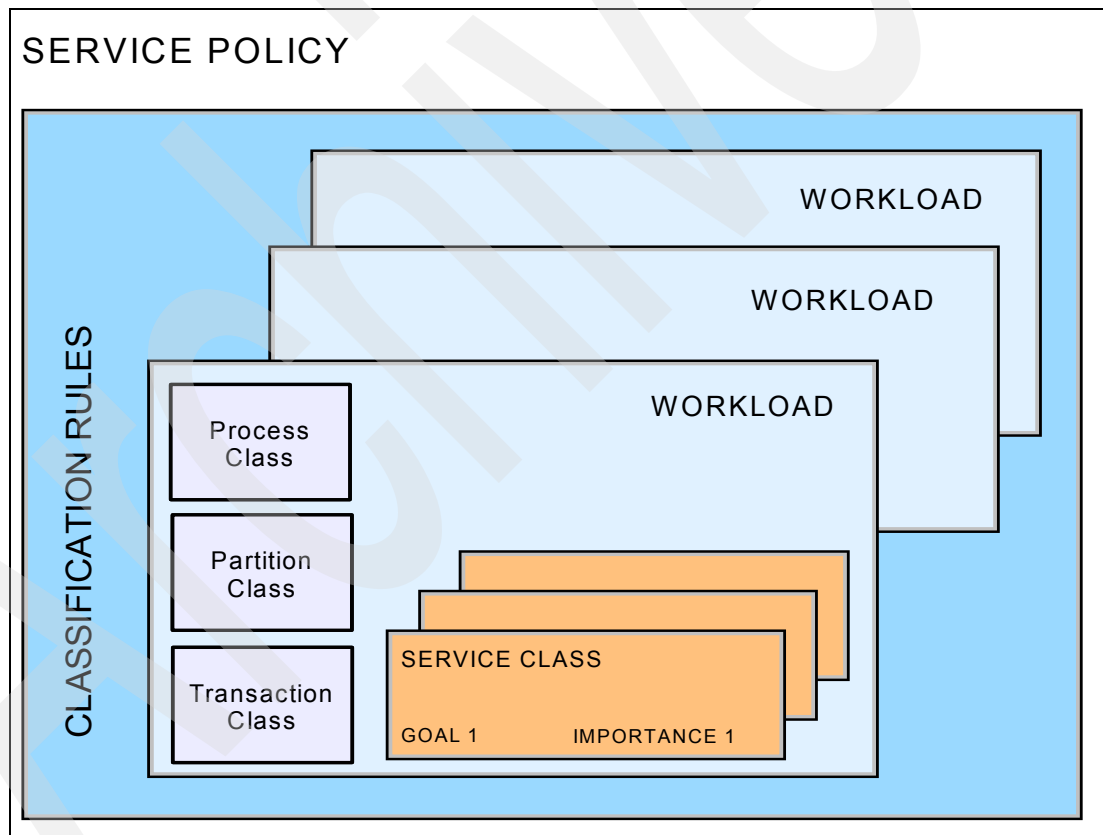


Figure 5-2 WLM service policy elements

Now we have all basic elements to define a policy, but we still have to clarify what kind of goals can be assigned to the service classes. From the traffic example, we see that the most natural goal is the time that is required for a car to go between two points. The same is true for work running in an enterprise: The time that the work needs between the points when it enters and leaves the enterprise can be used as a base to manage it. Since an enterprise deals with many work requests running in parallel, the time of an individual request becomes

less important. The average time the work needs to fulfill its task is more interesting because this is also the time the end user can observe while waiting for a response. This time is named the *response time*.

As we will see later, the average response time can be influenced by very few long-running transactions. In order to become more independent for such events, we want a more stable goal definition. For example, we would like a certain amount of work requests to end within a predefined time. Other work requests may run much longer, but we will consider the goal as achieved when the defined amount of requests ends in the expected time period. This goal is named a *percentile response time* goal and defines the percentage of all work requests that must end within a predefined time limit.

Using the time as a measure to manage the work on a system requires that the system has the ability to capture it. Usually, an operating system provides processes to execute the application programs; these are usually started once and live for a very long time. That means that EWLM needs assistance from the middleware to capture the response time. To capture the work requests behavior, EWLM relies on the middleware to be ARM instrumented. Using ARM instrumentation, EWLM supports an infrastructure that allows the middlewares to identify individual work requests, capture their response times, and thus allow EWLM to manage them in their own service classes. This means that EWLM is able to manage service classes of transactions that belong to the same application and that are executed by a set of servers.

At this point, the question that arises is what EWLM can do when it is not possible to capture the response times of the transactions. There are many reasons why this is necessary: A middleware might not have been ARM instrumented and be able to tell EWLM the response times of their transactions or there are too few and too sporadic transactions, so that it does not make sense to manage them towards a response time goal. Also, many processes provide services in the system, which cannot be easily attributed to a certain type of transaction. For those processes, it is also necessary to understand how fast they progress in the system and define a goal for them. Since the system is not able to capture the response time of this type of work, it must use the information it is able to collect: This is the information about the delay and using values it observes when the work tries to use resources in the system. This information can be obtained by observing where such delay situations may occur and constantly taking snapshots. After a certain time period, these snapshots are summarized and the result helps to identify how often individual work and the service classes were delayed for a resource and how often they were able to use it. The speed in the system can now be expressed by the acceptable amount of delay for the work when it executes. The speed is named *velocity*, a value from slowest to fastest, where fastest means that all measured samples are using samples and that the work did not encounter any delays for resources managed by EWLM, and slowest means that the work was completely delayed over the measurement interval.

Let us now try to compare the potential goals that can be defined for work in the system. On the one side we can have a response time goal. Independently of whether we use an average or percentile response time goal, we can always immediately understand what this means with respect to how fast the work progresses in the system. On the other hand, we have a velocity goal; clearly, we see that a velocity of Fastest means no delay and therefore it means maximal speed. But what does a velocity goal of Moderate mean? With respect to the traffic example, we know that it is always possible to go very fast by car in a city on a Sunday morning but not during rush hours where a fast speed always means that the car has to wait at intersections and traffic lights. That also means that the best we can do during rush hours implies that we have to wait very often and that a speed expressed as a velocity will be very low. If we want to define a reliable goal that can be achieved under all circumstances, we have to understand how the system behaves during rush hour. Such goals may be easily achieved

during other times because there is no contention and everybody and everything can progress very fast. When we have two service classes of work, one for which we can measure the response time and for the other where we cannot measure the response time, we need a way to compare the potential goal settings against each other. Because it is very difficult to understand how a velocity translates into a response time, we have to capture the velocity of the work being managed towards response time goals simply because we need to understand what this means as velocity.

At last, in addition to setting response time or velocity goals, you might want to have work running in the system for which no concrete goal must be achieved. This work should only run when plenty of resources are available. This work is classified to service classes with a *discretionary* goal and thus tells the system that it should only run when service classes with higher importance do not need the resources to achieve their goals.

In the next sections we analyze the EWLM concepts and the elements they use for workload managing in more depth; the starting point of the workload control process is to define a Service Level Agreement between installation and the end users. The Service Level Agreement for an installation can be either the contract between the end users and the IT environment or, in absence of a contract, can be derived by creating a model based on the normal application behavior. In case your installation needs to build the model, you need to observe your applications over their entire cycle to understand what should be considered an acceptable response time, which metrics will fit the best to measure the performance, and so forth.

The following items can give you an idea of which elements you should observe to understand the application behavior and build a performance model:

- ▶ Average transaction response time for network, I/O, CPU, or total
- ▶ Distribution of response time
- ▶ Throughput
- ▶ System availability (percentage of time the system is available to users)
- ▶ System load

Once you understand your workload, you can then go through the fundamental steps to classify, manage, and monitor the overall performance:

- ▶ Performance administration
 - Definition of work, goals, and business importance for all work.
 - Provide methods to allow others access to desired results.
 - Done via EWLM Control Center Setup function.
- ▶ Performance monitoring
 - Capture of results for each workload.
 - Where did the work spend its time.
 - Information about what got in the way of success.
 - Provide methods to allow others access to achieved results.
 - Done via EWLM sampling.
- ▶ Performance reporting
 - Single place where desired results and achieved results are reported
 - Done via the Control Center Monitor function

Even though EWLM management functions are currently limited to network load balancing and CPU resource distribution between partitions on a Power5 machine, we recommend that you build your policy in a such a way that your installation SLAs for the multiple workloads are already represented and monitored.

The following sections provide further details on these phases.

5.1.1 EWLM elements

Before going further with the discussion, let us get familiar with the terminology for the EWLM elements used to identify and track your workload.

You are already familiar with the concept of a *Management Domain*—a heterogeneous environment comprising a collection of managed servers and a domain manager. In this environment, the *domain manager* is the central point of control for a domain.

A Management Domain defines the environment on which a service policy runs. Only one service policy is active in a Management Domain at any given time and this policy applies to all managed servers in the domain. If no user-defined service policy has been activated, the domain runs with a default service policy provided by the installation.

Domain Policy

You can think of a *Domain Policy* as a contract that prescribes how EWLM should treat work within a Management Domain. This treatment may include how the work should be classified, prioritized, managed, reported, and so forth.

A Domain Policy is a collection of service policies, applications, transaction classes, and service classes. In addition, platforms, process classes, and partition classes can be optionally defined. The structural view of the components is shown in Figure 5-3. In the following sections we look at each component in turn and describe the relations of these components to each other.

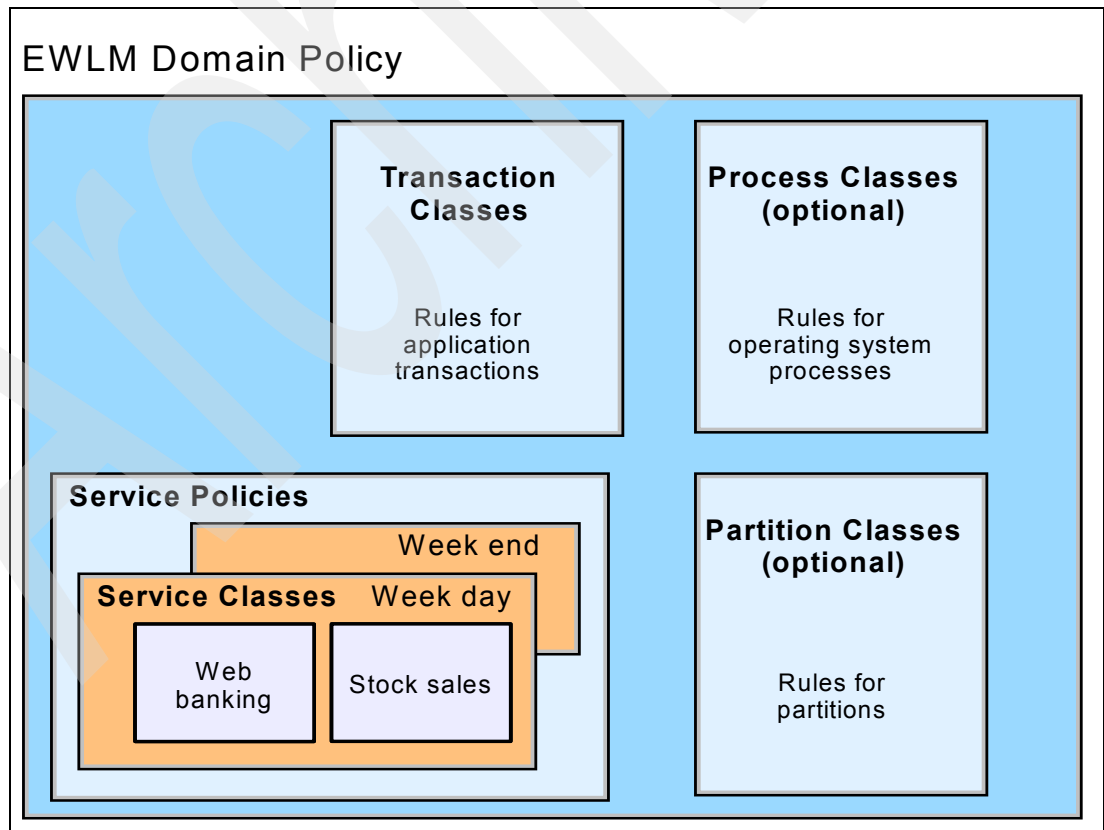


Figure 5-3 Structural view of Domain Policy

Service policy

A *service policy* describes the business performance objective and the business importance of work running in a heterogeneous installation. It describes the business view of workloads and ties it to the reporting and implicit management of the workloads. It does not contain any explicit linkage to server topology or middleware. Similar to a service level agreement contract, a service policy is defined by a name and consists of service classes, transaction classes, process classes, partition classes, and defined workloads.

While you can define multiple service policies depending on your installation objectives, *service classes* typically are the same for multiple policies. They usually differ in their goal settings only. However, in most cases a single service policy is sufficient.

As shown in Figure 5-4, you can have multiple service policies defined in the Domain Policy, representing different objectives for your enterprise. In this illustration, both service policies (weekend and week day) share the same service classes (Web banking and Stock sales); the only difference is that the performance goal for the service classes is slightly different to represent the different performance objective.

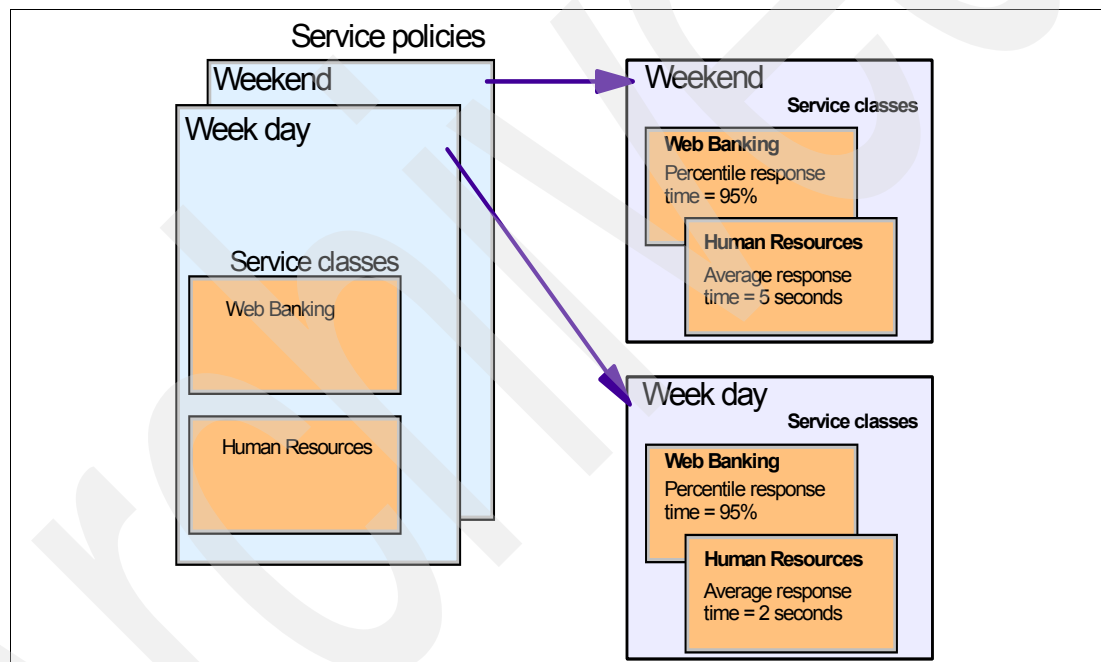


Figure 5-4 Service policy

Workload

An optional feature of the EWLM Control Center is *workload*. It is used to group a set of service classes that you want to put together for reporting purposes.

Service class

A *service class* is a central concept to EWLM and represents a group of work that has similar performance goals or business requirements. You must define a specific goal as well as an importance factor. A service class contains only one goal and one importance factor. The supported performance goals are the following:

- ▶ **Percentile Response Time:** What percentile of work requests should complete in a specified amount of time. The specified value is given as a percentage. The time value can be in hours, minutes, seconds, and microseconds.

- ▶ **Average Response Time:** The average amount of time in which work in the service class should complete. It is specified in hours, minutes, seconds, or microseconds.
- ▶ **Velocity:** Defines how fast work should run relative to other work requests when ready, without being delayed for CPU, storage, and I/O. Velocity goals are intended for work for which response time goals are not appropriate. In particular, this is the case for all kinds of processes that are not instrumented, such as server processes, daemons, or long-running batch-like work. The following are some velocity type values:
 - Fastest
 - Fast
 - Moderate
 - Slow
 - Slowest
- ▶ **Discretionary:** Low-priority work for which you do not have any particular performance goal or importance. EWLM processes discretionary work using resources not required to meet the goals of other service classes. It is used for workloads whose completion time is unimportant.

In addition, you must also specify how important it is to your business that the assigned goal is met through the *Importance* factor. Importance plays a role only when a service class is not achieving its goal. For instance, resources may be taken away from a less important service class in order for a more important service class to achieve its goal. You can specify five levels of importance settings:

- ▶ Highest
- ▶ High
- ▶ Medium
- ▶ Low
- ▶ Lowest

Tip: While it should not be a performance issue, we recommend that you limit the number of service classes to reflect real differences in performance objectives. A good rule of thumb is to limit your service classes to around 25 to 30 for ARM transactional work. If you have hundreds of service classes in mind, you might be trying to micro-manage the environment. Sampling is used to allocate resources so EWLM is more efficient when more work is present in any given service class. EWLM can make better decisions more quickly and hence be more responsive to changing system conditions.

The number of transaction and process classes should be limited to what is meaningful and manageable from a reporting perspective. The number of service classes should be limited to the classes of work that have distinguishably different performance objectives. If considering usage of more than 100 transaction, process, or service classes, start monitoring memory and processor usage on the domain manager as the configuration approaches 100 managed servers.

Regarding transaction classes, what is important from a performance perspective is the number of transaction classes active on each server. If there are many transaction classes in the policy, but only a few are active on each server, there should not be much of a performance impact. On the other hand, many transaction classes with every transaction class active on every server could have more of a performance impact.

Start with a simple policy and grow as required.

Measuring your workload

Now that we introduced the concepts of service class and goal objective, we would like to give you an idea on how EWLM understands whether these goals are met. For this scope, EWLM

uses the performance index; this is not a metric defined in Domain Policy, but it is a value calculated by EWLM to represent whether a certain service class is achieving the goals. It is independent of the type of goal associated with the service class. The performance index (PI) is in fact used to compare goals of different types and the calculated value is used to determine how well work is meeting its goal.

The following are the values for a performance index:

- ▶ PI = 1.0 means that the service class is meeting the goal.
- ▶ PI > 1.0 means that the service class is missing the goal.
- ▶ PI < 1.0 means that the service class is overachieving the goal.

Work classification

Now, before going further with the description of the remaining elements, we need to introduce the concept of work classification. EWLM uses the transaction class, process class, and partition class to identify the type of work coming into the enterprise. This association happens by using filters. A filter is what identifies a work request in the entry point middleware. Transaction classes, process classes, and partition classes will then be assigned to the service class that would represent the type of performance goal we want to achieve for this particular workload.

Figure 5-5 is a representation of the classification process.

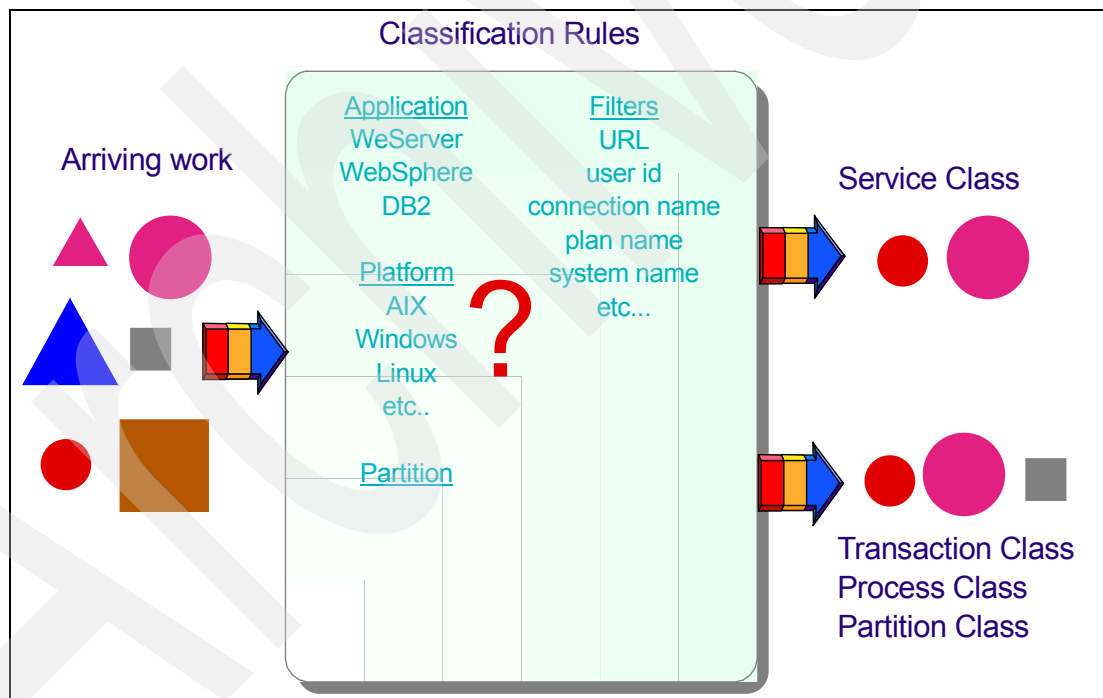


Figure 5-5 Classification process

Every time a work request enters the first middleware, the middleware issues an EWLM call passing the work request properties to EWLM. These properties are matched against the filters described by the installation in the policy classification rules. Filters are different depending on the middleware. After the matching, EWLM associates this work request to the service class through either a Transaction class, a Process class, or a Partition class. From this point, it is now a matter of enforcing the goals described in the service class.

Rule

A *rule* provides a way to identify and to classify the transaction, partition, or process classes to which work belongs. They consist of a filter type, a filter value, and an operation to apply to the filter type-value pair to determine if there is a match. At least one rule must be defined when creating a transaction, process, or partition class. The filter type specifies what attribute of work requests you want to use to identify the work.

Enterprise Workload Manager Control Center

Log out | Help User: ewlmdm Role: Administrator

Active service policy: **ITSO Trade6 Service Policy fo...** | Domain policy: **ITSO Trade6 Domain Policy** | [Domain Settings](#)

Domain Policies ⇒ Transaction Classes ⇒

Edit Transaction Class

Edit transaction class properties. Click Edit to change the rules for the class. Transactions that match the defined rules are assigned to the service class specified.

Application name: IBM Webserving Plugin Default service class: EWLM Service Class

*Name: ITSO_Redbook TC1

Description: ITSO Redbook sample transaction class1

*Position: (Enter a number from 1 to 3) 1 Service class: Gold_Trade6_SC

Rules

(EWLM:URI == "/trade(*)"
AND EWLM:Hostname == "ewlm1")

Edit... Delete...

OK Cancel

Figure 5-6 A sample of defined rules

Filter

A *filter* is used by EWLM to identify work requests that are processed in the applications and platforms that you are monitoring. The filters that you use to identify work requests depend on what was ARM-instrumented in the application or what was EWLM enabled in the platform. When you set up the application or platform in the EWLM Control Center, you specify what filters to use. The filters are available in the Domain Policy when you create transaction, partition, or process class rules. Example filters are EWLM: user name, EWLM: OS Platform, and EWLM: URI (Uniform Resource Identifier).

You can have nested rules to obtain a more granular classification, although you need to remember that performance is search sensitive. For this reason, you want to place highest probability of hit first.

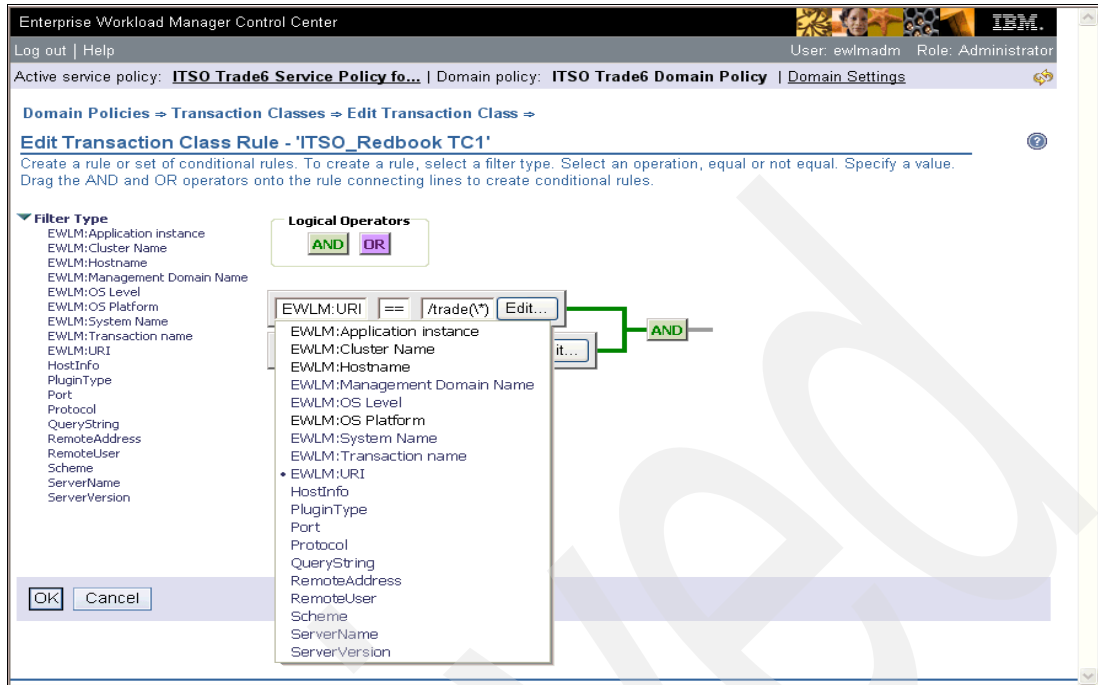


Figure 5-7 Defining filter example

Application

An *application* is the starting point for classification of transactions into transaction classes and must be ARM-instrumented in order for it to be monitored by EWLM as an application. Examples of application environments include WebSphere Application Server, HTTP Server, and DB2.

You can assign a default transaction class and service class to an application. Transactions that are not classified to a transaction class are then assigned to the default transaction class and service class.

Each application environment provides a specific set of rules to classify transactions and assign them to a transaction class.

Transaction class

Transaction classes are used to group transactions that have the same goal or that are reported together and are mapped to service classes. The classification of transactions into a transaction class is done through the use of rules. Complex rule sets can be built, consisting of multiple nested rules, in order to accurately classify transactions. The classification rules are specific to each application environment.

An application must be ARM-instrumented in order for it to be monitored by EWLM as an application. It is the starting point for classification of transactions into transaction classes. For example, WebSphere Application Server, HTTP Server, and DB2 are examples of applications that are ARM-instrumented. EWLM applications that are not ARM-instrumented can still be monitored, but as process or partition classes.

You can use transaction classes to not only define which work should be assigned to a service class, but also to define which work within a service class you would like to specifically monitor. Transactions that are not classified to a transaction class are then assigned to the default transaction class and service class.

Platform

The *platform* identifies the environments where process classes are supported. Currently, AIX, i5/OS, z/OS, Windows, Linux, HP-UX, and Solaris are the supported platforms. The names of all EWLM-monitored processes must be added to the Domain Policy.

The filter types available for process classes are specific to each platform.

Process class

A *process class* defines a group of system processes intended for reporting and management according to a service class. A process class monitors work requests that a system initiates, while a transaction class monitors work requests that users initiate from an application. Non-ARM instrumented transactional and application servers have to be monitored as process classes.

You can map several process classes to the same service class. The classification of processes into a process class is done through the use of rules. These classification rules are specific to each platform. You cannot map a process, a transaction class, or a partition class to the same service class. The mapping is mutually exclusive, which means you either map a process class *or* a transaction class to a service class.

Partition class

A *partition class* identifies all the work within a partition (at the current date only on an IBM System p5 or eServer P5 system), including both application-level transactions and operating system processes.

We suggest the use of a partition class when all the workload on a partition has the same performance goal. A partition class monitors the workload on the partition as a whole rather than as individual transactions and processes. If the type of workload is transactional, remember that a partition class includes all application-level transactions regardless of the application instrumenting ARM 4.0 standards. Consequently, the performance data for partition classes is not as granular as it is for transaction classes.

Work running within a partition can be also reported through a transaction or process class, if it meets the classification filters for either transaction or process class. For example, if a Domain Policy contains a transaction class for the WebSphere Application Server unit of work and contains a partition class for the system on which the WebSphere Application Server runs, EWLM will classify the work that WebSphere processes to both a transaction class and a partition class. This allows you to monitor work performed by the WebSphere Application Server as part of an overall partition workload and separately as transactions only.

When should a process class be used versus a partition class? When you define a process with a process class, keep in mind that you most likely map each process of this type running on every platform against the same performance goal (service class), and they will all be reported as an entity in the process class. For example, if your installation has 100 Oracle database servers running on AIX, you might decide they have the same performance goal, hence, they can be mapped to the same service class and have the same velocity goal. In such a configuration, if you want to monitor a specific Oracle database server, you need to add additional filters to isolate this single process and map it to a unique process class. In the situation where these processes are running on an IBM System p5 or eServer P5 system and you would like EWLM to manage the CPU resource, you would need to identify these single processes and isolate them through the classification process. Or, in a much simpler way, you can define a partition class with a respective velocity goal associated with the partition where the Oracle process is running. In this way, you can have a generic process class for all the Oracle processes in your configuration and a partition class for the ones running on an IBM System p5 or eServer P5 system.

Another way to see the partition class is where you want to manage all the work running on a partition as a whole entity without specifying classifications for each of them.

Now that all the different elements belonging to a policy have been introduced, let us take a step to understand what steps you need to take in your installation to build the Domain Policy.

5.2 Building the Domain Policy

In order to build a Domain Policy, you need to assess your applications and classify them according to their business goals. The following sections provide details for each phase of the process; once you complete, you will be ready to log in to the Control Center and build the policy.

5.2.1 Assessment

An assessment should be made to understand the applications and the platforms present in your environment, the type of workload, and the business goal either dictated by an SLA agreement or by workload behavior. You might also consider starting from an application and then moving on to another workload. The following is a list of tasks that you might want to consider during your assessment:

- ▶ Build an application list and choose an application to start from.
- ▶ Identify the edge server of the application. The starting point will be the edge server or the entry point of the application into the Management Domain. This means it is the first hop, or hop 0, and is where classification will take place. To accomplish this task, you might need to contact the application owners or use some tracing facilities where possible. Another practical method to discover the edge of your workload can be found by running it with a default policy. Since classifications are not present for your workload, it will be classified under the default service classes provided by EWLM. You can use this information to verify which middleware is the edge server and the application and server topology.
- ▶ If your installation runs multiple business applications, you are probably dealing with a large number of transaction types. You can classify these transactions into groups of transaction classes with similar goals and mapped to service classes. There can be a one-to-one mapping of transaction classes to service classes as well as a mapping of multiple transaction classes into one service class. You can also mix transaction classes from different business applications into the same service class if they have the same importance and response time goal. The question to think about carefully is how many different business goals do you have and therefore how many service classes do you want to define. Is it really necessary to split your business applications into hundreds of transaction classes or is it sufficient to classify in a more general and global way?
- ▶ A naming convention is also an important factor to think about when creating your Domain Policy. As soon as your environment processes more than a handful of transactions, it can be difficult to understand what workload you are monitoring or managing if you do not have a clear naming convention. For service classes, we recommend using a name that indicates the importance of your service class rather than the name of the business application. This makes it clear at first sight when you look at the performance of the service classes if missing its goal is of major or minor importance. Another reason for naming based on business importance is that you can map transaction classes from different business applications to one service class like Trade or Plants, for example, if they have the same or similar performance goals. Clearly naming a service class something like Trade6_SC does not help describe its business importance, while naming it *Gold* might immediately identify how important this service class is for your business. For

transaction classes the name should give an indication of the action of the transaction or the application they belong to so it is easier to identify which business process is involved.

- ▶ Identify whether the workload has predefined business objectives. For example, transaction-type work may have a previously established Service Level Agreement (SLA). An existing SLA should specify required levels of service and can be the basis for setting performance goals during the classification process. If you do not have an SLA for your workload, you might want to consider temporarily assigning a service class with a discretionary goal to this workload in order to monitor the workload and build a performance model that can be used to define the real-time performance goals.

In setting up the goals, remember that response-time goals should be used whenever possible, although they are not appropriate for all types of work. For example, we do not recommend response time goals when you have a very low number of transactions (the frequency of completion needs to be at least 10 completions in 20 minutes for a response time goal to be effective).

5.2.2 Classification

Once the assessment process is complete, it is time to define the work and its attributes to EWLM. When you first implement EWLM it is important to classify your work into distinct service classes, to set up an initial set of goals, and verify the applicability of your definitions on your running system. The following is a list of tasks that you might want to consider during your classification:

1. Create a service class for the application that defines the performance goal.

In setting up your service class, there are two objectives that you want to achieve:

- It is necessary to distinguish different work and to define service classes in a way that different groups of transactions/processes can get the service they need from the operating systems.
- You want keep your service classes to a minimum. If there are too many service classes it is possible that EWLM needs too many adjustment cycles to really calculate all possibilities to adjust the resources.

The results of these requirements say that on one hand we need to define service classes to distinguish all types of work from each other, and on the other hand we need to restrict ourselves to help EWLM to work efficiently. Here are some guidelines that can help in satisfying both requirements:

- Observe the default service class for each application environment. If you see transactions in the default service class, it means that you have workload in your environment that has not been classified and associated to a service class. The default service class is associated with a discretionary goal and might not suit the objective for this workload.
- It is not meaningful to create service classes for the same type of work with identical service objectives. For example, if you define process classes for different applications with exactly the same goals and importance, you can combine them into the same service class.
- Do not combine diverse types of work into one service class. For example, do not combine transaction and processes into one service class. EWLM sampling data, plots, and projections can become distorted when this is done.

The general rule is to have a sensible number of service classes. Transaction classes with similar performance goals would normally be mapped to one service class. If you know the performance goal for each transaction you are ready to do your mapping. In many cases this information is not available, which means you might want to start by mapping all

transaction classes to one service class and then monitor the response times for each transaction class. An alternative and more complicated approach would be to define a service class for each transaction class and to monitor the performance of each service class.

Based on these guidelines, it should be possible to stay within the range of 25 to 30 service classes with non-discretionary goals, even in environments with a diverse workload.

2. Identify the work as a transaction, a process, or a partition class.
3. Define all necessary rules to identify work requests.

Now, with all the information about your installation, you are ready to define your Domain Policy.

5.2.3 Building the Domain Policy

Once you have an idea of how to classify your workload, you are ready to build the Domain Policy. This may well be an iterative process; it may be necessary, especially in the early phase, to observe your workload environment and verify whether your definitions are meaningful. So, after the first deployment, you might want to revisit and reevaluate your Domain Policy until you obtain the desired management and reporting results. The following is a list of tasks that you might want to consider while building your policy:

1. Log in to the Control Center as described in “Administering EWLM through the Control Center” on page 150 as administrator.
2. Create the Domain Policy.

There are several ways to build your policy, as explained in “Using a sample policy” on page 142 and in “Using the wizard” on page 143. You can:

- Use the wizard, which will guide you through the entire process of building a new Domain Policy.

Create a New Domain Policy with the wizard, then click **Finish** immediately, rather than Next. This will save an empty Domain Policy where, at a later time, using the edit function, you can add the remaining definitions.

- Create a new policy from a sample one and alter it to the specific needs of your installation. This is the easy and recommended way if the sample policy is very close to what you expect from your installation.
- Use the Import function from a previous exported policy. The export function creates an xml document that you can save in any directory. To import the policy back, select **Domain policies** in the Set up menu and click the **Import** function. In the pop-up window returned, input the name and location of the exported xml policy file and click **Import**. This imports the Domain Policy back into the current database.

3. Deploy and activate the service policy in the Management Domain.

At this stage we recommend that you use the EWLM Control Center to monitor the workload. Once you know the actual performance achieved you can then establish realistic goals for the workload and map the transactions to the appropriate service class.

4. Revisit and reevaluate your Domain Policy.

The objective of revisiting your Domain Policy is comprised of the following steps:

- a. Determine whether your classification rules still work. This means whether all of your service classes are still used and whether the usage of your service classes has changed.

- b. Determine whether the goals you set are still being achieved and are still appropriate for your environment.
- c. Verify that the classification occurs as you expected. There may be mistakes in your classification filter no matter how carefully you implemented it. To verify the classification we recommend using the EWLM Control Center. Start your workload and go to the transaction class view. If your classification is correct you will see entries for response time and an entry application where the classification took place. You should also verify the application and server topology for the transactions. If you do not see any performance data for some of the transaction classes you need to insure that the transactions are actually running. The application log files may help in verifying if certain transactions are running. If they are running then you need to go back to your classification and adjust the filter values until you get the transaction classification correct.

Reevaluating the Domain Policy can be seen as a consequence of revisiting the service definition:

- a. If the classification rule filters no longer match or if the workload has been changed, it might be necessary to change the rules, remove service classes, and possibly add new service classes.
- b. If the goals are not in the expected range it might be required to tune the goal definitions.

Using a sample policy

If the sample banking Domain Policy resembles your installation definition, a good start is to make a copy of the policy and alter or add definitions to tailor it to your specific needs. Use the following steps to do this:

1. At the Control Center, select **Set up** → **Domain policies**. Select the **Sample Banking Domain Policy** and **New based on** from the pull-down menu. Click **Go**.
2. Enter the name of the new Domain Policy at the prompt. We entered ITSO_Redbook. A new policy ITSO_Redbook, with exactly the same definitions as the Sample Banking policy, is created.
3. This Domain Policy has a single service policy association: Banking 2004 Service Policy. If the name of the service policy does not apply to your installation, you can rename it. To do this, edit your new policy by doing the following:
 - a. Click **Service policies**.
 - b. Select the **Banking 2004 Service Policy**.
 - c. Select the **Edit** function from the pull-down menu and click **Go**.
 - d. Change the name of the policy to a name that is consistent with your installation standards.
4. Edit definitions or add new definitions to the policy as needed.
5. When you have finished defining your service policy, use EWLM to verify that it is formatted properly, contains valid syntax, and whether the managed servers will activate it successfully. To automatically check the policy, begin by selecting **Set up** → **Domain Policies**. Select **Sample Banking Domain Policy**, and then **Verify** from the action pull-down menu. Click **Go**.

The EWLM Control Center verifies the Domain Policy on all servers defined in the domain.

6. You are now ready to deploy the Domain Policy ITSO_Redbook. To do this, begin by selecting **Set up** → **Domain policies**. Select **ITSO_Redbook**, and then **Deploy** from the action pull-down menu. Click **Go**.

7. EWLM displays the Deploy Domain Policy page. In the Service Policy to Activate field, select the service policy you just created.
8. EWLM displays the Deploy Domain Policy and Activate Service Policy Status page. When you get to this point, you have defined, verified, and deployed a Domain Policy, and activated a service policy for your installation.

Deploying a Domain Policy may take a few minutes. Be prepared to wait until all servers have come up as active. After you make changes to your policy, you may find that it is quicker to deploy it again rather than just activating the service policy. The result should be the same, but it is an alternative to try when you feel the activation takes too long.

Using the wizard

The intent of this section is not to step through all of the wizard panels but to highlight some main views and show the important steps in creating a new policy.

The Domain Policy is the anchor level defined within the domain. Therefore, the Domain Policy is the first item that is defined. To perform this task, log in to the Control Center as administrator, then click **Set up** → **Domain policies**. You should see the Domain Policies screen shown in Figure 5-8.

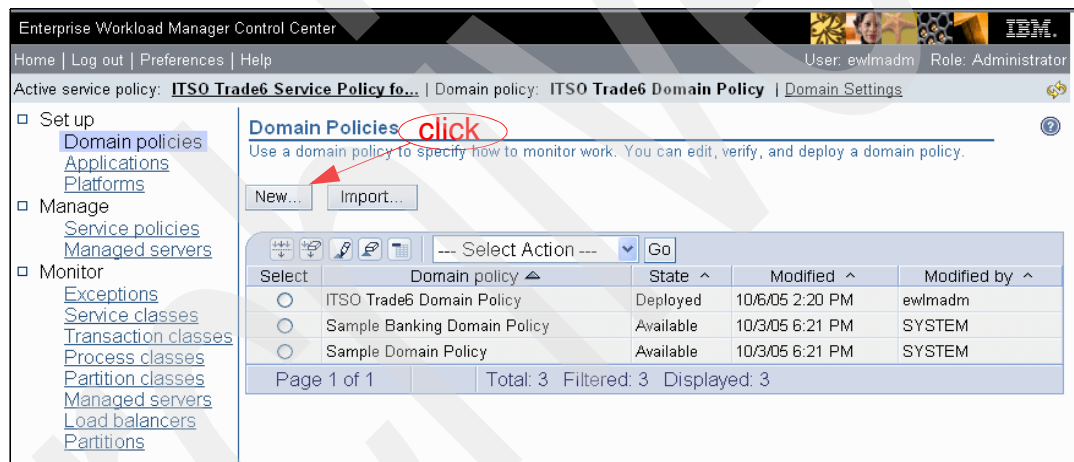


Figure 5-8 Initiating a new Domain Policy

The Domain Policies screen lists all the Domain Policies currently defined to EWLM. To define a new policy, click **New**. The Domain Policy wizard welcome screen shown in Figure 5-9 is presented.

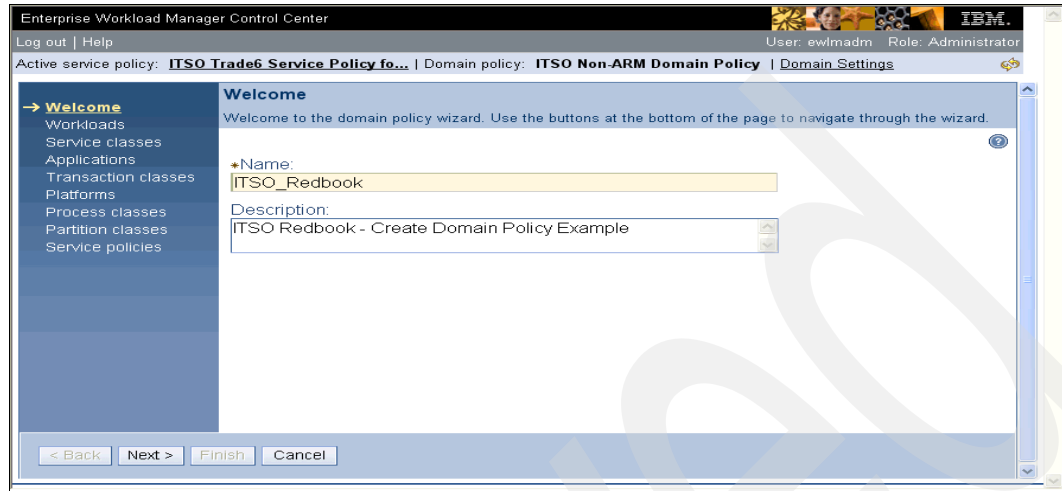


Figure 5-9 Domain Policy wizard welcome screen

The left side of the Domain Policy wizard screen lists all the resources that might be defined in the policy. The wizard keeps track as you progress through this list. Use the welcome panel to enter the name and description of the Domain Policy you want to create. In this example, we assigned the name ITSO_Redbook to the new Domain Policy being created. Click **Next** (not shown in the figure) at the bottom of the screen to continue.

Any time you click **Next**, the wizard proceeds to the next resource to be defined until it completes the list presented in the left panel. For this discussion, we go through the steps to produce just the workload definition, then we skip to the end of the wizard without going through all the other resource panels.

Figure 5-10 on page 145 shows the first screen used to define our first resource, the workload.

Tip: Creation of the workload is optional, and in fact, it may seem a bit strange to create the workload first. From a logical point of view the service policy seems to be the first step in your Domain Policy, but since the elements are related to each other in various ways it is suggested that you follow the order shown by the wizard.

For your production environment we recommend that you do proper planning before starting to build your policy. This includes preparing a list of service classes; mapping transaction, process, and partition classes to service classes; as well as preparing a list of platforms, applications, and workload. Once these elements are defined, the order in which you enter them into the Domain Policy Wizard is not important.

Defining a workload means assigning a logical set of service classes for the purpose of reporting. When a workload is defined, it is created and maintained at the Domain Policy level. To create a workload definition, click the **New** button shown in Figure 5-10.

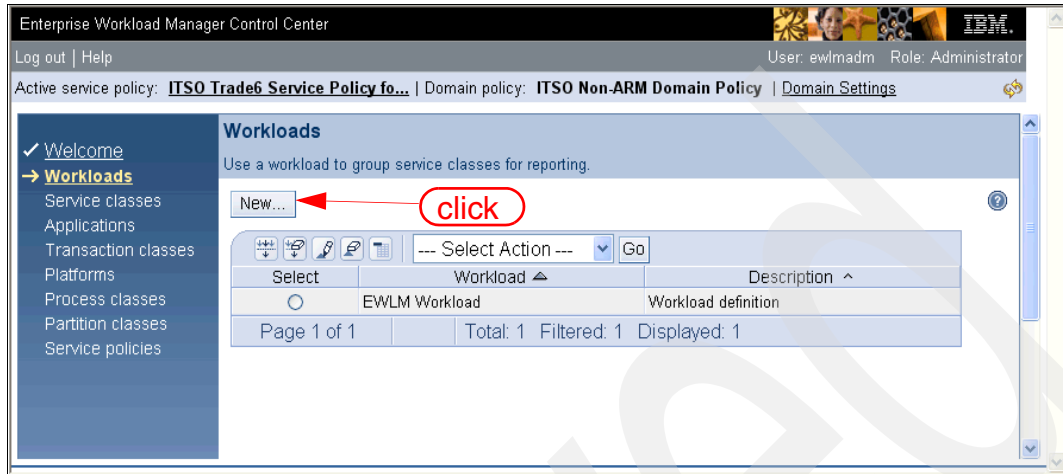


Figure 5-10 Workload definition

Click the **New** button, then enter the name and description to assign to this grouping of service classes. When done, click **OK** to continue.

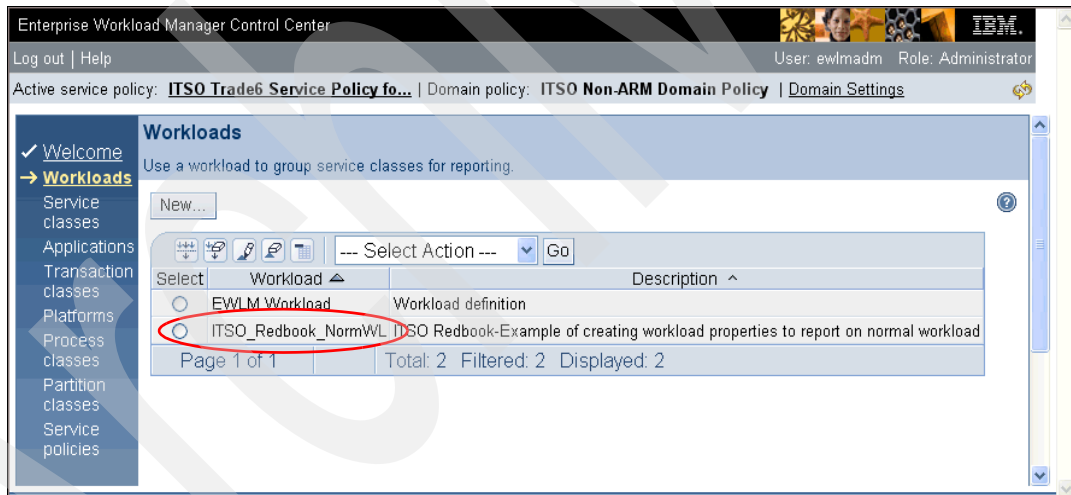


Figure 5-11 Workload list

As illustrated in Figure 5-11 on page 145, every time a definition is completed, the wizard shows the new definition (in this case the new workload ITSO_Redbook_NorWL just created).

Tip: You can keep on going through the setup, defining service classes next, and so forth, but during this process the information is not saved. When using the Domain Policy wizard, or any other part of the Control Center for that matter, if there is prolonged inactivity, a logon time out may occur and the updates will be lost. A similar loss will occur if you experience a browser problem during set up. The wizard does not save the updates, so be sure to enter all information from start to finish before stepping away.

An alternative is to click **Finish** after the definition of a workload, and then go through service class, transaction class, and so forth manually, clicking **Finish** after you define each element. Apart from producing incremental saves, there is no difference in defining the resources in this way; you still follow the panels provided by the wizard.

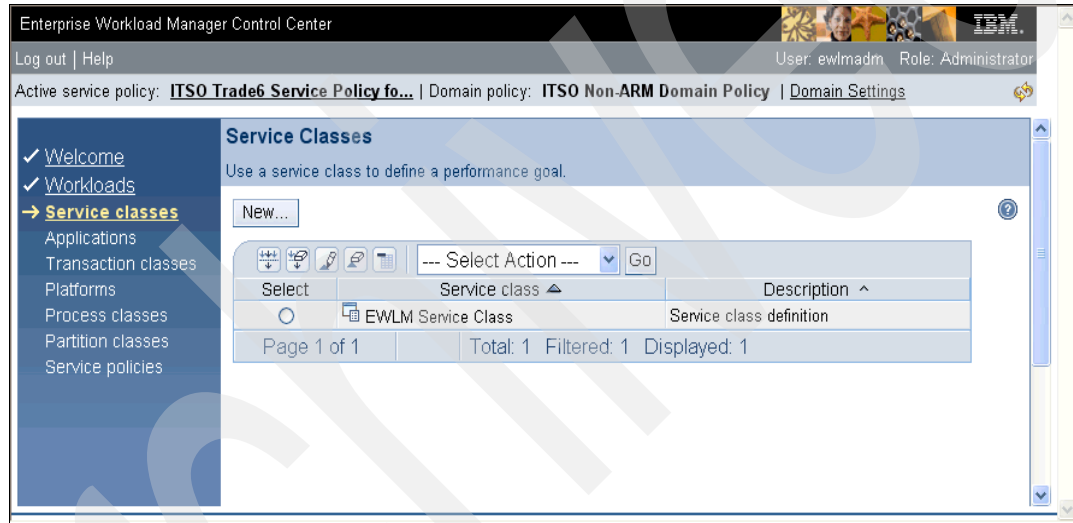


Figure 5-12 Service classes

The screen shown in Figure 5-12 gives you a place to create service classes. As you can see, as you proceed in the wizard the Domain Policy elements that have been defined have a check mark. You need to continue with the wizard to complete the definition of the elements that are not yet checked.

The last resource to be defined is the Service policy. When you complete the wizard, click **Apply** at the bottom of the screen to save your entries and changes, and then **OK** to save the Domain Policy.

Before EWLM can use the information just saved within the Domain Policy, the Domain Policy needs to be deployed. *Deployment* stages the service policies within the Domain Policy for activation. However, only one service policy within the Domain Policy can be active at any one time.

1. Select the Domain Policy to be deployed from the list of domain policies.
2. From the action pull-down menu, select the **Deploy** function.
3. Click **Go**.

In Figure 5-13, the Domain Policy ITSO_Redbook was selected for deployment.

The drop-down menu displays all the service policies available for activation within the ITSO_Redbook Domain Policy. It also presents an option of None. When None is selected, activation of the service policy is done as a second step. Here, the EWLM Service policy is selected to be activated once the Domain Policy is deployed. Click **Deploy** to deploy the ITSO_Redbook Domain Policy and activate the selected service policy on all the managed servers.

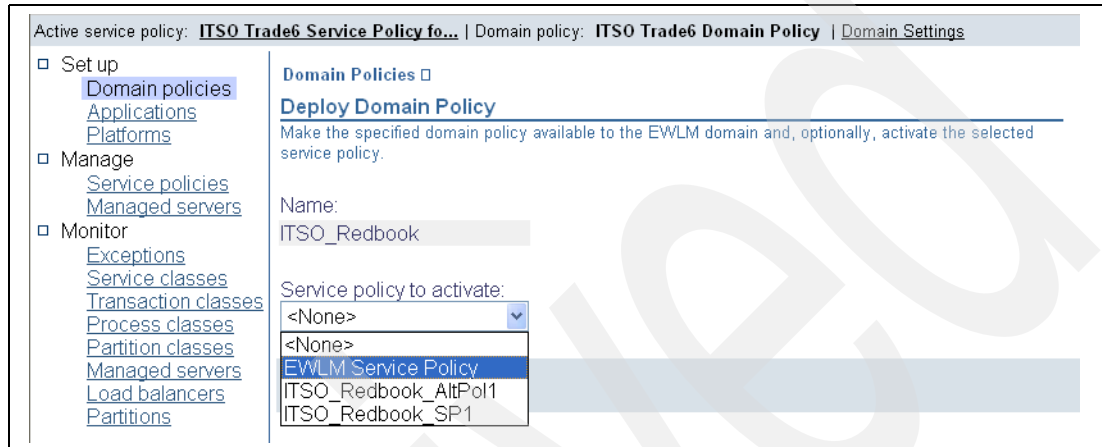


Figure 5-13 Policy deployment panel

Following deployment and activation, the screen in Figure 5-14 is shown. The first field notes that the Domain Policy deployment was successful. The service policy being activated is identified in the second field. The fields labelled 3 give the details about the service policy, specifically that activation was successful and that it was successful on all managed servers (5 of 5).

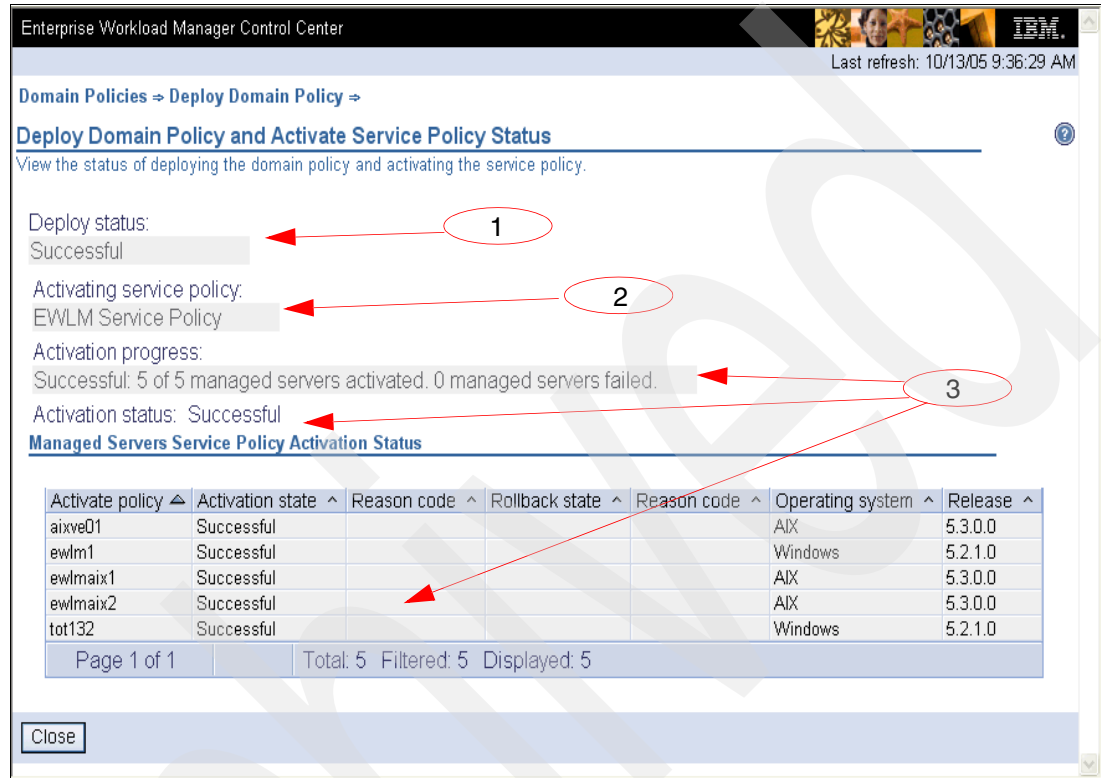


Figure 5-14 Policy activation

Adding a New based on service policy

If you want to use several service policies in your environment, for instance, at different times of the week or month, it is easiest to create the additional service policies using the *New based on* function. The advantage is that you have all service, transaction, partition, and process classes, and all other elements already defined. This includes classification filters as well as the original goals. All you need to do is to change the goals within the service classes included in this service policy.

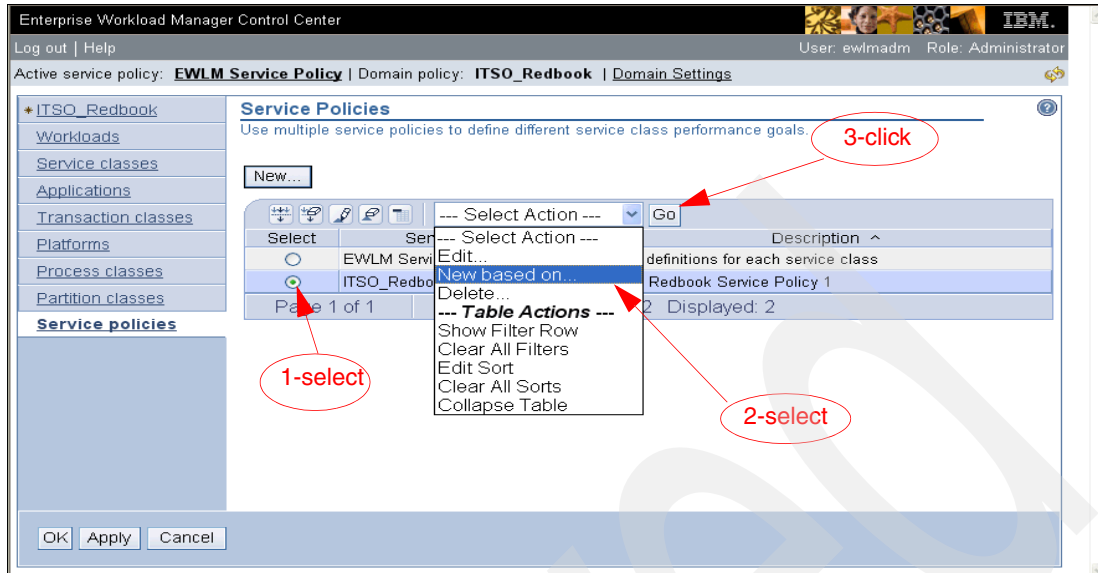


Figure 5-15 Define a Service policy using “New based on”

You need to have a base model service policy. You select the service policy and then the **New based on** function from the action pull-down menu and click **Go**. This creates a new service policy with the exact definitions of the based on policy. You still need to alter goals within the services classes in order to accommodate differing workload requirements during the day, weekend, shift, and so forth.

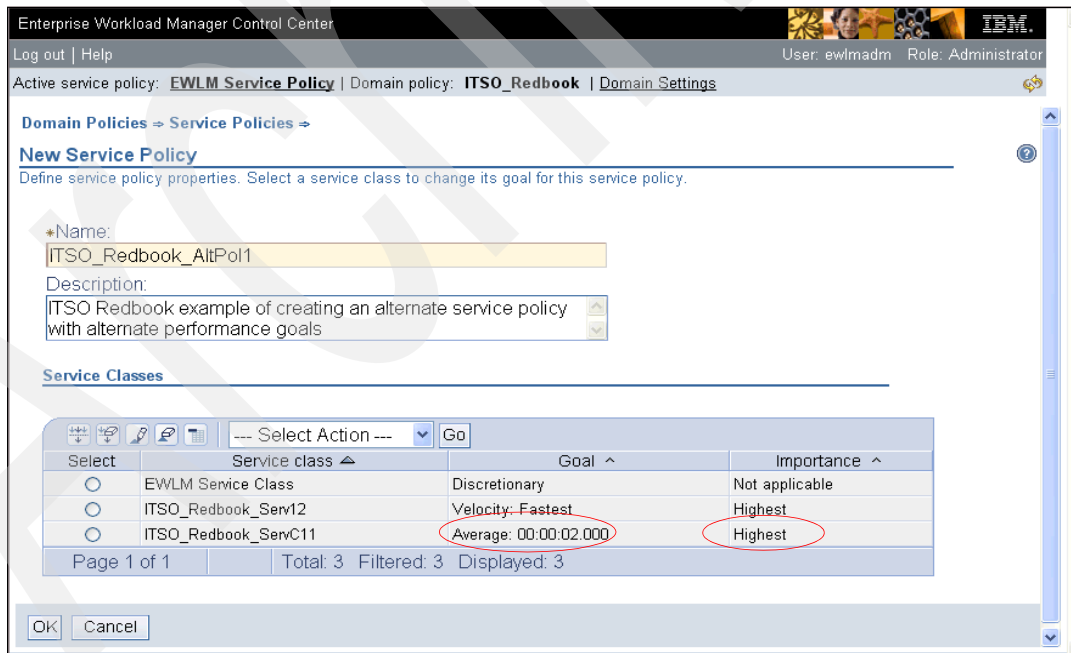


Figure 5-16 Defining a Service policy

Figure 5-16 shows that all the service classes included in the new service policy have the same goals as the original service policy. If you had chosen to use the New rather than the “New based on” function the goals would have been set to discretionary. Because the “New based on” choice is similar to a copy function, at this point you only need to change the goal for the service class you are interested in.

5.3 Administering EWLM through the Control Center

EWLM provides a browser interface to the Control Center. Figure 5-17 shows the Welcome panel.

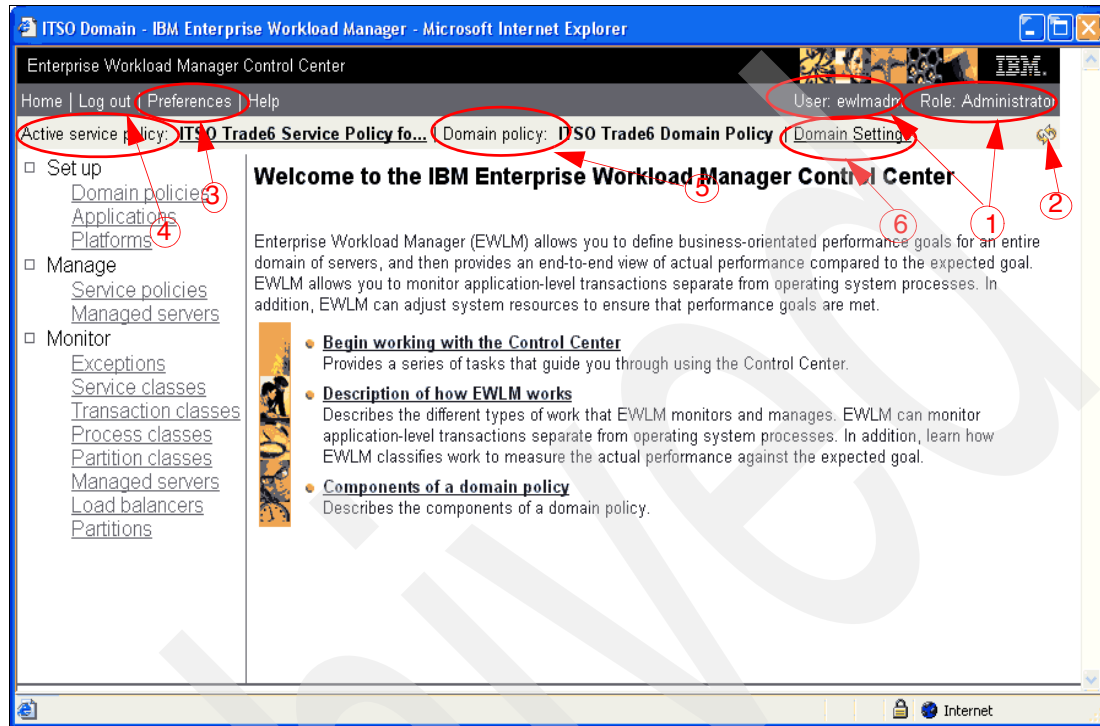


Figure 5-17 Welcome panel for EWLM administrator

Certain information displayed on the Welcome screen is carried forward to other screens throughout the Control Center including the following items identified by number in Figure 5-17:

- ▶ The logged-in user ID (User) and the associated authority (Role).
- ▶ The circling arrow icon. When clicked, this refreshes the content of the screen.
- ▶ Preferences - Used to change the reporting interval of the monitoring; more details in “Interval selection” on page 153.
- ▶ The name of the active service policy that was activated last and is currently running.
- ▶ The name of the Domain Policy currently in use.
- ▶ Domain Setting - You can use this option to view or set EWLM options for your domain. These settings are described in more detail next.

Domain settings

You can see the domain settings information by clicking the **Domain Settings** field. Figure 5-18 is a sample of the Domain properties screen.

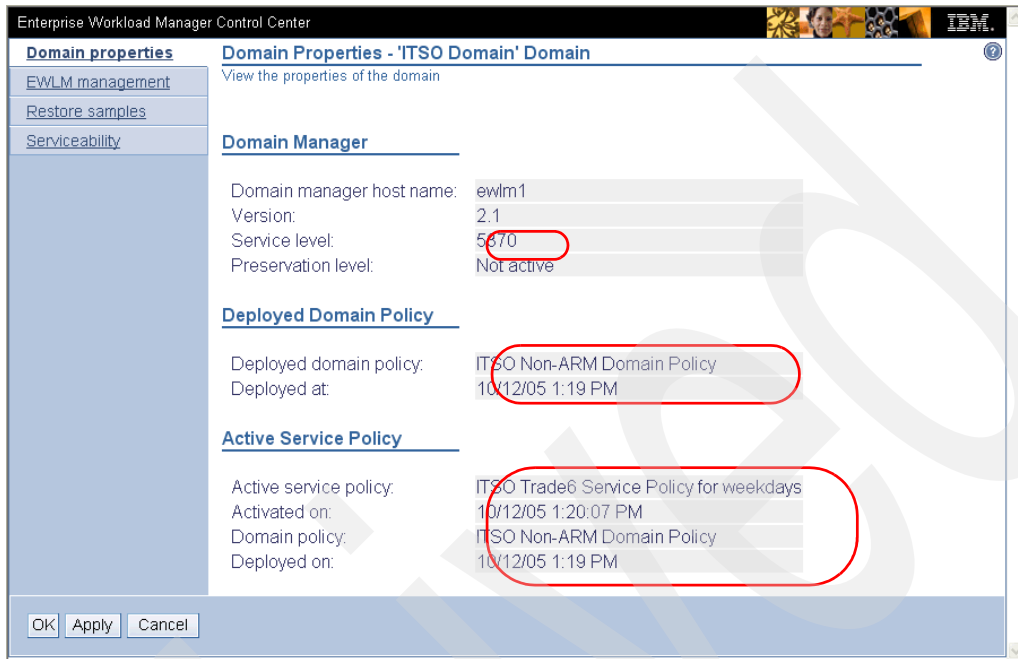


Figure 5-18 Domain properties screen

You can see the EWLM Domain Management configuration with the deployed and active policies and also the current service level of the EWLM code.

On the left pane of the screen you have listed the functions available with EWLM. Click the corresponding function to display the current setup for the function and to enter any change.

Figure 5-19 shows the content of the Restore Sample function. You can use this function if you need to return the sample policies to their original state, predefined applications to their original state, or predefined platforms to their original state.

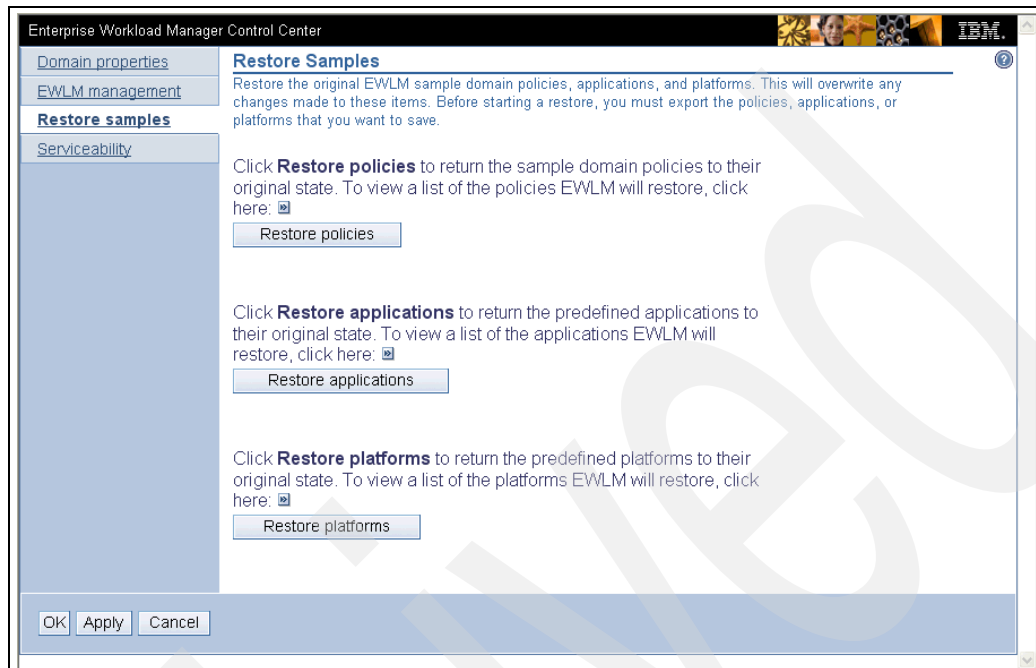


Figure 5-19 Restore Sample panel

Figure 5-20 shows the content of the Serviceability function. You can use this function if you need to set up tracing for problem analysis in the following areas:

- ▶ Log algorithm information
- ▶ Domain manager information
- ▶ Snapshot of domain manager information

You can find more information in Chapter 10, “EWLM traces and logs” on page 249.

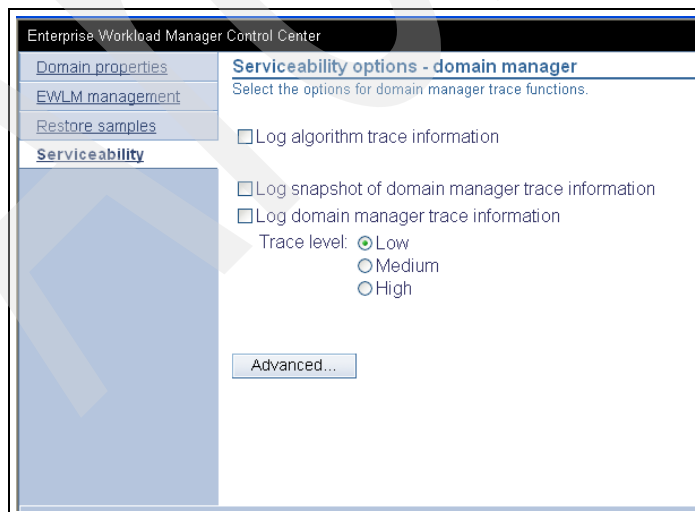


Figure 5-20 Serviceability panel

Figure 5-21 shows the content of the EWLM Management function. You can use this function if you need to enable load balancing and/or partition workload group management. For additional information about these functions, refer to Chapter 7, “EWLM Management” on page 191.

EWLM will activate the function only when the respective box is checked. During your testing periods, you can always return to the initial state by unchecking the function on this panel. For example, when you uncheck the partition workload group management, the hypervisor will return to use the values defined on the HMC.

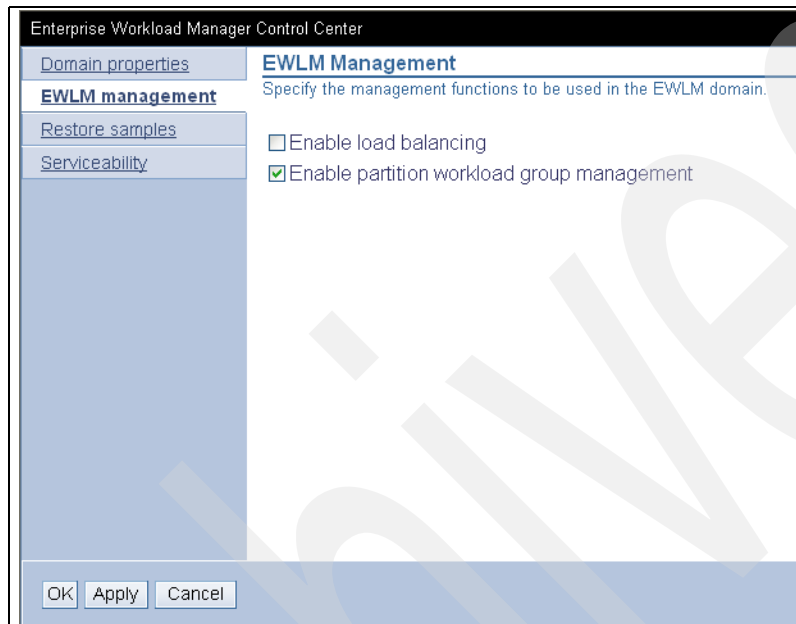


Figure 5-21 EWLM Management panel

Interval selection

You can select the reporting interval by clicking the option **Preferences** on the Welcome menu. Figure 5-22 shows the interval reporting selection.

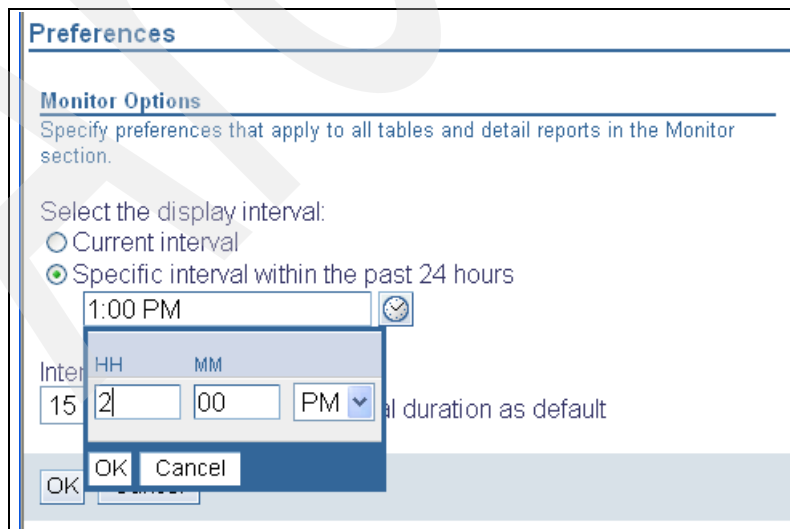


Figure 5-22 Interval setting

You can select the current interval and specify the interval options. The interval options are 1, 5, 10, 15, 30, 45, or 60 minutes. We recommend that you use:

- ▶ Shorter 1-minute or 5-minute intervals to observe only the recent performance of your workload
- ▶ 10-minute or 15-minute intervals for normal workload monitoring
- ▶ 30-minute, 45-minute, or 60-minute intervals for long-running work

When you select this option, the reporting information you will see on the EWLM panels will have the current time as the starting time and the data will apply to the interval prior to the current time.

You can select to see reporting data for an interval different than the current one. To do so, you need to click the **Specific interval** option and enter the starting time in the drop-down box. When you select this option, the reporting information you will see on the EWLM panels will have the specified time as the starting time and the data will apply to the interval following the initial time.

For example, you can see in Figure 5-23 the corresponding reporting interval starting from 2 p.m., as specified in the selection box for the duration of 15-minute intervals.

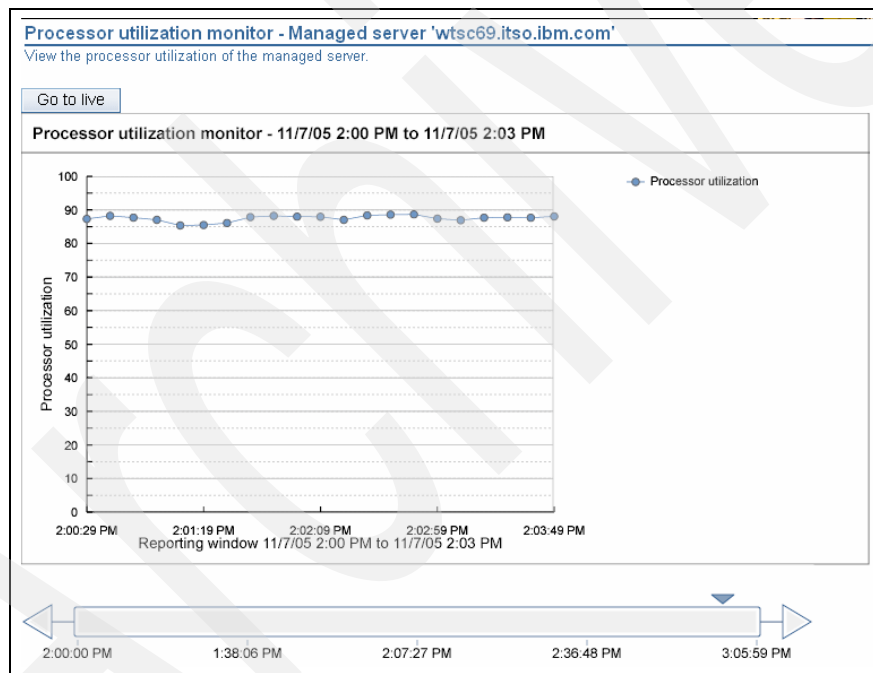


Figure 5-23 Reporting panel for a specific reporting interval

EWLM Control Center functions

On the left-hand panel you can see the three main functions. *Set up* is the function you use to define and modify your Domain Policy. This can only be performed by users with the role of Administrator. In addition, this user can perform all functions to manage and monitor the environment. *Manage* is the function you use to control your EWLM environment, and *Monitor* is the function you should use when you need to look at monitoring capabilities and to report performance in your end-to-end enterprise environment. Based on the assigned authority that a user has, the corresponding functionalities are shown after the login has completed. As you can see, some of the elements, for example, Managed Servers, are listed in both the Manage and Monitor functions. When you select an element, for example, Managed Server, from the Manage menu, you will see all the static information about the

element and you will be presented with management options such as disable the element. When you select the same element from the Monitor menu, you will not be able to change anything about the element, but you could retrieve performance data such as % CPU busy and so forth.

Currently the list of functions available under the Manage option is limited to:

- ▶ Service policies
- ▶ Managed servers

Service policies

For the service policies, in the main area you can see which service policy is currently active and you are provided with two options: View and Activate.

When you select to view the service policy, the summary panel shown in Figure 5-24 is displayed.

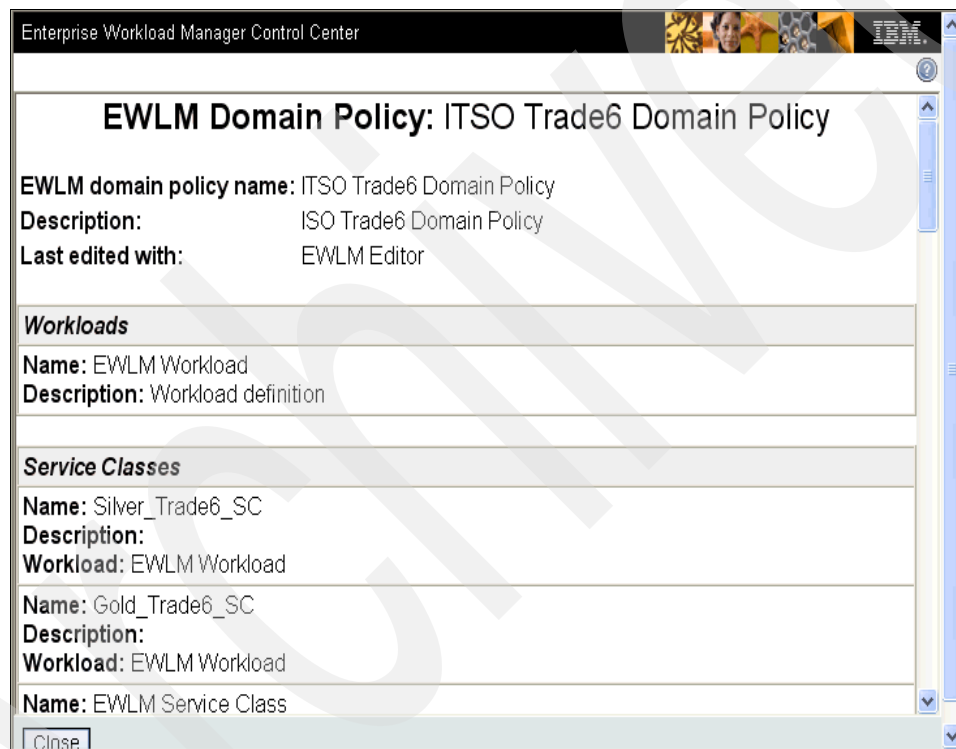


Figure 5-24 View Service policy

As you scroll through the panel all the definitions belonging to the service policy are shown: Workloads, Service Classes, Service policies, Application, Transaction Classes (with their filters), Platforms, Process Classes (with filters), and Partition classes.

When you select to activate, the service policy will be activated on all the managed servers. As a result of the activation task, the domain manager propagates the service policy to all the managed servers. Depending on how many managed servers are connected to the domain manager, the activation task might take some time.

Managed servers

When you select the Managed Servers options, a summary of all the connected managed servers is shown, as in Figure 5-25.

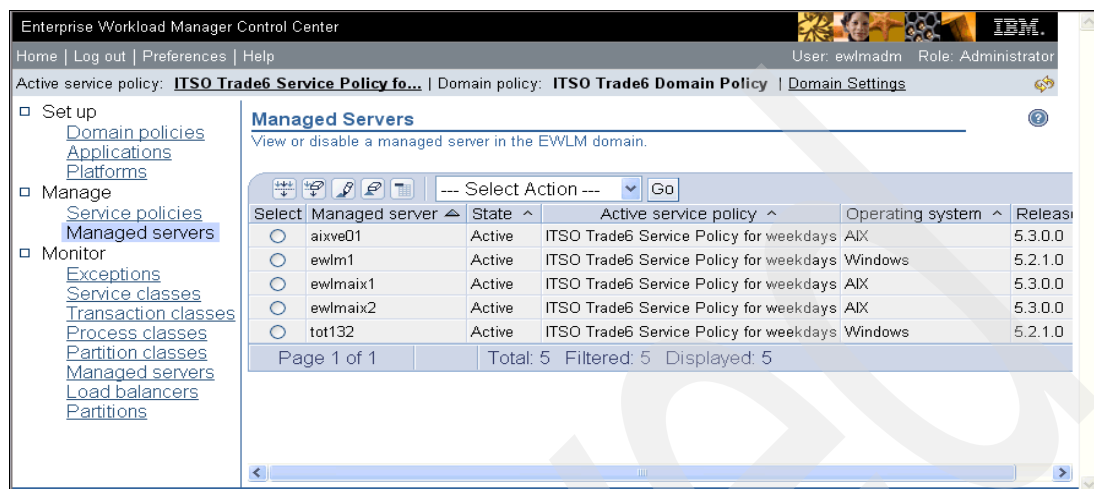


Figure 5-25 Managed servers view

From this summary, you can see the host name of each managed server, the status of the server, the active service policy name, the operating system running on this system, and the release of the OS.

One of the important details on this screen is the state of the managed server. The possible states of each server are as follows:

- ▶ **Joining:** The EWLM managed server is in the process of joining to the EWLM domain manager.
- ▶ **Join_pending:** The EWLM domain manager has accepted the join request from the EWLM managed server and is currently waiting for acknowledgement from the EWLM managed server.
- ▶ **Active:** The EWLM managed server has joined the EWLM Management Domain successfully and is functioning properly. The EWLM managed server might not have an active service policy on it yet. If this is the first time configuring the EWLM managed server, it is the default service policy.
- ▶ **Resynchronizing:** The EWLM managed server is currently resynchronizing with the EWLM domain manager to make sure that it is running with the current EWLM service policy.
- ▶ **Dormant:** The EWLM managed server has successfully joined the EWLM Management Domain, but since EWLM is disabled on the system, it is not providing any reporting data to the EWLM domain manager.
- ▶ **Activation_pending:** The EWLM domain manager has requested that the EWLM managed server activate an EWLM service policy and is currently waiting for acknowledgement.
- ▶ **Activating_policy:** The EWLM managed server is currently activating an EWLM service policy.
- ▶ **Leaving:** The EWLM managed server has requested to leave the EWLM Management Domain.
- ▶ **Left:** The EWLM managed server is not currently joined to the EWLM Management Domain.

- ▶ **Lost_contact:** Communications with the EWLM managed server are disrupted. Several attempts have been made to get the status of the EWLM managed server but no response has been received.
- ▶ **Confused:** The EWLM managed server is detected to be running with a different EWLM Domain Policy than the one currently active on the EWLM domain manager. All reporting data from the EWLM managed server will be ignored.
- ▶ **Disabling:** The EWLM managed server has requested to terminate itself.
- ▶ **Disabled:** The EWLM managed server is currently not running or has not attempted yet to join the EWLM Management Domain.

From the pull-down menu, two options are available:

- ▶ **Properties:** This option shows you details about the managed server such as the server name, the status, the Active Service Policy, the server architecture, whether LPAR is enabled, and the EWLM version and build number.
- ▶ **Disable:** Stops the EWLM managed server and removes it from the domain. This option is intended for removing a server that is not functioning properly. You need to restart it with the **startMS** command on the server itself; you cannot restart it from the Control Center.

Note: The EWLM Control Center does not support the use of the browser back button.

Archived



EWLM monitoring and reporting

This chapter describes how to use the following EWLM capabilities:

- ▶ First-level monitoring and reporting of exceptions, classes, managed servers, Load Balancers, and partitions
- ▶ Second-level details reporting of workloads and resource consumption

6.1 Overview

One of the main features of EWLM is its reporting capability. When a service policy has been activated, you can monitor how it is running in the EWLM Management Domain using:

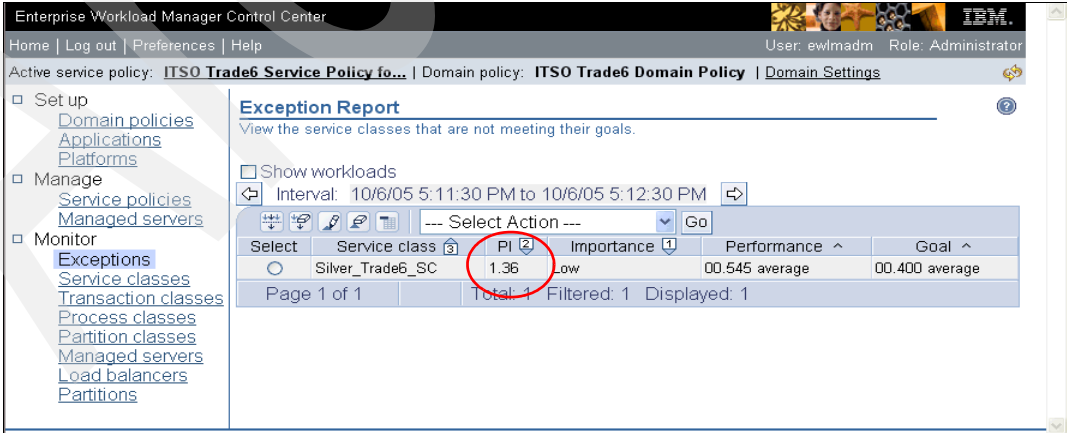
- ▶ First level:
 - Exceptions
 - Service classes
 - Transaction classes
 - Process classes (not supported by z/OS)
 - Partition classes
 - Managed servers
 - Load Balancers
 - Partitions
- ▶ Second level:
 - Details reports
 - Topology reports: Application and server
 - Real-time performance monitors
 - Goal achievement
 - Processor utilization
 - Transaction rate

6.2 First-level reports

The first-level reports have the objective of showing performance — to project the actual response time versus the desired goal, and to examine the CPU and the response time.

6.2.1 Exceptions report

You can start with the Exceptions report. It shows a list of service classes that are *not* meeting their goals. The report lists, by service class, the service class name, performance index, its importance, current performance, and the performance goal. This level of reporting shows, at a glance, the problem areas within the domain.



Enterprise Workload Manager Control Center

Home | Log out | Preferences | Help User: ewlmdm Role: Administrator

Active service policy: ITSO Trade6 Service Policy fo... | Domain policy: ITSO Trade6 Domain Policy | Domain Settings

□ Set up
 Domain policies
 Applications
 Platforms

□ Manage
 Service policies
 Managed servers

□ Monitor
 Exceptions
 Service classes
 Transaction classes
 Process classes
 Partition classes
 Managed servers
 Load balancers
 Partitions

Exception Report

View the service classes that are not meeting their goals.

Show workloads

Interval: 10/6/05 5:11:30 PM to 10/6/05 5:12:30 PM

Select Action --- Go

Select	Service class	PI	Importance	Performance	Goal
<input type="radio"/>	Silver_Trade6_SC	1.36	Low	00.545 average	00.400 average

Page 1 of 1 Total: 1 Filtered: 1 Displayed: 1

Figure 6-1 Exceptions report

Figure 6-1 shows a sample of the Exceptions report where you can see the service classes that are not achieving the goal. EWLM uses the performance index (PI) to indicate how well a service class is attaining the goal defined for it. The performance index is a ratio between the

response time goal and the actual response time. A PI greater than 1 means that the goal is being missed, while a PI less than 1 means that the goal is being met.

Table 6-1 Performance index

Situation	PI value
Service class is exceeding its defined goal.	Less than (<) 1
Service class is meeting its defined goal.	1
Service class is missing its defined goal.	Greater than (>) 1

On the screen, you can immediately see the PI, which can be the starting point for understanding what caused the problem.

6.2.2 Service Classes report

This report provides information about service classes both at the individual service class level, and, if the service class is also part of a workload, then the performance of the service class at the workload level can also be displayed. The service class report shown in Figure 6-2 displays the current performance index, the importance, the current performance, and the performance goal for all service classes. This level of reporting is useful to see, at a high level, how all service classes are performing.

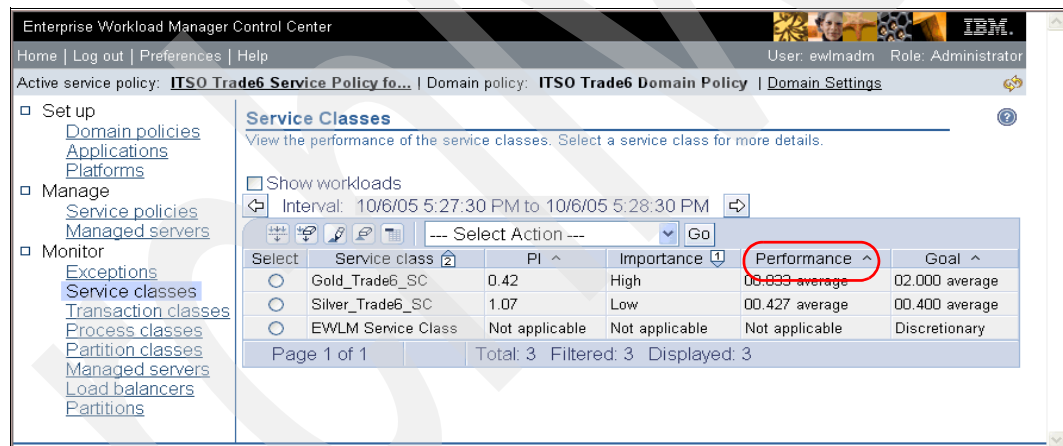


Figure 6-2 Service Classes report

In Figure 6-2 you can view performance statistics for each service class defined in our Domain Policy. You can see that our workload Trade6 uses only the first two service classes because we have the PI value along with the performance and achieved goal. The service class EWLM Service Class does not report a PI because of the discretionary goal.

6.2.3 Transaction Classes report

Just as the service class reporting is specific to service classes, the same exists for transaction classes. This report, shown in Figure 6-3, shows the average response time for the transactions. Individual transaction class information is available by selection. A response time distribution chart is available within the transaction class detail, as shown in 6.3.3, “Transaction class details” on page 166.

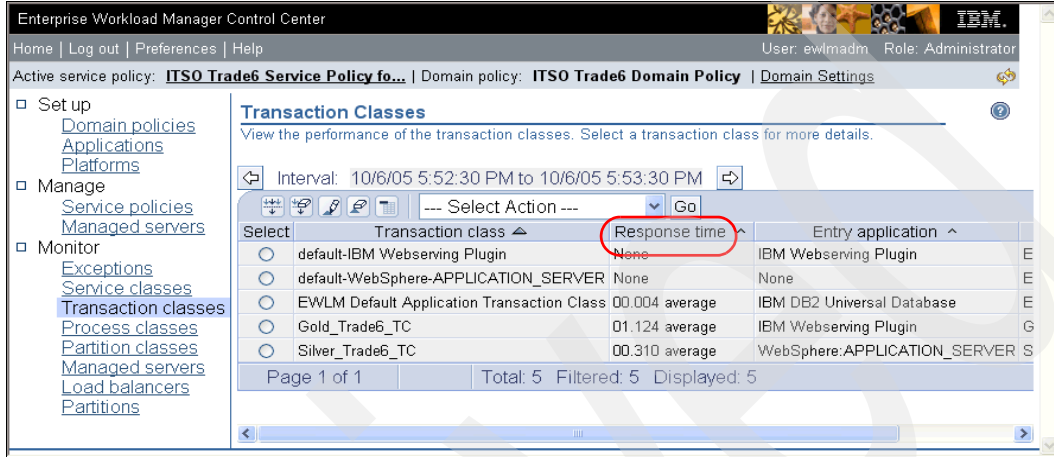


Figure 6-3 Transaction classes report

6.2.4 Process Classes report

Process class reporting provides attribute and statistical performance information about process classes. This includes processor usage and delay as well as storage and I/O delays.

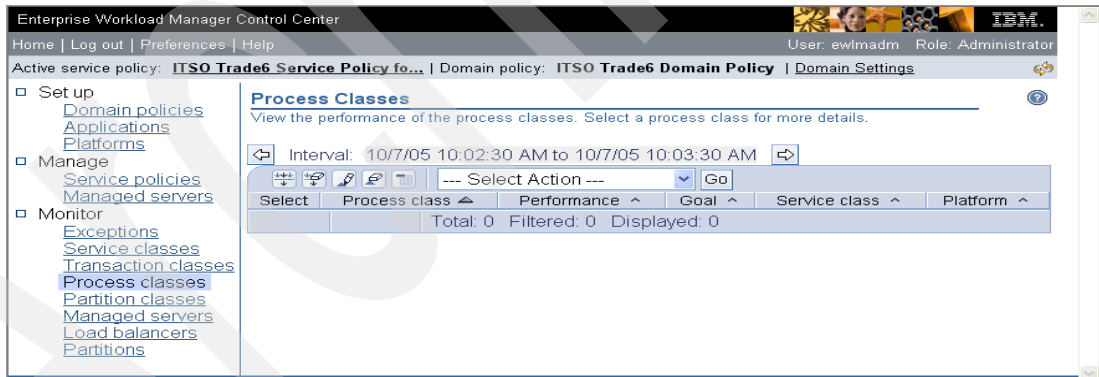


Figure 6-4 Process Classes report

6.2.5 Partition Classes report

The Partition Classes report shown in Figure 6-5 displays the current goal, PI, service class, and platform. A partition class must use a service class that specifies either a velocity or a discretionary goal.

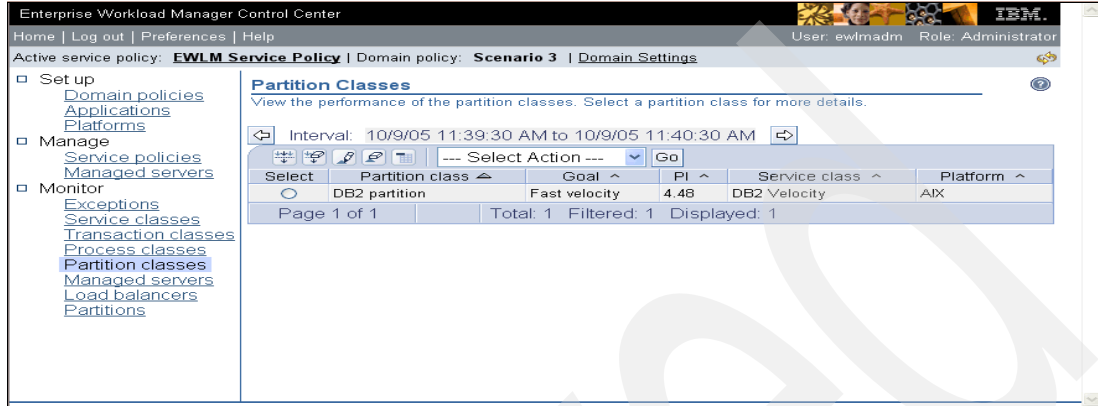


Figure 6-5 Partition classes report

6.2.6 Managed Servers report

The Managed Server report is shown in Figure 6-6.

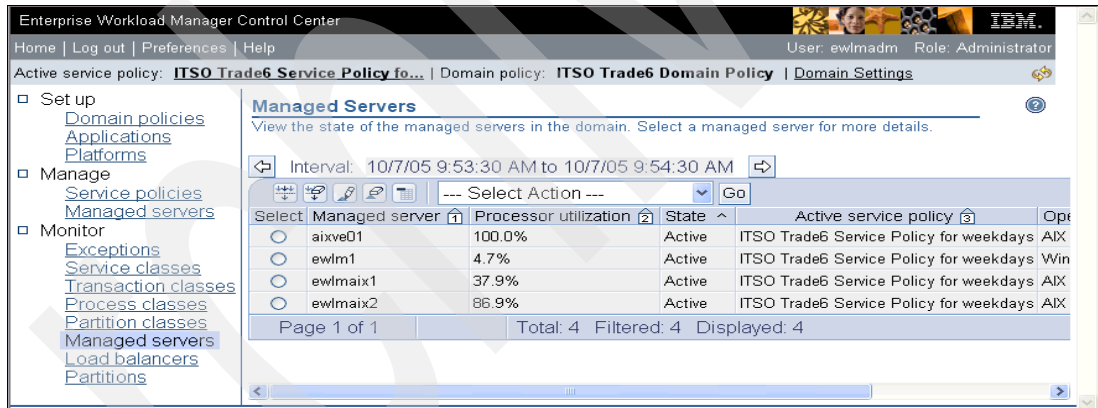


Figure 6-6 Managed Servers report

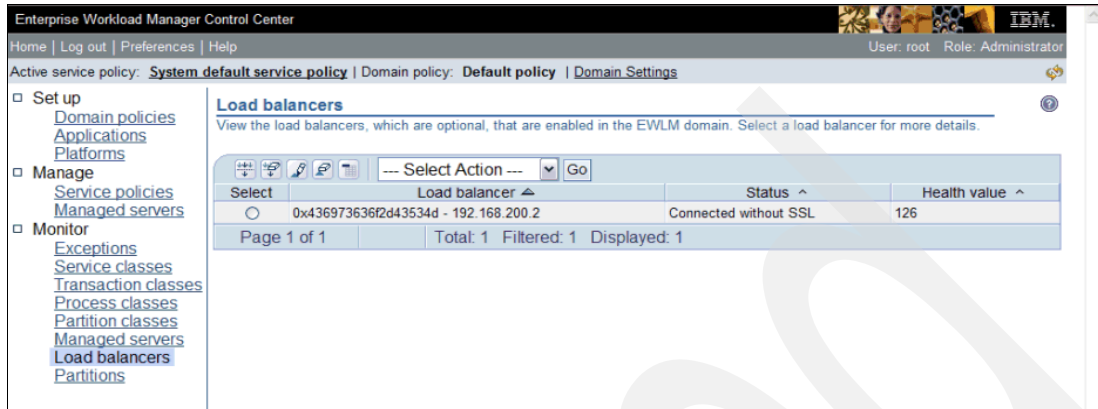
For each managed server, you can display the managed server detailed statistics (more information is in 6.3.5, “Managed server details” on page 168) and the Processor Utilization Monitor by selecting the desired task from the pull-down menu.

In Figure 6-6 you can view the status of the four servers of our configuration, the processor utilization percentage over the specified interval that the operating system is running, and the release level.

One of the important pieces of information from this screen is the state of the managed servers. The possible values and their meanings are described in 6.3.5, “Managed server details” on page 168.

6.2.7 Load Balancers report

The Load Balancers report is shown in Figure 6-7.



The screenshot shows the Enterprise Workload Manager Control Center interface. The active service policy is 'System default service policy' and the domain policy is 'Default policy'. The left navigation pane is expanded to 'Load balancers'. The main content area displays a table of load balancers.

Select	Load balancer	Status	Health value
<input type="radio"/>	0x436973636f2d43534d - 192.168.200.2	Connected without SSL	126

Page 1 of 1 | Total: 1 | Filtered: 1 | Displayed: 1

Figure 6-7 Load Balancers report

The report displays the name of the Load Balancer followed by its IP address. The status column indicates the current state of the connection (SSL or non-SSL) and is followed by the Load Balancer provided health value. If the Load Balancer does not provide a health value, then this field is blank.

6.2.8 Partitions report

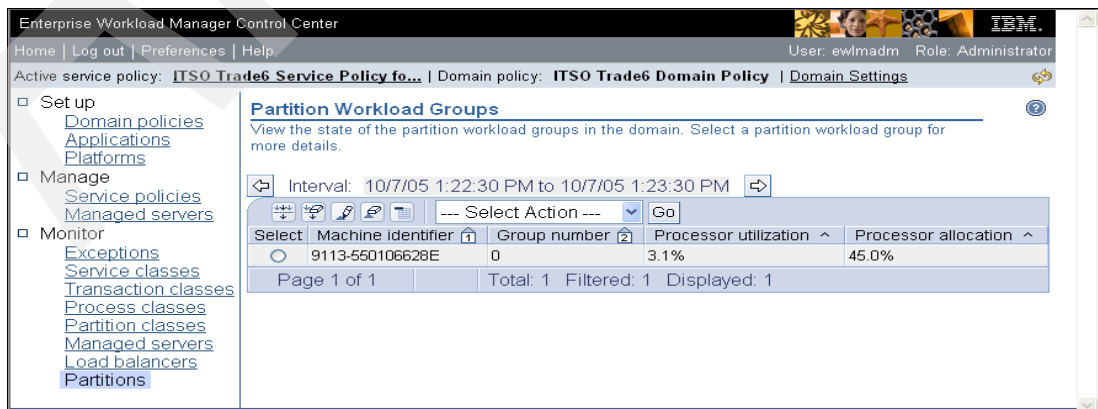
The Partitions report shown in Figure 6-8 displays the machine identifier, group number, processor utilization, and processor allocation.

The machine identifier represents the IBM System p5 or eServer p5 system containing this partition workload group.

The group number is the unique name given to the partition workload group defined to the system denoted by the machine identifier.

Processor utilization represents the percentage of the total processing units available, within the shared pool, that this partition workload group is currently utilizing.

Processor allocation represents the percentage of the total processing units available, within the shared pool, that are allocated to this partition workload group as entitled processing units.



The screenshot shows the Enterprise Workload Manager Control Center interface. The active service policy is 'ITSO Trade6 Service Policy fo...' and the domain policy is 'ITSO Trade6 Domain Policy'. The left navigation pane is expanded to 'Partitions'. The main content area displays a table of partition workload groups.

Interval: 10/7/05 1:22:30 PM to 10/7/05 1:23:30 PM

Select	Machine identifier	Group number	Processor utilization	Processor allocation
<input type="radio"/>	9113-550106628E	0	3.1%	45.0%

Page 1 of 1 | Total: 1 | Filtered: 1 | Displayed: 1

Figure 6-8 Partitions report

6.3 Second-level reports

EWLM provides detailed reports about the functioning of various elements. This section includes a sample of each of those reports so you can become familiar with the type of information available.

6.3.1 Process class details

You obtain the Process class details report (Figure 6-9) by selecting the process class details on the pull-down menu. The following information is returned:

- ▶ Service class associated with this process class
- ▶ Goal type and the related goal
- ▶ Platform
- ▶ Managed server where it is active
- ▶ Statistics calculated in the displayed interval for CPU usage, paging delay, and I/O delay

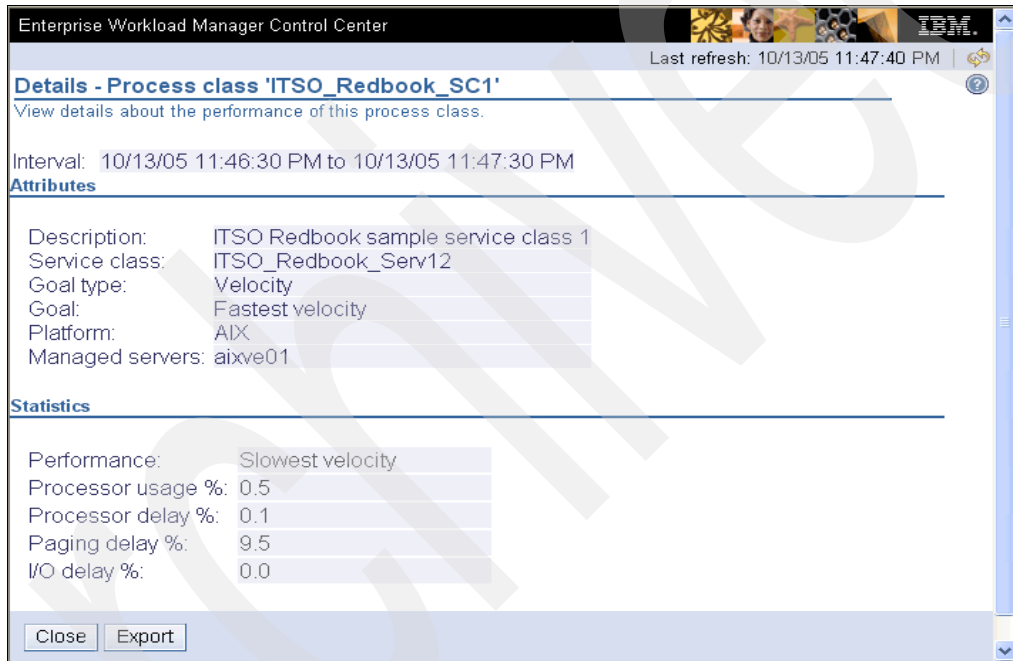


Figure 6-9 Process class detail report

6.3.2 Service class details

You obtain the Service class details report (Figure on page 166) by selecting a service class from the list of service classes and then choosing **Details** from the Select Action pull-down menu.

The view has three sections: Attributes, Statistics, and a Response Time Distribution Chart.

- ▶ Attribute: This section includes the following items:
 - A description of the service class
 - The workload to which the service class belongs
 - The goal type, the goal, and the importance
 - The transaction classes that are associated with this service class
- ▶ Statistics: This section specifies performance data that is specific to this service class.

- Performance: A value that specifies the actual performance and response time of the transaction processed by the service class in the interval. A transaction starts and terminates in the outer hop 0 at the middleware level, and the response time covers the end-to-end elapsed time.
 - Performance index: A value that indicates how well the service class is meeting the goal. If the PI is less than or equal to 1, the goal is being met; if it is greater than 1, the goal is not met.
 - Transactions: Completed successfully: The number of transactions associated with this service class that successfully completed processing at hop 0 level.
 - The number of transactions that failed, stopped, ended in unknown state, and the total. Failed or stopped transactions can be caused by middleware or operating system problems or user errors. The hop where the error may have occurred is not indicated.
- Response Time Distribution Chart: For service classes specifying average or percentile response time, a graph is presented for the selected interval length.

On the bottom of the screen (not shown here) there is also a graphical view of the response time distribution. It is only shown when the goals are average or percentile response time goals.

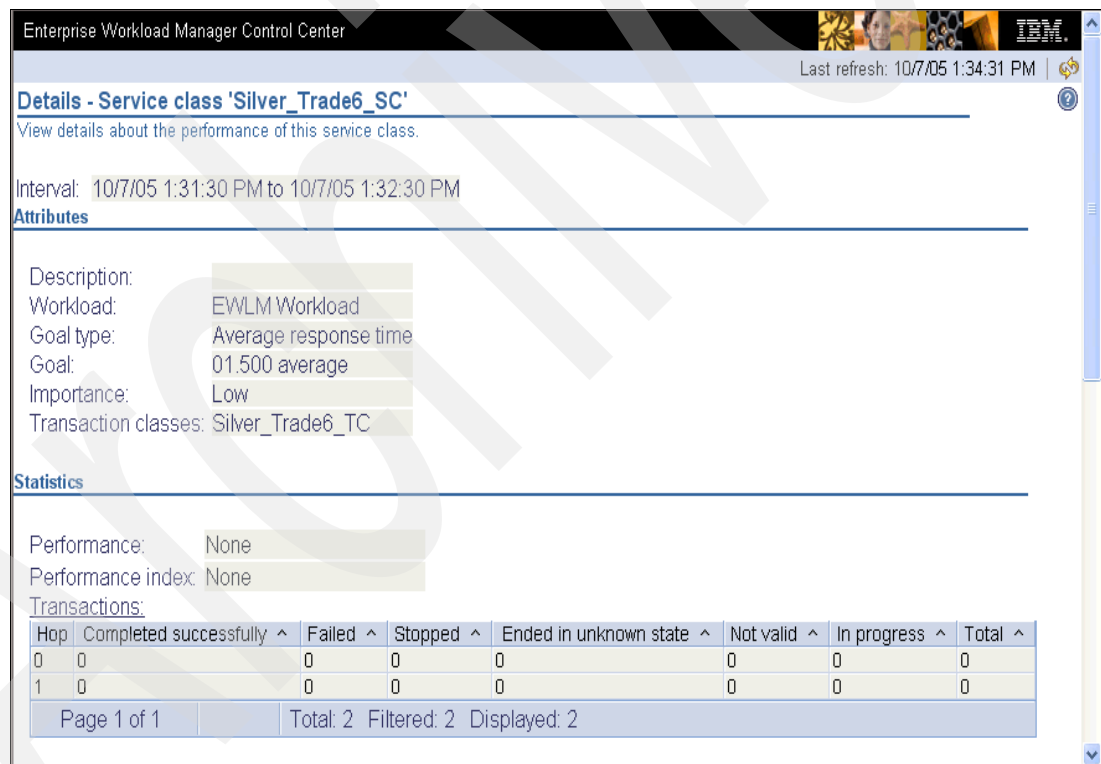


Figure 6-10 Service class details report

6.3.3 Transaction class details

You obtain the Transaction class details report (Figure 6-11 on page 167) by selecting a transaction class from the list of transaction classes and then choosing **Details** from the Select Action pull-down menu. The following information is returned:

- Service class associated with the transaction class
- Service class goal type and the goal

- ▶ Service class importance level
- ▶ Statistics calculated in the displayed interval for:
 - Performance
 - Standard deviation
 - Transactions (total, successfully completed, failed, stopped, ended unknown)

On the bottom of the screen (not shown here) there is also a graphical view of the response time distribution. It is only shown when the associated service class goals are average or percentile response time goals.

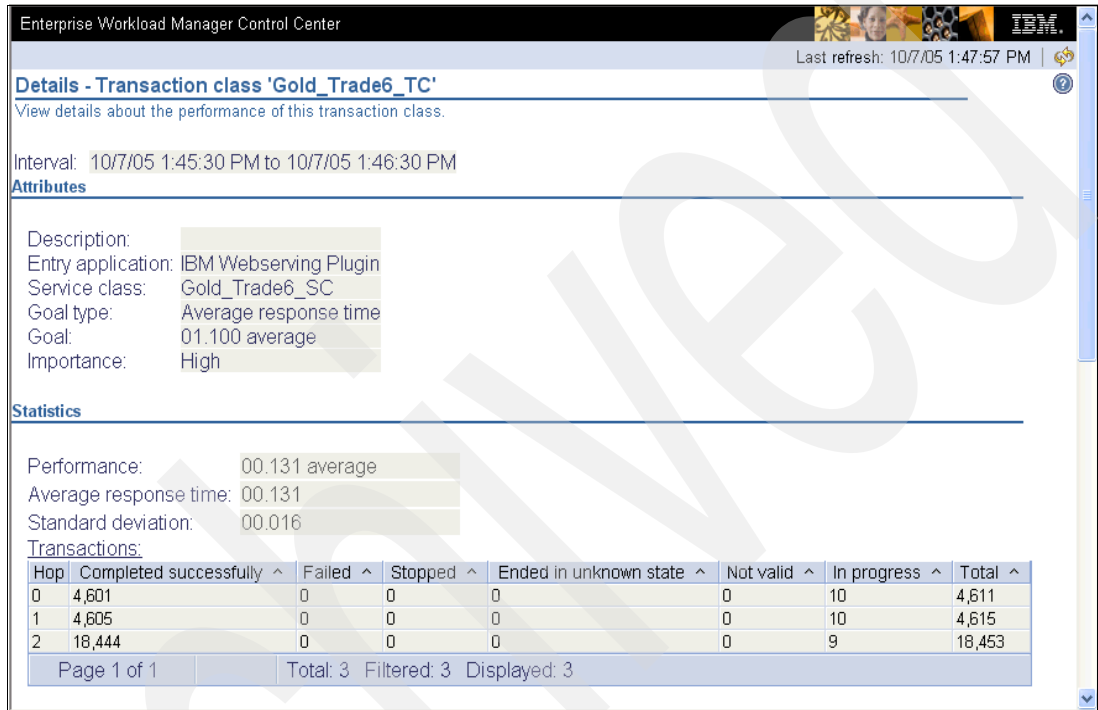


Figure 6-11 Transaction class details report

6.3.4 Partition class details

You obtain the Partition class details report (Figure 6-12) by selecting a partition class from the list of partition classes and then choosing **Details** from the Select Action pull-down menu. The following information is returned:

- ▶ Service class associated with the partition class
- ▶ Service class goal type and the goal
- ▶ Platform
- ▶ Managed servers
- ▶ Statistic calculated in the displayed interval for PI

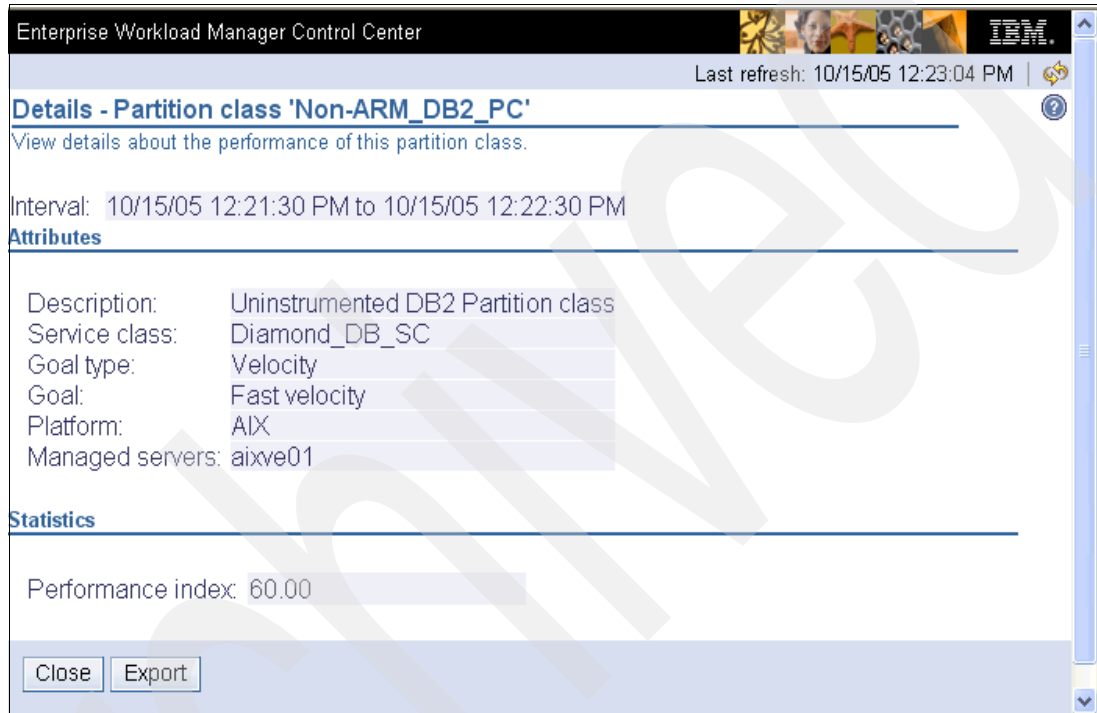


Figure 6-12 Partition class details report

6.3.5 Managed server details

You obtain the Managed server details report (Figure 6-13 on page 170) by first selecting a managed server and then choosing **Details** from the Select Action pull-down menu. The following information is returned:

- ▶ **Statistics:** This section specifies performance data that is specific to this managed server.
- ▶ **Managed server name:** Unique name by which a server is known on a network. In our case the name consists of the host name aixve01.
- ▶ **Average processor utilization %:** Average processor utilization (98.6 percent) of the managed server that includes all work requests that the server is processing, not just the EWLM monitored work.

Important: For a z/OS managed server, the reported average processor utilization percentage includes only the captured time that corresponds to the APPL% of workload activity RMF report for policy.

- ▶ **Real memory:** Total amount of memory of the LPAR hosting the server.

- ▶ Number of logical processors: Number of logical CPs of the LPAR.
- ▶ Page fault rate: Number of page faults per second managed by the operating system. A high number means that the memory is not sufficient to sustain *all* of the workloads the server is running, not just the EWLM monitored work.
- ▶ Application (Group): Application instances that support EWLM monitored work. In our domain the application environment name is IBM DB2 Universal Database, instance db2inst1.
- ▶ Service classes processed in server: This section reports all service classes associated with the managed server.
- ▶ Service class name: Name of the EWLM service class associated to a transaction by the *Entry application*.

Important: For a z/OS managed server, the using and delay percentages are derived from z/OS WLM sampling status data for the z/OS WLM service classes managing the EWLM monitored workload.

Status samples categories are: (processor using), (processor, I/O, storage, others delays), and idle. The percentages are derived from using and delay status samples with the exclusion of the idle status samples.

Because there may not be a one-to-one relationship between an EWLM service or transaction class and a z/OS WLM service class, the using and delay sample data provided by z/OS WLM are apportioned to EWLM service or transaction classes with a weighted average based on EWLM service classes transaction rate and active time.

- ▶ Processor usage %: This value indicates the percentage of the number of samples when the EWLM monitored workload was found using the processor.

Important: This information should not be interpreted as the percentage of the average server processor utilization used by the service class.

For a z/OS managed server, the CPU utilization reported by RMF Monitor I for a time frame that corresponds to the EWLM Control Center Processor utilization interval will show a little difference because the EWLM Control Center does not report the uncaptured time.

- ▶ Processor delay %: This value indicates the percentage of the number of samples when the EWLM monitored workload was found delayed because the processor was busy with other workloads.
- ▶ Paging delay %: This value indicates the percentage of the number of samples when the EWLM monitored workload was found delayed because of page faults.
- ▶ I/O delay %: This value indicates the percentage of the number of samples when the EWLM monitored workload was found delayed waiting for an I/O operation to complete.

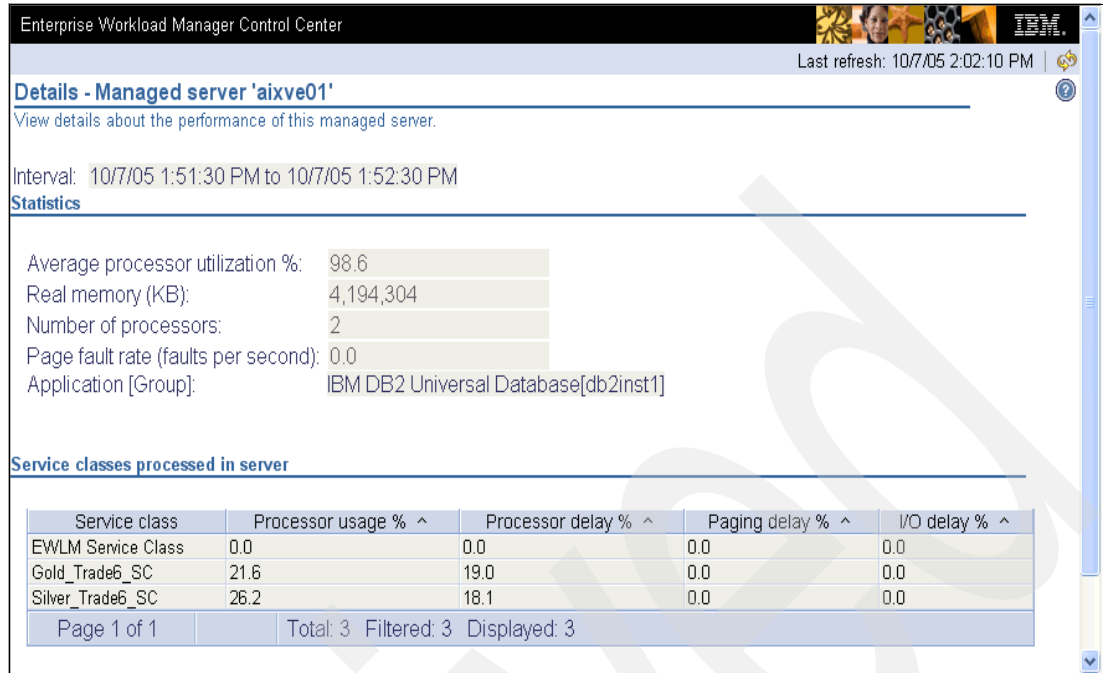


Figure 6-13 Managed server details report

6.3.6 Load Balancers details

You obtain the Load Balancer details report (Figure 6-14 on page 171) by selecting the Load Balancer detail statistics on the pull-down menu. The Load Balancer defined groups are displayed and a details report for the members of the group selected is available. The following detail information for a Load Balancer group is returned:

- ▶ The host name of the server associated with the IP address defined for this group member
- ▶ The IP address, port, and protocol that define the application represented by this group member
- ▶ The ARM-enabled application name that is listening for connections on the IP address, port, and protocol
- ▶ EWLM relative Weight recommendation - used by the Load Balancer in determining the distribution of workload to the members of this Load Balancer group
- ▶ EWLM reporting statistics that were used to compute the weight:
 - Average response time
 - Transaction count
 - Failed transactions
 - Stopped transactions
- ▶ Status of the load-balanced application

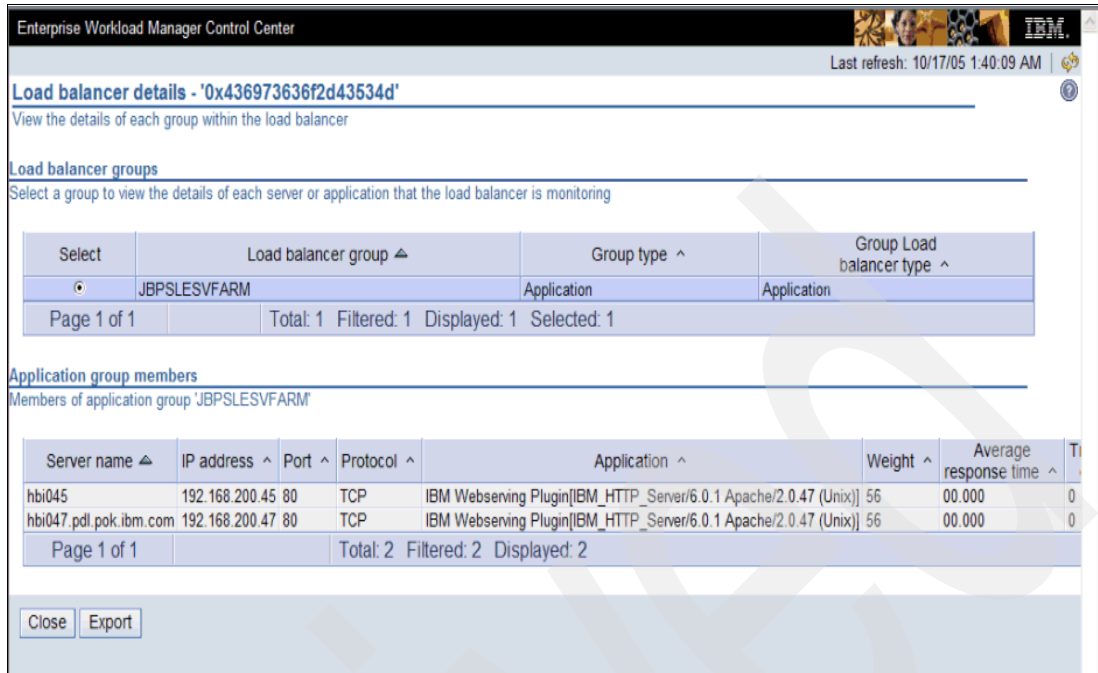


Figure 6-14 Load Balancers details report

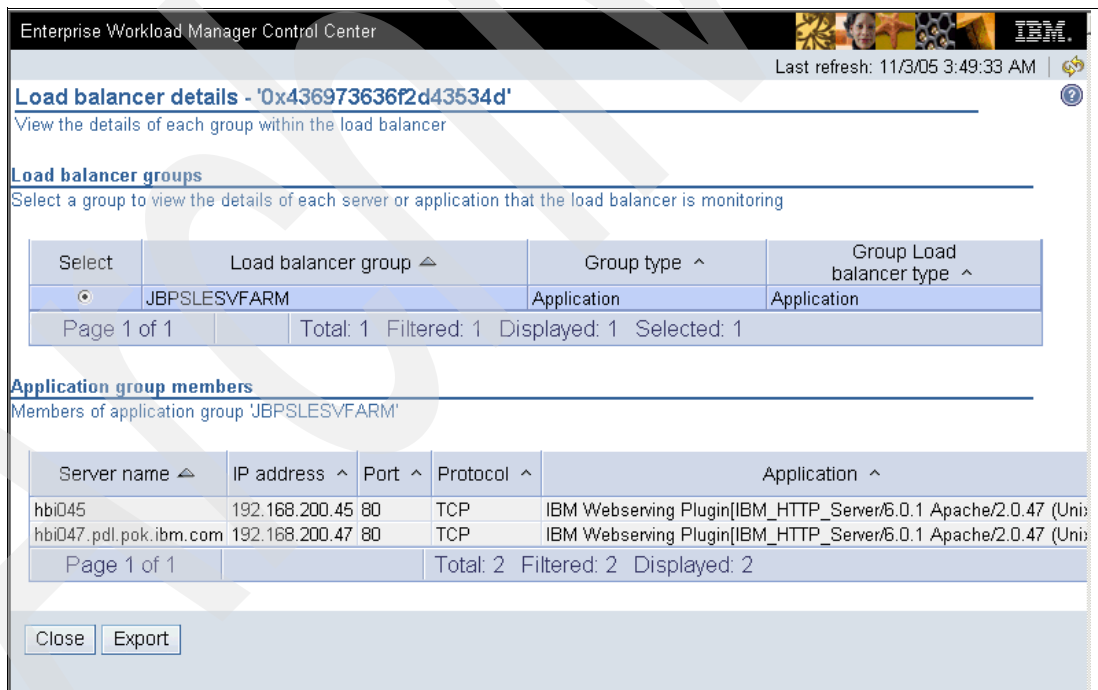


Figure 6-15 Load Balancers details report

Weight ^	Average response time ^	Transaction count ^	Failed transactions ^	Stopped transactions ^	Status ^
58	00.000	0	0	0	Active
54	00.000	0	0	0	Active

Figure 6-16 Load Balancers details report cont.

6.3.7 Partitions details

You obtain the Partitions details report (Figure 6-17) by selecting the Partitions detail statistics on the pull-down menu. The following information is returned:

- ▶ Partition name
- ▶ Weight
- ▶ List of the processor information about this partition with the following details:
 - Virtual processors
 - Physical processors
 - Processor utilization
 - Processing unit
 - Capped
 - Minimum processing units
 - Maximum processing units
- ▶ List of the memory information about this partition with the following details:
 - Allocated memory
 - Minimum memory
 - Maximum memory
- ▶ Managed server

Enterprise Workload Manager Control Center

Last refresh: 10/15/05 12:18:41 PM

Partition Workload Group Details - 9113-550106628E-0

View the details of each partition in the partition workload group.

Interval: 10/15/05 12:17:30 PM to 10/15/05 12:18:30 PM

Statistics

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^	Allocated memory ^
EWLM_AIX_DB2	128	3	3	0.69%	0.90	No	0.10	3.00	4,096 MB
EWLM_AIX_WAS1	128	3	3	0.71%	0.90	No	0.10	3.00	4,096 MB
EWLM_AIX_WAS2	128	3	3	0.52%	0.90	No	0.10	3.00	4,096 MB

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 6-17 Partitions details report

6.3.8 Application and server topology reports

The application and server topology reports are built by EWLM dynamically following the flow of your transaction or your process. The *application topology* is the logical representation of the application flow while the *server topology* is the physical representation of which server you application has been going through during its processing. You can obtain these reports by selecting Application topology or Server topology from the pull-down menu and clicking **Go**.

Following are samples of these reports.

Application topology

Figure 6-18 is an example of the application topology graphic. EWLM, as it detects instrumented data from the various applications, makes this graphic possible. The graphic on the far left represents the *hop0* or edge server where the transactions originate. This is where the end-to-end response time is measured and represents the round-trip for the transaction. Hop0 is where the EWLM classification takes place, and an ARM correlator is assigned to the transaction and flows with the transaction through the subsequent hops. So, what EWLM is mainly doing in the application topology is building a logical view of your application showing all the middleware that your application's flow is going through from start to end. In order to have a complete topology view of your application, all the middleware elements must be ARM instrumented. If you click the logical view of the application's icon, you can zoom in on the physical server's icon and, furthermore, if you click the server's icon, you can see the application instances.

From the application topology, you can obtain an additional report called the Tableview by clicking the matrix icon, as shown in Figure 6-18.

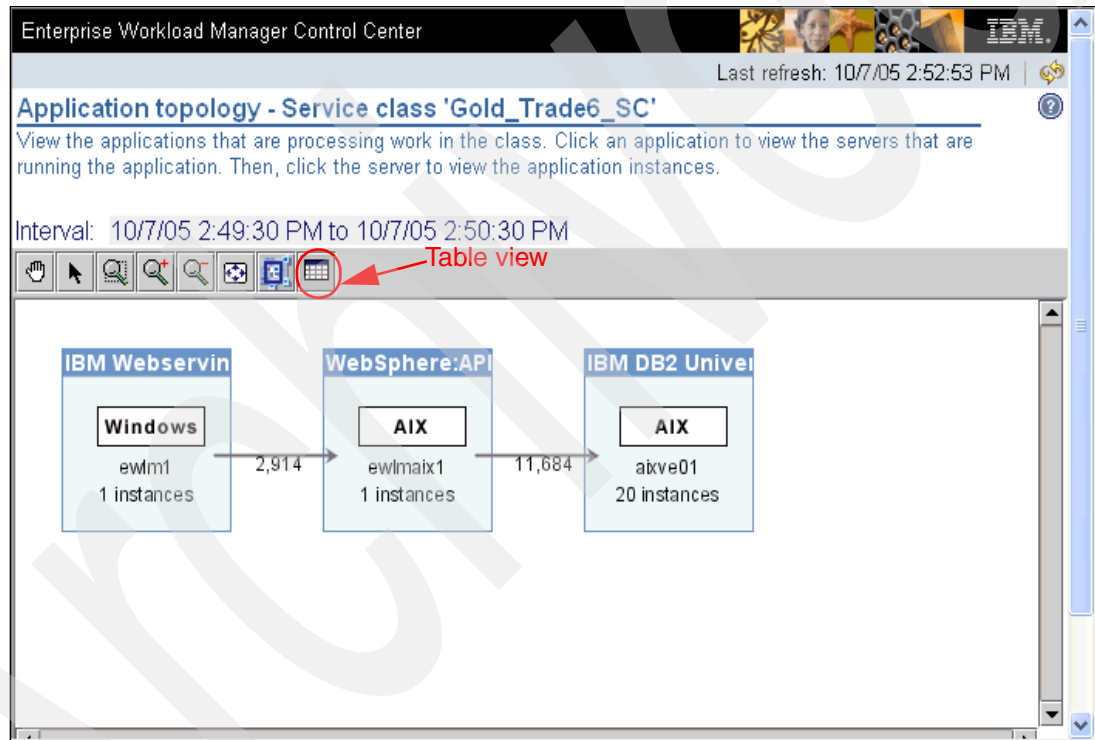


Figure 6-18 Application topology report

The example in Figure 6-20 on page 176 is a partial view of the Topology tableview. The first column lists the hops (0, 1, and 2). These represent the individual segments where the transactions are processed. Hop0 identifies the edge server that is the starting and endpoint for the transactions, where the EWLM classification takes place. For each Hop, information is provided at application, server, and each instance level. This makes the flow of the transaction and where the time is being spent very apparent.

The data in the tableview is retained for a maximum of 24 hours.

Following is the meanings for some of the most significant fields (let us use the DB2 tier as an example):

- ▶ Hop: Indicates the application or server where a transaction is processed. For example, in our case, the DB2 is the third tier identified as Hop2.
- ▶ Name: Name of the server, application, and instance to which the transaction moved.
- ▶ Type of Node: Describes whether the Name field is server (instrumented operating system), application environment (instrumented middleware), or instance (DB2 instance). In our case, the application environment IBM DB2 Universal Database has multiple instances representing all the DB2 threads.
- ▶ Platform: Specifies the operating system on which the server is running (AIX).
- ▶ Average response time: The field indicates the average amount of time in seconds that the transactions classified to the service class spent at this middleware hop plus any time spent in downstream hops. The average response time is the elapsed time from the arm_start_transaction and the arm_stop_transaction API calls. In our sample, the average response time is the same as the active time because this is the last hop. It includes the queue time within DB2 (no queue time in our case).
- ▶ Average active time: The field indicates in seconds how long the server took to process the request. Active time does not include the time the application indicates that the request is blocked because it is waiting for a subtransaction sent to another server (hop) to complete.

In our example the active and response times are the same because DB2 middleware is the last hop. See Figure 6-19 for response and active times concepts. Our example shows a one-to-one relationship among hops, though this is not always the case.

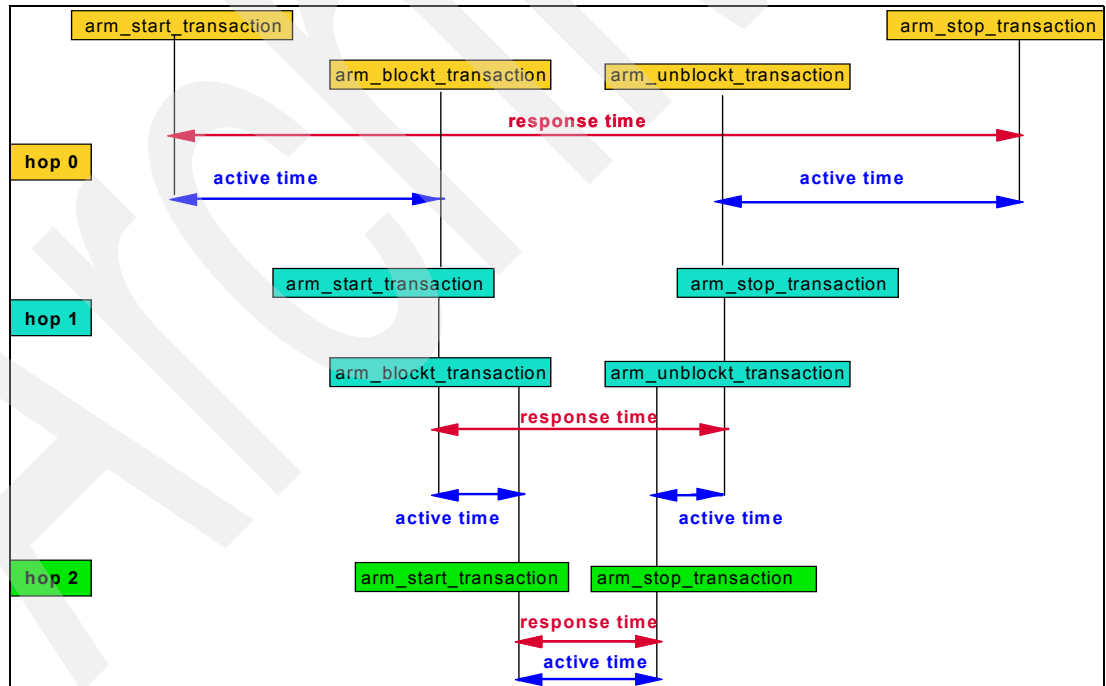


Figure 6-19 Transaction response and active time flow

Important: For a z/OS managed server, the processor time, using, and delay percentages are derived from z/OS WLM data for the z/OS WLM service classes managing the EWLM monitored workload.

Status samples categories are: (processor using), (processor, I/O, storage, others delays), and idle. The percentages are derived from using and delay status samples with the exclusion of the idle status samples.

Because there may not be a one-to-one relation between an EWLM service or transaction class and a z/OS WLM service class, the processor time, using, and delay sample data provided by z/OS WLM are apportioned to EWLM service or transaction classes with a weighted average based on EWLM service classes transaction rate and active time.

Processor time (seconds): This field shows the actual CPU time used over the interval.

- ▶ Processor delay%: This field shows the percentage of time the application was not able to obtain processor resources when requested over the interval.
- ▶ Processor using%: This field shows the percentage of time the application was able to obtain processor resources when requested over the interval.
- ▶ Successful transactions: This field specifies the number of transactions associated with the service class that successfully terminated its processing in hop 2, the database server on AIX. A transaction is delimited by issuing `arm_start_transaction`, `arm_stop_transaction` at the middleware level, in our case DB2.

Note: This transaction count is for hop 2 middleware; it should not expect a one-to-one relationship with the hop 0 transaction count in the first row of the tableview or in the transaction count of the Service Class detail view.

- ▶ I/O delay%. This field shows the percentage of time the application was waiting for an I/O to complete.
- ▶ Page delay%: This field shows the percentage of time the application was delayed because of a page fault: Central storage not available.
- ▶ Idle%: This field shows the percentage of time the application threads were in idle state. This status is not used to evaluate the processor using and delay percentages.
- ▶ Other%: This field shows the percentage of time the application was found in a non-idle state that could not be accounted to page, processor, or I/O delay.
- ▶ Average queue time (seconds): This is the time spent in the middleware preceding the actual processing of the transaction `arm_start_transaction`.
- ▶ Standard deviation (seconds): This field specifies a numerical value that indicates how much the transactions deviate from average response time.

Hop number	Name	Type of node	Platform	Average response time	Standard deviation	Average active time	Average queue time
0	IBM Webserving Plugin [IBM_HTT...	Application		00.417690	00.039000	00.007634	00.0072
0	ewlm1	Server	Windows	00.417690	00.039000	00.007634	00.0072
0	EWLM1/PID=0000000380	Instance		00.417690	00.039000	00.007634	00.0072
1	WebSphere:APPLICATION_SERV...	Application		00.407166	00.037000	00.002729	00.0000
1	ewlmaix1	Server	AIX	00.407166	00.037000	00.002729	00.0000
1	ewlmaix1Node01.TradeSe...	Instance		00.407166	00.037000	00.002729	00.0000
2	IBM DB2 Universal Database [db...	Application		00.100076	00.173000	00.100076	00.0000
2	aixve01	Server	AIX	00.100076	00.173000	00.100076	00.0000
2	287	Instance		00.100870	00.174000	00.100870	00.0000
2	149	Instance		00.100111	00.173000	00.100111	00.0000
2	337	Instance		00.100204	00.173000	00.100204	00.0000
2	504	Instance		00.098990	00.171000	00.098990	00.0000
2	634	Instance		00.100993	00.174000	00.100993	00.0000
2	684	Instance		00.099806	00.173000	00.099806	00.0000
2	859	Instance		00.100076	00.173000	00.100076	00.0000
2	1050	Instance		00.100280	00.173000	00.100280	00.0000
2	1203	Instance		00.100022	00.173000	00.100022	00.0000

Figure 6-20 Tableview report

Server topology

Figure 6-21 is an example of a server topology report. If you click a server icon, the ARM instrumented middleware running on that server is displayed. The server to the left is considered the hop0 or edge server and is the server where the transaction is classified to EWLM. The topology displayed here should be what you would expect. If not, then there is most likely work that should be classified that was not known previously.



Figure 6-21 Server topology report

6.3.9 Real-time performance monitors

These reports are available for service classes, transaction classes, process classes, and managed servers. The objective for these reports is to monitor the achievement of the goal, showing performance in real time. To use this graphical reporting, the minimum Java Runtime Environment is IBM Java Runtime Environment (JRE) 1.4.2 with Service Release 2 or later. All the following graphical reporting is based on the last 10-second interval:

- ▶ Application topology
- ▶ Server topology
- ▶ Hop details
- ▶ Performance index monitor
- ▶ Goal achievement monitor
- ▶ Transaction rate monitor

Each of the graphs has a variety of views in addition to those shown; the views are configured via the sliders on each. For further information about using these monitors, refer to the help in the Control Center for each screen shown.

Let us look at a sample of these graphical monitors. The performance index monitor is shown in Figure 6-22, where the goal of $PI=1$ is indicated by the horizontal yellow line. The current PI is calculated and projected on the chart against the target line: A $PI > 1$ is when the goal is being missed and a $PI < 1$ is when the goal is being met. Everything below the line is an indication that the goal is being exceeded (as in this example). Anything above the line (>1.00) indicates the service goal is being missed. A $PI > 1$ through several monitoring intervals may indicate a problem.

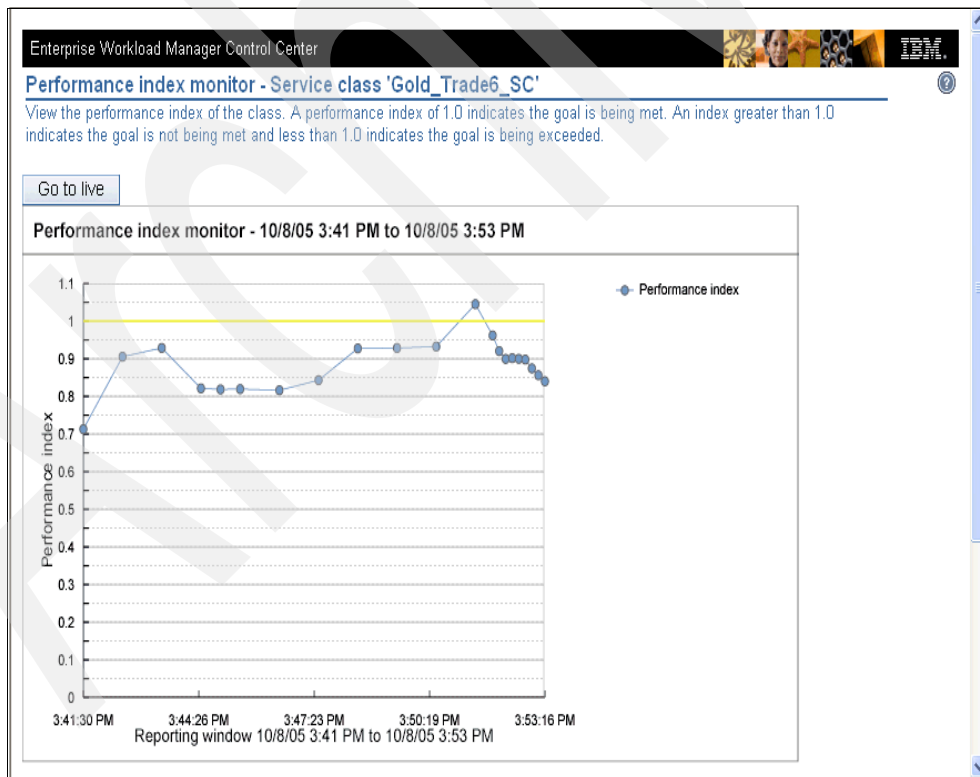


Figure 6-22 PI monitor

In 8.1, “Running Trade6 in the ITSO environment” on page 212, we present a sample scenario in our ITSO environment where we use the EWLM monitoring capability to help determine whether we have a performance problem related to a service class. You can see

how easy and quick it is to identify in the servers farm which applications or servers are experiencing the problem.

Figure 6-23 shows the average response time report. The goal is represented by the flat line in the center and the jagged line is the current trend of the service class. Thus, it is easy to see if the installation is achieving the goal in the displayed interval of time.

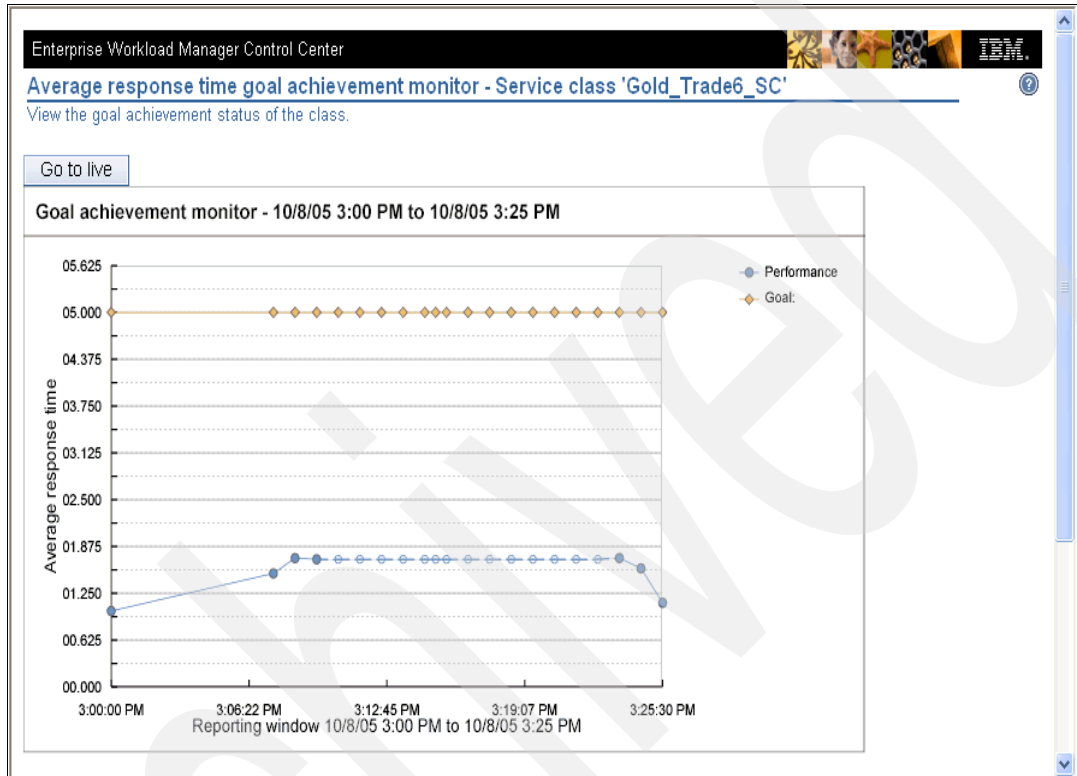


Figure 6-23 Average response time report

The last report (Figure 6-24) shows a sample of a Transaction rate report where you can see how many transactions per second are processed during a specific time interval. You can compare the transaction rate monitor to the PI monitor to determine whether there is a direct relationship between specific time intervals. For example, you can determine if the performance index increases when the number of transactions increases. Also, if a work request begins during one time interval and continues processing over multiple time intervals, the work request is included in each subsequent interval, where you can see the absolute number of successfully processed transactions in the interval of time.

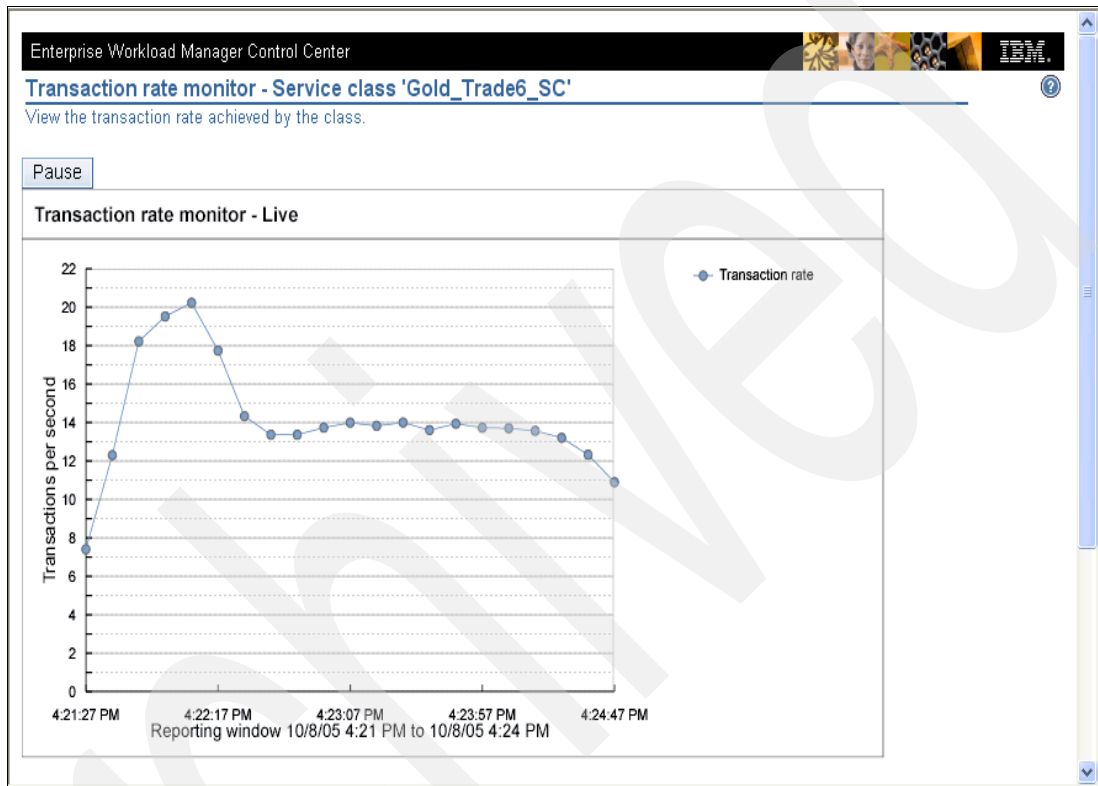


Figure 6-24 Transaction rate report

Analyzing PI in a partition management environment

If you configured EWLM partition management, yet the PI for a service class indicates that it is still not meeting its goal, then the following discussion is intended to show what factors may be contributing to the missed goal.

The first place of interest is the Service Class or Exceptions view to see which, if any, of your service classes are not meeting their goals.

The Partition Workload Group Details (see 6.3.7, "Partitions details" on page 172) allows you to monitor your partition's processor utilization and view any adjustments made by EWLM to the entitled processing units within the partition workload group. It must be remembered that the situation of a poor PI is only addressed by EWLM partition management when the root cause is the application being CPU starved. In that case, EWLM will attempt to increase the entitled processing units for the partition contributing to the poor PI by lowering the entitlements for the partitions doing less important work. See 7.1, "pSeries and IBM System p5 partition management" on page 192.

When EWLM realizes that the performance index is not being met by a service class it will contact the managed servers and perform a check of how much CPU resource each partition

could provide and still meet the goal for transactions flowing through it. If it gets a positive response from one or more managed servers that resources are available, EWLM initiates a dynamic reconfiguration of entitled capacity. If it gets a negative response from all managed servers or when it realizes that giving the suffering work stream more resources will not improve performance, EWLM will not make any adjustments to partition entitlements. Furthermore, EWLM will not initiate any changes to entitlements when there are resources available from the free pool for distribution using the POWER5 hypervisor algorithm.

Therefore, if you have a situation where your performance index for one or more service classes is not being met and EWLM is unable to satisfy the goal by shifting entitled processing units, then you need to explore your environment further for other factors that may be contributing to the service class missing its goal. You should check the *domain settings* to verify whether EWLM is in fact running in management mode. If partition management is enabled then your environment is most likely not CPU bound, so shifting CPU resources will not improve the overall performance index.

There are several possibilities of what might be the cause of the problem. Maybe your goals are set too high, such that with the resources available they cannot be met. Another cause could be application boundness. Depending on how work transactions are passed from one tier to the next, it might be that a transaction has to wait for the previous transaction to be finished before it can run. In this case, increasing the processing units on any one server will be unlikely to improve the PI. Maybe there are parameters within the application that can help limit the queuing time. Another contributor to a bad PI could be the network if it does not allow sufficient traffic to be passed between servers.

For most of these cases it is worth looking at the local PI of the partitions. When the end-to-end PI for the service class is greater than 1, which means we are missing the goal but the local PI of the individual servers is good, then EWLM determines that the overall PI will not improve simply by reallocating entitlements. One way to approach this bad PI value is by looking at the average active time associated with each server at a given hop. The one with the highest active time will have the highest local performance index value compared to the other servers at the same hop. EWLM shifts resources if the bad PI is due to CPU delays. Therefore, CPU delay experienced by a service class at a specific server is a useful indicator; however, it is not necessarily sufficient for EWLM to react.

Another approach is to look at the application topology for the service class missing its goal. If possible, EWLM identifies the server that is the most likely cause of the bad PI by a yellow triangle containing an exclamation mark, as shown in Figure 6-25. To get a more detailed view of the performance data, go to the tableview within the application topology or view the hop details for the specific service class.

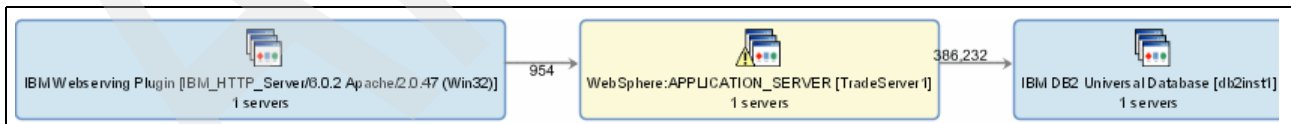


Figure 6-25 Application topology for a problem situation

Tableview

The data in the tableview can indicate how well the transactions are flowing or if they are not, the data may help you determine why not. In Figure 6-26 we show an example of the first few columns of the table view. When you scroll to the right view you can view additional performance data.

Hop number	Name	Platform	Average response time	Standard deviation	Average active time	Average queue time	Successful transactions	Failed transactions	Stopped transactions
0	IBM Webserving Plugin (IBM_HTT...		01.866154	00.292000	00.007628	00.007250	1,287	0	0
0	ewlma1	Windows	01.866154	00.292000	00.007628	00.007250	1,287	0	0
0	EWLM1/PID=0000000508		01.866154	00.292000	00.007628	00.007250	1,287	0	0
1	WebSphere.APPLICATION_SERV...		01.859992	00.290000	00.759205	00.000000	1,288	0	0
1	ewlmaix1	AIX	01.859992	00.290000	00.759205	00.000000	1,288	0	0
1	ewlmaix1Node01.TradeSe...		01.859992	00.290000	00.759205	00.000000	1,288	0	0
2	IBM DB2 Universal Database [db...		00.000888	00.008000	00.000888	00.000000	519,199	0	0
2	aixve01	AIX	00.000888	00.008000	00.000888	00.000000	519,199	0	0
2	31		None	None	None	None	0	0	0
2	34		00.000952	00.008000	00.000952	00.000000	12,548	0	0
2	45		00.000850	00.006000	00.000850	00.000000	12,951	0	0

Figure 6-26 Example of table view for performance analysis

A good starting point is to look at the *average active time* and see if any hop has a large value compared to the other application hops. In this example the value for the WebSphere Application Server running on ewlmaix1, which is part of the *gold* workstream, is much higher than the other two tiers. This would tend to indicate, since EWLM is not currently shifting resources, that we are not CPU bound but that there might be a problem with the WebSphere instance or how it is set up to work.

When looking for possible causes of bad PI you should also check the *average queue time* entries in the table view. In the example, this number for the IBM HTTP Server is quite large, especially compared with the average active time. The recommendation in such a case is to verify that all parameters influencing queuing time are properly established for the application.

Other good sources of information are the *successful*, *failed*, and *stopped transactions*. If the value for the successful transaction count in the table view shows a much lower value for one hop relative to the others, it may indicate a performance problem. When looking at our table view the number of successful transactions for the IBM HTTP Server is much lower than for the other two hops. This might be attributed to the queuing problem mentioned in the previous paragraph. The failed transaction count is worth looking at to see if it is higher than you would expect or if one hop has a much higher count relative to the other hops.

Within the table view if you scroll further to the right, you see a column called processor delay. A high value for the processor delay for an application is usually a good indicator of a performance problem.

6.4 Performance statistics logging

The EWLM Control Center allows you to view statistics only over the last 24 hours. For trending and capacity planning in an enterprise environment, we need data for longer periods of time. Performance statistics logging is a new feature in EWLM 2.1 that allows you to log performance data to disk. The data, collected and stored by the domain manager, can be analyzed in a variety of ways, and the resulting information can then be used for capacity planning, trending, and assessment of the configuration.

This section describes how we enabled and configured performance statistics logging in the ITSO environment. It also provides a sample scenario of how you might use this data for capacity planning.

6.4.1 Configuring performance statistics logging

The domain manager is responsible for collecting performance statistics for all managed servers in the EWLM management domain. This is configured in the xml file. The default location is <Install Dir>\servers\<configID>\Plugins\DataHardeningPlugin.xml.

This file is automatically created the first time that the domain manager is started with the startDM command. Example 6-1 shows an example of the DataHardeningPlugin.xml.

Many key/value pairs are self-explanatory. However, there are several configuration options you should configure based on your goals, environment, and skills. These are described in the subsequent sections.

Example 6-1 DataHardeningPlugin.xml sample

```
<?xml version="1.0" encoding="windows-1252" ?>
<Plugin Type="Runtime" >
  <Implementation Class="com.ibm.wlm.dh.DataHardeningPlugin" />
  <Origin origin="IBM" />
  <Arguments
    RefreshablePluginState="Enabled"
    OutputFormat="SER"
    OutputEncoding="UTF-8"
    OutputFilePrefix="SER_EWLMReportData"
    DestinationDir="C:\Program Files\IBM\VE2\EWLM\ewlmdm\servers\tot132\Diagnostics"
    DeltaMode="True"
    IntervalLength="5"
    FileSizeLimit="NO LIMIT"
    RecordsPerFile="NO LIMIT"
    MaxFileCount="NO LIMIT"
    OLCTraceEnabled="False"
  />
</Plugin>
```

RefreshablePluginState

To enable performance statistic collection, set RefreshablePluginState to Enabled.

OutputFormat

The OutputFormat parameter in DataHardeningPlugin.xml allows you to specify the output format of the data file. The possible values are XML, SER, or OLC.

XML Performance statistics are written in raw XML. There are several advantages of this format. XML can be programatically accessed using XML parsers, or

transformed using eXtensible Style Sheet Language Transformations (XSLT). At the most basic level, the XML is readable with no tools.

Because many spreadsheet applications support XML, you could quickly import the performance statistics and create charts and graphs. However, this can be problematic if your XML documents are too large. When we attempted to import XML performance statistics into Microsoft Excel® for a workload scenario that lasted several days, we encountered an Excel limitation on the number of rows. Therefore, we recommend using smaller data sets or pruning the data sets. This method requires the most disk space.

SER SER is short for *serialized*. In this format records are written to a file as serialized Java objects. This format requires that you use Java and the EWLM Reporting API to deserialize the records into EWLM Reporting API objects. Then you can call methods on the EWLM objects to retrieve specific statistics. In 6.5, “Scenarios and examples” on page 185, we provide an example using this format.

The SER format also provides methods to *manipulate* the object. For example, given two total mode objects, you can generate a delata mode object using the methods provided.

OLC OLC stands for offset length count. With this format, records are written to file in binary. Using any language, such as C or Java, statistics can be accessed by the offsets and lengths of the fields.

This method requires the least disk space.

For a quick summary of this information, see Table 6-2.

Table 6-2 Format comparison

Output format	Disk space consumption	Required tools	Complexity of parsing	Domain manager overhead
XML	High	-Knowledge of XML -Importing into spreadsheet	Low	High
SER	Medium	Java programming	Medium	Medium
OLC	Low	Any basic programming language can be used to parse the output file	High	Low

IntervalLength

IntervalLength allows you to specify the data collection and aggregation interval, in minutes. The default value is 5 minutes. Depending on the analysis you are performing with the data, you can adjust the interval. For example, in a capacity planning exercise, you might want to keep this as a small value to be sure that you are capturing every peak or valley or the workload. In a policy assessment, this interval might be a little wider.

DeltaMode

The DeltaMode parameter allows you to represent the data in two ways. If DeltaMode is set to true, the data written to the output file contains statistics for the current interval only. We will refer to this mode as *delta mode*.

If DeltaMode is set to False, the data written to the output file contains cumulative statistics since one of the following events (whichever is most recent):

- ▶ The domain manager started.
- ▶ A new EWLM Policy was activated.
- ▶ A managed server joined the domain.

We refer to this mode as *total mode*.

Delta mode is helpful in capacity planning, as it captures data for shorter intervals. For example, suppose that you want to better understand performance during the evening hours, when system usage is the heaviest. Delta mode would be the best choice, as it will allow you to specifically analyze data for that time of day. Total mode would be less helpful in this situation because morning and afternoon statistics would also be factored into the data.

On the other hand, total mode can be very useful for accounting purposes since it will give you the overall totals for a period of time.

To further illustrate the differences between delta and total mode, consider Table 6-3, which contains example data collected at 5-minute intervals.

Notice how Sum of Response Time (RT) and Transaction Count (TC) are cumulative in total mode but interval-specific in delta mode. This has a dramatic effect on the Average Response Time (RT/TC) values.

Table 6-3 Total mode versus delta mode

	13:05:00	13:10:00	13:15:00	13:20:00	13:25:00	13:30:00
TOTAL MODE (5-minute interval)						
Sum of Response Time (RT)	100	1100	1200	1400	2100	2700
Transaction Count (TC)	10	20	30	50	70	90
Average Response Time (RT/TC)	10	55	40	28	30	30
Processor Utilization	25	70	25	20	40	30
DELTA MODE (5-minute interval)						
Sum of Response Time (RT)	100	1000	100	200	700	600
Transaction Count (TC)	10	10	10	20	20	20
Average Response Time (RT/TC)	10	100	10	10	35	30
Processor Utilization	25	70	25	20	40	30
DELTA MODE (15-minute interval)						
Sum of Response Time (RT)		1200		1500		
Transaction Count		30		60		

	13:05:00	13:10:00	13:15:00	13:20:00	13:25:00	13:30:00
Average Response Time	40			25		
Processor Utilization	$(25+70+25)/3 = 40$			$(20+40+30)/3 = 30$		

6.5 Scenarios and examples

There are different situations in which the performance statistics logging can be useful in your installation. In this section we give you scenario examples of when this feature can be useful.

Capacity planning

A time where the performance statistics logging can be very useful is during the capacity planning task.

There are several methods and tools that you can use when performing capacity planning. The performance statistics logging can be used to input performance data into your capacity planning exercise.

With the performance statistics logging you can understand the trend of your workload, verify whether there are peaks of utilization, and be able to project the future behavior.

As you can see in the example below, you can extract current performance data and make linear projections to understand if any resources, especially CPU, are going to become a bottleneck. Transaction rate and CPU utilization are good indicators to understand the growth and potential consumption of the workload.

Topology

Another situation in which performance statistics logging can be useful is the mapping of the workload distribution versus the application/server topology. If your installation needs to verify how the transactions are distributed by servers or by applications, the performance statistics logging can be useful in providing this type of information.

Building and assessing the EWLM policy

The performance statistics logging can also be used to verify the efficiency of the EWLM policy. For example, as described in 5.2.3, “Building the Domain Policy” on page 141, a recommended and simple way to start working with EWLM is by deploying the sample policy provided with the product and make consequent adjustments to reflect your environment.

You can do this by using the Control Center and analyzing the performance data online, although this task might be difficult at times since you can only view a certain interval of data or you can use the performance statistics logging. With this tool, you can analyze the data by Application Environment and verify how the transactions are distributed compared to their response time and, based on this information, be able to adjust the transaction class/service class for this workload. Similarly, you can extract data for the process and evaluate their response times compared to a velocity goal and make adjustments.

Once you have adjusted the EWLM policy, you can redeploy it and perform many iterations of this process until you reflect the correct behavior and performance goal in the EWLM policy.

6.5.1 Capacity planning example

In this example we use the SER file with Java and the EWLM API to extract statistics for one WebSphere Application Server. The example Java program is downloadable. Refer to Appendix B, "Additional material" on page 307 for instructions.

First, we configure the domain manager's DataHardeningPlugin.xml. The important values we used are shown in Example 6-2.

Example 6-2 DataHardeningPlugin.xml sample

```
RefreshablePluginState="Enabled"  
OutputFormat="SER"  
OutputEncoding="UTF-8"  
DeltaMode="False"  
IntervalLength="5"  
FileSizeLimit="NO LIMIT"  
RecordsPerFile="NO LIMIT"  
MaxFileCount="NO LIMIT"
```

Then we collected five hours of performance data while increasing the workload running on these servers.

We wrote a simple Java application and used the EWLM API to extract processor utilization, transaction rate, transaction response time, performance index, and the interval time for one of our WebSphere Application Servers. The Java program writes the data to a Comma Separated Value (CSV) file so that it can easily be imported into a spreadsheet application. Example 6-3 provides an example of the CSV file with data from only the first three intervals.

Example 6-3 CSV file of statistics for three 5-minute intervals

```
ProcessorUtilization,2.128847599029541,2.124846935272217,2.114020347595215  
TransactionRate,288.955223880597,447.18791046954397,605.1089680813982  
TransactionRT,21.0,21.0,21.0  
PerformanceIndex,0.03200872987508774,0.03403092548251152,0.03527209162712097  
Time,2006/06/06 11:11:50,2006/06/06 11:16:52,2006/06/06 11:21:53
```

We then imported the CSV file into Microsoft Excel and generated several graphs. The following are samples of the graphs you can plot with the collected data that could be useful in a capacity planning exercise to understand the trend of your workload and when there will be a need for additional resources.

The first graph, shown in Figure 6-27, charts processor utilization and transaction rate in a given time period. This is helpful in capacity planning: for a given transaction rate we can see the impact to CPU utilization. From the chart you can see the CPU consumption trend compared to the transaction rate growth. This graph can help you understand the correlations between the two elements and how fast your workload can grow before saturating the current resources. In our example, you can see that running with a transaction rate of 1000 transactions per second will saturate the current available CPU.

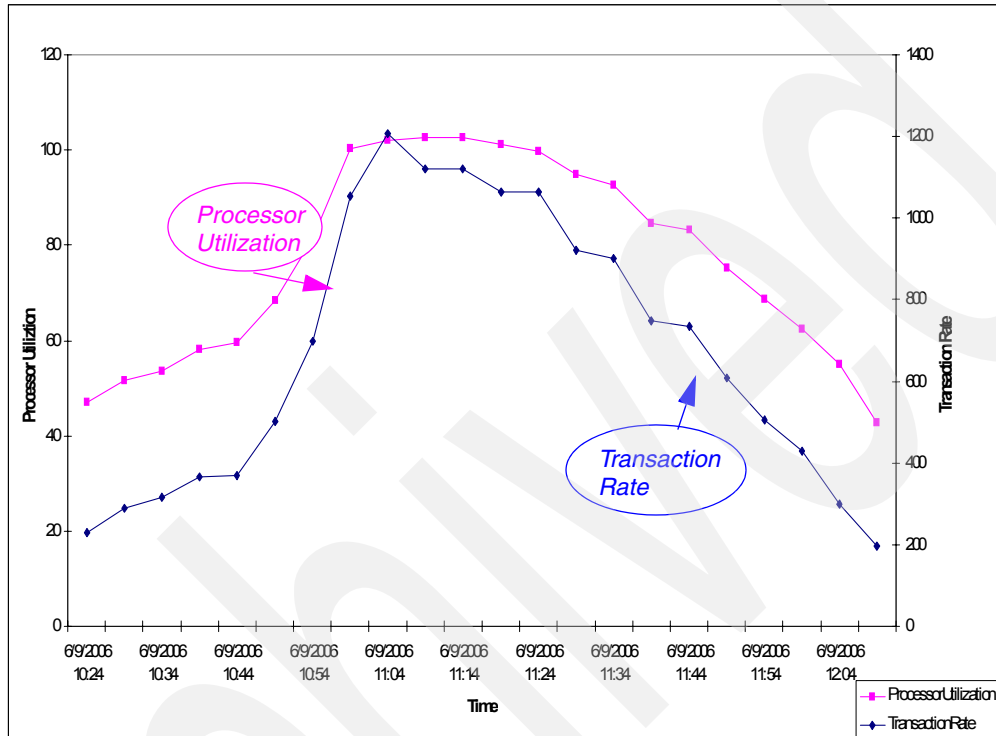


Figure 6-27 CPU utilization versus transaction rate

Additional useful information you can obtain from the performance data and plot on a graph is shown in Figure 6-28, where the performance index (PI) and the transaction response time are traced. In this sample, you can see the behavior of the response time being consistent with the performance goal.

From the chart you can also observe that the workload trend is not evenly distributed and has spikes at certain times, for example, in our case around 11:04 a.m. This information could be useful if you are planning a server consolidation scenario where you know you can mix workloads onto the same machine that have workload spikes that occur at different time intervals.

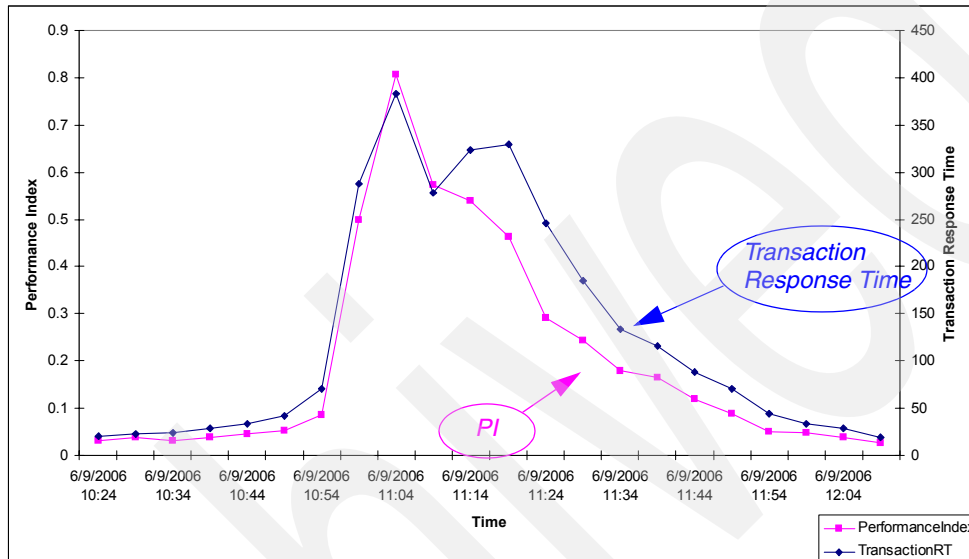


Figure 6-28 Performance index versus transaction response time

In the last graph in Figure 6-29, we show another example on how you can plot the performance data with the CPU utilization and the transaction response time. This chart can help you understand when you are reaching the knee of the curve and the CPU resource might become a bottleneck in your installation. In our example, you can see that as the CPU utilization increases, the transaction response time gets higher and higher, potentially starting to accumulate queuing time. From this graph, you can see how the workload behaves and how it can use the CPU resource without causing a constraint.

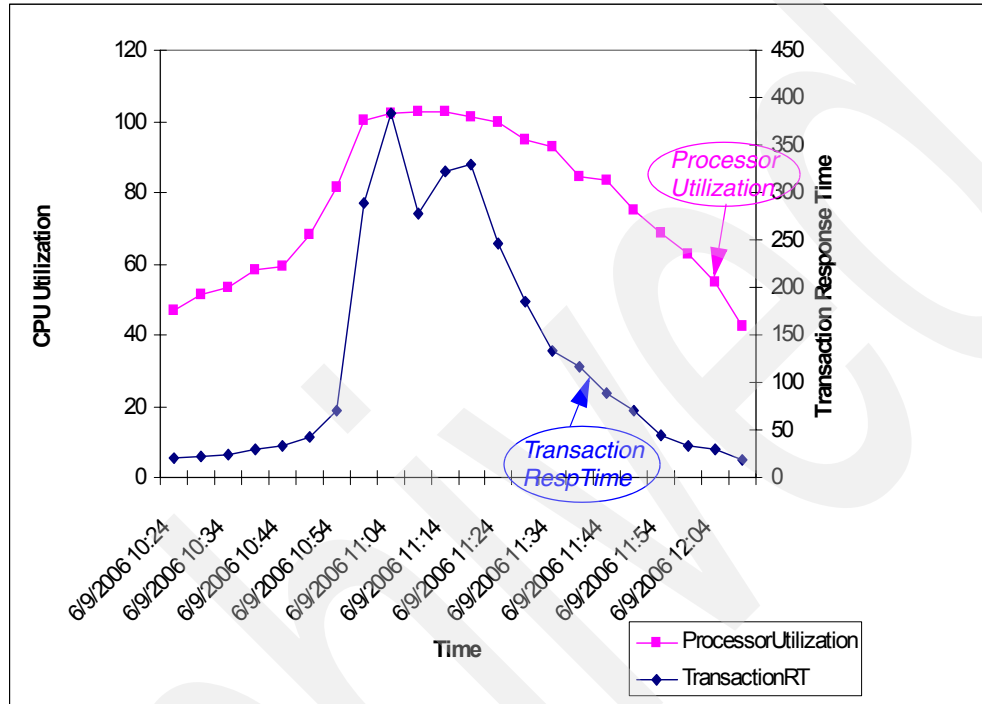


Figure 6-29 Processor utilization versus transaction response time

Archived



EWLM Management

This chapter describes how EWLM is managing the following resources:

- ▶ Processing units within a POWER5 partition workload group
- ▶ Network load balancing - Provides a description of how EWLM influences load balancing through the use of relative weight recommendations

7.1 pSeries and IBM System p5 partition management

EWLM collects and aggregates performance statistics that can be viewed using the EWLM Control Center. EWLM captures relevant performance statistics on each operating system instance within a Management Domain. It then reports an aggregated view of the real-time state of the domain to the EWLM domain manager. Detailed statistics of the actual results as opposed to the desired results can be viewed in the EWLM Control Center.

In this section we concentrate on statistics, reports, and performance data related to IBM POWER5 partition management. We focus on the POWER5 partition management because that is the platform where EWLM V2.1 offers management functionality. We include a discussion on situations when EWLM is maybe shifting resources in a different way than we might expect or EWLM is not shifting resources at all even though we do not meet the performance goal for specific service classes. The purpose of this discussion is to make you aware and understand what EWLM wants to achieve and how it interferes with partition management.

7.1.1 Terminology and general concept of POWER5 partitions

EWLM V2.1 is capable of automated POWER5 partition management. In the ITSO environment, we use a p-550, which means we base the general and ITSO-related discussion on partition management on IBM System p5 and IBM @server® p5 systems. However, this concept applies similarly to POWER5-based partitionable iSeries systems.

Before we go into a detailed discussion of the actions EWLM initiates and the data it displays, we need to clarify the concept of Micro-Partitioning™ available on the POWER5-based platform. This includes a discussion on the POWER5 hypervisor as well as terms like *virtual* and *physical processors*, *capacity entitlements*, and *weight*. In this chapter we present an overview on this topic but for a more detailed discussion we refer you to *Advanced POWER Virtualization on IBM System p5*, SG24-7940.

Micro-Partitioning allows for multiple partitions to share one physical processor by dividing it into fractions of processing units. Through the more efficient allocation of processing resources, the overall CPU utilization within the managed server is increased. In the partition configuration of Micro-Partitioning, two modes, *capped* and *uncapped*, are available. The capped mode prevents the hypervisor from making any idle processing units available to the partition above its entitled capacity. When a partition is uncapped the hypervisor can make additional processing units available to the partition up to the value specified for its maximum processing units of capacity.

Terminology

The concept of Micro-Partitioning includes important terms like *virtual* and *physical processors*, *entitled capacity*, and *weight*. Some basic definitions are listed as follows:

- Virtual processor** It defines the way that a partition's entitlement is spread over physical processors. In other words, it is the whole number of concurrent operations that the operating system can execute. From the operating system point of view, a virtual processor is indistinguishable from a physical processor.
- Physical processor** It is the number of actual physical hardware resources; that is, the number of unique processor cores but not the number of processor chips because each chip contains two processing cores.
- Processing capacity** It is specified in terms of processing units and can be configured in fractions of 1/100 of a processor. The minimum capacity for a partition is 1/10 of a processor specified as 0.1 processing units. If you want to

assign 1/2 of a processor to your partition the value you put would be 0.50 processing units.

Entitled capacity

When a partition is started, the system assigns the partition's entitlement to be within the capacity range defined by the minimum and desired processing units. After a partition is activated, the processing capacity it is assigned is usually referred to as capacity entitlement or entitled capacity. This entitled capacity, as the terminology implies, represents the portion of the shared pool that "cannot be denied" to the partition when it requires the processing capacity.

Weight

If multiple uncapped logical partitions simultaneously require more idle processing units than are available from the shared pool, the POWER5 hypervisor distributes the idle processing units in proportion to each partition's weight. The higher the weight value the more processing units the partition gets. A partition's share is computed by dividing its variable capacity weight by the sum of the variable capacity weights of all uncapped partitions.

Processing units of capacity

On the Hardware Management Console (HMC), where you define your partition settings, the processing capacity is specified in terms of *processing units*. A partition may be defined with a processor capacity as small as 1/10 of a processor and is specified as 0.1 processing units. This means that if you want to assign 50 percent of a processor to a partition, you specify 0.50 processing units in your partition profile. Each processor can be shared by up to 10 shared processor partitions. Additional processing capacity can be configured in fractions of 1/100 of a processor. The shared processor partitions are dispatched and time-sliced on the physical processors under the control of the POWER5 hypervisor.

When setting up a partition you need to define both its memory and I/O resources. Additional values to be specified are shown below:

- ▶ Minimum, desired, and maximum processing units of capacity
- ▶ Minimum, desired, and maximum virtual processors
- ▶ Capped or uncapped

When a partition is started, preference is given to the *desired* value of processing units of capacity. This value is then referred to as *entitled capacity*. If there are not enough processing units available in the system when starting the partition to satisfy the desired capacity, a value between the *minimum* and desired defined is chosen for the entitled capacity. For uncapped partitions the *maximum* limit specifies the upper limit for dynamic operations only. It defines the maximum processing units that the hypervisor is allowed to make available to the partition.

Virtual processors

The second value that needs to be specified for each partition as listed above is the *minimum, desired, and maximum* number of *virtual processors*. In Micro-Partitioning there is no fixed relationship between the virtual and physical processors. The POWER5 hypervisor can use any physical processor in the shared pool when it schedules a virtual processor. By default, it attempts to use the same physical processor that was last used by the partition, but this cannot always be guaranteed. From the operating system point of view, a virtual processor is indistinguishable from a physical processor. The POWER5 Hypervisor uses a dispatch wheel with a fixed time slice of 10 milliseconds to guarantee that each virtual processor receives its share of entitlement in a timely fashion. When a partition is completely busy the partition entitlement is evenly distributed among its online virtual processors.

By default, the number of processing units that you specify is rounded up to the minimum number of virtual processors needed to satisfy the assigned number of processing units. If you specify 0.75 processing units, for example, one virtual processor will be assigned. If you specify 2.60 processing units, three virtual processors will be assigned.

The number of virtual processors has to be chosen sensibly. The optimal number of virtual processors depends on the workload in the individual partitions. The default settings maintain a balance of virtual processors to processing units. If you define too small a number of virtual processors you may limit the processing capacity of the partition. For example, if you specify 0.50 processing units and a maximum of one virtual processor, the partition cannot exceed 1.00 processing units and therefore be limited to executing one operation at a time. If, on the other hand, you assign the same partition two virtual processors, the partition could use an additional 1.50 processing units and execute two concurrent operations.

However, if you assign a very high number of virtual processors to your partition you might impact your performance negatively because of extensive context switching. For this, starting with maintenance level 3 of AIX 5L V5.3, an improved management of virtual processors is introduced. It minimizes the use of virtual processors that are idle and therefore enhances the utilization of a shared processor pool. These idle virtual processors are not removed from the partition but they are put to sleep or disabled. This means that with this feature, even when you assign a high number of virtual processors, your overall performance will not be negatively influenced.

Unless you have applications that you specifically need to bind to a small number of processors, we recommend that you set the number of virtual processors equal to the number of available processors in the shared pool.

7.1.2 Partition workload group

Within a partitionable POWER5-based server you can have several shared partitions and group these together in a partition workload group. These settings are defined on the Hardware Management Console (HMC) when you create a partition. These partition definitions are available to EWLM for all partitions running the managed server code. If you have partitions within one partition workload group, of which some are managed by EWLM and others are not, the EWLM *Partition Workload Group Details* view monitors and manages only the partitions participating in the EWLM domain.

Figure 7-1 shows an example of a possible workload group assignment. If you have six shared processor partitions defined, for example, all six partitions could be assigned to one partition workload group. Alternatively, you could have three partitions belonging to workload group 0, two others belonging to workload group 1, and the last partition is not participating in any EWLM workload group at all. This partition still belongs to a workload group when you look at the HMC definitions. If you have not assigned a specific group, it belongs by default to workload group 0.

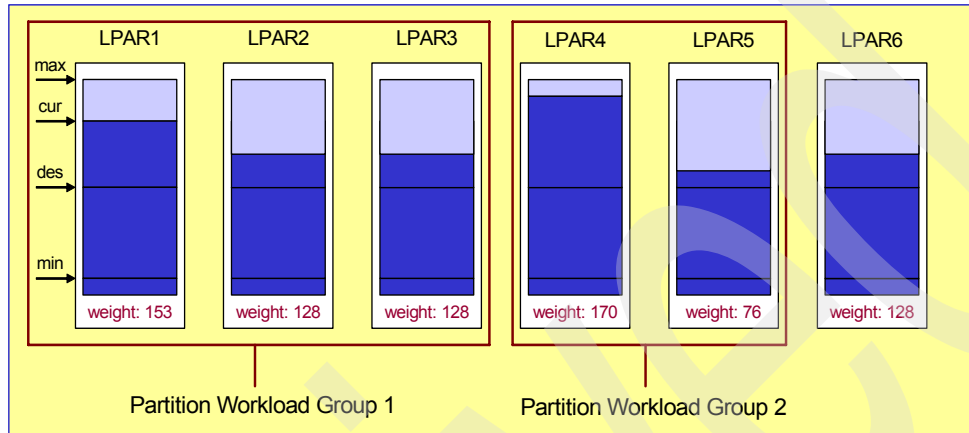


Figure 7-1 Possible workload group assignment

Note that EWLM shifts resources only between partitions belonging to the same group but not between partitions of different workload groups. Therefore, you might want to group all Web servers or applications servers together, or maybe it is a better idea to group all partitions belonging to one line of business together. Alternatively, you can work with only one partition workload group for all your partitions participating in the EWLM concept. We do not provide a general recommendation because this setup will be very customer specific.

In the *Partitions* view of the EWLM Control Center you can see your workload groups listed as displayed in Figure 7-2. To view and monitor individual members of the groups go into the *Details* view of each group.

Partition Workload Groups				
View the state of the partition workload groups in the domain. Select a partition workload group for more details.				
Interval: 10/16/05 12:03:30 PM to 10/16/05 12:04:30 PM				
--- Select Action --- Go				
Select	Machine identifier	Group number	Processor utilization	Processor allocation
<input type="radio"/>	9113-550106628E	0	2.2%	90.0%
<input type="radio"/>	9113-550106628E	1	15.4%	10.0%

Page 1 of 1 Total: 2 Filtered: 2 Displayed: 2

Figure 7-2 Example of partition workload groups

In this example, we have two workload groups within one p5-550. If you do not specify a workload group number when creating the partition profile, the partitions belong by default to partition workload group 0. These assignments can be changed at a later time, but when you move the last member of a specific group, the workload group will not exist anymore and you will not be able to move other partitions to this workload group. It is sufficient though to create a new partition profile with a new partition number. This partition does not need to actually run

in order to create a new workload group number. Bear in mind though, that for changing the partition workload group number a shutdown and reactivation of the partition is required.

Processor allocation

If you have a setup where all partitions belong to one workload group, the recommendation for the entitled capacity for the partitions is that the sum of the entitlements is equal or close to the number of processors in the shared pool. The reason for this is that EWLM shifts resources only within the group. It will only reallocate resources within the group such that the sum of the entitlements is always equal to the sum of the entitlements we started with. It quickly settles down to the sum of entitlements as originally defined in the partition profile. This means that if you chose the sum of entitled capacity for all partitions to be less than the available processing units in the shared pool, you are not using CPU resources efficiently. If you chose the sum of entitlements equal to the available resources in the free pool it is not possible to start new partitions without manually initiating dynamic reconfigurations.

If you have a more complex environment where you have some partitions belonging to the EWLM workload group but also other partitions not involved in the EWLM setup at all, or you have several partition workload groups defined, you need to think about the entitlement settings a little bit more carefully. If you chose the sum of the entitlements for your EWLM workload group too small or the desired number of entitlements too close to the minimum number of entitlements, then EWLM may not be able to shift many resources around within the group to improve performance goals for specific workloads.

The initial entitlement settings should reflect a stable working environment for the normal workload you are running. If your environment then experiences increased workload on any of your workload streams, EWLM can shift resources within the group to achieve the goals. At a later time you can always change the entitlement of one or more partitions within your group using dynamic reconfiguration and hence change the amount of entitled CPU resource within the workload group.

One method to view the entitlements for the partitions participating in the EWLM domain is shown in Figure 7-3. The Partition view provides information about the *server*, *workload group number*, *processor utilization*, and *processor allocation*. In this example, there is only one workload group, the default group 0. The value for the processor allocation is 90 percent. This means that 90 percent of the available processing units from the shared pool are allocated as entitled capacity to the partitions participating in this EWLM workload group.

Select	Machine identifier ¹	Group number ²	Processor utilization [^]	Processor allocation [^]
<input type="radio"/>	9113-550106628E	0	69.9%	90.0%
Page 1 of 1		Total: 1 Filtered: 1 Displayed: 1		

Figure 7-3 Partition view

An important partition management concept to be aware of is that EWLM will only shift resources when it believes that shifting CPU resources will improve the performance of the workload not meeting its goal. The EWLM algorithms calculate how much CPU resource each partition can relinquish and still operate within its performance goals and what effect adding these CPU resources to the partition in need will improve its workload performance. If EWLM concludes that moving CPU resource will not improve the PI, then EWLM probably concluded that the failure to achieve the goal is not due to a lack of processing power, that is, the bottleneck is caused by network delays or application queuing.

As discussed previously, EWLM readjusts values according to changing workload, but the maximum amount of resources within the workload group is unchanged. For an individual partition, however, it may be shifted from 0.9 to 0.5, for example, and for another partition

from 0.9 to 1.3. After a shift of resources and an improved performance for all service classes, EWLM does not reshift resources back to the original entitlements. It only moves resources when another service class is not achieving the goal. If there is no need for any resource change, EWLM keeps the resource assignment as it is even when the workload is stopped altogether.

7.1.3 Monitoring the performance of partition workload groups

The best view to monitor your POWER5 partitions is to go to the Partition Workload Group Details. Log in to the EWLM Control Center, go to **Monitor** → **Partitions**, and open the details view of your server. This view is shown in Figure 7-4.

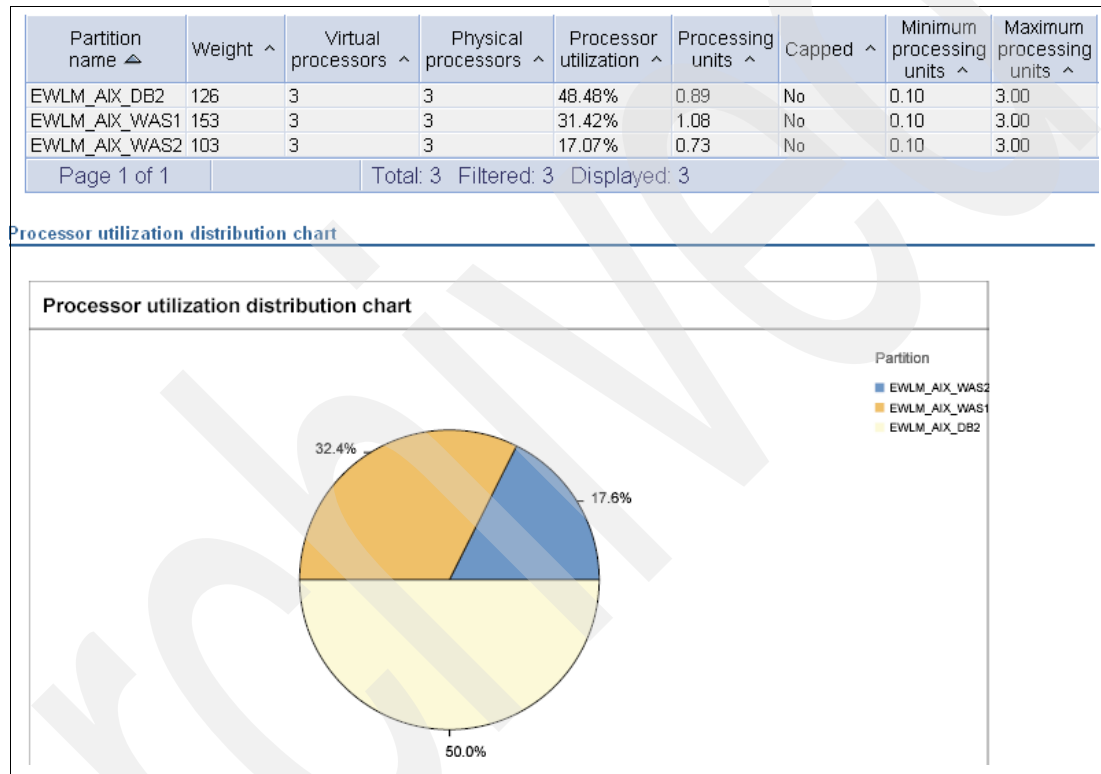


Figure 7-4 Partition workload group details

We have only one workload group, which consists of three servers. If you have different workload groups defined you see these individually listed. To view the Partition Workload Group Details, as shown above, you need to open the Details view of each workload group separately. The Partition Workload Group Details view shows the *partition name*; the *weight*; the number of *virtual and physical processors*; the *partition mode*; the *minimum, entitled, and maximum processing units*; and the *processor utilization*. It also includes the memory settings for each partition further to the right, but these values are cut off in the figure.

The columns where values can change due to EWLM adjusting resources are *weight*, *virtual processors*, *processing units*, and *processor utilization*. All other values are static after a partition is started and can only be changed using either dynamic reconfiguration operations or a partition shutdown and reactivation.

The value for the processing units represents the entitled capacity. As discussed before, for a given workload group the sum of these entitlements will always be the same or close to the sum of the entitlements the group started with. These values should reflect what you see when running the `lparstat` command with the `-i` option. The `lparstat` output is shown in Example 7-1 for the partition called `EWLM_AIX_WAS1`. The important numbers that are also displayed in the EWLM Workload Group Details view in Figure 7-4 on page 197 are highlighted below and are as follows: *Partition Name*, *Entitled Capacity*, *Variable Capacity Weight*, and *Active CPUs in Pool*.

Example 7-1 `lparstat` output for comparison to EWLM Workload Group Details view

```
# lparstat -i
Node Name                : ewlmaix1
Partition Name          : EWLM_AIX_WAS1
Partition Number         : 2
Type                     : Shared
Mode                    : Uncapped
Entitled Capacity      : 1.08
Partition Group-ID       : 0
Shared Pool ID           : 0
Online Virtual CPUs      : 3
Maximum Virtual CPUs     : 32
Minimum Virtual CPUs     : 1
Online Memory            : 4096 MB
Maximum Memory           : 4096 MB
Minimum Memory           : 2048 MB
Variable Capacity Weight : 153
Minimum Capacity      : 0.10
Maximum Capacity      : 3.00
Capacity Increment       : 0.01
Maximum Physical CPUs in system : 4
Active Physical CPUs in system : 4
Active CPUs in Pool    : 3
Unallocated Capacity     : 0.00
Physical CPU Percentage   : 36.00%
Unallocated Weight       : 2
```

Processor utilization

The order in which we have seen EWLM react is to change the number of virtual processors first to increase performance and to change the weight and capacity entitlement in the next step. A very interesting number in the Partition Workload Group Details view, as shown in Figure 7-4 on page 197, is the percentage of *processor utilization*. It is displayed as a number and as a graphical view for all partitions. The percent processor utilization shows what percent of the total physical processors available in the shared pool was used by the partition over a specific time interval. This number does not by itself imply a resource adjustment. Processor capacity is redistributed if EWLM determines degradation in performance by analyzing the performance index and the CPU delays. On the other hand, EWLM does not start adjusting resources for a partition until the partition uses at least 90 percent of its entitled capacity. This means that the value of the processor utilization is very useful for verifying this threshold.

The processor utilization allows you to get reliable performance numbers for all partitions in one screen. There is no need to additionally check the partition utilization using AIX performance monitoring tools for each individual partition. The following example shows how to deduce from the available values in the Partition Workload Group Details view, the number of physical processors consumed shown as *physc* value, and the percentage of entitled

capacity shown as the *entc* value in AIX monitoring tools. We use the values displayed in Figure 7-5 for an example calculation.

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^
EWLM_AIX_DB2	126	3	3	48.28%	0.89	No	0.10	3.00
EWLM_AIX_WAS1	153	3	3	31.41%	1.08	No	0.10	3.00
EWLM_AIX_WAS2	103	3	3	17.16%	0.73	No	0.10	3.00

Figure 7-5 Numerical view of partition workload group details

The number of physical processors consumed (*physc*)

The processor utilization is 17.16 percent for partition EWLM_AIX_WAS2. The number of available physical processors in the free pool is 3, as displayed above in the column called physical processors. Calculating 17.16 percent of 3 processors gives a value of 0.52, which is similar to the *physc* values you see when running **topas** or **lparstat** commands, for example, on the AIX partition itself. It is shown in Example 7-2 and equals the number of physical processors consumed. We compare the calculated value with the displayed value of the example output below. This confirms the formula for calculating the *physc* value using the Partition Workload Group Details view provided by EWLM. It looks as follows:

$$\text{physc} = \frac{\text{processor utilization} \times \text{number of physical processors}}{100}$$

Example 7-2 *lparstat* output for partition EWLM_AIX_WAS2 for comparison of *physc* and *entc* values

```
ewlmaix2@/> lparstat 1 5
System configuration: type=Shared mode=Uncapped smt=Off lcpu=3 mem=4096 psize=3 ent=0.73
```

%user	%sys	%wait	%idle	physc	entc	lbusy	vcsw	phint
48.3	22.0	0.0	29.7	0.53	73.0	33.8	1442	450
44.9	20.8	0.0	34.2	0.50	68.7	34.8	1532	514
52.6	17.2	0.0	30.2	0.53	72.3	33.4	1378	399
49.7	19.5	0.0	30.8	0.52	71.7	35.6	1340	421
43.7	20.0	0.0	36.3	0.49	66.5	33.4	1491	468

The percentage of entitled capacity (*entc*)

Another important value displayed on AIX performance monitors is the *entc* value, the percentage of entitled capacity. This number can also be calculated using the numbers displayed in the EWLM Partition Workload Group Details view. We multiply the processor utilization by the number of physical processors and divide the value by the entitled processing units. This gives for partition EWLM_AIX_WAS2 with a *processing utilization* of 17.16 percent, three *physical processors* and *entitled processing units* of 0.73, a *percentage of entitled capacity (entc)* of 70.5, which is similar to the values shown in Example 7-2. The general formula looks as follows:

$$\text{entc} = \frac{\text{processing utilization} \times \text{number of physical processors}}{\text{processing units}} = \frac{\text{physc} \times 100}{\text{processing units}}$$

These numbers will not be of great interest for non-AIX-installations. However, for technical support teams who are used to working with AIX performance monitors, it helps a lot being able to extract these numbers from the EWLM monitor. This provides them with a better understanding of what the processor utilization numbers provided by EWLM mean.

7.2 EWLM and workload balancing overview

The EWLM Domain Manager has a global view of all the servers and middleware technologies that make up the Management Domain that supports the business applications, as described in 1.2.1, “Domain manager” on page 5. The Domain Manager obtains performance information about each server and application instance, and knows how much time a request is spending at each instance. The Domain Manager can use this knowledge to influence network routing decisions made by a Load Balancer in order to achieve the end-to-end business response time goal.

Without EWLM, when incoming requests arrive at a Load Balancer, the Load Balancer typically has little information about the application’s health or the performance of the servers it is routing to. It can use pure algorithmic techniques such as round-robin, or general system statistics represented by static, preconceived weights to route the incoming requests. Some Load Balancers have algorithms that allow them to sense network state and forward requests to the most capable instance, like the instance with the least number of sessions or connections.

EWLM does not route the work itself, but provides recommendations to the routing entity using the IBM Server/Application State Protocol (SASP). Through SASP messages, a Load Balancer can tell the domain manager which systems and applications it wishes to load balance and the domain manager can make recommendations to Load Balancers regarding how to distribute work. However, it is up to the Load Balancer to actually use EWLM’s recommendations to route incoming requests to the members. Today, the Load Balancers that exploit EWLM are from Cisco Systems, Inc. and Nortel Networks.

7.2.1 Configuring an EWLM load balancing setup

Figure 7-6 shows the components of an EWLM Management Domain that consists of Managed Servers, a Domain Manager, and a Load Balancer.

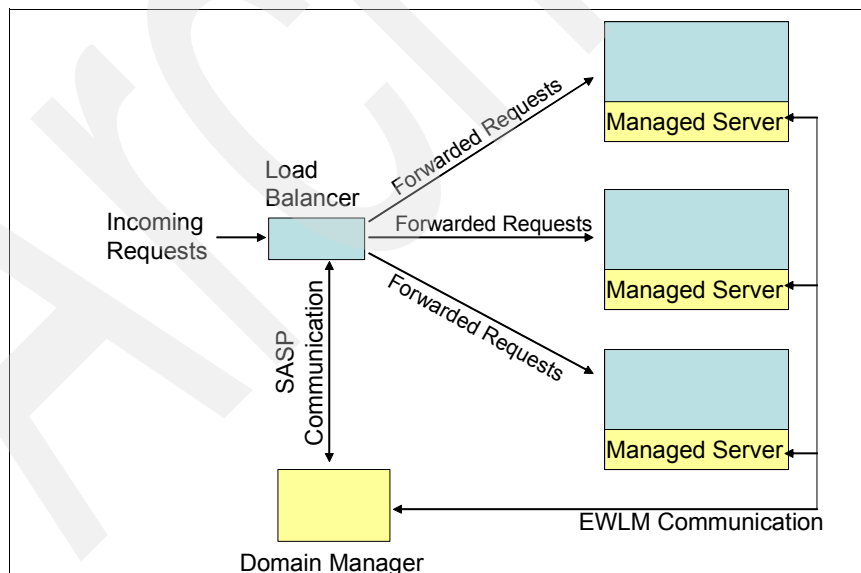


Figure 7-6 EWLM workload balancing components

Although it is assumed that customers who are using EWLM will most likely have instrumented applications using ARM, the Domain Manager supports both application-level and system-level load balancing. In application-level load balancing, the instances are identified by IP, port, and protocol. In system-level load balancing, the instances are identified

solely by IP address. Once the Managed Servers are configured to run in this environment, system-level performance data is sent to the Domain Manager by each of the Managed Servers in the Management Domain. If the managed server system is running ARM-instrumented applications, additional performance data can be sent regarding the performance of the applications themselves.

The Load Balancer registers those instances (system or application) as a group that it wishes to load balance to and associates them to the Domain Manager. The Load Balancer can then start obtaining EWLM distribution recommendations for each instance within the group. More than one group may be registered at any one time. Using the EWLM recommended distribution, the Load Balancer can forward incoming requests in a manner that provides better end-to-end business response time. At this time, the Load Balancers provided by Cisco Systems, Inc. and Nortel Networks both support system-level balancing. Application-level balancing is only performed by Cisco Systems, Inc. Load Balancers.

In the next section we discuss installation and configuration.

Managed server

Currently, the managed servers running on the following operating systems have load balancing support:

- ▶ IBM AIX 5.2 ML03
- ▶ Microsoft Windows 2000 server (32-bit)
- ▶ Microsoft Windows 2003 server (32-bit)
- ▶ Sun Microsystems Solaris 8 (SPARC Platform Edition)
- ▶ Sun Microsystems Solaris 9 (SPARC Platform Edition)

With this level of support, the EWLM Domain Manager together with the Load Balancer can perform system-level load balancing from the information provided by the Managed Server. If the applications running on these operating systems are ARM-enabled, then application-level load balancing can occur as well.

Installation and configuration

Installation and configuration are specific to each operating system.

AIX

AIX 5.2 ML03 has the platform support installed. There is a special fileset on the EWLM CD that must be installed. See 2.6.2, “Configuring EWLM managed server” on page 46.

Windows and Solaris

The Virtualization Engine Install Shield installs the system-specific code and the EWLM common code. The installation of the system-specific package makes all changes to the operating system required for it to run.

Solaris only

Additionally, in order for applications running on a Solaris operating system to participate in load balancing with EWLM, they need to have a linkage to the shared library object `/usr/lib/libnetwlm_applib.so`. One method to accomplish the link is via a shell script. Example 7-3 shows an application called `Webserv`, which has set the environment variable `LD_PRELOAD` to point to the `libnetwlm_applib.so` library. Using this shell script method allows the `libnetwlm_applib.so` library to be used only on those applications that the administrator selects to participate in the EWLM load balancing function. Further information about the `LD_PRELOAD` environment variable can be found in the Solaris manual *Linker and Libraries Guide*, 816-0559-10.

Example 7-3 Example script of a load balanced application running on Solaris

```
#!/bin/ksh
LD_PRELOAD=/usr/lib/libnetwlm_applib.so
export LD_PRELOAD
/opt/application/bin/Webserv
```

Domain manager

There are only minor configuration changes that need to be done in the Domain Manager so that it can participate in the load balancing environment.

Installation and configuration

As part of the initial creation of the Domain Manager via the `createDM` command, an IP address (`-ma address`) identifying the IP address or host name of the Domain Manager used by the Managed Servers to connect to is required. The Load Balancer will use this address in the load balancing environment to connect to the EWLM Domain Manager.

For load balancing, a port needs to be defined for communications between the Domain Manager and the Load Balancer or Load Balancers using the `changeDM` command. The port can be any valid port number, or it can be set to `Off` if no load balancing capability is desired. `Off` is the default. It can be an SSL port (`-lbs port`) or a non-SSL port (`-lbp port`). If the `-lbs port` is used, then `-sslks path` and `-sslpw password` must be supplied. The `-sslks` defines the path to an SSL keystore and the `-sslpw` defines the password for the SSL keystore (defined by the `-sslks` property).

If desired, both ports can be defined. All Load Balancers that want to communicate with the Domain Manager using SSL will use the `-lbs port` and all Load Balancers that want to communicate non-SSL with the Domain Manager will use the `-lbp port`. Currently, Load Balancers from Cisco Systems, Inc. and Nortel Networks only support the non-SSL port.

Note: The Load Balancer has to support at least one of the two port options for communicating with the Domain Manager. It is up to the Load Balancer to decide which port option will be used or if both port options will be used (secure and public) for different applications.

Load Balancer

Today, without EWLM, a Load Balancer routes work to multiple servers (or applications). The Load Balancer would define a server group that would include all of the servers identified by their real server IP address. The Load Balancer would also define some type of virtual IP address that is associated with the server group. Requests come in to the Load Balancer via the virtual IP address. Based on some type of predetermined algorithm, the Load Balancer selects an individual server from the server group that the request should get routed to.

To exploit EWLM, configuration changes are needed in the Load Balancer to accept EWLM's weight recommendations and to modify its algorithm, or create a new algorithm, based on these recommendations. This interaction may be a new addition for some vendors, requiring a firmware upgrade. For communications, it needs to define the IP address (ma address) and port number (the lbp port or lbs port) of the Domain Manager.

The Load Balancer needs to be able to communicate with the Domain Manager using the SASP protocol. The way in which this is configured is vendor-specific and sometimes version-specific. In general, the Load Balancer has a way in which the connection criteria for the domain manager can be given and identified as the way to get dynamic weights for a particular group.

The current implementation from Cisco System, Inc. does this by creating a special DFP agent that supports the SASP protocol. This SASP agent is configured with the domain manager's IP address and port information and is identified by a special BIND ID. This BIND ID is also assigned to the server farm to identify the agent to be used to get dynamic weights for this group.

7.2.2 Interfaces

This section describes the types of interfaces between the different elements.

Load Balancer to Domain Manager (SASP communication)

The Load Balancer connects to the Domain Manager to:

- ▶ Register groups and instances within a group.
- ▶ Deregister groups and instances.
- ▶ Quiesce instances.
- ▶ Set the Load Balancer configuration parameters to:
 - Allow instances to register/deregister themselves by setting a trust flag.
 - Define the way weights are sent (pushed or pulled from the Domain Manager).
- ▶ Get (or receive) the current weights from the Domain Manager.

Load balanced instances to Domain Manager (SASP communication)

A load balanced instance connects to the Domain Manager to:

- ▶ Register itself (provided the trust flag has been set by the Load Balancer)
- ▶ Deregister itself (provided the trust flag has been set)
- ▶ Quiesce itself

Domain manager to load balanced instances (Platform APIs)

A Domain Manager can communicate to the load balanced instances to:

- ▶ Obtain all the IP addresses that are defined to the platform.
- ▶ Obtain the process ID (PID) of the application the Load Balancer was registering.

7.2.3 Load balancing algorithm

In system-level balancing, either the Load Balancer or a member application can register the instance. When the Managed Server comes up, it connects to the EWLM Domain Manager. Statistical data at the server level is obtained for each instance of the server group. This data is analyzed and weighted (prioritized) by the Domain Manager. The weights can then be passed to the Load Balancer to help achieve a better distribution of the work. If application-level balancing is desired, the Load Balancer must register the applications. The Domain Manager obtains statistical data at the application-level and calculates the weights of the applications in the group. EWLM dynamically tracks the topology of the work and how

much time is spent at each member or instance, as well as how much delay occurs at each member due to a bottleneck at the next hop. EWLM can help the Load Balancer to determine the best server or application instance to send the work to by recommending a higher weight for one instance over another instance.

The following is a step-by-step flow of the identification process done at the system level and at the application level.

System-level identification

The flow illustrated in Figure 7-7 shows how system-level identification is performed. The identification mechanism organizes the appropriate data by all possible identification characteristics in the Domain Manager so that statistical data can be obtained based on the Load Balancer's registered identification.

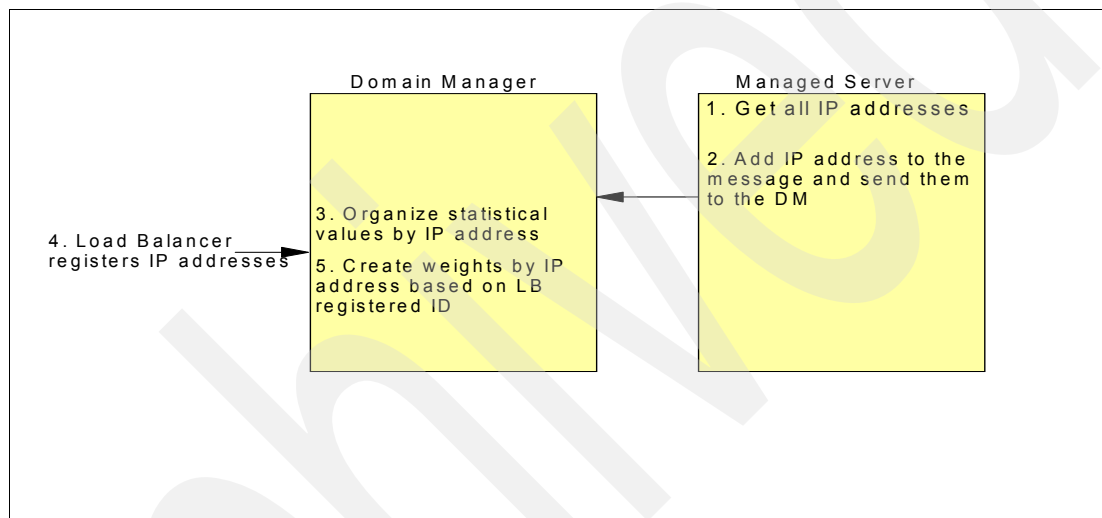


Figure 7-7 System-level identification

The process is:

1. The Managed Server determines all the IP addresses associated with the platform. This occurs every 30 seconds.
2. The IP addresses are sent from the managed server to the domain manager.
3. The Domain Manager organizes statistical values by the IP addresses passed by the Managed Server in the message.
4. The Load Balancer registers IP addresses representing system instances with the Domain Manager.

Note: The Load Balancer can register the system instances at any time. However, if the Load Balancer registers an instance before this time (step 5), then a NO_CONTACT flag is set. When seeking new information to calculate the next iteration of weights, the Domain Manager will try to resolve this instance again.

5. The Domain Manager determines the relative weights for the Load Balancer registered IP address using the statistical data collected from the managed server.

7.2.4 Application-level identification

The flow illustrated in Figure 7-8 shows the mechanism for application-level identification. The application-identification mechanism organizes similar data by the corresponding application criteria registered by the Load Balancer.

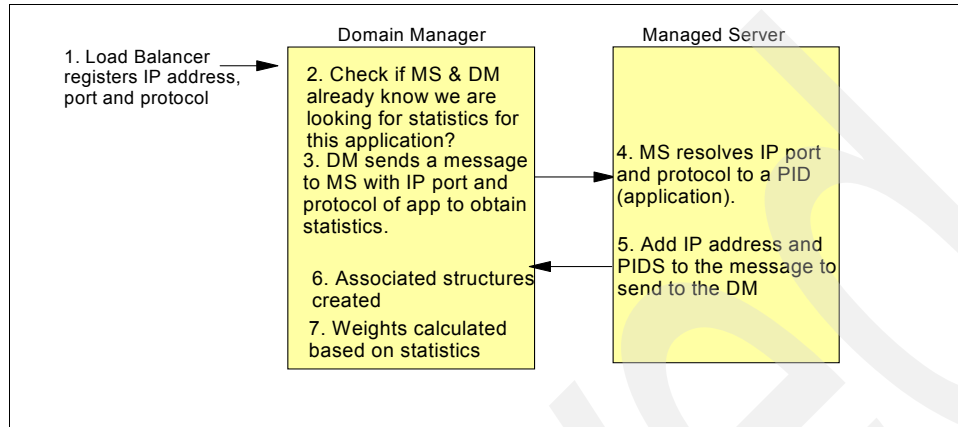


Figure 7-8 Application-level identification

The process is:

1. The Load Balancer registers the IP address, port, and protocol representing applications with the Domain Manager.
2. The Domain Manager determines if it knows we are already looking for statistics for this application. If the Domain Manager does not know, the server ID of the system the application is running on will be obtained from the information that was sent from the previous system-level identification. The server ID identifies which Managed Server to send the message to in step 3. If the result of this check reveals that no corresponding server is registered, the contact flag for this instance is set to `NO_CONTACT`. When seeking new information to calculate the next iteration of weights, the Domain Manager will attempt to resolve this instance again.
3. The Domain Manager sends a message to the Managed Server with the port and protocol of the applications we want application-level statistics from.

Note: All applications we are interested in tracking on the receiving system are included in this message. It will replace the list kept in the managed server.

4. The Managed Server takes the port and protocol we want to resolve and sets a particular PID. Now, the Managed Server will not only periodically determine all the IP addresses associated with the machine (as described in the “System-level identification” on page 204), it will also get the PIDs associated with previously set ports and protocols.
5. All IP addresses and IP/port/protocol combinations with their corresponding PIDs (if they have been determined) for the interested applications are periodically sent to the Domain Manager.
6. Structures are created in the Domain Manager to associate IP, port, and protocol to the application’s corresponding statistics.
7. Periodically, the statistics are retrieved containing the Load Balancer-registered instance identification and the weights calculated.

7.2.5 Where to send the work (weight calculations)

This section describes the process in which weights are computed for the servers and applications registered with the Domain Manager. A Load Balancer may register specific applications or specific systems in the groups. Because different data is used to compute weights in these cases, the current assumption is that instances in one group must be either all system instances or all application instances.

As mentioned earlier, the domain manager is capable of two different types of load balancing. If the Load Balancer registers system-level members, the domain manager will perform system-level load balancing. If the Load Balancer registers application-level members, the domain manager will attempt to perform ARM-instrumented application load balancing. However, if any of the applications registered in the group are not ARM-instrumented, the entire group will revert to system load balancing.

Initially, all instances in a group are assumed to be equivalent as far as available resources, and start out with an equal number of credits to be used for weights. To maintain stability in the weight generation algorithm, a finite number of *weight credits* will be used. An instance's weight is nothing but an accumulation of its weight credits. If one instance of the group is awarded an extra credit, it must be taken from another instance. Likewise, if an instance is punished and fined a credit, it must be given to another.

System-level balancing

The system-level balancing algorithm is best understood in two stages: The absolute stage and the relative adjustment stage. The absolute stage has the highest priority and factors in elements such as when instances are determined to be down or quiesced. The behavior of the weight algorithm in these cases is predetermined, and does not depend on the condition of any other systems. System-level balancing in the absolute stage occurs when:

- ▶ Instance is down: Set weight = 0, and contact flag = 0 → will receive no credits.
This may occur when the Domain Manager has missed the last several statistical update messages or if the managed server has not yet connected to the domain manager.
- ▶ Instance is quiesced: Set weight = 0, and quiesced flag = 1 → will receive no credits.
By setting the quiesced flag, the system is excluded from the group member weight calculations.

Statistical metrics gathered about each group instance are used to form a preference of using one instance relative to the other instances. The statistical data available to us at the system level are:

- ▶ System CPU utilization: Total number of CPUs and the amount of CPU time used.
Utilization is currently determined by:
$$\text{CPU Utilization} = \text{Total CPU Time used} / (\text{CPU Count} * \text{Time Interval})$$
- ▶ Importance level of work running on systems.
This is specified in the service policy for each service class.
- ▶ Performance index (PI) of work running on systems.
The PI is shown on the EWLM Control Center if you view the performance data of the service classes under the Monitor section.

In system-level balancing, our goal is to favor the systems with the most capacity for handling new requests, systems with histories of meeting their goals, and systems doing the least important work.

System-level balancing algorithm

The actual algorithm starts with all of the machines weighted equally and with an initial capacity model. At each interval, credits will be distributed as weights in a group according to the following steps:

1. Reward under-utilized systems.

In this step, a normalized reward distribution is developed for these instances based on remaining capacity with the higher rewards for instances with the most remaining capacity.

2. Reward systems with better histories of meeting goals.

This step will develop a reward distribution for the group instances weighted towards those with the best history of meeting goals.

3. Reward systems doing the least important work.

This step will develop a reward distribution for the group instances weighted towards those doing the most amounts of the least important work.

Once steps 1 through 3 are complete, the three normalized reward distributions will be statistically combined to form a general reward distribution, and weights will be calculated and prioritized in the following order:

1. The highest weights should go to the most under-utilized systems.
2. Of these systems, preferences should be given to those that are meeting the goals of the majority of their work.
3. Of these systems, preferences should be given to those that are meeting goals of the most important work.
4. Of these systems, preferences should be given to those that are running the least important work.

Application-level balancing

Application-level balancing is similar to system-level balancing in that requests or transactions are routed to applications that are most likely to successfully complete the transaction in a timely manner while getting the best use out of the systems. However, in application load balancing, there is more information about the application and the transactions being sent there. Once again, if any of the applications in the group are not ARM instrumented, balancing will default to system level. The current data available at the application level is the following:

- ▶ Number of successful transactions
- ▶ Performance index (PI) ratios
- ▶ Application response times
- ▶ Application topology
- ▶ Importance of transactions being processed
- ▶ Time the application is blocked waiting for resources
- ▶ Resource consumption data

Application load balancing can also be separated into the same stages as the system-level balancing stages, with the addition of one additional stage known as the failure recovery stage, which allows us to slowly ramp up the volume of requests to very troubled machines. Once again, the absolute stage has the highest priority. In addition to the fields used in the absolute stage of system-level balancing, we also consider whether an application is listening on the port mentioned. If there is no process listening to that port, then we should not send work there.

No PID listening to Application port - Return weight of 0 → will receive no credits.

Application-level balancing algorithm

The steps involved in application-level balancing are similar to those of the system level; however, there is more detailed information that can be obtained and analyzed at a more granular level. These steps include:

1. Reward under-utilized systems.

In this step, a normalized reward distribution will be developed for these instances based on remaining capacity, with higher rewards for instances with the most remaining capacity.

2. Reward success.

This step rewards applications that have had successful transactions. To do this, a Success Factor (SF) is created that measures the success rate:

$$SF = 1 - (\text{Failed Transaction Count} / \text{Total Transaction Count})$$

Note: If the SF is too low, the instance is placed in sick mode. In sick mode, the instance's credits are reduced to 1, thus reducing the number of transactions that will get routed there. One credit for each successful transaction completed will be added until a recovery threshold has been hit. Once passed, the instance will start a *capacity learning period* and eventually fully participate again.

3. Reward systems with better histories of meeting goals.

This step will develop a reward distribution for the group instances weighted towards those with the best history of meeting goals.

4. Reward applications that were not blocked.

This step develops a reward distribution for the group instances weighted towards those spending the least percentage of their time blocked.

5. Reward applications and systems doing the least important work.

This step will develop a reward distribution for the group instances weighted towards those doing the most amounts of the least important work. Two statistical reward distributions are developed (system-level importance and application-level importance) because the application-level member may process work at one importance level, while the whole system may be more diverse. Note that the system-level importance is the same as was used in step 3 in "System-level balancing algorithm" on page 207.

Once steps 1 through 5 have been completed, the normalized reward distributions will be statistically combined to form the general reward distribution.

Recall that a few applications may be given values in this distribution that do not fit their statistics:

- ▶ If an application is in failure recovery mode, its distribution values will be determined by the stage of the failure recovery it is in.
- ▶ If an application has been determined to be down, quiesced, or otherwise not listening on the port given in the absolute phase, it will have a value of zero.

At the end of each interval, the weights will be calculated using the general reward distribution, and prioritized in the following order:

- ▶ The highest weights should be given to under-utilized systems.
- ▶ Of these systems, preference should be given to those that have been very successful in completing transactions.

- ▶ Of these systems, preference should be given to applications that meet their goals for the highest percentage of their work.
- ▶ Of these systems, preferences should be given to applications that meet their goals for their most important work.
- ▶ Of these systems, preference should be given to those that are blocked the least.
- ▶ Of these systems, preference should be given to those that are running the least important work.

7.2.6 SSL implementation

The domain manager will provide a choice to the administrator to select the security modes it would like to function in:

- ▶ Non-encrypted - This mode uses standard TCP/IP socket communication.
- ▶ Authentication with encryption - This mode uses an SSL TCP/IP socket with mutual client-server authentication.

In EWLM's load balancing authentication with encryption mode, the Load Balancer and the Domain Manager must develop trust for each other without the help of a third-party component. To establish the authentication credentials for both the Load Balancer and the Domain Manager, an administrator would have to do the following:

1. Create public/private key combinations for the Domain Manager and the Load Balancer. This step should create a keystore for the domain manager holding its public/private keys, as well as a keystore for the Load Balancer holding its public/private keys.
2. Export the public certificate from the Load Balancer's keystore and import it into the keystore of the Domain Manager with trust.
3. Export the public certificate from the Domain Manager's keystore and import it into the keystore of the Load Balancer with trust.
4. Load the domain manager's keystore into the Domain Manager during configuration and setup.

This requires the addition of the following parameters to the EWLM configuration process:

- ▶ Non-encrypted port (-lbp <port>)
- ▶ SSL Port (authentication with encryption, -lbs <port>)
- ▶ Both ports (-lbp <port> -lbs <port>)

If option b or c is selected, the domain manager's keystore (-sslks <keystore path>) and the keystore password (-sslpw <keystore password>) are needed. These options can be specified as configuration parameters on the **changeDM** command.

7.2.7 Monitoring the routing environment

Although most of the monitoring is done from the Load Balancer, there are commands available from the Managed Server and the Domain Manager to verify load balancing can occur at the application level.

Load Balancer

The Load Balancer should provide the following:

- ▶ The ability to display the server groups that have been defined to it.
- ▶ Commands to identify all of the instances of a server group by the real IP address.

- ▶ A command to list the virtual IP address associated with the server group.
- ▶ A command or display to show which instances are actively receiving work.
- ▶ Commands to display what the current weights are, as well as the default weights (if provided by the Load Balancer), for each instance of the server group.
- ▶ A command to display how many requests have been forwarded to which instance and, of those requests, how many have ended.
- ▶ The ability to *ping* the Domain Manager to verify that communication can occur.

Managed server

To list the applications registered to ARM on an AIX:

```
/usr/sbin/larm -a
```

To see if the netwlm daemon is started on AIX:

```
ps -ef | netwlm
```

To determine if the common code and the netwlm code were installed:

```
AIX - smitty ewlm
```

To display the current properties of the Managed Server:

```
displayMS commands
```

Determining how many transactions a particular system or application has processed is application specific. Each application may have tools or commands to show the number of transactions that have been processed. For the HTTP Server, there are logs that can be manually processed via a script to determine how many transactions were processed and how many have failed:

- ▶ access.log
- ▶ plug-in.log
- ▶ error.log

Domain manager

From the EWLM Control Center, there are various screens to show how work is performing. From the left pane, under Monitor, there are several options available to see the details of either the service class or the transaction class that will show how many transactions completed successfully, how many failed, and how many have stopped.

You can also see the current performance index (PI) of a service class to see if the goals are being met.

For process information, you can see the system-level information by looking in the Process class or the Managed servers detail screen. See 6.2, “First-level reports” on page 160.

Commands

The `displayDM` command displays current properties of the Domain Manager.

Use `netstat` to see if the port that was identified for communication to the Load Balancer is in use.



The ITSO EWLM environment

In this chapter we provide information about the Web application Trade6, which we use throughout the book. We include information and steps on the assessment, planning, and setup of the ITSO Domain Policy.

This chapter provides an introduction of the ITSO environment. We guide you through the assessment of our EWLM domain and discuss some considerations for designing and building the ITSO Domain Policy. This policy serves as a basis for future expansion for different test scenarios we run and explain in Chapter 9, “Scenarios” on page 223.

8.1 Running Trade6 in the ITSO environment

Trade6 is the fourth generation of the WebSphere end-to-end benchmark and performance sample application. It is designed and developed to cover the significantly expanding programming model and performance technologies associated with WebSphere Application Server. Trade6 provides a real-world workload and demonstrates several new features in WebSphere Application Server V6. All Trade-versions are WebSphere-version dependent, which means Trade6 only works with WebSphere Application Server V6.

The Trade6 installation package can be downloaded from the following Web site:

<http://www.ibm.com/software/webservers/appserv/was/performance.html>

More detailed information about Trade6 as well as descriptions of the installation, setup, and configuration can be found in Chapter 8 of the IBM Redbook *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

8.2 ITSO Trade6 base environment

Most customer environments and configurations are more complex than the ITSO base environment. The setup we work with is fairly simple, but it is still sufficient to give you guidance through the important planning and assessment steps and to demonstrate the main functionalities of EWLM V2.1. In particular, if this is your first experience with EWLM, we recommend that you allow plenty of time for this preparation and planning phase. You need to carefully collect all information needed to build the EWLM policy to ensure a successful usage of the workload capabilities of EWLM.

8.2.1 Assessment of ITSO environment

Before we can build our EWLM Domain Policy we need to assess the ITSO environment. The design of the Domain Policy really depends on the number of platforms and applications, the type of workload we are running, and the Service Level Agreements we want to achieve.

8.2.2 Identifying the EWLM boundaries

The first step is to define the EWLM Management Domain and its boundaries. The ITSO Management Domain consists of one domain manager running on Windows and five managed servers, of which three are running on p5 partitions with AIX and two are running on xSeries servers with Windows. The topology of the ITSO environment is shown in Figure 8-1. It provides information about the host name, the operating system, and the middleware applications that are running on each server.

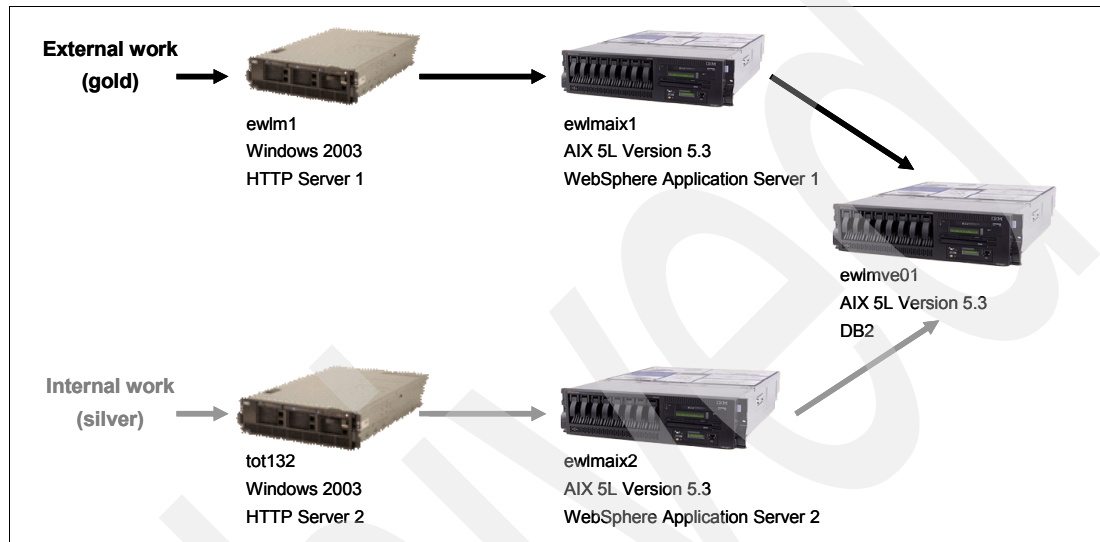


Figure 8-1 Topology of ITSO environment

In this topology view of the ITSO environment we also show the two streams of workload, that is, gold and silver. Gold is the external Trade6 workload that is classified to be of higher importance than the silver stream, the internal Trade6 workload.

As shown in the figure, both streams run through different Web and application server instances but share the same DB2 database. In the base scenario, all applications (that is, the Webserving plugin, WebSphere Application Server, and DB2) are ARM instrumented. This means that the processing segments performed by each participating middleware instance are correlated. EWLM can use the information to create topologies and to construct an end-to-end view of each type of business transaction.

We just identified all servers and middleware in the ITSO environment. The next step is to make an inventory of applications and client programs that we want to monitor and manage using EWLM. For the ITSO environment we identified the Trade6 Web application. For more complex environments we recommend a table listing all applications and their related middleware applications and platforms.

8.2.3 Identifying the edge server

It is not an easy task to identify all transactions in a distributed environment even when the EWLM Management Domain consists of only a few servers such as in our ITSO environment. The easiest approach for obtaining this information is to interview the application developers. When detailed information concerning the transactions within the domain is only partially available, an alternative approach is demonstrated in the following paragraphs.

Before we can start creating our Domain Policy and any EWLM classes, we need to identify the edge server of our work requests. This is necessary because all work requests get classified to EWLM service classes by its entry point, the edge application.

With every installation of EWLM there are by default two domain policies as samples that you can start with. They are called *Sample Domain Policy* and *Sample Banking Domain Policy*. You need to deploy one of these policies. Open the EWLM Control Center, log in, go to **Set up** → **Domain policies** and deploy one of the two policies. For the ITSO environment we have deployed the *Sample Banking Domain Policy*. Go to the Transaction Class view and you see some transaction classes are already defined. At this point, we are only interested in the four default transaction classes for each application:

- ▶ Default - IBM DB2 Universal Database
- ▶ Default - IBM Web serving plug-in
- ▶ Default - WebSphere Application Server 6.0.2 and later
- ▶ Default - WebSphere Application Server 5.0 through 6.0.1

These transaction classes have the same classification filter, (^*) Equal (^*), which matches with all work requests that do not match with any other transaction class filters. The difference between these three types of transaction classes is the entry application. This means that if the edge server is DB2, then transactions get classified in the default DB2 transaction class. We see response times and entry application information for this class and similarly for the others.

We generate work requests to determine what the edge servers are for the Trade6 Web application that runs in the ITSO environment. We expect to see a classification in either of the four default transaction classes because the classification of the other example transaction classes will most likely not fit our environmental needs. This is shown in Figure 8-2 and confirms that the edge server for the Trade6 Web application is the Webserving plugin. We see a response time value as well as an entry in the Entry application field when viewing the Transaction classes.

Select	Transaction class ▲	Response time ^	Entry application ^
<input type="radio"/>	Account Query	None	None
<input type="radio"/>	Default - IBM DB2 Universal Database	00.000	IBM DB2 Universal Database
<input checked="" type="radio"/>	Default - IBM Web serving plug-in	00.359 average	IBM Webserving Plugin
<input type="radio"/>	Default - WebSphere Application Server 5.0 through 6.0.1.	None	None
<input type="radio"/>	Default - WebSphere Application Server 6.0.2 and later.	None	None
<input type="radio"/>	EWLM Default Application Transaction Class	None	IBM DB2 Universal Database
<input type="radio"/>	NorthEast	None	None
<input type="radio"/>	NorthWest	None	None
<input type="radio"/>	Stock Query	None	

Figure 8-2 Identifying the edge server for the ITSO environment

When you create your own policy you see that in addition to the user-defined transaction classes there are also predefined default transaction classes. These different transaction classes are helpful for verifying that your transactions are being correctly classified. When you run your workload and you see an entry for response time in the default transaction classes you may need to check your classification. If you do not get an entry, your classification configuration is working well.

In addition to checking the values for response time and entry application in the transaction class view we can also look at the application topology for the transaction class. This view is automatically generated by EWLM when there are transactions flowing through the domain. In our example we see transactions getting classified to the default, the IBM Web serving

plug-in transaction class. We run the application topology on this class, which is shown in Figure 8-3. It confirms that Trade6 is running through a 3-tier environment. The transactions enter the domain at the Webserving plugin and are forwarded to the WebSphere Application Server and subsequently to the DB2.



Figure 8-3 Application topology view of ITSO environment

You will then need to follow similar steps for all other business applications you want to include in your EWLM domain individually and make a note of the entry application, which is the left-most application, as shown in the topology for each application.

The ITSO environment consists of two workload streams, *gold* and *silver*, both running the same Trade6 transactions and both entering the domain at the Webserving plugin. See Figure 8-3. As you can see in Figure 8-1 on page 213, we have two different physical servers, ewlm1 and tot132, each configured as an IBM HTTP server and containing a Webserving plugin. To identify our entry applications we first ran the *gold* and *silver* workloads separately and observed that the resulting application topology for both the *gold* and *silver* workloads displayed the Webserving plugin as the entry application. One way to differentiate between these two workload streams is to observe the resulting server topology. The server topology when only the *gold* stream is running is shown in Figure 8-4.

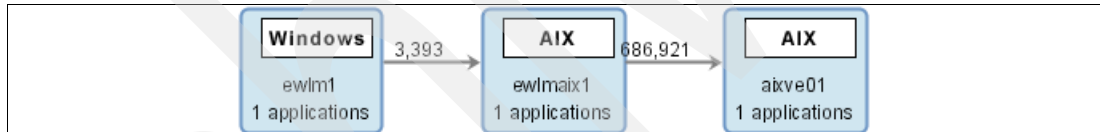


Figure 8-4 Server topology of the gold workload stream

The server topology when only the *silver* stream is running looks very similar to the *gold* stream; however, both the edge server (tot132) and the WAS server (ewlmaix2) are different servers with different host names. This is shown in Figure 8-5. For the ITSO Domain Policy we need to record the two different host names of the edge servers, where the Webserving plugin is our entry application for both workload streams. We then use the combination of these two values when we define our two service classes for the *gold* and *silver* workload stream.

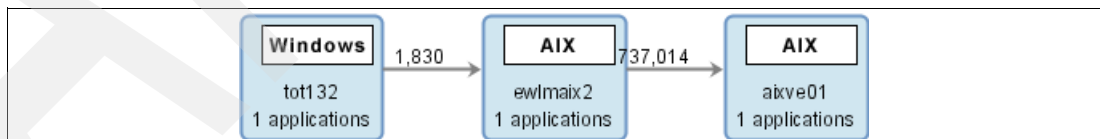


Figure 8-5 Server topology of the silver workload stream

8.2.4 Identifying transactions

We identified our edge server for the applications we want to monitor and manage using EWLM, that is, the *gold* and the *silver* workload stream both running Trade6 transactions, which we now have to identify. At this point you need to think carefully how detailed you want to monitor the business transactions. In theory, you can look at the HTTP server access log and classify each individual transaction into a transaction class. Classifying transactions this

way for our Trade6 environment means that we can easily define up to 10 transaction classes.

For the ITSO environment we assigned a very simple classification, which serves the purpose of monitoring and managing two workload streams of different importance. In the following paragraphs we demonstrate how we created a Domain Policy for our ITSO environment and for completeness describe a more complex environment with different filtering options in 8.4, “Example of more complex Trade6 classification” on page 218.

In the ITSO environment we define two transaction classes—one for each of the work streams *gold* and *silver*, which are both associated with the Webserving plugin as entry applications. We defined for both a nested **AND** filter including the host name of the edge server and the URI of Trade6. This is shown below and looks similar for both transaction types with the only difference in *Hostname*.

- ▶ Filter value for transactions collecting the *gold* work stream:

EWLM URI = /trade(*) **AND** *EWLM Hostname* = ewlm1

- ▶ Filter value for transactions collecting the *silver* work stream:

EWLM URI = /trade(*) **AND** *EWLM Hostname* = tot132

We map these two transaction classes to two different service classes, which represent the importance and the goal for the *gold* and *silver* workload streams. The *gold* stream represents external Web requests, which have a higher importance than the *silver* stream representing internal Web requests. Both service classes have the same response time goal. We expect the transactions flowing through the 3-tier architecture to each finish on average within 1.5 seconds.

8.3 ITSO Domain Policy

We have now collected all of the necessary information and made the decisions about the structure and complexity of our policy. We can start thinking about the naming convention and make a list of all definitions we want to assign. The next step is to go to the EWLM Control Center and to define and deploy our Domain Policy.

All EWLM policy components such as transaction classes, process classes, partition classes, service classes, workloads, and service policies require names that are in accordance with the EWLM naming requirements. For a complete description of the rules for naming, we refer you to:

<http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/ewlminfo/eicaakickoff.htm>

The naming convention for the ITSO environment looks as shown Figure 8-6.

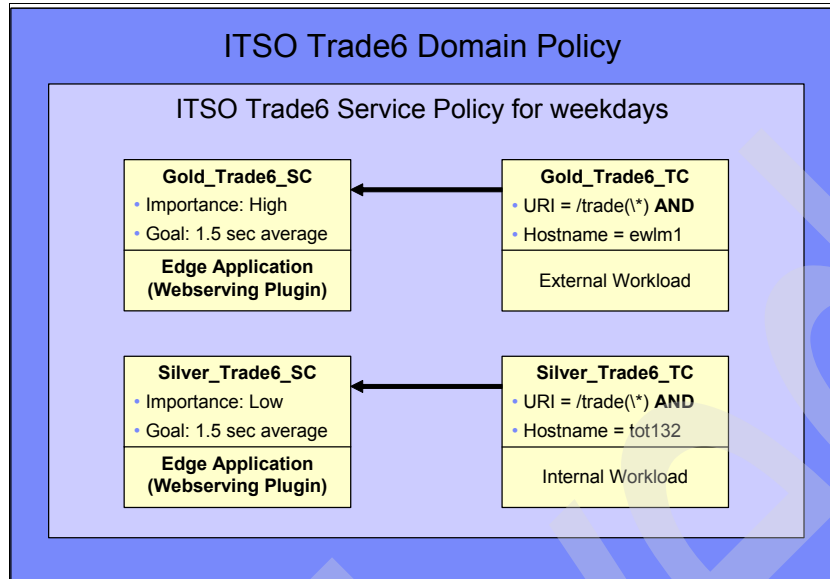


Figure 8-6 ITSO Domain Policy

For the ITSO base scenario we have no platforms, process class, partition class, or workload defined. In 9.3, “Partition class for non-ARM instrumented application” on page 230, we expand this base policy and include most of the above components. Depending on the complexity of your environment and the EWLM objectives you wish to achieve, you may find using all of the available policy components useful or, as in our case, just a small set of policy components is sufficient. For every policy though you must define at least a Domain Policy, a service policy, and a service class. If you want to monitor and manage a transactional type of work you also need to specify applications and transaction classes. If you want to monitor and manage a process type of work or include any non-ARM-instrumented applications, you need to define platforms and process or partition classes. The workload component is an optional feature that allows you to collect specific service classes and group them together for reporting purposes.

8.3.1 Partition settings for the ITSO environment

The ITSO p5-550 environment consists of a Virtual I/O Server assigned one dedicated processor and three partitions belonging to one EWLM workload group using a shared pool of three processors. The partition settings for the three shared processor partitions are as follows:

- ▶ Processing units:
 - Minimum: 0.1
 - Desired: 0.9
 - Maximum: 3
- ▶ Virtual processors:
 - Minimum: 1
 - Desired: 1
 - Maximum: 3

- ▶ Further settings:
 - Processing mode: Shared
 - Partition mode: Uncapped
 - Weight: 128

The minimum and maximum settings can only be changed when rebooting the partitions. The values that can be manually changed or that EWLM can adjust in order to achieve the performance goals are:

- ▶ Desired processing units
- ▶ Desired virtual processors
- ▶ Weight

As discussed in the previous sections, EWLM shifts resources between partitions belonging to the same workload group, but does not exceed the sum of entitlements that was originally assigned to the partitions belonging to the workload group. In the ITSO environment we have three partitions in one workload group with 0.9 processing units assigned, giving a total of 2.7 entitlements. This means that even though we have three processors in the free pool we can use these processing units only through the hypervisor but not with the entitlement settings we chose. This means that for the ITSO environment we could have used 1.0 entitlement for each partition, but because we wanted to demonstrate that EWLM will not exceed the sum of entitlements, we chose a setting of 0.9 entitled capacity each. After EWLM has shifted entitled processing resources to help achieve the service class goals, the total entitled capacity of the workload group is still 2.7 processing units. This is shown in Figure 8-7.

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^
EWLM_AIX_DB2	128	3	3	48.21%	0.90	No	0.10	3.00
EWLM_AIX_WAS1	153	3	3	32.34%	1.08	No	0.10	3.00
EWLM_AIX_WAS2	102	3	3	16.3%	0.72	No	0.10	3.00

Figure 8-7 Example of EWLM entitlement shifting

8.4 Example of more complex Trade6 classification

In 8.3, “ITSO Domain Policy” on page 216, we built the ITSO Domain Policy, which is a fairly simple setup. In this section we want to discuss a more complex classification of Trade6 to give you an example of how much more detailed you can monitor and manage transactions if it makes sense in your environment. The example discusses Trade6 transactions but applies similarly to other Web applications.

8.4.1 Finding classification values for transactions

Web applications provide business functions to customers by processing the HTTP requests received from a Web browser, then invoking servlets or JSPs, and finally returning the results to the Web browser. For example, when the client sends an HTTP request, the application server forwards it to an appropriate servlet using a requested URI, which might be as follows:

```
/context-root/servlet-name?key1=value&key2=value2
```

Context root (context-root) is the root part of the URI and is unique among all Web applications within the same application server cell or application server. Servlet name (servlet-name) or query string (key1 or key2) often represents which business function is to be invoked in the Web application specified by the context root. These can be found in the IBM HTTP Server access log and can be used to identify and classify transactions.

For Trade6 on Windows the log file is located at C:\Program Files\IBM HTTP Server\logs and is called access.log. We see in Example 8-1 that Trade6 uses the servlet name (/trade) and action parameters like *portfolio*, *buy*, *account*, etc. to determine which business function is to be invoked. These seem to be good differentiators for our transactions.

Example 8-1 IBM HTTP Server access log file

```

9.12.4.141 - - [17/Oct/2005:11:10:18 -0400] "GET /trade/app?action=quotes&symbols=s:0,...
9.12.4.141 - - [17/Oct/2005:11:10:20 -0400] "GET /trade/app?action=buy&symbol=s%3A1&quan...
9.12.4.141 - - [17/Oct/2005:11:10:24 -0400] "GET /trade/app?action=portfolio HTTP/1.1" 2...
9.12.4.141 - - [17/Oct/2005:11:10:25 -0400] "GET /trade/app?action=account HTTP/1.1" 200...
9.12.4.141 - - [17/Oct/2005:11:10:27 -0400] "GET /trade/app?action=home HTTP/1.1" 200 13122
9.12.4.141 - - [17/Oct/2005:11:10:32 -0400] "GET /trade/app?action=logout HTTP/1.1" 200...
9.12.4.141 - - [17/Oct/2005:11:10:37 -0400] "GET /trade/register.jsp HTTP/1.1" 200 4115

```

8.4.2 Defining transaction classes

For the ITSO base scenario we only defined our transaction classes at a generic level, that is, two transaction classes for the two workload streams. In this section we want to further break down the Trade6 transactions and define a transaction class for each transaction we run. The servlet name is the same for all and is specified as /trade. The action parameter is different for each transaction, which we list in Table 8-1.

Table 8-1 Definition of transaction classes

Position	Transaction class name	Filter type	Filter operation	Filter value
1	TC_Trade6_buy	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	action=buy(*)
2	TC_Trade6_sell	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	action=sell(*)
3	TC_Trade6_account	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	action=account(*)
4	TC_Trade6_register	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	action=register(*)
5	TC_Trade6_update_profile	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	action=update_profile(*)
6	TC_Trade6_home	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	action=home(*)
7	TC_Trade6_logout	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	action=logout(*)
8	TC_Trade6_portfolio	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	action=portfolio(*)
9	TC_Trade6_quotes	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	action=quotes(*)

Position	Transaction class name	Filter type	Filter operation	Filter value
10	TC_Trade6_general	EWLM:URI	Equal	/trade(*)

Note that *TC_Trade6_general* does not correspond to any *action* parameter of the query string. This transaction class is used for work requests that correspond to requests for static content such as HTML pages or images. If, for example, the *home* transaction does not need to be monitored by EWLM, remove the definition of *TC_Trade6_home* and it will be classified to *TC_Trade6_general*. The advantage of having a general Trade6 transaction class is that for this class we can set a velocity or response time goal. Without this class these transactions would fall into the default Webseving plugin transaction class, which has a discretionary goal.

This is one reason why we need a nested classification filter with the servlet **AND** action parameter. Another reason is that if we had other applications running like Plants for WebSphere, for example, which have the same action parameters like login, we need nested classification to differentiate between the login transaction of Trade6 and Plants for WebSphere. An example for *TC_Trade6_buy* is shown in Figure 8-8.

Application name: IBM Webseving Plugin Default service class: EWLM Service Class

*Name: TC_Trade6_buy

Description: Buy stocks

*Position: (Enter a number from 1 to 10) 1 Service class: Silver

Rules

(EWLM:URI == "/trade(*)" **AND** QueryString == "action=buy(*)")

Figure 8-8 Example of nested classification for Trade6 transaction class

8.4.3 Verification of transactional classification and goal settings

In our example with Trade6 we do not see any response time value for the two transaction classes *TC_Trade6_register* and *TC_Trade6_update_profile*. No work request is classified to these transaction classes, although there are work requests to *register* and *update_profile* action. This is due to wrong classification filters. We found that the query string of these work requests is a bit different than we originally thought when we created the classification filter. We assumed that all query strings *begin* with *action=actionname* and defined filters value as *action=actionname(*)*, as shown in Table 8-1 on page 219. However, query strings for *register* and *update_profile* actions *end* with *action=actionname*. Thus, transaction classes *TC_Trade6_register* and *TC_Trade6_update_profile* could not classify any work requests.

We need to fix the classification. The obvious method would be to use a wildcard (*) before the action=register value, as shown below. However, this is not possible because the wildcard can only be used at the end.

```
(\*)action=register
(\*)action=update_profile
```

The correct filter definition looks as shown in Table 8-2. We verify running our workload again and now we get classification for all user-defined transaction classes. This example shows that classification is not always straightforward and often it means changes in your classification a few times until it is all running smoothly.

Table 8-2 Revised definition of filters for transaction classes

Position	Transaction class name	Filter type	Filter operation	Filter value
4	TC_Trade6_register	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	Full+Name=(*)
5	TC_Trade6_update_profile	EWLM:URI	Equal	/trade(*)
		+ QueryString	Equal	userID=(*)

When all of your classification is working you should see response times for each user-defined transaction class and values of *None* for the EWLM default transaction classes. The next step is to make a note of the response times for all of the transactions classes. You then group them together and map them to service classes with the same or similar goals. Change your transaction to service class mapping accordingly and assign achievable and realistic goals for each service class based on either the monitored response times for the transactions or on your Service Level Agreements. Run your workload again and carefully monitor your service classes and their associated performance index (PI). Adjust your performance goals until you achieve your goals. If you now run EWLM in management mode it will automatically monitor how well your transactions are doing and react accordingly if goals are not being met. It shifts resources, that is, changes the weights, the capacity entitlements, and/or the number of virtual processors of your partitions if that will improve the performance of your service classes.

Archived

Scenarios

This chapter presents four scenarios based on the ITSO Trade6 Domain Policy, which we described in detail in Chapter 8, “The ITSO EWLM environment” on page 211. The first three scenarios describe POWER partition management functionalities of Enterprise Workload Manager V2.1. The fourth scenario includes z/OS hardware and discusses the EWLM and WLM monitoring functionalities for z/OS.

Additional scenarios that describe how EWLM can dynamically shift processing capacity to manage workloads using an eServer i5 system running i5/OS logical partitions can be found at:

http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/veicinfo/eicar_i5p5_scenario.htm

The scenarios include information and steps on the assessment, planning, and setup of an EWLM Domain Policy, as well as information about how EWLM performs partition management.

9.1 Improve performance of most important workload

We now describe the first of the three scenarios we used to demonstrate the POWER partition functionality of EWLM V2.1.

9.1.1 Objective and expectation

The objective of the first scenario is to analyze how EWLM partition management functions when we have an increase in workload for the *gold* workload stream. This is the workload stream that we classify as being of high importance.

We expect to see that when the performance index for the *gold* workload stream is getting closer to 1 and finally the *gold* service class is not meeting the goal, EWLM should shift resources from either the server running the *silver* WebSphere Application server and/or from the server running the *gold* and *silver* DB2.

9.1.2 Starting point

The starting point is our base setup of the ITSO scenario. We have the server topology as shown in Figure 9-1, where the *gold* workload stream runs from the HTTP server on Windows to WebSphere Application Server 1 on AIX to DB2 on AIX. The *silver* workload stream runs from a different HTTP server on Windows to WebSphere Application Server 2 on AIX and to the same DB2 on AIX as we use for the *gold* stream.

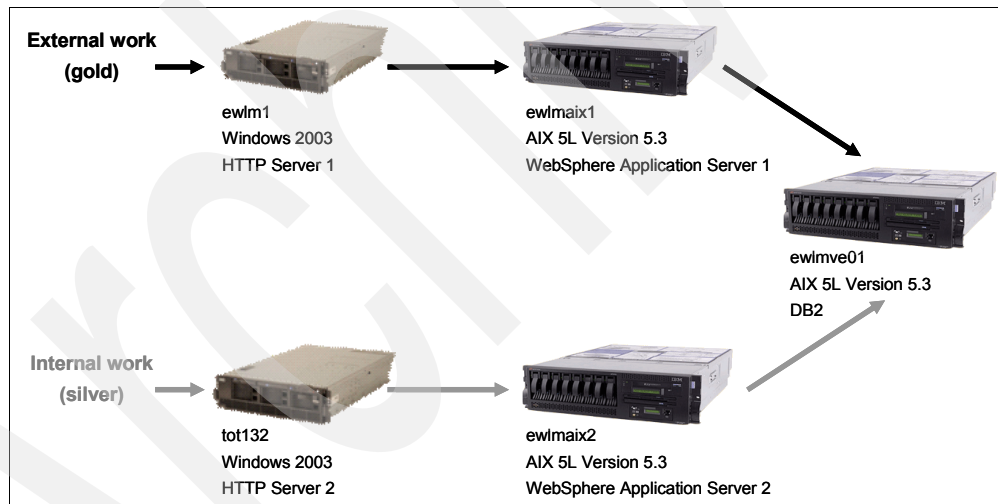


Figure 9-1 ITSO server and application topology

The *gold* stream is associated with the external and therefore more important workload. *Silver* is associated with internal and less important work. We therefore have two service classes called *Gold_Trade6_SC* and *Silver_Trade6_SC*. Both have the same average response time goal of 1.5 seconds but a different importance setting. The *gold* stream is classified as being of high importance and *silver* as being of low importance.

Our pSeries partition settings are explained in detail in Chapter 8, “The ITSO EWLM environment” on page 211. We start with having an *entitlement* of 0.9 for each partition with 1 *virtual processor* as the desired value and a *weight* of 128. This looks as shown in the Partition Workload Group Details view in Figure 9-2.

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^
EWLM_AIX_DB2	128	1	3	33.3%	0.90	No	0.10	3.00
EWLM_AIX_WAS	128	1	3	20.73%	0.90	No	0.10	3.00
EWLM_AIX_WAS2	128	1	3	14.13%	0.90	No	0.10	3.00

Figure 9-2 Partition settings - Starting point

We run EWLM in partition management mode with a stable workload for both the *gold* and the *silver* streams. The performance index for both service classes is less than 1, which means we are meeting the goals. This is shown in Figure 9-3.

Select	Service class ^	PI ^	Importance ^	Performance ^	Goal ^
<input type="radio"/>	Gold_Trade6_SC	0.92	High	01.375 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	0.49	Low	00.739 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Figure 9-3 Performance index for the gold and silver workload stream

As you can see, the performance index for our important class *Gold_Trade6_SC* is getting close to 1, but EWLM does not see a need to shift resources at this point. However, we see a change in the number of virtual processors for our three partitions, as shown in Figure 9-4. EWLM decides that increasing the number of virtual processors already helps to improve overall performance, and we see a slight improvement of the PI.

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^
EWLM_AIX_DB2	128	3	3	31.74%	0.90	No	0.10	3.00
EWLM_AIX_WAS1	128	3	3	13.34%	0.90	No	0.10	3.00
EWLM_AIX_WAS2	128	3	3	11.98%	0.90	No	0.10	3.00

Figure 9-4 Partition Workload Group Details after change of number of virtual processors

9.1.3 Running the test case

We have a stable environment set up now where both service classes reach their performance goal and EWLM has no reason to interfere in partition management. As long as there are enough resources in the free pool, all partitions can use more than their entitlement and the transactions are running smoothly.

Our next step is to increase the workload of our high importance *gold* transaction stream while keeping the *silver* workload running constant. As seen in Figure 9-5, the performance index for both classes increases. There is a very small increase in the PI for the low importance *silver* service class, but we are still meeting the goal without any problems. The reason is most likely that *gold* and *silver* are both using the same database server, which means an increase in load for either stream will also slightly affect the other stream.

Select	Service class ²	PI [^]	Importance ¹	Performance [^]	Goal [^]
<input type="radio"/>	Gold_Trade6_SC	1.03	High	01.552 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	0.55	Low	00.828 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Figure 9-5 Increased workload for gold stream leads to increased performance index

The important thing to notice though is that the PI for the high importance *gold* service class is greater than 1 and is therefore no longer achieving its goal. It is very close to 1, but we can already see EWLM reacting to not meeting the goal. This is shown in Figure 9-6. We see a small resource shift from WebSphere Application Server 2 being part of the *silver* stream to WebSphere Application Server 1, which is part of the *gold* stream.

Partition name [^]	Weight [^]	Virtual processors [^]	Physical processors [^]	Processor utilization [^]	Processing units [^]	Capped [^]	Minimum processing units [^]	Maximum processing units [^]
EWLM_AIX_DB2	128	3	3	48.11%	0.90	No	0.10	3.00
EWLM_AIX_WAS1	129	3	3	27.02%	0.91	No	0.10	3.00
EWLM_AIX_WAS2	125	3	3	20.48%	0.88	No	0.10	3.00

Figure 9-6 Capacity entitlement changes to help the gold service class meet its PI again

The workload for the *gold* workload is still very high. This means that the performance index for this service class keeps growing, as shown in Figure 9-7.

Select	Service class ²	PI [^]	Importance ¹	Performance [^]	Goal [^]
<input type="radio"/>	Gold_Trade6_SC	1.12	High	01.687 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	0.60	Low	00.898 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Figure 9-7 Performance index for the gold stream is still growing

This increase in performance index is sufficient to have EWLM react. It communicates with all managed servers running within the partition workload group to determine if shifting resources would help to get the PI for the important *gold* stream down to 1. It now shifts more and more resources from one WebSphere Application Server partition to the other, as shown in Figure 9-8.

Partition name [^]	Weight [^]	Virtual processors [^]	Physical processors [^]	Processor utilization [^]	Processing units [^]	Capped [^]	Minimum processing units [^]	Maximum processing units [^]
EWLM_AIX_DB2	128	3	3	48.21%	0.90	No	0.10	3.00
EWLM_AIX_WAS1	153	3	3	32.34%	1.08	No	0.10	3.00
EWLM_AIX_WAS2	102	3	3	16.3%	0.72	No	0.10	3.00

Figure 9-8 Changes in capacity entitlement to help the gold stream to meet performance goals

9.1.4 Results and considerations

For this test case we expected to see a resource shift from the WebSphere Application Server 2 partition running the *silver* workload and possibly from the DB2 partition running both workload streams to the WebSphere Application Server 1 partition with the *gold* stream.

We can see that the server running DB2 with both the *gold* and the *silver* streams has no change in entitled capacity. In order to achieve the performance goal for the high importance *gold* stream, EWLM decided it is sufficient to move entitled processing units from the WebSphere Application Server 2 running the *silver* stream to WebSphere Application Server 1 partition running the *gold* stream. For the *gold* WebSphere Application Server partition the entitled capacity increased from 0.9 to 1.08, while for the *silver* WebSphere Application Server partition the entitled capacity decreased from 0.9 to 0.72. With these settings we are now meeting the performance goal for both service classes.

In Chapter 8, “The ITSO EWLM environment” on page 211, we discussed the partition settings as a starting point for a stable workload environment where the performance goals for both partitions are being met. We explained that EWLM will only shift resources within the partition workload group and that the sum of the entitlements is held constant. In this case, the starting point for each partition was an entitlement of 0.9, which gives 2.7 as the sum for all three partitions. If we look at the numbers after the EWLM resource shift, the sum of entitlements is still 2.7.

9.2 Improve performance of less important workload

We now present the second of three scenarios to demonstrate the POWER partition functionality of EWLM V2.1.

9.2.1 Objective and expectation

The objective of this scenario is to analyze how EWLM partition management works when we have an increase in workload for the *silver* workload stream that is classified as being of low importance.

We expect to see that when the performance index for the *silver* workload stream is getting closer to 1 and finally the *silver* service class is not meeting the goal, EWLM should shift resources from either the server running the *gold* WebSphere Application server and/or from the server running the *gold* and *silver* DB2.

9.2.2 Starting point

As for test case one, the starting point is our base setup of the ITSO scenario. We have the server topology, as shown in Figure 9-1 on page 224, where the *gold* workload stream runs from the HTTP server on Windows to WebSphere Application Server 1 on AIX to DB2 on AIX. The *silver* workload stream runs from a different HTTP server on Windows to WebSphere Application Server 2 on AIX and to the same DB2 on AIX as was used for the *gold* stream.

The *silver* stream is associated with internal and therefore less important workload. We have two service classes called *Gold_Trade6_SC* and *Silver_Trade6_SC*. Both have the same average response time goal of 1.5 seconds but a different importance setting. The *gold* stream is classified as being of high importance and *silver* as being of low importance.

Our partition settings are explained in detail in Chapter 8, “The ITSO EWLM environment” on page 211. We start with having an *entitlement* of 0.9 for each partition with 1 *virtual processor* as the desired value and a *weight* of 128. Once we enable partition management in EWLM, the number of virtual processors changed to 3 for all partitions. We can see this in the Partition Workload Group Details view in Figure 9-9.

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^	Allocated memory ^
EWLM_AIX_DB2	128	3	3	0.62%	0.90	No	0.10	3.00	4,096 MB
EWLM_AIX_WAS1	128	3	3	0.57%	0.90	No	0.10	3.00	4,096 MB
EWLM_AIX_WAS2	128	3	3	0.46%	0.90	No	0.10	3.00	4,096 MB

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 9-9 Partition workload group starting point for scenario 2

9.2.3 Running the test case

We now start a consistent workload for our scenario until we see the environment stabilize. Since there are enough available resources in the free pool, the partitions are running well, as you can see in Figure 9-10.

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^
EWLM_AIX_DB2	128	3	3	48.64%	0.90	No	0.10	3.00
EWLM_AIX_WAS1	128	3	3	20.44%	0.90	No	0.10	3.00
EWLM_AIX_WAS2	128	3	3	30.58%	0.90	No	0.10	3.00

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 9-10 Partition detail for the steady workload

Also, the PI is good for all service classes, as you can see in Figure 9-11.

Select	Service class ^	PI ^	Importance ^	Performance ^	Goal ^
<input type="radio"/>	Gold_Trade6_SC	0.36	High	00.538 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	0.68	Low	01.026 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 9-11 PI for the steady workload

At this point we increase the load for the *Silver* stream to see how it effects the PI. Figure 9-12 shows the PI values for this increased workload.

Select	Service class ^	PI ^	Importance ^	Performance ^	Goal ^
<input type="radio"/>	Gold_Trade6_SC	0.61	High	00.914 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	1.17	Low	01.761 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 9-12 PI for increased workload on silver

As you can see, the PI is now above 1 for *silver* and increased a little for *gold*. This increase in PI for gold is probably due to both streams using the same database running in the DB2 partition. This increase in PI for the silver stream is sufficient to have EWLM react. It communicates with all managed servers running within the partition workload group to

determine if shifting resources would help to get the PI for the silver stream down to 1. Let us see how EWLM has shifted processing units. Figure 9-13 shows this shift of entitled processing units.

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^
EWLM_AIX_DB2	110	3	3	48.77%	0.78	No	0.10	3.00
EWLM_AIX_WAS1	128	3	3	16.37%	0.90	No	0.10	3.00
EWLM_AIX_WAS2	162	3	3	33.97%	1.14	No	0.10	3.00

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 9-13 Partition details for increased workload on silver

Notice that initially EWLM took entitlement from the DB2 partition (EWLM_AIX_DB2) and gave it to the WebSphere Application Server partition (EWLM_AIX_WAS2) that is running the *silver* stream. We now waited a bit longer to observe how EWLM continues to redistribute the entitled processing units within the partition workload group to improve the PI for the silver stream. After a few minutes we again check the PI. See Figure 9-14.

Select	Service class ^	PI ^	Importance ^	Performance ^	Goal ^
<input type="radio"/>	Gold_Trade6_SC	0.49	High	00.742 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	0.90	Low	01.355 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 9-14 PI after EWLM changes

The PI is now good for both streams. Notice that the PI for the *gold* stream also decreased. Let us take a look at how EWLM redistributed the entitled processing units. See Figure 9-15.

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^
EWLM_AIX_DB2	110	3	3	48.04%	0.78	No	0.10	3.00
EWLM_AIX_WAS1	99	3	3	14.02%	0.70	No	0.10	3.00
EWLM_AIX_WAS2	173	3	3	34.34%	1.22	No	0.10	3.00

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 9-15 CPU usage after a good PI

Notice that EWLM again changed the weight of the partitions and removed additional resources from the WebSphere partition that runs the *gold* stream (EWLM_AIX_WAS1) and added them to the WebSphere partition that runs the *silver* stream (EWLM_AIX_WAS2).

9.2.4 Results and considerations

As expected, we saw EWLM moving resources from the high-importance service class to the low-importance service class. The interesting thing in this scenario is that EWLM did not initially change the partition that runs the WebSphere for the *gold* stream but instead moved resources from the DB2 partition to the partition that is running the *silver* stream. Later, EWLM realizes that the DB2 partition is also in a lack of resources and then it decides to move resources from the *gold* WebSphere partition to the DB2 partition.

This assumption being that if EWLM removes resources only from the DB2 partition, the gold stream might experience an increase in PI above 1.

This scenario demonstrates that EWLM maintains a global view of the workload and will even remove resources from a partition that runs high importance workload and allocate it to a partition running less important work.

9.3 Partition class for non-ARM instrumented application

We now discuss the last of three scenarios to demonstrate the partition management functionality of EWLM in a non-ARM instrumented environment. We ran two kinds of workloads to observe and analyze the manner in which EWLM responded to changing conditions.

9.3.1 Objective and expectation

The objective of this scenario is to analyze how EWLM partition management works when we have a non-ARM instrumented application in the EWLM Domain. EWLM can manage the partition on which the non-ARM instrumented application runs and monitors the resources of the application as well by defining Partition class or Process class and setting the performance goal for that class. When you define a Process class, you can assign it to a Service class that has either velocity goals (most likely, good fit for long-running/never-ending processes) or response time goals (applies to things like fairly short-running batch). However, when you define a Partition class assigning to a Service class, it is allowed to define only a velocity goal.

To create an ITSO environment that includes a non-ARM instrumented backed end database, we disabled the ARM support from the DB2 partition (EWLM_AIX_DB2). In addition, we defined a new Service class (Diamond) that has one Partition class for the EWLM_AIX_DB2 partition that is running a single primary DB2 application that can be viewed as having a single business importance.

We would expect to see that when the performance index of the new Diamond Service class for the EWLM_AIX_DB2 partition is over 1, meaning that it is missing its goal, then the capacity of the DB2 partition will be increased by allocating more resources from the other partitions in the shared pool. On the other hand, if the partition running the gold or silver service class workload requires additional entitled capacity, EWLM shifts resources from the EWLM_AIX_DB2 partition to the other partitions.

9.3.2 Starting point

The starting point is our base setup of the ITSO scenario. We have the server topology, as shown in Figure 9-1 on page 224, where the gold workload stream runs from the IBM HTTP server (ewlm1) on Windows to WebSphere Application Server 1 (ewlmaix1) on AIX to DB2 on (ewlmve01) on AIX. The silver workload stream runs from a different IBM HTTP server on Windows (tot132) to WebSphere Application Server 2 (ewlmaix2) on AIX and to the same DB2 (ewlmve01) on AIX as is the gold stream.

The gold stream is associated with external users and therefore more important workload. The silver is associated with internal users and less important work. We therefore have two service classes called Gold_Trade6_SC and Silver_Trade6_SC. Both have the same average response time goal of 1.5 seconds but a different importance setting. The gold stream is classified as being of high importance and silver as being of low importance.

In addition, we create a new Domain Policy called ITSO Non-ARM Domain Policy that has an additional service class called Diamond_DB_SC based on the ITSO Trade6 Domain Policy. In this new Diamond_DB_SC service class, we define a Partition class for the DB2 system on

an AIX platform. The *diamond* service class has the velocity goal of fast and is classified as being of highest importance.

Create a Domain Policy

The naming Domain Policy for scenario 3 looks as shown below and the changes from the base Domain Policy are highlighted:

- ▶ Domain Policy:
 - Policy name: *ITSO Non-ARM Domain Policy*
 - Policy description: *ITSO Trade6 Domain Policy with non-ARMed Application*
- ▶ Service Policy:
 - Policy Name: *ITSO Trade6 Domain Policy for weekdays*
 - Policy description: *Weekday Service Policy*
- ▶ Applications:
 - IBM Webserving Plugin
- ▶ Service Class:
 - Class Name: *Diamond_DB_SC*
 - Class description: *Diamond non-ARM DB2 service class*
 - Class type: *Partition*
 - Class Importance: *Highest*
 - Class Goal: *Velocity: Fast*
 - Class Name: *Gold_Trade6_SC*
 - Class description: *High importance for external workload*
 - Class type: *Transaction*
 - Class Importance: *High*
 - Class Goal: *AverageResponseTime: 1.5s*
 - Class Name: *Silver_Trade6_SC*
 - Class description: *Low importance for internal workload*
 - Class type: *Transaction*
 - Class Importance: *Low*
 - Class Goal: *AverageResponseTime:1.5s*
- ▶ Platform
 - Platform name: *AIX*
 - Platform description: *Contains filters for AIX*
- ▶ Transaction Class:
 - Class Name: *Gold_Trade6_TC*
 - Class Description: *External workload*
 - Filter: *EWLM URI = /trade(*) AND EWLM Hostname = ewlm1*
 - Mapping to Service Class: *Gold_Trade6_SC*
 - Position: *1*
 - Class Name: *Silver_Trade6_TC*
 - Class Description: *Internal workload*
 - Filter: *EWLM URI = /trade(*) AND EWLM Hostname = tot132*
 - Mapping to Service Class: *Silver_Trade6_SC*
 - Position: *2*

- ▶ Partition Class:
 - Class Name: *Non-ARM_DB2_PC*
 - Class Description: *Non-ARM instrumented DB2 Partition class*
 - Filter: *EWLM Hostname = aixve01*
 - Mapping to Service Class: *Diamond_DB_SC*
 - Position: *1*

Disable ARM instrumentation on DB2

For the test purpose only, we disabled the ARM instrumentation on DB2 by removing the user capabilities required for ARM. The command is shown in Example 9-1.

Example 9-1 Remove ARM capabilities from DB2 instance owner

```

aixve01@/> chuser "capabilities=" db2inst1
aixve01@/> lsuser db2inst1
db2inst1 id=109 pgrp=db2grp1 groups=db2grp1,staff,dasadm1 home=/home/db2inst1
shell=/usr/bin/ksh login=true su=true rlogin=true daemon=true admin=false sugroups=ALL
admgroups= tpath=nosak ttys=ALL expires=0 auth1=SYSTEM auth2=NONE umask=22 registry=files
SYSTEM=compat logintimes= loginretries=0 pldwarntime=0 account_locked=false minage=0
maxage=0 maxexpired=-1 minalpha=0 minother=0 mindiff=0 maxrepeats=8 minlen=0 histexpire=0
histsize=0 pwdchecks= dictionlist= fsize=-1 cpu=-1 data=491519 stack=32767 core=-1 rss=-1
nofiles=2000 time_last_login=1129397438 time_last_unsuccessful_login=1129394172
tty_last_login= tty_last_unsuccessful_login= host_last_login=aixve01
host_last_unsuccessful_login=aixve01 unsuccessful_login_count=0 roles=

```

Initial Partition settings

Our partition settings are explained in detail in 8.3.1, “Partition settings for the ITSO environment” on page 217. We start with having an entitlement of 0.9 as the desired value for each partition with 1 virtual processor and a weight of 128. This looks as shown in the Partition Workload Group Details view in Figure 9-16.

Note: By the time of the details display EWLM has already changed the desired value for Virtual Processors to 3.

Partition name ^	Weight ^	Virtual processors ^	Physical processors ^	Processor utilization ^	Processing units ^	Capped ^	Minimum processing units ^	Maximum processing units ^	Allocated memory ^
EWLM_AIX_DB2	128	3	3	0.62%	0.90	No	0.10	3.00	4,096 MB
EWLM_AIX_WAS1	128	3	3	0.57%	0.90	No	0.10	3.00	4,096 MB
EWLM_AIX_WAS2	128	3	3	0.46%	0.90	No	0.10	3.00	4,096 MB
Page 1 of 1		Total: 3		Filtered: 3	Displayed: 3				

Figure 9-16 Partition settings - Starting point

9.3.3 Running the test case

We have set up the environment to start the workload, and the EWLM does not interfere in the partition management as long as the performance goals defined in the Domain Policy are met.

Case 1: Capacity shift to DB2

We increase the workload from gold and silver transaction streams and it causes the workload surge to the DB2 system. As we can see in Figure 9-17, the performance index of the Diamond service class is over 1, which means that it begins to miss the goal, but the goals for gold and silver service classes are still met.

Select	Service class ²	PI [^]	Importance ¹	Performance [^]	Goal [^]
<input type="radio"/>	Diamond_DB_SC	1.28	Highest	Moderate velocity	Fast velocity
<input type="radio"/>	Gold_Trade6_SC	0.43	High	00.638 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	0.44	Low	00.657 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Figure 9-17 Increased workload for gold and silver stream leads to increase PI of diamond

The EWLM detects CPU delay in DB2 system and it determines that the DB2 system will do better by taking resources from EWLM_AIX_WAS1 and EWLM_AIX_WAS2 partitions and giving them to EWLM_AIX_DB2. The number of processing units adjusted is shown in Figure 9-18 in order to help ensure that the goals defined in the Domain Policy are met.

Partition name [^]	Weight [^]	Virtual processors [^]	Physical processors [^]	Processor utilization [^]	Processing units [^]	Capped [^]	Minimum processing units [^]	Maximum processing units [^]
EWLM_AIX_DB2	130	3	3	34.57%	0.92	No	0.10	3.00
EWLM_AIX_WAS1	126	3	3	27.64%	0.89	No	0.10	3.00
EWLM_AIX_WAS2	126	3	3	23.23%	0.89	No	0.10	3.00

Figure 9-18 Processing units shifted to DB2,

The EWLM keeps adjusting the entitled processing units until the optimal settings are found, as shown in Figure 9-19.

Partition name [^]	Weight [^]	Virtual processors [^]	Physical processors [^]	Processor utilization [^]	Processing units [^]	Capped [^]	Minimum processing units [^]	Maximum processing units [^]
EWLM_AIX_DB2	194	3	3	48.44%	1.37	No	0.10	3.00
EWLM_AIX_WAS1	91	3	3	24.19%	0.64	No	0.10	3.00
EWLM_AIX_WAS2	98	3	3	24.46%	0.69	No	0.10	3.00

Figure 9-19 Processing units shifted to DB2 again

After the capacity changed, the performance index of the *diamond* service class becomes 1, as shown in Figure 9-20.

Select	Service class ²	PI [^]	Importance ¹	Performance [^]	Goal [^]
<input type="radio"/>	Diamond_DB_SC	1.00	Highest	Fast velocity	Fast velocity
<input type="radio"/>	Gold_Trade6_SC	0.47	High	00.702 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	0.46	Low	00.685 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Figure 9-20 Goals in the Domain Policy are all met now

Case 2: Capacity shift from DB2

To see the capacity movement in a partition group, we initialize the partition settings by disabling EWLM partition management using the EWLM Control Center and then enabling it again. This resets the partitions to the settings specified in the partition policy. Each partition now has an entitled capacity of 0.9 and a weight of 128.

We now increase the workload on the gold stream while the workload on the silver stream remains constant. As a result of this workload change, the gold service class begins to miss its goal, as shown in Figure 9-21.

Select	Service class ²	PI [^]	Importance ¹	Performance [^]	Goal [^]
<input type="radio"/>	Diamond_DB_SC	0.81	Highest	Fastest velocity	Fast velocity
<input type="radio"/>	Gold_Trade6_SC	1.12	High	01.685 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	0.60	Low	00.897 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Figure 9-21 Gold service class missing goal

EWLM detects CPU delay in the EWLM_AIX_WAS1 partition supporting the gold service class and it takes resources from the EWLM_AIX_WAS2 partition and gives them to EWLM_AIX_WAS1. The number of processing units adjusted is shown in Figure 9-22.

Partition name [^]	Weight [^]	Virtual processors [^]	Physical processors [^]	Processor utilization [^]	Processing units [^]	Capped [^]	Minimum processing units [^]	Maximum processing units [^]
EWLM_AIX_DB2	128	3	3	45.23%	0.90	No	0.10	3.00
EWLM_AIX_WAS1	147	3	3	30.41%	1.04	No	0.10	3.00
EWLM_AIX_WAS2	108	3	3	21.61%	0.76	No	0.10	3.00

Figure 9-22 Processing units shifted from EWLM_AIX_WAS2

EWLM also determines that the EWLM_AIX_WAS1 system might do better by taking resources from the EWLM_AIX_DB2 partition and giving them to EWLM_AIX_WAS1. The number of processing units adjusted is shown in Figure 9-23.

Partition name [^]	Weight [^]	Virtual processors [^]	Physical processors [^]	Processor utilization [^]	Processing units [^]	Capped [^]	Minimum processing units [^]	Maximum processing units [^]
EWLM_AIX_DB2	126	3	3	26.74%	0.89	No	0.10	3.00
EWLM_AIX_WAS1	149	3	3	18.07%	1.05	No	0.10	3.00
EWLM_AIX_WAS2	108	3	3	12.86%	0.76	No	0.10	3.00

Figure 9-23 Processing units shifted from EWLM_AIX_DB2

EWLM continues to move entitled processing units from the EWLM_AIX_DB2 partition to the EWLM_AIX_WAS1 partition, as shown in Figure 9-24.

Partition name [^]	Weight [^]	Virtual processors [^]	Physical processors [^]	Processor utilization [^]	Processing units [^]	Capped [^]	Minimum processing units [^]	Maximum processing units [^]
EWLM_AIX_DB2	119	3	3	45.12%	0.84	No	0.10	3.00
EWLM_AIX_WAS1	156	3	3	32.11%	1.10	No	0.10	3.00
EWLM_AIX_WAS2	108	3	3	19.64%	0.76	No	0.10	3.00

Figure 9-24 Keeps the CPU movement from EWLM_AIX_DB2

Although the performance index is still not a 1 for the gold stream, as shown in Figure 9-25, EWLM does not move any additional entitled processing units. The assumption is that EWLM has determined that additional processing unit resources would not improve the PI of the gold service class.

Select	Service class ²	PI [^]	Importance ¹	Performance [^]	Goal [^]
<input type="radio"/>	Diamond_DB_SC	0.80	Highest	Fastest velocity	Fast velocity
<input type="radio"/>	Gold_Trade6_SC	1.02	High	01.537 average	01.500 average
<input type="radio"/>	Silver_Trade6_SC	0.54	Low	00.807 average	01.500 average
<input type="radio"/>	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Figure 9-25 PI values following processing unit adjustments

9.3.4 Results and considerations

For this scenario we expected to see EWLM shift processing units to or from the aixve01 partition, which has a non-ARM instrumented DB2 application, to meet the goal defined in the Domain Policy. These expectations were met. We can see that the DB2 partition supporting the diamond service class can get more CPU resources from both the EWLM_AIX_WAS1 partition running the gold workload and the EWLM_AIX_WAS2 partition running the silver workload. In addition, the DB2 partition can give CPU resources to either the EWLM_AIX_WAS1 or the EWLM_AIX_WAS2 partition, which requires CPU resources to meet the goal if it misses the goal.

In this scenario, we observed that the EWLM managed at the partition level. The entire partition is given a goal and importance and management of the partition is based on the partition's achievement of this goal. Assignment of a goal and importance allows EWLM to put the partition performance in the perspective of the work it is managing.

Partition is assigned a velocity goal that measures how well it is being treated by the hypervisor. With a partition class, there is no need to understand the role of individual processes compared to a process class. In addition, it allows a partition to be viewed as a single entity from the point of view of the performance administrator.

9.4 Monitoring a multi-tiered workload with EWLM and z/OS

In this scenario, we move the DB2 on a z/OS system and then compare the traditional z/OS and DB2 reporting with the new EWLM monitoring facility.

9.4.1 Objective and expectation

The objective of this scenario is to run a multi-tiered transactional workload and to understand how the monitoring data presented by EWLM compares with the data presented by the conventional RMF and DB2 PE reports. We also describe the difference in the classification methodologies between EWLM and z/OS policies and how, when possible, to relate the data presented by the two workload management facilities.

The attribute values for work requests received by the entry application of an EWLM Management Domain might be different from those passed to WLM by JDBC type 4 if the business transaction has to be sent to a z/OS hop to get data, especially if the classification on the distributed platform happened in an application environment other than DB2. For this reason it may not always be possible to have a one-to-one mapping for EWLM service classes and WLM service classes.

As an example refer to Figure 9-26, where the EWLM and WLM classification logic is shown for application transactions whose topology uses a z/OS hop as a database server.

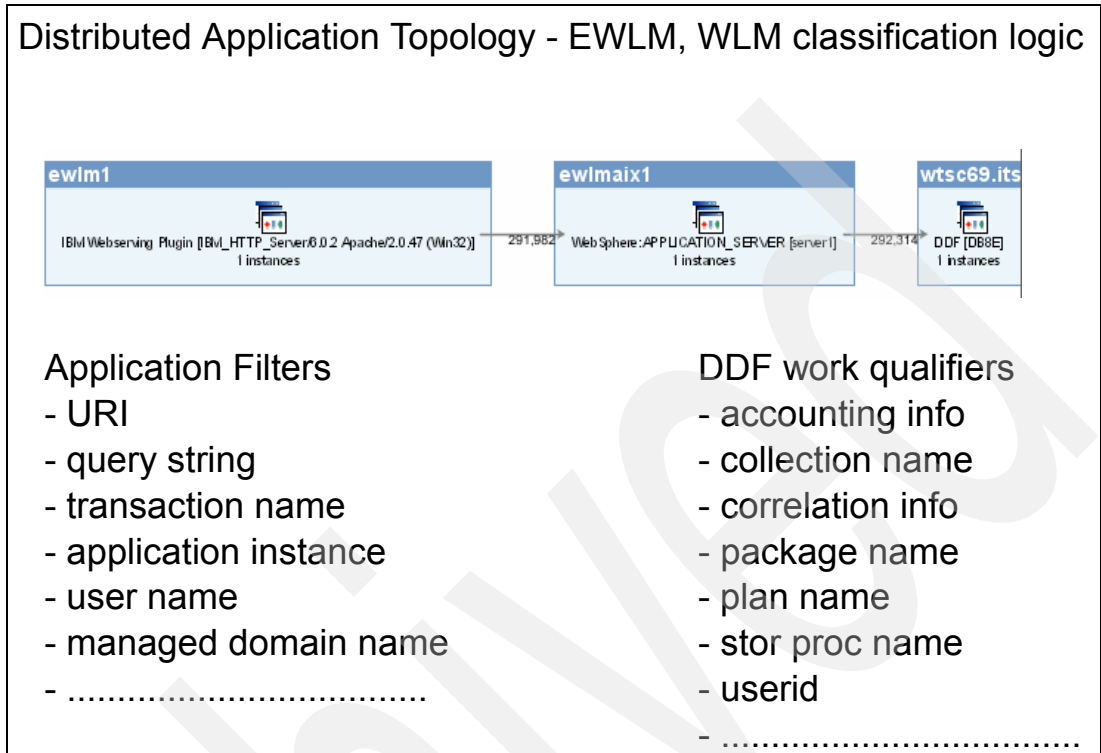


Figure 9-26 Classification flow

Figure 9-27 is a summary of the filters and classification rules used in the EWLM and z/OS WLM policies.

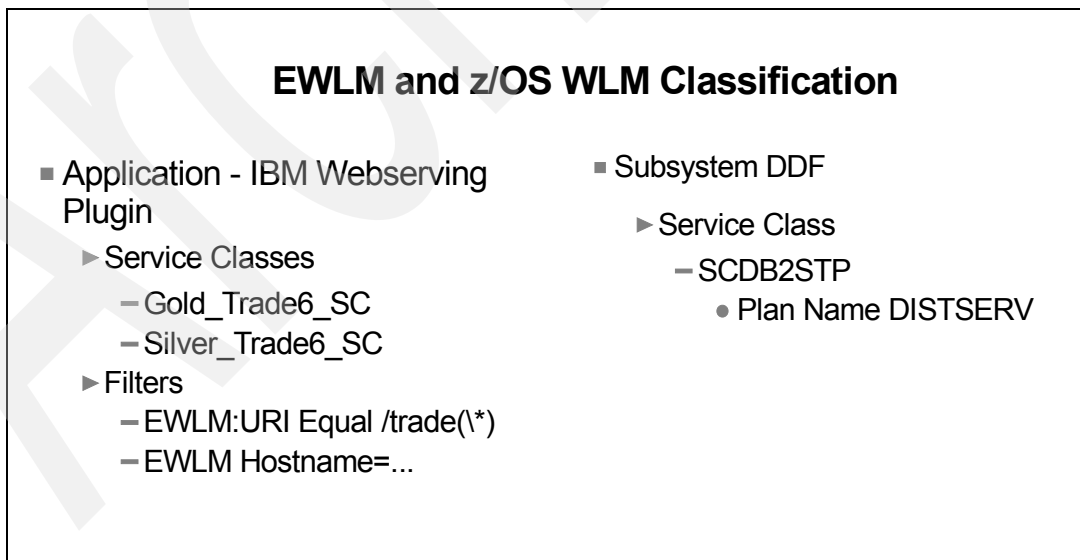


Figure 9-27 Classification rules

The reason we were not able to put together a one-to-one relationship among EWLM and WLM service classes is that, for a JDBC type 4 application, attribute values passed to WLM,

such as PLAN, PACKAGE, CONNECTION, and CORRELATION, are the same for each message sent to DDF. In our scenario these are the attribute values passed to DDF:

- ▶ PLAN = DISTSERV
- ▶ PACKAGE = SYSLN200
- ▶ CONNECTION = SERVER
- ▶ CORRELATION = db2jccSe

Example 9-2 is a DB2 PE accounting report (short type) where you can see the work qualifier for these transactions.

Example 9-2 DDF work qualifiers

PRMAUTH PLANNAME	#OCCURS #DISTR	#ROLLBK #COMMIT	SELECTS FETCHES	INSERTS OPENS	UPDATES CLOSES	DELETES PREPARE	CLASS1 CLASS1	EL.TIME CPUTIME	CLASS2 CLASS2	EL.TIME CPUTIME	GETPAGES BUF.UPDT	SYN.READ TOT.PREF	LOCK #LOCKOUT	SUS
MCONNOL DISTSERV	795323 795323	0 800531	0.00 1.01	0.00 1.01	0.00 0.00	0.00 1.01		0.008215 0.000720		0.004821 0.000546	3.02 0.00	0.00 0.00	0.04 0	

PROGRAM NAME	TYPE	#OCCURS	SQLSTMT	CL7	ELAP.TIME	CL7 CPU TIME	CL8	SUSP.TIME	CL8 SUSP
SYSLN200	PACKAGE	774023	4.03		0.003989	0.000540		0.000192	0.11

REQUESTER	METH	#DDFS	TRANS	#ROLLBK	#COMMIT	SQLRECV	ROWSENT	CONVI
9.12.4.177	DRDA	263K	0.00	0	262930	2.00	0.00	0.00
9.12.4.178	DRDA	532K	0.00	0	537601	2.02	0.00	0.00

ACCOUNTING REPORT COMPLETE

Usually, you can classify based on the first stored procedure called in a transaction, but this was not our case.

Tip: If you need to differentiate DDF transactions arriving from a distributed environment, you can use one of the following programming approaches:

- ▶ Use a different user ID to connect to DB2 through DDF.
- ▶ Use a different data source within WebSphere Application Server for each classification and use a specific data source within the application. Each data source could have either a different AUTHID associated with it, or a JDBC driver collection. It is possible to classify based on either of those.
- ▶ Use DB2 client strings, which are text attributes associated with the connection that can be used for classification of the workload. The client strings can be set as a part of the datasource definition, or they can be set within the application so that every transaction can have a different value. Client strings can be used for workload classification and for end-to-end auditing. The fields are written in DB2 accounting reports. The recommended client strings are *application name* and *end user ID*. Additional information is available in the white paper *Managing Enterprise Java Applications for DB2*, G507-1457.

9.4.2 Starting point

For this scenario, we moved the database on a z/OS LPAR that consists of a z/OS 1.6 with two logical CPs on an IBM zSeries 990 2084-318. The LPAR is a sysplex image in which a DB2 data sharing group is active. Only the DB2 member DB8E, active in our partition, is used

as a database server by the distributed application. Figure 9-28 shows the configuration used to run this scenario.

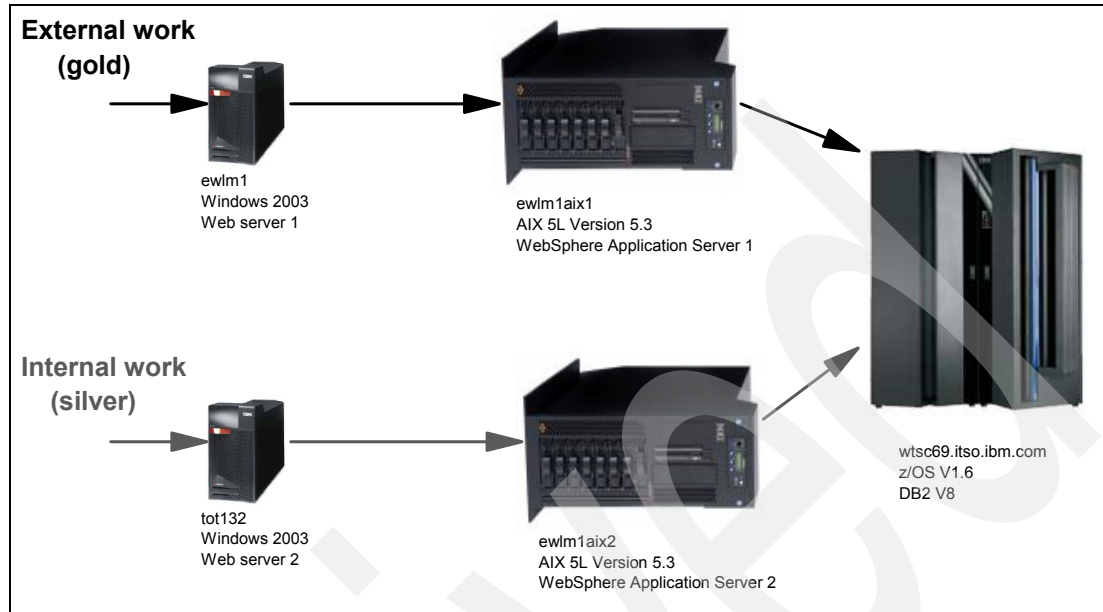


Figure 9-28 Configuration for the scenario with z/OS as managed server

We used the basic ITSO policy with the two workload streams, gold and silver, running through two IBM HTTP servers, two WebSphere Application Servers, and both streams sharing the DB2 database on z/OS.

During our scenario there was no contention with the other logical partitions defined in the processor. Figure 9-29 shows the service class goals for the z/OS workloads that were run during our scenario.

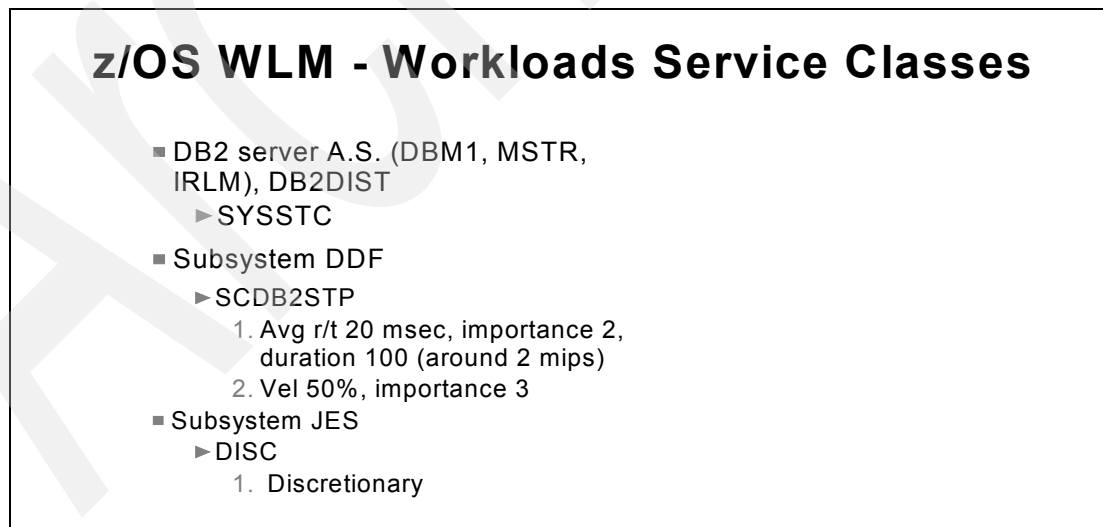


Figure 9-29 DDF service class on z/OS

We classify the DB2, MSTR, DBM1, DIST, and IRLM address space in the SYSSTC service class, although in a production environment we suggest assigning the DB2, MSTR, DBM1, and DIST address spaces to a managed service class with an IMP of 1, velocity of 50 or 60, and IRLM to the SYSSTC service class.

The RMF recording interval is set to one minute. To collect DB2 monitor data, accounting and statistic traces have been started, as shown in Example 9-3.

Example 9-3 DB2 trace activation

```
-DB8E DISPLAY TRACE
RESPONSE=SC69
DSNW127I -DB8E CURRENT TRACE ACTIVITY IS -
TNO TYPE CLASS DEST QUAL
01 STAT 01,03,04,05, SMF NO
01 06
02 ACCTG 01 SMF NO
03 ACCTG 01,02,03,07, SMF NO
03 08
```

Note: To be able to relate the EWLM Control Center and z/OS RMF and DB2 monitors information, we confirmed that the platform clocks were synchronized at one minute boundary.

9.4.3 Running the test case

Now we correlate EWLM Control Center monitoring data with data provided by RMF and DB2 PE monitors for the application environment DDF. The scenario is based on a 10-minute reporting interval.

Example 9-4 contains the control statements used to obtain the RMF and DB2 PE report.

Example 9-4 RMF and DB2 PE control statements

```
--- RMF-----
SUMMARY(INT,TOT)
RTOD(1020,1030)
DINTV(0010)
REPORTS(CPU)
SYSRPTS(WLMGL(POLICY,WGROUP,SCPER,RCLASS,SYSNAM))
-----
--- DB2 PE Accounting long -----
GLOBAL
TIMEZONE(+5)
INCLUDE(SUBSYSTEMID(DB8E)
CONNECT(SERVER)
)
FROM(,10:20) TO(,10:30)
ACCOUNTING
REPORT
LAYOUT(LONG)
EXEC
-----
--- DB2 PE Statistics long -----
GLOBAL
TIMEZONE(+5)
INCLUDE(SUBSYSTEMID(DB8E)
CONNECT(SERVER)
)
FROM(,10:20) TO(,10:30)
```

9.4.4 EWLM Control Center reports

Before examining the application reporting data, let us have a look at the overall processor utilization. For comparison, Example 9-5 shows CPU utilization reported by RMF Monitor I for the time frame corresponding to the EWLM Control Center Processor utilization view shown in Figure 9-30. You can see that the reporting data, for the intervals in which the workload is running, is very similar in both reports. You can display more processor utilization detailed information about the EWLM Managed Server panel, shown in Figure 9-31 on page 241 and in the Details panels shown in Figure 9-32 on page 241. In some cases, you might observe a small difference between the RMF reported values and the EWLM reported values because the EWLM Control Center does not report the uncaptured time.

Example 9-5 RMF Monitor I CPU utilization

RMF OVERVIEW REPORT										
z/OS V1R6		SYSTEM ID SC69		START 11/21/2005-10.10.00		INTERVAL 00.05.00		PAGE 001		
		RPT VERSION V1R5 RMF		END 11/21/2005-10.30.00		CYCLE 1.000 SECONDS				
0		TOTAL LENGTH OF INTERVALS 00.20.00								
NUMBER OF INTERVALS 4	NUMPROC	CPUBUSY	APPLPER	BATCH	DDF	STC	TSO	SYSTEM	SYSSTC	
-DATE TIME INT	MM/DD HH.MM.SS MM.SS									
11/21 10.10.00 05.00	2.0	37.7	66.4	0.0	29.8		0.3	5.3	29.3	
11/21 10.15.00 05.00	2.0	66.4	120.6	0.0	61.4		0.3	3.2	52.8	
11/21 10.20.00 04.59	2.0	68.9	124.8	0.0	74.2		0.1	3.5	44.1	
11/21 10.25.00 05.00	2.0	68.9	124.8	0.0	74.4		0.0	3.5	43.8	

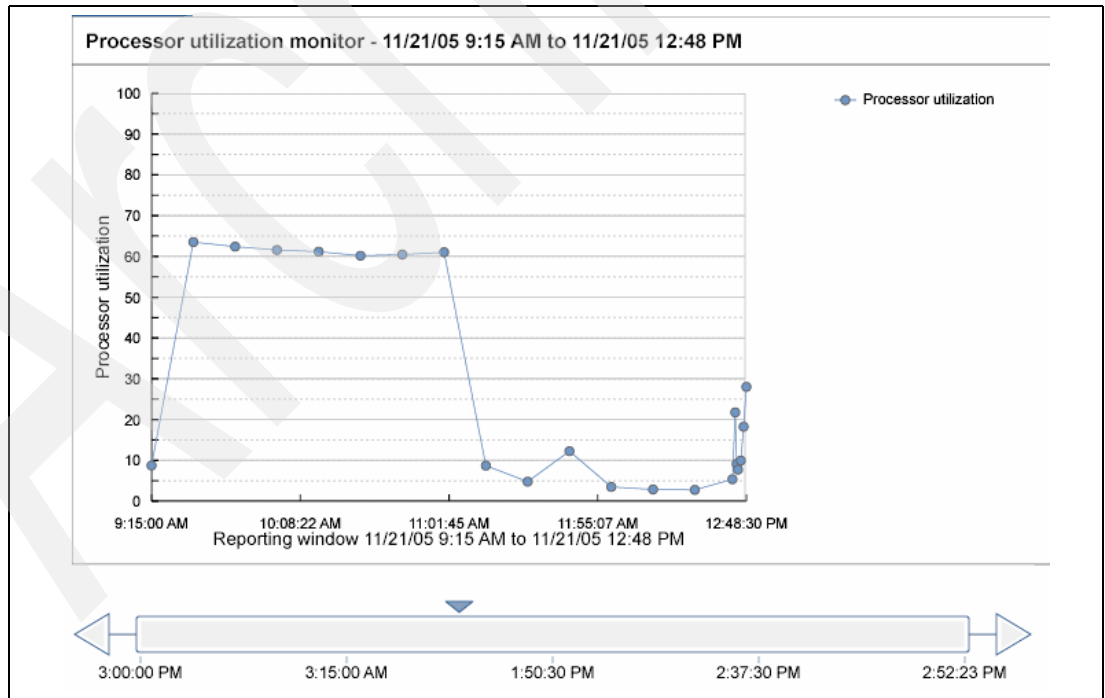


Figure 9-30 Processor utilization monitor

Select	Managed server	Processor utilization	State	Active service policy	Operating system
<input type="radio"/>	aixve01	0.9%	Active	ITSO Trade6 Service Policy for weekdays	AIX
<input type="radio"/>	ewlm1	29.0%	Active	ITSO Trade6 Service Policy for weekdays	Windows
<input type="radio"/>	ewlmaix1	25.1%	Active	ITSO Trade6 Service Policy for weekdays	AIX
<input type="radio"/>	ewlmaix2	23.2%	Active	ITSO Trade6 Service Policy for weekdays	AIX
<input type="radio"/>	tot132	84.5%	Active	ITSO Trade6 Service Policy for weekdays	Windows
<input type="radio"/>	wtsc69.itso.ibm.com	63.5%	Active	ITSO Trade6 Service Policy for weekdays	z/OS

Figure 9-31 Managed Server reporting

Details - Managed server 'wtsc69.itso.ibm.com'				
View details about the performance of this managed server.				
Interval: 11/21/05 10:15:00 AM to 11/21/05 10:30:00 AM				
Statistics				
Average processor utilization %:	63.5			
Real memory (KB):	4,194,304			
Number of processors:	2			
Page fault rate (faults per second):	0.0			
Application [Group]:	DDF[DB8E]			
Service classes processed in server				
Service class	Processor usage %	Processor delay %	Paging delay %	I/O delay %
Gold_Trade6_SC	13.0	11.8	0.0	0.0
Silver_Trade6_SC	17.6	19.4	0.0	0.0
Page 1 of 1 Total: 2 Filtered: 2 Displayed: 2				

Figure 9-32 Managed Server details

The Trade6 workload uses two EWLM service classes. The processor utilization, transaction data, transaction response, and queue time can be found in the table view for the Service Class Application topology views.

- ▶ Service Class Gold_Trade6_SC Application topology table view (Figure 9-33, Figure 9-34 on page 242, and Figure 9-35 on page 242)

Hop number	Name	Type of node	Platform	Average response time	Standard deviation	Average active time	Average queue time	Successful transactions	Failed transactions
0	IBM Webserving Plugin (IBM_HTTP...	Application		00.016621	00.008000	00.007858	00.007393	418,261	0
0	ewlm1	Server	Windows	00.016621	00.008000	00.007858	00.007393	418,261	0
0	EWLM1/PID=0000004628	Instance		00.016621	00.008000	00.007858	00.007393	418,261	0
1	WebSphere:APPLICATION_SERV...	Application		00.006133	00.005000	00.002443	00.000000	418,370	0
1	ewlmaix1	Server	AIX	00.006133	00.005000	00.002443	00.000000	418,370	0
1	ewlmaix1Node02.server1	Instance		00.006133	00.005000	00.002443	00.000000	418,370	0
2	DDF [DB8E]	Application		00.000414	00.000000	00.000414	00.000000	840,860	0
2	wtsc69.itso.ibm.com	Server	z/OS	00.000414	00.000000	00.000414	00.000000	840,860	0
2	DB8E	Instance		00.000414	00.000000	00.000414	00.000000	840,860	0

Figure 9-33 Application topology for Gold_Trade6_SC service class - Right segment

Hop number	Name	Stopped transactions	Unknown transactions	Not valid transactions	Topology uncertainty count	Reporting gap count	Processor time	Processor using (%)
0	IBM Webserving Plugin [IBM_HTT...	0	0	0	0	0	38.553779	22.6%
0	ewlrm1	0	0	0	0	0	38.553779	22.6%
0	EWLM1/PID=0000004628	0	0	0	0	0	38.553779	22.6%
1	WebSphere:APPLICATION_SERV...	0	0	0	0	0	06:33.989209	9.1%
1	ewlmaix1	0	0	0	0	0	06:33.989209	9.1%
1	ewlmaix1Node02.server1	0	0	0	0	0	06:33.989209	9.1%
2	DDF [DB8E]	0	0	0	0	0	01:52.619346	57.0%
2	wtsc69.itso.ibm.com	0	0	0	0	0	01:52.619346	57.1%
2	DB8E	0	0	0	0	0	01:52.619346	57.1%

Figure 9-34 Application topology for Gold_Trade6_SC service class - Middle segment

Hop number	Name	Processor delay (%)	Page delay (%)	I/O delay (%)	Idle (%)	Other (%)	In progress transactions	In progress elapsed time
0	IBM Webserving Plugin [IBM_HTT...	1.0%	0.0%	0.0%	0.0%	76.3%	0	None
0	ewlrm1	1.0%	0.0%	0.0%	0.0%	76.3%	0	None
0	EWLM1/PID=0000004628	1.0%	0.0%	0.0%	0.0%	76.3%	0	None
1	WebSphere:APPLICATION_SERV...	9.9%	0.0%	0.0%	92.5%	81.0%	4	00.000001
1	ewlmaix1	9.9%	0.0%	0.0%	92.5%	81.0%	4	00.000001
1	ewlmaix1Node02.server1	9.9%	0.0%	0.0%	92.5%	81.0%	4	00.000001
2	DDF [DB8E]	43.0%	0.0%	0.0%	48.6%	0.0%	0	None
2	wtsc69.itso.ibm.com	42.9%	0.0%	0.0%	48.6%	0.0%	0	None
2	DB8E	42.9%	0.0%	0.0%	48.6%	0.0%	0	None

Figure 9-35 Application topology for Gold_Trade6_SC service class - Left segment

- ▶ Service Silver_Trade6_SC Application topology table view (Figure 9-36, Figure 9-37, and Figure 9-38 on page 243)

Hop number	Name	Type of node	Platform	Average response time	Standard deviation	Average active time	Average queue time	Successful transactions	Failed transactions
0	IBM Webserving Plugin [IBM_HTT...	Application		00.010418	00.008000	00.000859	00.000000	360,458	0
0	tot132	Server	Windows	00.010418	00.008000	00.000859	00.000000	360,458	0
0	TOT132/PID=0000005860	Instance		00.010418	00.008000	00.000859	00.000000	360,458	0
1	WebSphere:APPLICATION_SERV...	Application		00.006197	00.005000	00.002472	00.000000	356,125	0
1	ewlmaix2	Server	AIX	00.006197	00.005000	00.002472	00.000000	356,125	0
1	ewlmaix2Node02.server1	Instance		00.006197	00.005000	00.002472	00.000000	356,125	0
2	DDF [DB8E]	Application		00.000416	00.000000	00.000416	00.000000	716,072	0
2	wtsc69.itso.ibm.com	Server	z/OS	00.000416	00.000000	00.000416	00.000000	716,072	0
2	DB8E	Instance		00.000416	00.000000	00.000416	00.000000	716,072	0

Figure 9-36 Application topology for Silver_Trade6_SC service class - Right segment

Hop number	Name	Stopped transactions	Unknown transactions	Not valid transactions	Topology uncertainty count	Reporting gap count	Processor time	Processor using (%)
0	IBM Webserving Plugin [IBM_HTT...	0	0	0	0	0	02:58.455988	35.5%
0	tot132	0	0	0	0	0	02:58.455988	35.5%
0	TOT132/PID=0000005860	0	0	0	0	0	02:58.455988	35.5%
1	WebSphere:APPLICATION_SERV...	0	0	0	0	0	05:53.945891	10.3%
1	ewlmaix2	0	0	0	0	0	05:53.945891	10.3%
1	ewlmaix2Node02.server1	0	0	0	0	0	05:53.945891	10.3%
2	DDF [DB8E]	0	0	0	0	0	01:36.394339	57.1%
2	wtsc69.itso.ibm.com	0	0	0	0	0	01:36.394339	57.0%
2	DB8E	0	0	0	0	0	01:36.394339	57.0%

Figure 9-37 Application topology for Silver_Trade6_SC service class - Middle segment

Hop number	Name	Processor delay (%)	Page delay (%)	I/O delay (%)	Idle (%)	Other (%)	In progress transactions	In progress elapsed time
0	IBM Webserving Plugin (IBM_HTTP...	49.2%	0.0%	0.0%	0.0%	15.3%	7	00.121853
0	tot132	49.2%	0.0%	0.0%	0.0%	15.3%	7	00.121853
0	TOT132/PID=0000005860	49.2%	0.0%	0.0%	0.0%	15.3%	7	00.121853
1	WebSphere:APPLICATION_SERV...	11.5%	0.0%	0.0%	92.0%	78.2%	5	00.000005
1	ewlmaix2	11.5%	0.0%	0.0%	92.0%	78.2%	5	00.000005
1	ewlmaix2Node02.server1	11.5%	0.0%	0.0%	92.0%	78.2%	5	00.000005
2	DDF [DB8E]	42.9%	0.0%	0.0%	48.6%	0.0%	0	None
2	wts69.itso.ibm.com	43.0%	0.0%	0.0%	48.6%	0.0%	0	None
2	DB8E	43.0%	0.0%	0.0%	48.6%	0.0%	0	None

Figure 9-38 Application topology for Silver_Trade6_SC service class - Left segment

The EWLM Control Center aggregation interval for the table views is 10 minutes starting at 10:20am November 21. We normalize for DDF application environment (hop 2) the processor time seconds and the successful transactions count to a second to be able to relate to RMF monitors data. We did not experience stopped or failed transactions during our test run.

- ▶ Processor time seconds:
 - Gold_Trade6_SC 112.61 (53.8% of total)
 - Silver_Trade6_SC 96.39 (46.2% of total)
 - Total 209 seconds that divided by 600 seconds equals 34.83% utilization of one CP
- ▶ Successful transactions:
 - Gold_Trade6_SC 840860 (54% of total)
 - Silver_Trade6_SC 716072 (46% of total)
 - Total 1556932 transactions that divided by 600 seconds equals 2594.88 transactions per second
- ▶ Transaction response time and queue time (in both cases it is a negligible value):
 - Gold_Trade6_SC r/t = 0.4 millisecond, queue time = 0 millisecond
 - Silver_Trade6_SC r/t = 0.4 millisecond, queue time = 0 millisecond

Note: Those are hop 2 transactions delimited by issuing an arm_start_transaction, arm_stop_transaction at the middleware level (DDF).

9.4.5 RMF postprocessor workload activity reports

From the RMF postprocessor, we extrapolate reports for WLM workload DB2RES and for service class SCDB2STP that have been set up for DDF workload.

Example 9-6, Example 9-7 on page 244, and Example 9-8 on page 245 show excerpts from the RMF reports based on a 10-minute monitoring interval that matches the EWLM Control Center interval.

Example 9-6 RMF workload report

```

1                                WORKLOAD ACTIVITY                                PAGE 7
z/OS V1R6                        SYSPLEX WTSCPLX1          START 11/21/2005-10.20.00 INTERVAL 000.09.59  MODE = GOAL
                                RPT VERSION V1R5 RMF          END   11/21/2005-10.30.00
                                POLICY ACTIVATION DATE/TIME 11/01/2005 22.45.33
===== WORKLOAD
REPORT BY: POLICY=POLTEST        WORKLOAD=DB2RES
                                DB2 stored procedures

```


<= 00.00.00.018	769K	797	100	0.1	>
<= 00.00.00.020	770K	444	100	0.1	>
<= 00.00.00.022	770K	111	100	0.0	>
<= 00.00.00.024	770K	129	100	0.0	>
<= 00.00.00.026	770K	71	100	0.0	>
<= 00.00.00.028	770K	59	100	0.0	>
<= 00.00.00.030	770K	78	100	0.0	>
<= 00.00.00.040	771K	293	100	0.0	>
<= 00.00.00.080	771K	169	100	0.0	>
> 00.00.00.080	772K	862	100	0.1	>

Example 9-8 RMF DDF workload report - Period 2

REPORT BY: POLICY=POLTEST	WORKLOAD=DB2RES	SERVICE CLASS=SCDB2STP CRITICAL =NONE	RESOURCE GROUP=*NONE	PERIOD=2	IMPORTANCE=3			
TRANSACTIONS	TRANS.-TIME	HHH.MM.SS.TTT	--DASD I/O--	---SERVICE---	--SERVICE TIMES--	PAGE-IN RATES	----STORAGE----	
AVG 0.00	ACTUAL	20	SSCHRT 0.0	IOC 0	TCB 0.0	SINGLE 0.0	AVG 0.00	
MPL 0.00	EXECUTION	20	RESP 0.0	CPU 0	SRB 0.0	BLOCK 0.0	TOTAL 0.00	
ENDED 20	QUEUED	0	CONN 0.0	MSO 0	RCT 0.0	SHARED 0.0	CENTRAL 0.00	
END/S 0.03	R/S AFFINITY	0	DISC 0.0	SRB 0	IIT 0.0	HSP 0.0	EXPAND 0.00	
#SWAPS 0	INELIGIBLE	0	Q+PEND 0.0	TOT 0	HST 0.0	HSP MISS 0.0		
EXCTD 0	CONVERSION	0	IOSQ 0.0	/SEC 0	IFA N/A	EXP SNGL 0.0	SHARED 0.00	
AVG ENC 0.00	STD DEV	34			APPL% CP 0.0	EXP BLK 0.0		
REM ENC 0.00				ABSRPTN 0	APPL% IFACP 0.0	EXP SHR 0.0		
MS ENC 0.00				TRX SERV 0	APPL% IFA N/A			
GOAL: EXECUTION VELOCITY 50.0%	VELOCITY MIGRATION:	I/O MGMT N/A	INIT MGMT N/A					
SYSTEM	RESPONSE TIME EX	PERF AVG	--- USING% ---	----- EXECUTION DELAYS % -----	---DLY%---	-CRYPTO%--	---CNT%--	%
SC69	--N/A--	N/A 0.0 0.0	0.0 N/A 0.0 0.0		0.0 0.0 0.0 0.0	0.0 0.0 0.0 0.0	0.0 0.0	0.0

The reports for the two periods of WLM service class SCDB2STP confirm the CPU utilization and the number of enclaves. In addition, they show the response time distribution and the low number of transactions (0.03 per second) completed in period 2. Period 1 duration is 100 SU, roughly equivalent to 2 MIPS.

WLM sample state percentages for CPU (there are no storage or I/O delays) are:

- ▶ CPU using = 0.1%
- ▶ CPU delay = 0.5%

If we relate those two values, it turns out that CPU using samples are 83 percent and CPU delay states are 16 percent.

This information can be used to adjust the WLM service class goals if necessary.

9.4.6 DB2 PE accounting and statistics reports

Now we look at the DB2 PE accounting and statistics reports and cross-check with the EWLM Control Center and RMF reports. The interval duration is 10 minutes.

We refer to Example 9-9, which shows a DB2 PE accounting report excerpt. The interval and the duration correlates with EWLM and RMF data.

Example 9-9 DB2 PE accounting report extract

1 LOCATION: DB8E	DB2 PERFORMANCE EXPERT (V2)	PAGE: 1-1
GROUP: N/P	ACCOUNTING REPORT - LONG	REQUESTED FROM: ALL 10:20:00.00
MEMBER: N/P		TO: DATES 10:30:00.00
SUBSYSTEM: DB8E	ORDER: PRIMAUTH-PLANNAME	INTERVAL FROM: 11/21/05 10:20:00.10
DB2 VERSION: V8	SCOPE: MEMBER	TO: 11/21/05 10:29:59.99
PRIMAUTH: MCONNOL	PLANNAME: DISTSERV	
ELAPSED TIME DISTRIBUTION	CLASS 2 TIME DISTRIBUTION	
APPL =====> 59%	CPU =====> 33%	
DB2 =====> 40%	NOTACC =====> 65%	

SUSP > 1%				SUSP => 2%			
AVERAGE	APPL (CL.1)	DB2 (CL.2)	IFI (CL.5)	CLASS 3 SUSPENSIONS	AVERAGE TIME	AV.EVENT	HIGHLIGHTS
ELAPSED TIME	0.004009	0.001642	N/P	LOCK/LATCH(DB2+IRLM)	0.000031	0.06	#OCCURRENCES : 771713
NONNESTED	0.004009	0.001642	N/A	SYNCHRON. I/O	0.000000	0.00	#ALLIEDS : 0
STORED PROC	0.000000	0.000000	N/A	DATABASE I/O	0.000000	0.00	#ALLIEDS DISTRIB: 0
UDF	0.000000	0.000000	N/A	LOG WRITE I/O	0.000000	0.00	#DBATS : 771713
TRIGGER	0.000000	0.000000	N/A	OTHER READ I/O	0.000000	0.00	#DBATS DISTRIB. : 0
				OTHER WRTE I/O	0.000000	0.00	#NO PROGRAM DATA: 0
CPU TIME	0.000714	0.000536	N/P	SER.TASK SWTCH	0.000000	0.00	#NORMAL TERMINAT: 771713
AGENT	0.000714	0.000536	N/A	UPDATE COMMIT	0.000000	0.00	#ABNORMAL TERMIN: 0
NONNESTED	0.000714	0.000536	N/P	OPEN/CLOSE	0.000000	0.00	#CP/X PARALLEL. : 0
STORED PRC	0.000000	0.000000	N/A	SYSLGRNG REC	0.000000	0.00	#IO PARALLELISM : 0
UDF	0.000000	0.000000	N/A	EXT/DEL/DEF	0.000000	0.00	#INCREMENT. BIND: 0
TRIGGER	0.000000	0.000000	N/A	OTHER SERVICE	0.000000	0.00	#COMMITTS : 771852
PAR.TASKS	0.000000	0.000000	N/A	ARC.LOG(QUIES)	0.000000	0.00	#ROLLBACKS : 0
				ARC.LOG READ	0.000000	0.00	#SVPT REQUESTS : 0
SUSPEND TIME	0.000000	0.000031	N/A	DRAIN LOCK	0.000000	0.00	#SVPT RELEASE : 0
AGENT	N/A	0.000031	N/A	CLAIM RELEASE	0.000000	0.00	#SVPT ROLLBACK : 0
PAR.TASKS	N/A	0.000000	N/A	PAGE LATCH	0.000000	0.00	MAX SQL CASC LVL: 0
STORED PROC	0.000000	N/A	N/A	NOTIFY MSGS	0.000000	0.00	UPDATE/COMMIT : 0.00
UDF	0.000000	N/A	N/A	GLOBAL CONTENTION	0.000000	0.00	SYNCH I/O AVG. : N/C
				COMMIT PH1 WRITE I/O	0.000000	0.00	
NOT ACCOUNT.	N/A	0.001075	N/A	ASYNCH CF REQUESTS	0.000000	0.00	
DB2 ENT/EXIT	N/A	14.00	N/A	TOTAL CLASS 3	0.000031	0.06	
EN/EX-STPROC	N/A	0.00	N/A				
EN/EX-UDF	N/A	0.00	N/A				
DCAPT.DESCR.	N/A	N/A	N/P				
LOG EXTRACT.	N/A	N/A	N/P				

The number of occurrences and commits per second are 771852/600 = 1286.42, and it should be close to the RMF number of enclaves per second (Example 9-6 on page 243).

- ▶ Class 2 CPU utilization is 1286.42 * 0.000714 = 0.9185, or 9.185 percent of one CP.
- ▶ Class 1 average elapsed time per commit is 4 milliseconds, and it should correspond with RMF average transaction (enclave) response time.

Example 9-10 shows the distributed activity from the DB2 PE accounting report.

Example 9-10 DB1 PE Distributed Activity report

---- DISTRIBUTED ACTIVITY ----						
REQUESTER	: 9.12.4.177	TRANSACTIONS RECV. :	0.00	MESSAGES SENT :	2.00	MSG.IN BUFFER: 1.00
PRODUCT ID	: JCC	#COMMIT(1) RECEIVED:	416796	MESSAGES RECEIVED:	2.00	ROWS SENT : 0.00
METHOD	: DRDA PROTOCOL	#ROLLBK(1) RECEIVED:	0	BYTES SENT :	798.90	BLOCKS SENT : 2.00
CONV.INITIATED	: 0.00	SQL RECEIVED :	2.00	BYTES RECEIVED :	754.00	#DDF ACCESSES: 416735
#COMMIT(2) RECEIVED:	0	#COMMIT(2) RES.SENT:	0	#PREPARE RECEIVED:	0	#FORGET SENT : 0
#BCKOUT(2) RECEIVED:	0	#BACKOUT(2) RES.SENT:	0	#LAST AGENT RECV.:	0	
#COMMIT(2) PERFORM.:	0	#BACKOUT(2) PERFORM.:	0	#THREADS INDOUBT :	0	
REQUESTER	: 9.12.4.178	TRANSACTIONS RECV. :	0.00	MESSAGES SENT :	2.00	MSG.IN BUFFER: 1.00
PRODUCT ID	: JCC	#COMMIT(1) RECEIVED:	355056	MESSAGES RECEIVED:	2.00	ROWS SENT : 0.00
METHOD	: DRDA PROTOCOL	#ROLLBK(1) RECEIVED:	0	BYTES SENT :	798.96	BLOCKS SENT : 2.00
CONV.INITIATED	: 0.00	SQL RECEIVED :	2.00	BYTES RECEIVED :	754.05	#DDF ACCESSES: 354978
#COMMIT(2) RECEIVED:	0	#COMMIT(2) RES.SENT:	0	#PREPARE RECEIVED:	0	#FORGET SENT : 0
#BCKOUT(2) RECEIVED:	0	#BACKOUT(2) RES.SENT:	0	#LAST AGENT RECV.:	0	
#COMMIT(2) PERFORM.:	0	#BACKOUT(2) PERFORM.:	0	#THREADS INDOUBT :	0	

The number of commits is split between the two application servers, and they match the number of DDF accesses.

DDF middleware is enabled for ARM instrumentation, and a pair of arm_start_transaction, arm_stop_transaction is associated with each database request message from a distributed application—in our environment the WebSphere Application Servers (hop1). You can see that the number of messages received per commit is 2.00 from both servers.

The total number of messages received per second is ((416796 * 2.0)+(355056 * 2.02)) / 600 = 2572.75, which is very close to the 2594.88 successful transactions per second for DDF middleware (hop 2) given by EWLM Control Center, as shown in Figure 9-33 on page 241 and Figure 9-36 on page 242.

The DB2 PE statistics report also has a section related to DRDA. Example 9-11 shows the report excerpt.

Example 9-11 DB2 PE statistics report

1	LOCATION: DB8E	DB2 PERFORMANCE EXPERT (V2)	PAGE: 1-15
	GROUP: N/P	STATISTICS REPORT - LONG	REQUESTED FROM: ALL 10:20:00.00
	MEMBER: N/P		TO: DATES 10:30:00.00
	SUBSYSTEM: DB8E		INTERVAL FROM: 11/21/05 10:20:23.22
	DB2 VERSION: V8	SCOPE: MEMBER	TO: 11/21/05 10:29:30.59

---- HIGHLIGHTS -----

INTERVAL START :	11/21/05 10:20:23.22	SAMPLING START:	11/21/05 10:20:23.22	TOTAL THREADS :	0.00
INTERVAL END :	11/21/05 10:29:30.59	SAMPLING END :	11/21/05 10:29:30.59	TOTAL COMMITS :	708.7K
INTERVAL ELAPSED:	9:07.363693	OUTAGE ELAPSED:	0.000000	DATA SHARING MEMBER:	N/A

DRDA REMOTE LOCS	SENT	RECEIVED
-----	-----	-----
TRANSACTIONS	0.00	2.00
CONVERSATIONS	0.00	2.00
CONVERSATIONS QUEUED	0.00	
SQL STATEMENTS	0.00	1417.4K
SINGLE PHASE COMMITS	0.00	708.7K
SINGLE PHASE ROLLBACKS	0.00	0.00
ROWS	0.00	0.00
MESSAGES	1417.5K	1417.5K
BYTES	566.2M	534.3M
BLOCKS	1417.5K	0.00
MESSAGES IN BUFFER	708.7K	
CONT->LIM.BLOCK FETCH SWTCH	0.00	
STATEMENTS BOUND AT SERVER	0.00	
PREPARE REQUEST	0.00	0.00
LAST AGENT REQUEST	0.00	0.00
TWO PHASE COMMIT REQUEST	0.00	0.00
TWO PHASE BACKOUT REQUEST	0.00	0.00
FORGET RESPONSES	0.00	0.00
COMMIT RESPONSES	0.00	0.00
BACKOUT RESPONSES	0.00	0.00
THREAD INDOUBT-REM.L.COORD.	0.00	
COMMITTS DONE-REM.LOC.COORD.	0.00	
BACKOUTS DONE-REM.L.COORD.	0.00	

The number of commits and messages received per second match the accounting report:

- ▶ Commits: 708700 / 600 = 1181
- ▶ Messages received = 1417500 / 600 = 2362.5

In the EWLM Application Topology, the transaction count field represents the number of ARM transactions executed in the DB2 middleware and it should correspond to the distributed messages reported in the DB2 PE reports.

If you sum the transactions spread across the two service classes serving the trade application, you get 1556932 total transactions (2594.88 transaction/second), which is very close to the DB2 report. Figure 9-33 on page 241 and Figure 9-36 on page 242 show the EWLM Application topology table view for the measured interval.

9.4.7 Result and considerations

From our experiences with our ITSO EWLM Management Domain, we offer some tips for implementing EWLM in your installation:

- ▶ Carefully plan your EWLM policy versus the WLM policy if you want to relate performance data between the z/OS and the distributed environment.

- ▶ When you evaluate data, either from EWLM or from the z/OS monitors, keep in mind the different concept of a *transaction*, as illustrated in Figure 9-39.

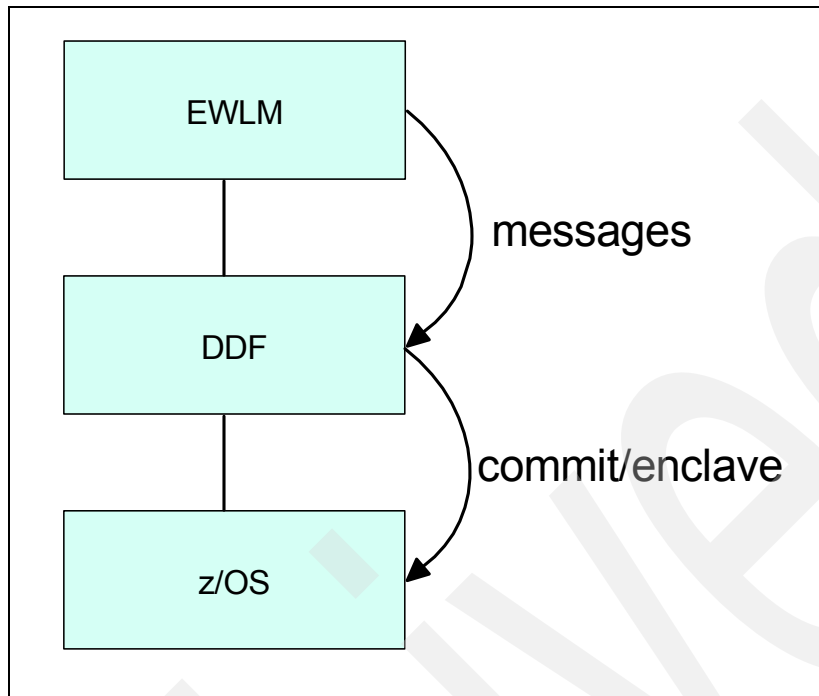


Figure 9-39 Transaction concept



EWLM traces and logs

This chapter provides information about resources and tools to help you diagnose problems and debug your systems.

In this chapter, we describe:

- ▶ The method of finding the performance problem that EWLM has monitored
- ▶ Commands to confirm the current properties
- ▶ Virtualization Engine and EWLM logs
- ▶ EWLM traces and snapshots
- ▶ AIX syslog facility
- ▶ ARM Serviceability Adapter

10.1 EWLM Control Center

In this section, we describe the method of finding the performance problem that EWLM has monitored.

The EWLM Control Center can be used to monitor the Management Domain when a service policy has been activated, as described in 5.2.3, “Building the Domain Policy” on page 141. From the Monitor option, you can check the current performance status of the Management Domain from various points of view, such as service classes, transaction classes, process classes, and partition classes. You can also check the status of managed servers, such as active, or whether there is a communication error. The status of Load Balancers and LPAR partitions can also be checked. From the sub menus, you can check more detailed information regarding the various monitoring results.

The EWLM Control Center provides a consolidated view of how well things are performing across your Management Domain. If the performance goal is not met (the PI is greater than 1) in a service class, it will show up in the Exceptions Report page, as shown in Figure 10-1. You can select an item from the pull-down list to view the following:

- ▶ Service class details
- ▶ Hop details
- ▶ Application topology
- ▶ Server topology
- ▶ Performance index monitor
- ▶ Goal achievement monitor
- ▶ Transaction rate monitor

Figure 10-1 shows an example of the exception report.

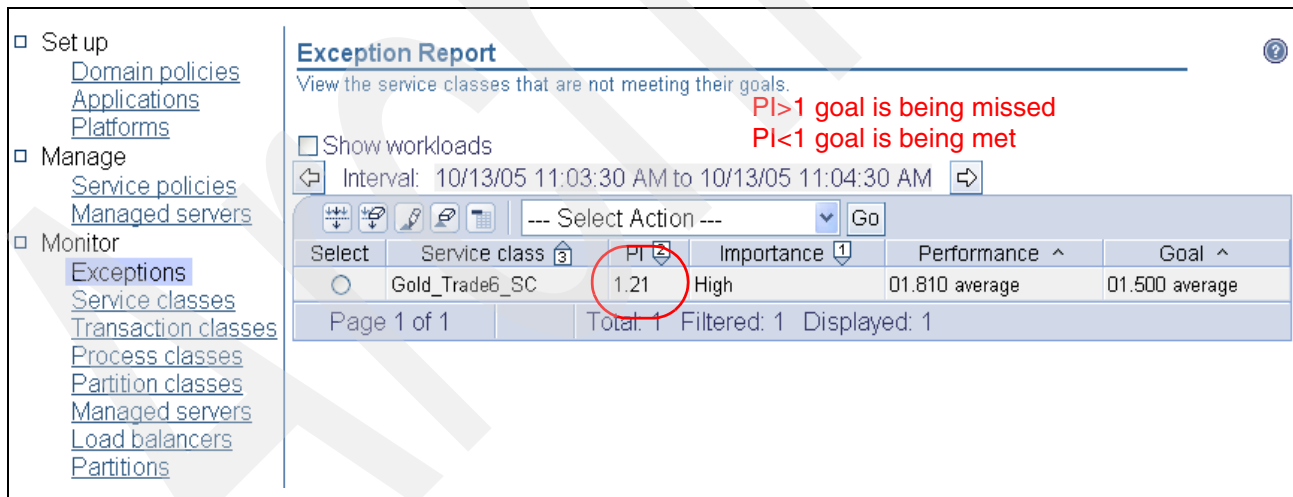


Figure 10-1 Exception report panel

Note: EWLM Control Center requires JRE1.4.2 to display the application topology and the server topology, and requires Adobe SVG Viewer to display the graph of details of the Service class. Refer to the 2.3, “Before installing” on page 22.

10.2 Commands

In this section, we describe the way to confirm the current properties for the elements of the EWLM domain. In troubleshooting and problem determination, it is important to understand these properties.

You can display the current properties for the following elements of an EWLM domain by using display commands.

displayDM	Displays the current properties of the domain manager. This includes serviceability options such as trace sizes, failure limits, trace levels, and dump retention age.
displayMS	Displays the current properties settings of the managed server. This includes serviceability options such as trace sizes, failure limits, trace levels, and dump retention age.
displayFB	Displays the current properties of the Firewall Broker.
displaySSEWLM	Displays the current properties of the Single System EWLM.
displayCC	Displays the ports, users, groups, and security currently used by WebSphere Application Server for the EWLM Control Center.

You can also use the configuration wizard GUI to display most of the current properties of domain manager and managed server just like using the **display** command. For more details, see 2.4, “Virtualization Engine common components installation” on page 29. If you want to know the serviceability options such as trace sizes, failure limits, trace levels, and dump retention age, use the **display** command.

To run the **display** command, use the configuration ID for an EWLM instance as a parameter.

Example 10-1 shows a sample output of the **displayDM** command and Example Example 10-2 shows a sample output of the **displayMS** command. The properties are displayed as abbreviations; therefore, we added the meanings of them on the right hand.

Example 10-1 Sample output of displayDM command and the meanings of each property

```
C:\Program Files\ibm\VE2\EWLM\bin> displayDM.bat DM1
```

auth(None)	Level of authorization between the DM and the MS or firewall broker
sslks(null)	The keystore path if security is used
sslpw(***password suppressed***)	Password to access the keystore
ma(ewlm1.itso.ibm.com)	DM ip address or dns name
mp(5773)	Port used by the dm to listen from the MS
jp(5800)	Port used to communicate between DM and CC
dn(ITSO Domain)	Domain name
fp(null)	Port used by DM to listen from the firewall broker
fb(null)	Firewall broker/SOCKS server ip address list
sa(null)	IP address of the SOCKS server
sp(null)	Port of the SOCKS server
lbp(Off)	Load balancing public port
lbs(Off)	Load balancing Secure port
nt(250)	Number of communication trace messages
ct(250)	Number of component trace messages
et(250)	Number of Event Trace messages
rt(250)	Number of reporting trace messages
ml(250)	Number of messages in the log
lbt(250)	Number of Load balancing trace messages
tcomm(0)	Trace distHub Broker enable/disable
tl(Min)	Trace level

tlog(Off)	PlugIn trace
dpa(30)	Dump retention age (days)
dpn(25)	Dump retention quantity
fl(50)	Failure limit

Example 10-2 Sample output of displayMS command and the meanings of each property

aixve01 @/opt/IBM/VE2/EWLMMS/bin> ./displayMS.sh AIXMS

auth(None)	Level of authorization between the DM and the MS or firewall broker
sslks(null)	The keystore path if security is used
sslpw(**password suppressed**)	Password to access the keystore
ma(9.12.4.141)	DM ip address or dns name
mp(5773)	Port used by the DM to listen from the MS
va(null)	Proxy firewall ip address or dns name
vp(null)	Proxy firewall port
jp(Off)	Port used to communicate between DM and CC
nt(250)	Number of communication trace messages
ct(250)	Number of component trace msgs
et(250)	Number of Event Trace msgs
jt(250)	Number of the platform interface trace messages
ml(250)	Number of messages in the log
tl(Max)	Trace level
tlog(Off)	PlugIn trace
dpa(30)	Dump retention age (days)
dpn(25)	Dump retention quantity
fl(50)	Failure limit

Note: Stop the EWLM service before displaying its property. For each display command, there is a corresponding change command to alter the values from the display command, if needed. The configuration commands do not allow for dynamic changes; therefore, you need to stop the EWLM service, issue the display or change commands, then restart the service.

10.3 Virtualization Engine and EWLM logs

This section contains lists of the log files that are created after you install and configure the EWLM. Log files contain various levels of information about the processing of the product and other software that is used to complete a task. These log files are grouped into the following categories:

- ▶ Virtualization Engine and EWLM domain manager installation log files
- ▶ EWLM managed server installation log files
- ▶ EWLM configuration log files
- ▶ Application server log files
- ▶ Web server log files
- ▶ Database log files
- ▶ Directory server log files

We provide the summary table of the log files in 10.5, “Summary table of log files” on page 265. If you want to confirm them quickly, refer to the section.

10.3.1 Virtualization Engine and EWLM domain manager installation log files

Virtualization Engine installer log messages and EWLM domain manager installation messages are sent to the same log files.

Error messages generated during the install are sent to installLog.txt. Both InstallShield and Virtualization Engine's installer errors are written to this file.

Debug messages generated during install are sent to installDebug.txt. Both InstallShield and Virtualization Engine's installer debug information is written to this file.

For example, the log files are created in the following directory.

- ▶ Windows - <Virtualization Engine-install-location>/Logs, where <Virtualization Engine-install-location> is the installation root directory of the Virtualization Engine products. By default it is C:\Program Files\IBM\VE2.
- ▶ AIX - /var/adm/sw

For z/OS, you can check in the joblog output of the SMP/E RECEIVE and APPLY jobs.

Example 10-3 shows a sample of installLog.txt when install failed because of no space left on device.

Example 10-3 Sample of installLog.txt

```
(Oct 17, 2005 1:41:42 PM), Install, com.ibm.ve.install.common.BootstrapJRE, err,
java.io.IOException: No space left on device
(Oct 17, 2005 1:41:42 PM), Install, com.ibm.ve.install.common.UnpackerThread, err, No space
left on device
(Oct 17, 2005 1:41:43 PM), Install, com.installshield.wizardx.actions.TriggerEventAction,
wrn, java.io.IOException: /cdrom/FILES/bin/java: not found
(Oct 17, 2005 1:41:43 PM), Install, com.ibm.ve.install.common.BootstrapSI, err,
java.io.IOException: /cdrom/FILES/bin/java: not found
(Oct 17, 2005 1:41:43 PM), Install, com.installshield.wizardx.actions.TriggerEventAction,
wrn, java.io.IOException: /cdrom/FILES/bin/java: not found
(Oct 17, 2005 1:41:43 PM), Install, com.ibm.ve.install.common.BootstrapSI, err,
java.io.IOException: /cdrom/FILES/bin/java: not found
(Oct 17, 2005 1:41:44 PM), Install, com.ibm.ve.install.common.VEHelp, wrn,
java.lang.Exception: Unknown exception.
(Oct 17, 2005 1:58:05 PM), Install, com.ibm.ve.install.common.PrivateData, err,
java.io.IOException: No space left on device at java.io.FileOutputStream.writeBytes(Native
Method)
```

Summary information from the common runtime installers on Windows is saved to <Virtualization Engine-install-location>/Logs/install/CRTInstallSummary.txt.

Properties set during the install that can be useful in determining the state of the resultant install on Windows are saved to <Virtualization Engine-install-location>/VEinstall.properties.

WebSphere Application Server logs

Install logs of the WebSphere Application Server, which EWLM utilizes, are written to <Virtualization Engine-install-location>/WebSphere/AppServer/logs/log.txt, where <Virtualization Engine-install-location> is C:\Program Files\IBM\VE2 on Windows and /opt/IBM/VE2 on AIX.

Virtualization Engine installs fixes for the WebSphere Application Server during the install. The failure of the fix installation causes Virtualization Engine's WebSphere Application Server installation to fail. These log files are located under <Virtualization Engine-install-location>/WebSphere/AppServer/logs/update. Each fix has a directory that contains its logs and you will find the log files in a *tmp* or *unknown* directory if a fix install has failed. The main log file for a fix install is named *update.log.txt*.

IBM Tivoli Directory Server logs

The `ldapinst.log`, which is in the installation log of the ITDS, is in the `<Virtualization Engine-log-directory>\ITDS` on Windows, where `<Virtualization Engine-log-directory>` is the installation log directory of the Virtualization Engine products. For example, it is located in `C:\Program Files\IBM\VE2\Logs\ITDS\ldapinst.log`.

The logs written while doing post-installation configuration tasks can be found in the following locations:

- ▶ Linux/AIX: `/var/idsldap/V6.0` and some are in `/<OwnerHomeDir>/<idsldap-ownerid>/logs`
- ▶ Windows: `<install-location>\ldap\V6.0\var` (and ITDS also puts a copy of `ldapinst.log` to the directory) and in `C:\<idsldap-ownerID>\logs`

10.3.2 Configuration summary logs

You can see the summary information regarding successful configuration of domain manager and managed servers that have been created or updated by using common commands such as `createdM` and `changeCC`.

The information includes executed date and time, command name, user ID, and parameters. It is shown in Figure 10-2 on page 255.

The information is added to the existing log file. Therefore, it is useful to confirm the history of configuring domain manager and managed server.

For example, the log files that contain how the commands were executed are:

- ▶ Domain manager
 - Windows - `C:\Program Files\ibm\VE2\EWLM\ewlmdm\logs\config.log`
 - i5/OS - `<user_specified_directory>/logs/config.log`, where `<user_specified_directory>` is addressed by the `EWLM_DATA_ROOT` variable set in the `/etc/ewlmdm_environment.conf`
 - z/OS - `<user_specified_directory>/logs/config.log`, where `<user_specified_directory>` is addressed by the `EWLM_DATA_ROOT` variable set in the `/etc/ewlmdm_environment.conf`
- ▶ Managed server
 - Windows - `C:\Program Files\ibm\VE2\EWLMMS\logs\config.log`
 - AIX - `/var/adm/ewlm/ewlmms/config.log`
 - i5/OS - `<user_specified_directory>/logs/config.log`, where `<user_specified_directory>` is addressed by the `EWLM_DATA_ROOT` variable set in the `/etc/ewlmms_environment.conf`
 - z/OS - `<user_specified_directory>/logs/config.log`, where `<user_specified_directory>` is addressed by the `EWLM_DATA_ROOT` variable set in the `/etc/ewlmdm_environment.conf`

```
9.12.4.176 - PuTTY
aixve01@/var/adm/ewlm/ewlmms> cat config.log
Monday, September 26, 2005 12:19:14 PM CDT  command: createMS  user: root
  ConfigID: AIXMS  Parameters: -ma 9.12.4.141  -mp 5773  -auth None

Thursday, October 6, 2005 10:31:49 AM CDT  command: createMS  user: root
  ConfigID: zOSMS  Parameters: -ma 9.12.6.22  -mp 5773  -auth None

aixve01@/var/adm/ewlm/ewlmms> █
```

Figure 10-2 Sample of config.log

10.3.3 Common command logs

For each common command, there are corresponding log files that get created in the Diagnostics directory. The common commands are:

- ▶ createXX
- ▶ changeXX
- ▶ deleteXX
- ▶ displayXX
- ▶ startXX
- ▶ stopXX
- ▶ wizardDM, wizardMS

Where XX should be replaced by each of the EWLM components:

- ▶ DM - Domain Manager
- ▶ MS - Managed Server
- ▶ CC - EWLM Control Center
- ▶ WAS - WebSphere Application Server
- ▶ FB - Firewall Broker
- ▶ SSEWLM - single system EWLM

The log information of these commands is written to new log files, which are created each time these commands are executed. When you execute the command, a new log file, which has the name followed by thirteen numeric characters, is created, as shown Figure 10-3. The file can be distinguished at the date and time when it has been created.

For example, the log files are located in:

- ▶ Domain manager
 - Windows - C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\Diagnostics
 - z/OS - <user_specified_directory>/logs/Diagnostics, where <user_specified_directory> is addressed by the EWLM_DATA_ROOT variable set in the /etc/ewlmdm_environment.conf
- ▶ Managed server
 - Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\logs\Diagnostics
 - AIX - /var/adm/ewlm/ewlmms/Diagnostics
 - z/OS - <user_specified_directory>/logs/Diagnostics, where <user_specified_directory> is addressed by the EWLM_DATA_ROOT variable set in the /etc/ewlmms_environment.conf

```

9.12.4.176 - PuTTY
aixve01@/var/adm/ewlm/ewlmms/Diagnostics> ls -ltr
total 120
-rw-r----- 1 root    system    4394 Sep 26 12:17 wizardMS__1127755053262.log
-rw-r----- 1 root    system    3837 Sep 26 12:19 createMS__1127755130381.log
-rw-r----- 1 root    system    5379 Sep 26 12:22 startMS__1127755348516.log
-rw-r----- 1 root    system    5378 Oct  5 10:16 startMS__1128525362534.log
-rw-r----- 1 root    system    4394 Oct  6 10:31 wizardMS__1128612663806.log
-rw-r----- 1 root    system    3836 Oct  6 10:31 createMS__1128612690625.log
-rw-r----- 1 root    system    3181 Oct  6 10:49 stopMS__1128613770057.log
-rw-r----- 1 root    system    3193 Oct  6 18:34 displayMS__1128641668801.log
-rw-r----- 1 root    system    5378 Oct  7 09:27 startMS__1128695239790.log
-rw-r----- 1 root    system    3766 Oct 11 13:58 changeMS__1129057097225.log
aixve01@/var/adm/ewlm/ewlmms/Diagnostics>
  
```

Figure 10-3 Sample list of common command logs

10.3.4 Configuration Wizard logs

For configuration wizard command, there is a corresponding log file that gets created each time you use the wizard. The log file contains the last information that you create or change to the domain manager and managed server by using the wizard. The commands to kick the wizard are:

- ▶ **configWizardDM**
- ▶ **configWizardMS**

The corresponding log files are:

- ▶ ConfigurationWizard.log

For example, the log files are located in:

- ▶ Domain manager
 - Windows - C:\Program Files\ibm\VE2\EWLM\ewlmdm\logs\ConfigWizardDiagnostics
- ▶ Managed server
 - Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\logs
 - AIX - /var/adm/ewlm/ewlmms/ConfigWizardDiagnostics

10.3.5 WebSphere Application Server logs

The Control Center runs on the WebSphere Application Server instance; therefore, you need to check the WebSphere Application Server log files when you encounter the problems in Control Center.

Here is the list of log types and where they are stored:

- ▶ JVM logs are created by redirecting the System.out and System.err streams of the JVM. The System.out log is used to monitor the health of the running application server. The System.err log contains exception stack trace information.

By default, these files are stored as

<EWLMDM_HOME>/servers/<DM_ConfigID>/WAS/logs/<server_name>/SystemOut.log and SystemErr.log.

- ▶ Process (native) logs are created by redirecting the stdout and stderr streams of the process's native module (.dlls, .so, UNIX libraries, and other JNI native modules), including the JVM native code itself. These logs can contain information relating to problems in native code, or diagnostic information written by the JVM.

By default, these files are stored as <EWLMDM_HOME>/servers/<DM_ConfigID>/WAS/logs/<server_name>/native_stderr.log and native_stdout.log.

For example, the log files on a Windows system might look like this:

C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\WAS\logs\server1\SystemOut.log and SystemErr.log

10.3.6 Error logs

If an error occurs while EWLM is starting, causing EWLM to terminate, the error logs will go in <EWLM_root>\Diagnostics.

10.3.7 Middleware components log files

All middleware components contain log files, including:

- ▶ Web server log files
- ▶ Database server log files
- ▶ Directory server log files

Refer to the documentation associated with the middleware for information about what log files are available to use for troubleshooting.

10.4 EWLM traces and snapshots

EWLM keeps internal states and various information in memory. That information can be written to a file as trace or snapshot by user operation from EWLM Control Center. These files are for IBM support use and are not intended to be analyzed by users.

EWLM provides the logging facilities to the following trace and snapshot information.

- ▶ Trace of algorithm information
- ▶ Trace and snapshot of domain manager and managed server information
- ▶ Trace of EWLM Control Center information

We provide the summary table of the log files that contains trace and snapshot files in 10.5, “Summary table of log files” on page 265. If you want to confirm them quickly, refer to that section.

10.4.1 Trace of algorithm

You can log the algorithm that EWLM uses when making resources adjustments to managed servers that belong to partition workload groups. Algorithm information provides details on how the EWLM domain manager calculated the data that it receives from the managed servers. The algorithm information can be written out to a trace file. This file is continuously updated with algorithm information as the domain manager calculates the data that it receive from the managed servers.

There are two places where algorithm trace logging should be set, the domain manager page and the partition workload groups page of EWLM Control Center, as shown Figure 10-4 on page 259 and Figure 10-5 on page 260. When you enable the logging facility on the domain manager page shown Figure 10-4 on page 259, the trace file will be created on the domain manager diagnostic folder. When you enable the facility on the partition workload groups page shown Figure 10-5 on page 260, the trace file will be created on each managed server diagnostic folder.

For example, the trace file that gets created in the domain manager is located in Windows in C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\Diagnostics\AlgTrace_YYYY_MM_DD_SS_OOO_CDT.dat.

For example, the trace file that gets created in managed server is located in AIX /ewlmData/ewlmms/servers/AIXMS/Diagnostics/AlgTrace_YYYY_MM_DD_SS_OOO_CDT.d

Where YYYY_MM_DD_SS is year_month_date_seconds and where OOO is three numeric characters.

Note: This trace is used for EWLM partition management. Currently, partition management is supported on IBM System p5 and eServer p5 systems. Therefore, the algorithm trace files are only generated for managed servers running on IBM System p5 and eServer p5 systems.

Setting of logging algorithm trace on domain manager

To create the trace file of algorithm information about domain manager, do the following steps and refer to the Figure 10-4:

1. Click **Domain Settings** in the EWLM Control Center home page.
2. Click **Serviceability** from the menu list on the left-hand pane.
3. Check **Log algorithm trace information**.
4. Click **OK**.
5. Recreate the problem by generating transactions.
6. To stop the trace, uncheck **Log algorithm trace information** and click **Go**.

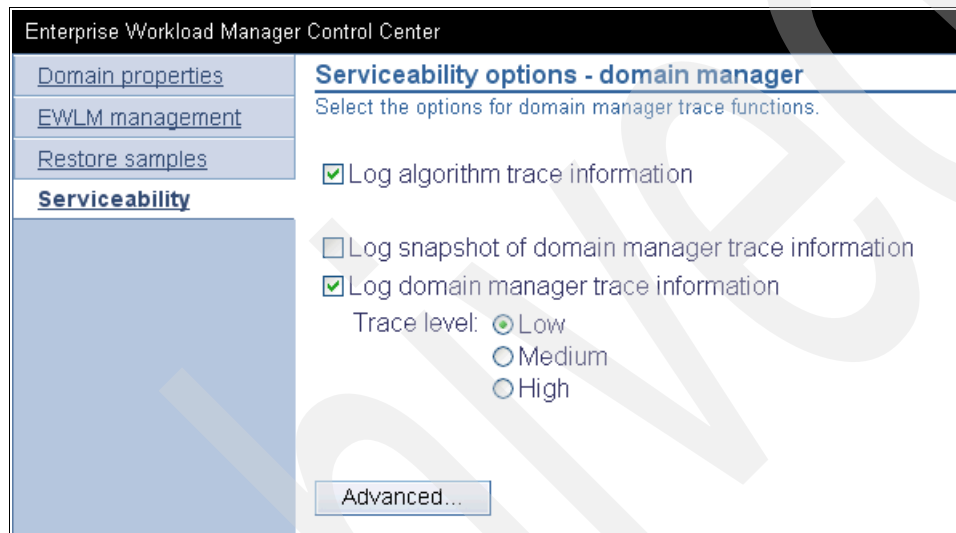


Figure 10-4 Domain manager traces panel

Setting of logging algorithm trace on partition workload groups

To create trace files of algorithm information about managed servers of partition workload groups, do the following steps and refer to Figure 10-5.

1. Click **Advanced** in the Serviceability options - domain manager page.
2. Click **Partition workload groups** from the menu list on the left-hand pane, and the Advanced serviceability of partition workload groups page appears in the right-hand pane.
3. Select the partition workload group that you want to make trace files.
4. Select the Enable function from the pull-down list and click **Go**. Or check the **Algorithm logging** column.
5. Click **OK**.
6. Recreate the problem by generating transactions.
7. To stop the trace, select the Disable function and click **Go**, or uncheck the **Algorithm logging** column, and then click **OK**.

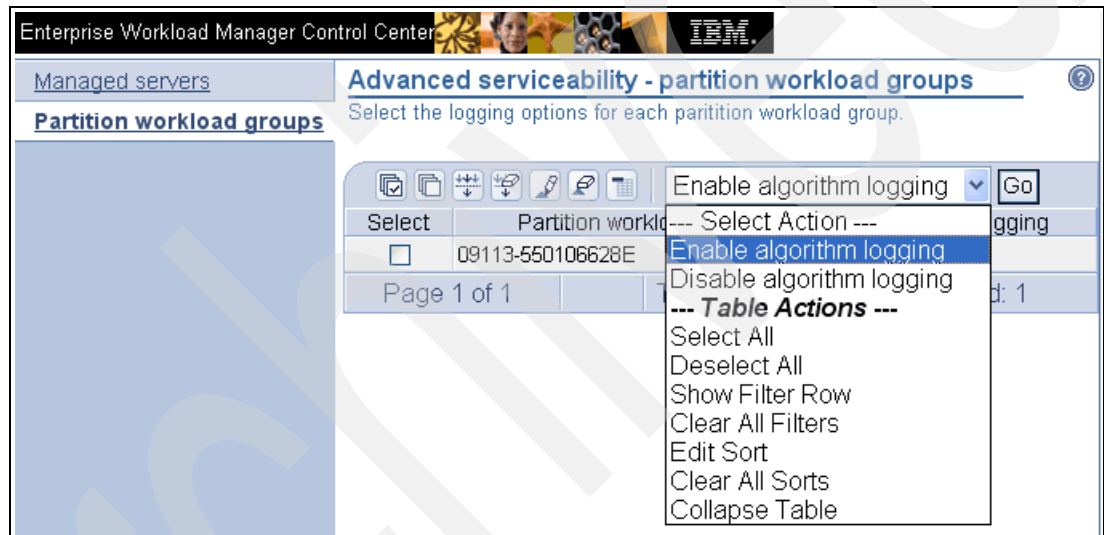


Figure 10-5 Partition workload group traces panel

Note: After the required tracing information has been gathered, you should disable the logging trace. The trace file can become very large.

10.4.2 Trace and snapshot of domain manager and managed server

You can generate log files of trace and a snapshot of the domain manager and the managed server from the EWL Control Center.

Snapshot of domain manager and managed server

Snapshot captures the internal states and information of domain manager and managed server currently existing in memory and outputs them to a file in XML format. The information includes such things as the following:

- ▶ State of the managed server
- ▶ Activity policy
- ▶ Operating system version and release
- ▶ Java properties

For example, the snapshot file of a domain manager is located in:

- ▶ Windows - C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\Diagnostics\EWLM_diagnostics_NNN.xml

For example, the snapshot file of a managed server is located in:

- ▶ Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\servers\WinDM1\EWLM_diagnostics_NNN.xml
- ▶ AIX - /ewlmData/ewlmms/servers/AIXMS/Diagnostics/EWLM_diagnostics_NNN.xml

Where NNN is thirteen numeric characters.

When EWLM detects a problem, EWLM automatically requests a dump. The information when the dump request has been initiated or completed is added to the files named System.log and Service.log. Therefore, you can use these files to monitor the health of EWLM. If System.log reaches the maximum size of 1 MB, the file is renamed as Previous.log. If a Previous.log file already exists, EWLM deletes it from the file system.

For example, the log file of the domain manager is located in:

- ▶ Windows - C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\System.log, Previous.log and Service.log

For example, the log file of the managed server is located in:

- ▶ Windows - C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\System.log, Previous.log and Service.log
- ▶ AIX - /ewlmData/ewlmms/servers/AIXMS/System.log, Previous.log and Service.log

Trace of domain manager and managed server

Trace information contains data such as the state of the domain manager and managed servers and the performance data that the domain manager sends to the EWLM Control Center. It can be written out to a log file in XML format.

For example, the trace file of the domain manager is located in:

- ▶ Windows - C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\Diagnostics\EWLM_trace_NNN.xml

For example, the trace file of managed server is located in:

- ▶ Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\servers\WinDM1\EWLM_trace_NNN.xml
- ▶ AIX - /ewlmData/ewlmms/servers/AIXMS/Diagnostics/EWLM_trace_NNN.html

Where NNN is thirteen numeric characters.

Setting of logging domain manager trace and snapshot

To create trace and snapshot files of a domain manager, do the following steps and refer to the Figure 10-6:

1. Click **Domain Settings** in the EWLM Control Center home page.
2. Click **Serviceability** from the menu list on the left-hand pane.
3. Check the items that you want to make files. To create trace files of domain manager, you can choose trace level. Possible values are low, medium, and high.
4. Click **OK**.
5. Recreate the problem by generating transactions.
6. To stop the trace, uncheck the items if you check **Log algorithm trace information** or **Log domain manager trace information**, and then click **OK**.

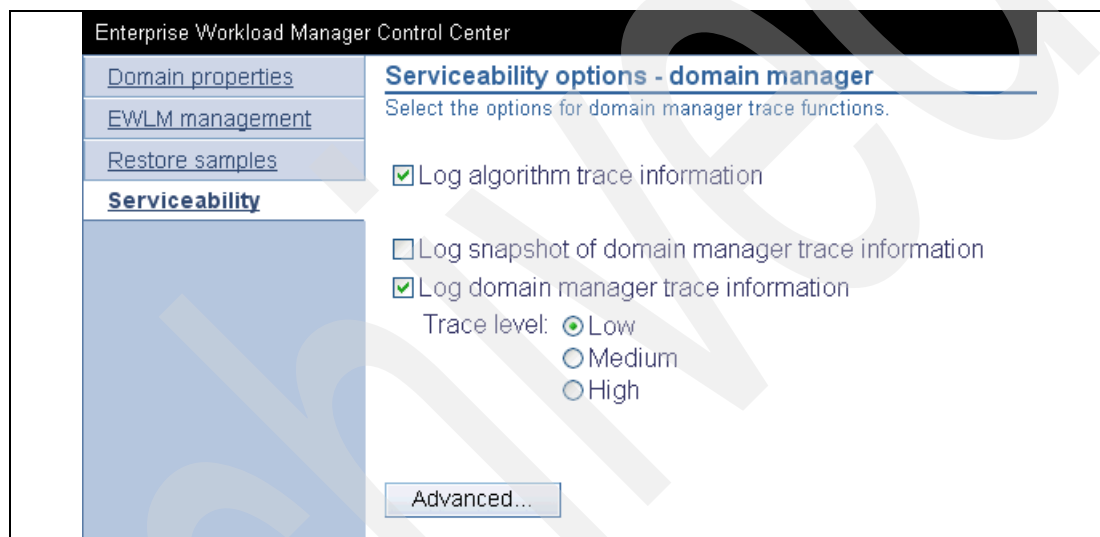


Figure 10-6 Domain manager traces panel

Note: After EWLM creates the log file of snapshot, the field is automatically deselected. You must select it again if you want to create another snapshot log file.

Setting of logging managed server trace and snapshot

To create trace and snapshot files of managed server trace information, do the following steps and refer to Figure 10-7:

1. Click **Domain Settings** in the EWLM Control Center home page.
2. Click **Serviceability** from the menu list on the left-hand pane.
3. Click **Advanced** in the Serviceability options - domain manager page.
4. Make sure that **Managed servers** are selected, and the Advanced serviceability of managed server page appears in the right-hand pane.
5. Select the managed servers that you want to make files.
6. Select the Enable function from the pull-down list, and click **Go**.
7. Click **OK**.
8. Recreate the problem by generating transactions.
9. To stop the trace, select the Disable function if you enabled log managed server trace information.

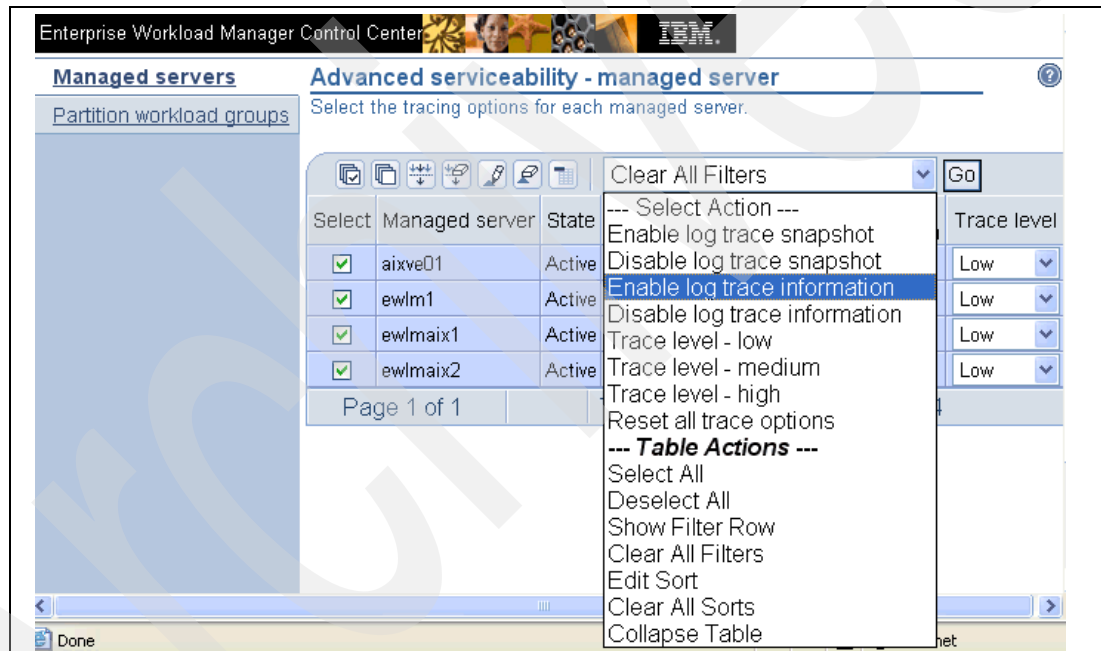


Figure 10-7 Managed server traces panel

After EWLM creates the snapshot file, the snapshot field is automatically deselected. You must select it again if you want to create another snapshot log file.

You can configure not only using the pull-down list but also using the check box.

Select the managed servers that you want to make files. If you want a snapshot, check **Log trace snapshot**. If you want trace information, check **Log trace information**. Next, select trace level from the pull-down list of Trace Level column. Click **OK**. When you configure few servers or when you configure various settings on each managed server, this step is simpler and more useful.

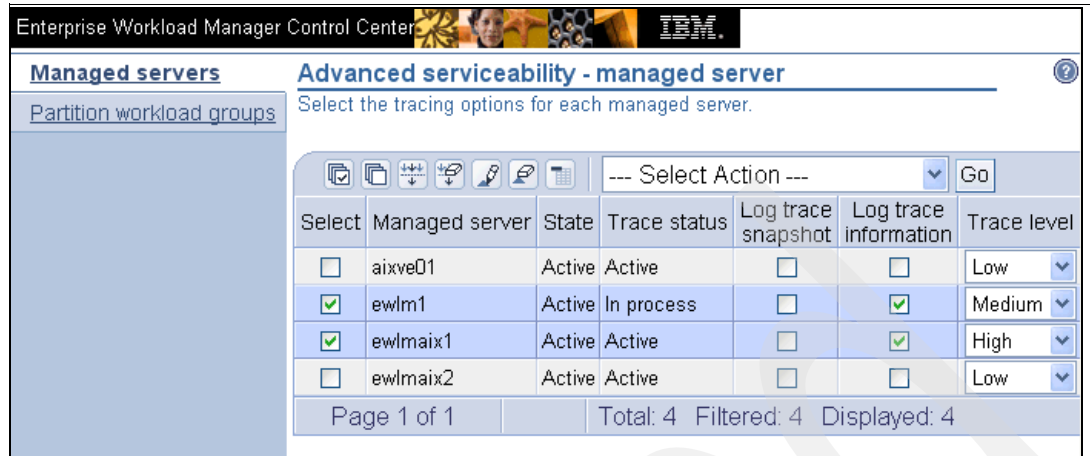


Figure 10-8 Managed server traces panel

Note: If the Trace status shows In process, EWLM indicates that the status of the logging request itself is in process, it does not indicate the status of trace. Because traces themselves of the managed servers are always stored in memory, you cannot stop it. After EWLM completes the request, this field turns to show Active.

10.4.3 EWLM Control Center traces

You can log trace information of the Control Center by using the **changeCC** command from the server where the domain manager is located. The WebSphere Application Server instance under which the EWLM Control Center runs must be active before you run this command. The information is sent to WebSphere Application Server logs, which are SystemOut.log and SystemErr.log. For more details about these log files, see 10.3.5, “WebSphere Application Server logs” on page 257.

This logging facility is not enabled by default. To enable this, do the following:

```
changeCC -trace configID -adminUser userid -adminPW password -state [None | Event | All]
```

Where each parameter is defined as follows:

► **adminUser**

The user ID of the administrator of the WebSphere Application Server instance on which the EWLM Control Center runs.

► **adminPW**

The password associated with the WebSphere Application Server administrator user ID.

► **state**

- None: No tracing is enabled.
- Event: Event tracing is enabled.
- All: Full tracing, including event tracing, is enabled.

The state is reset to None every time WAS is restarted. When you restart WAS, you must set it again if you want to create another log file of trace.

10.5 Summary table of log files

As we describe in 10.3, “Virtualization Engine and EWLM logs” on page 252, and 10.4, “EWLM traces and snapshots” on page 257, various log files are available to help debug problems you may encounter. This section provides the summary table of the log files so that you can confirm them quickly. Refer to the each section for details of each log file.

Table 10-1 shows Virtualization Engine and EWLM Domain manager log files. The directories are default values. Replace them to the values you specified during install if you change them.

Table 10-1 Virtualization engine and domain manager installation log files

File name	Directory (for example)	Description
installLog.txt	Windows - C:\Program Files\IBM\VE2\Logs AIX - /var/adm/sw	Contains error information from InstallShield and Virtualization Engine's installer
installDebug.txt	Windows - C:\Program Files\IBM\VE2\Logs AIX - /var/adm/sw	Contains debug information from InstallShield and Virtualization Engine's installer
CRTInstallSummary.txt	Windows - C:\Program Files\IBM\VE2\Logs\install	Contains summary information from the CRT common runtime installers
Veinstall.properties	Windows - C:\Program Files\IBM\VE2\WebSphere\AppServer\logs\update	Contains CRT properties set during install
log.txt	Windows - C:\Program Files\IBM\VE2\WebSphere\AppServer\logs AIX - /opt/IBM/VE2/WebSphere/AppServer/logs	Contains information from WAS install
updateLog.txt	Windows - C:\Program Files\IBM\VE2\WebSphere\AppServer\logs\update AIX - /opt/IBM/VE2/WebSphere/AppServer/logs/update	Contains information from WAS fix install
Idapinst.log	Windows - C:\Program Files\IBM\VE2\Logs\ITDS	Contains information from ITDS install on Windows

Table 10-2 shows EWLM domain manager log files. The directories are default values. Replace them to the values you specified during install if you change them.

Table 10-2 EWLM domain manager log files

File name	Directory (for example)	Description
config.log	<ul style="list-style-type: none"> ▶ Windows - C:\Program Files\ibm\VE2\EWLM\ewlmdm\logs ▶ z/OS - <user_specified_directory>/logs 	Contains summary information from successful configuration by using common commands
changeDM_NNN.log ^a createDM_NNN.log ^a deleteDM_NNN.log ^a displayDM_NNN.log ^a startDM_NNN.log ^a startWAS_NNN.log ^a stopDM_NNN.log ^a stopWAS_NNN.log ^a wizardDM_NNN.log ^a	<ul style="list-style-type: none"> ▶ Windows - C:\Program Files\ibm\VE2\EWLM\ewlmdm\logs\Diagnostics ▶ z/OS - <user_specified_directory>/logs/Diagnostics 	Contains information from configuration by using common commands

File name	Directory (for example)	Description
configurationWizard.log	C:\Program Files\ibm\VE2\EWLM\ewlmdm\logs\ConfigWizardDiagnostics	Contains information from the creation and change of DM by using ConfigWizard
Previous.log Service.log System.log	C:\Program Files\ibm\VE2\EWLM\ewlmdm\logs\Diagnostics	Contains information from the dump request when EWLM detects a problem
AlgTrace_YYYY_MM_DD_SS_n_n_n_CDT. ^b dat	C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\Diagnostics	Contains information from algorithm Trace
EWLM_diagnostics_NNN.xml ^a	C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\Diagnostics	Contains information from snapshot of domain manager trace
EWLM_trace_NNN.xml ^a	C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\Diagnostics	Contains information from domain manager trace
SystemOut.log	C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\WAS\logs\server1	Contains JVM logs that is used to monitor the health of the running application server
SystemErr.log	C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\WAS\logs\server1	Contains process logs that includes exception stack trace information
native_stderr.log	C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\WAS\logs\server1	Contains information relating to problems in native code, or diagnostic information written by the JVM
native_stdout.log	C:\Program Files\ibm\VE2\EWLM\ewlmdm\servers\DM1\WAS\logs\server1	Contains information relating to problems in native code or diagnostic information written by the JVM

a. NNN is thirteen numeric characters.

b. YYYY_MM_DD_SS is year_month_date_seconds, and n_n_n is three numeric characters.

Table 10-3 shows EWLM managed server log files. The directories are default values. Replace them to the values you specified during install if you change them.

Table 10-3 EWLM managed server log files

Name	Directory (for example)	Description
config.log	<ul style="list-style-type: none"> ▶ Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\logs ▶ AIX - /var/adm/ewlm/ewlmms ▶ z/OS - <user_specified_directory>/logs 	Contains summary information from successful configuration by using common commands
changeMS_NNN.log ^a createMS_NNN.log ^a deleteMS_NNN.log ^a displayMS_NNN.log ^a startMS_NNN.log ^a stopMS_NNN.log ^a wizardMS_NNN.log ^a	<ul style="list-style-type: none"> ▶ Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\logs\Diagnostics ▶ AIX - /var/adm/ewlm/ewlmms/Diagnostics ▶ z/OS - <user_specified_directory>/logs/Diagnostics 	Contains information from configuration by using common commands

Name	Directory (for example)	Description
ConfigurationWizard.log	<ul style="list-style-type: none"> ▶ Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\logs ▶ AIX - /var/adm/ewlm/ewlmms/ConfigWizardDiagnostics 	Contains information from the creation and change of MS by using ConfigWizard
Previous.log Service.log system.log	<ul style="list-style-type: none"> ▶ Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\servers\WinDM1 ▶ AIX - /ewlmData/ewlmms/servers/AIXMS 	Contains information from the dump request when EWLM detects a problem
AlgTrace_YYYY_MM_DD_SS_OOO_CDT.dat ^b	<ul style="list-style-type: none"> ▶ Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\servers\WinDM1\Diagnosics ▶ AIX - /ewlmData/ewlmms/servers/AIXMS/Diagnostics 	Contains information from algorithm trace
EWLM_diagnostics_NNN.xml ^a	<ul style="list-style-type: none"> ▶ Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\servers\WinDM1\Diagnosics ▶ AIX - /ewlmData/ewlmms/servers/AIXMS/Diagnostics 	Contains information from snapshot of domain manager trace
EWLM_trace_NNN.xml ^a	<ul style="list-style-type: none"> ▶ Windows - C:\Program Files\ibm\VE2\EWLMMS\ewlmms\servers\WinDM1\AIX - Diagnostics ▶ AIX - /ewlmData/ewlmms/servers/AIXMS/Diagnostics 	Contains information from domain manager trace

a. NNN is thirteen numeric characters.

b. YYYY_MM_DD_SS is year_month_date_seconds, and OOO is three numeric characters.

10.6 AIX syslog facility

This section describes a useful facility that can assist in finding the date and time when CPU resources have changed in an AIX LPAR partition.

You can confirm the number of processing units, the value of the uncapped weight, and the number of virtual processors of each partition that belongs to the partition workload group from the Partition Workload Group Details view of the EWLM Control Center. But the view does not dynamically display the latest information; therefore, you need to refresh the view by clicking the refresh button to know latest information. To know when and how EWLM domain manager makes resource adjustments to partition workload groups, the logging algorithm trace facility is provided, but it is for IBM supports and the viewing tool is not provided.

You can confirm the date and time when CPU resources are added or removed by using AIX syslog facility. In addition, you can confirm if the operation has succeeded or failed.

On AIX, the syslog facility is not enabled by default. To enable recording CPU adjustment events using syslog, do the following:

1. Edit the /etc/syslog.conf file as the root user.
2. Add the required syslog entries to the end of the file.

For example, add the following:

```
*.debug /tmp/syslog.out rotate size 100k files 4
```

This directive line instructs the syslog facility to log all messages of priority debug (LOG_DEBUG) and above to the /tmp/syslog.out file. After the file reaches 100 k, the file is copied to /tmp/syslog.out.0. In the same way, four files that are /tmp/syslog.out.1, /tmp/syslog.out.2, and /tmp/syslog.out.3 are created, then log files are rotated. The default is not to rotate the log files. If you do not specify the number of files, the files are not rotated.

3. Create the file explicitly:

```
# touch /tmp/syslog.out
```

4. Restart the syslogd subsystem:

```
# stopsrc -s syslogd
# startsrc -s syslogd
```

Example 10-4 shows a sample of syslog output when a processing unit addition operation is successfully performed. In the same way, you can see output of a processing units removal operation and a weight addition or removal operation using the AIX syslog facility.

Example 10-4 Sample of syslog output for a processing unit addition request

```
Oct  7 14:12:39 aixve01 syslog:info syslogd: restart
Oct  7 14:13:04 aixve01 local1:info DRMGR:  ==== Start: SPLPAR addition operation ====
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting CHECK phase for capacity Add operation.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Phase CHECK started for scripts, kernel
extensions and applications.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting CHECK phase for Scripts.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Completed the phase for Scripts.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting the phase for kernel extensions.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Completed the phase for kernel extensions.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting the phase for application signal
handlers.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Completed the phase for kernel extensions.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting PRE phase.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Phase PRE started for scripts, kernel extensions
and applications.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting PRE phase for scripts.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Completed the phase for Scripts.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting the phase for application signal
handlers.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Completed the phase for kernel extensions.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting POST phase.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Phase POST started for scripts, kernel extensions
and applications.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting the phase for application signal
handlers.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Completed the phase for kernel extensions.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Starting POST phase for scripts.
Oct  7 14:13:04 aixve01 local1:info DRMGR: Completed the phase for Scripts.
Oct  7 14:13:04 aixve01 local1:info DRMGR: ~~~~ End: SPLPAR addition operation ~~~~
```

10.7 ARM Serviceability Adapter

The EWLM ARM Serviceability Adapter is a useful facility that can capture and record ARM classification data and ARM transaction data.

The EWLM Serviceability Adapter consists of a shared library and two executables. This facility should be deployed on a managed server; therefore, these parts are shipped with the EWLM managed server product and can be found under the EWLM installation directory.

The shared library is located in the lib folder and the executables are located in the bin folder of the ewlm managed server install. For UNIX systems it is typically /opt/IBM/VE2/EWLMMS/usr/lib and /opt/IBM/VE2/EWLMMS/usr/bin.

The executables are **ewlmarmsrv** and **ewlmarmsrvconv**. **ewlmarmsrv** is used to enable or disable the ARM Serviceability Adapter. **ewlmarmsrvconv** is used to convert the output file from binary format to text. You can specify the directory where the output file will get created when using **ewlmarmsrv** command with the -f option. This file is intended for use by IBM Supports.

To deploy the EWLM ARM serviceability adapter, you need to copy or link the shared library into /usr/lib. This file is different depending on the platform. Table 10-4 shows a list for the different platforms, although you might contact the IBM Service representative for the most current information.

Table 10-4 Shared library on each platform

Platform	Shared library
AIX	libarm4adap.a
i5/OS	No library move required
Linux	ewlm_arm4_adap.so, ewlm_arm4_adap64.so
Solaris	libarm4adap.so (64 bit library is contained under /opt/IBM/VE2/EWLMMS/usr/lib64 with same name)
Windows	Not supported for Windows
z/OS	libarm4adap_31.so, libarm4adap_3x.so, libarm4adap_64.so

Deploying the EWLM ARM Serviceability Adapter

To deploy this facility in order to capture and record ARM classification data and ARM transaction data, for example, on AIX, do the following as the root user:

1. Copy or link the adapter library from its install location into /usr/lib.
2. Enable the ARM Serviceability logging.


```
# ./ewlmarmsrv -on -f /tmp/armoutput.dat -l 4
```
3. Restart the applications that are being traced.
4. Generate transactions for these applications.
5. Alert the serviceability adapter to flush all captured data to file on the next ARM transaction.


```
# ./ewlmarmsrv -fb
```
6. Generate one more transaction for each application being traced.
7. Disable the serviceability tracing. The output is saved to /tmp/armoutput.dat in binary format.


```
# ./ewlmarmsrv -off
```
8. Convert the /tmp/armoutput.dat to text format. The text file contains the captured data in UTF-8 encoding.


```
# ./ewlmarmsrvconv /tmp/armoutput.dat /tmp/armoutput.txt
```

```
armoutput - Notepad
File Edit Format View Help
Sub-Buffer Type      : App Identity
Identity Count       : 1
Identity Name        : PluginType
Identity Value       : websphere
-----
LogRecord 2          : Start Application
GroupName            : IBM_HTTP_Server/6.0.1 Apache/2.0.47 (unix)
InstanceName         : hci246/PID=0000618568
Process ID           : 618568
TimeStamp             : 1121875498 s
AppID                : 0x01000000009E46FB00000000000000000
App Handle           : 576460752313796349
-----
LogRecord 3          : Register Transaction
Trans Name           : WebRequest
Process ID           : 618568
TimeStamp             : 1121875498 s
TranID               : 0x02000000009E46FE00000000000000000
AppID                : 0x01000000009E46FB00000000000000000
-----
Sub-Buffer Type      : Transaction Identity
Context Name Count   : 9
Context Name         : HostInfo
Context Name         : Port
Context Name         : RemoteAddress
Context Name         : Protocol
Context Name         : Scheme
Context Name         : QueryString
Context Name         : RemoteUser
Context Name         : ServerName
Context Name         : ServerVersion
-----
LogRecord 4          : Start Transaction
Process ID           : 618568
TimeStamp             : 1121876101 s
App Handle           : 576460752313796349
TranID               : 0x02000000009E46FE00000000000000000
Tran Class ID        : 2
Start Tran Handle    | : 216172782124156732
EWLM Time            : 175191301202894
```

Figure 10-9 Generated output from ARM Serviceability Adapter

Performance considerations

This chapter provides a description of initial performance and sizing considerations for the Enterprise Workload Manager elements.

In this environment there are three elements that require some sizing consideration:

- ▶ The domain manager
- ▶ The managed server
- ▶ The instrumented middleware that is actually divided into two elements: the ARM processing in the native operating system support and in the infrastructure

This chapter focuses on the general factors that will affect the domain manager resources with additional performance data for the domain manager running on z/OS with EWLM V2.1 and on Windows, AIX, and i5/OS with EWLM V1.1.

It includes also the general factors that will affect the managed server resources with additional performance data for the managed server running on z/OS.

In Version 2.1, the development team included enhancements that should improve both the numbers of servers supported on a domain manager and the relative resource investment needed on managed servers, making any Version 1 performance information delivered here somewhat conservative.

Updated information may be found in the Virtualization Engine Performance Reference, located in the Printable PDF section of the IBM eServer Information Center at:

<http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp?topic=/veicinfo/>

Attention: Performance data in this chapter was obtained in a controlled environment with specific performance benchmarks and tools. This information is presented along with general recommendations to assist the reader to have a better understanding of IBM products. Results obtained in other environments may vary significantly. The data presented here does not predict performance in a specific installation's environment.

11.1 Domain manager resource description

The domain manager functionality is essentially provided by two Java applications. The domain manager application is responsible for aggregating data statistics from the managed servers, evaluating performance goals (versus actuals), distributing policies to the managed servers, and providing reporting data to the Control Center and other potential requestors. The WebSphere Application Server runs the EWLM Control Center application servlet, which provides the user interface into the domain manager. On independent time-intervals, each managed server sends server statistics messages to the domain manager. The size of the statistics transmitted can differ depending on the work running on the given managed server and how it maps to the active service policy. The domain manager regularly houses a live version of the aggregated statistics in memory as well as one day's worth of historical data, hardened to disk. Both the live and hardened data are utilized to provide reporting data to the Control Center or other requestors.

Data at the managed server contains information about the specific server only, while at the domain manager, data is aggregated to provide you with an end-to-end view of the transactions by EWLM transaction or service class.

The domain manager reports aggregated statistical data and not single transaction response time or behavior. If you need more detailed performance information, you will need to utilize additional monitoring products.

11.1.1 Sizing factors

The key resources you must size for the domain manager are CPU, memory, and disk storage for the internal database. Below is a list of the key elements that contribute to domain manager resource usage. The number of permutations of these elements is virtually endless and unpredictable and depends on any given domain configuration and active policy. Hence, in our testing we selected a subset of these that spans a reasonable number of probable environments. The elements are:

- ▶ Number of managed servers in the domain.
- ▶ Size and usage of the service policy in terms of number of transaction classes, process classes, and service classes. By usage we mean service, transaction, and process classes that actually have work qualified (classified into). For instance, if you have 80 transaction classes defined, but only 10 are typically utilized, then resource utilization will be much less than having all 80 utilized.
- ▶ Application/server topology in terms of number of application environments, number of hops, how transaction classes map to application environments/servers, and others.

The size of the service policy, combined with the level of complexity of the application/server topology, has a larger impact on the resource requirements than does the mere number of managed servers in the domain.

Measurements were conducted to evaluate the performance and scalability of the EWLM domain manager. This following information contains the detailed performance results of these tests and some general sizing recommendations based on those results. This information can be utilized for a general appreciation of computing resources necessary to run the domain manager.

11.2 Performance tests

We conducted a set of tests varying the number of managed servers and policy complexity, while keeping the actual workload transaction rate (per server) relatively constant.

11.2.1 Environment

The following platforms were chosen and configured as domain managers, as illustrated in Table 11-1.

Table 11-1 Domain manager server configuration

EWLM version	Platform	Processor model	Processor speed	Physical memory	Network adapter
V2.1	z/OS V1.6	z990 2084-316	4-way LPAR with 4 dedicated processors	6.0 GB	OSA Gbit
V1	Windows Server 2003 Enterprise Edition	x255	Xeon® 4-way @2 GHz w/ht	3.8 GB	Gbit
V1	AIX 5.2 ML03 (32bit)	p630 7028-6C4	Power 64bit 1.4Ghz 4-way	7.0 GB	Gbit
V1	i5/OS	4-way of an i825 6-way	4-way of a 6-way rated at 6600 CPW	5.7 GB	Gbit

11.2.2 Key measurement variables

This provides an explanation of the different variables affecting the key measurements.

Managed server number

The number of managed servers varied from 60 to 600.

Policy and workload composition

Two different service policies were used during the measurements and they are summarized in Table 11-2. All the service classes are defined with response time goals.

For all measurements, if a service class or transaction class was defined, then it was also utilized (that is, had transactions classified to it). When we moved from the simple to the complex policy, we further filtered the workload to utilize the additional transaction and service classes.

The number of rules (filters) or type of rule is essentially not a factor in domain manager resource consumption because classification occurs on the *edge* managed server. We utilized one filter per transaction class.

Table 11-2 Service policies summary

Policy	Number of service classes	Number of transaction classes	Number of process classes	Number of service classes utilized	Number of transaction classes utilized	Number of process classes utilized
Simple	5	20	1	5	20	1
Complex	20	80	1	20	80	1

In order to simulate high numbers of real EWLM managed servers, we utilized an internal tool for all of our tests. The tool essentially simulated the performance data being collected every 2 seconds on each of the individual managed servers. Hence, the messages being sent from the managed servers to the domain manager were real, though based on simulated input into the individual managed servers. Special code was added to the EWLM managed servers, allowing multiple managed servers to run on the same OS image. No part of the actual domain manager processing was scaffolded or simulated.

Our model environment is an EWLM Management Domain with many operating system instances (servers), with each server responsible for a single function (for example, Web serving, application serving, database), as illustrated in Figure 11-1.

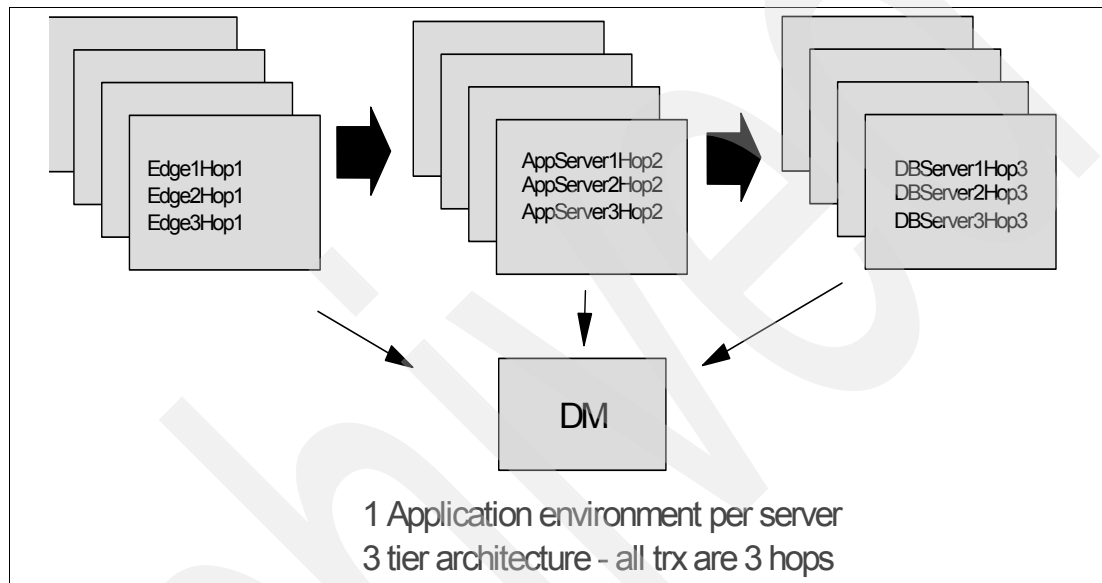


Figure 11-1 Application/server topology

A given server in the domain consisted of one application environment. Each server is simulating roughly between 10 and 50 sub-transactions per second, which are spread across these three application environment instances. Here we use the term *sub-transaction* to represent a piece (or hop) of a multi-tiered transaction. The actual transaction rate is not a factor in the domain manager resource requirements. The number of unique combinations of application environment instance — application environment — hop — transaction classes having actual transaction counts (for example, completions) is a factor. To simplify, this could be thought of as the number of different transaction classes that have qualified transaction counts on a given server.

Table 11-3 exhibits additional characteristics of the combined policy/application/server topologies utilized in our testing.

Table 11-3 Workload complexity

Policy	Average number of service classes utilized per server	Average number of transaction classes utilized per server	Average number of servers a given transaction class is utilized on
Simple	1	3	1/7th of all managed servers
Complex	3	10	1/7th of all managed server

In other words, although the domain manager was monitoring five service classes and twenty transaction classes in the total network for our simple policy definition, we assumed that an

individual managed server would only report on a single service class with three transaction classes.

We do not recommend at an installation trying to predict all of these different combinations depending on the variation of the influencing factors. This serves to give a better appreciation for our topology characteristics and some of the additional influencers of resource consumption. Generally, as these values increase, so does the amount of data being transmitted to the domain manager along with domain manager processing and memory resources.

11.2.3 Domain manager on z/OS

Figure 11-2, Figure 11-3 on page 276, and Figure 11-4 on page 276 depict the measured domain manager resource usage when running on z/OS. For a precise understanding of these metrics as well as additional performance metrics captured, we recommend reading this section in its entirety.

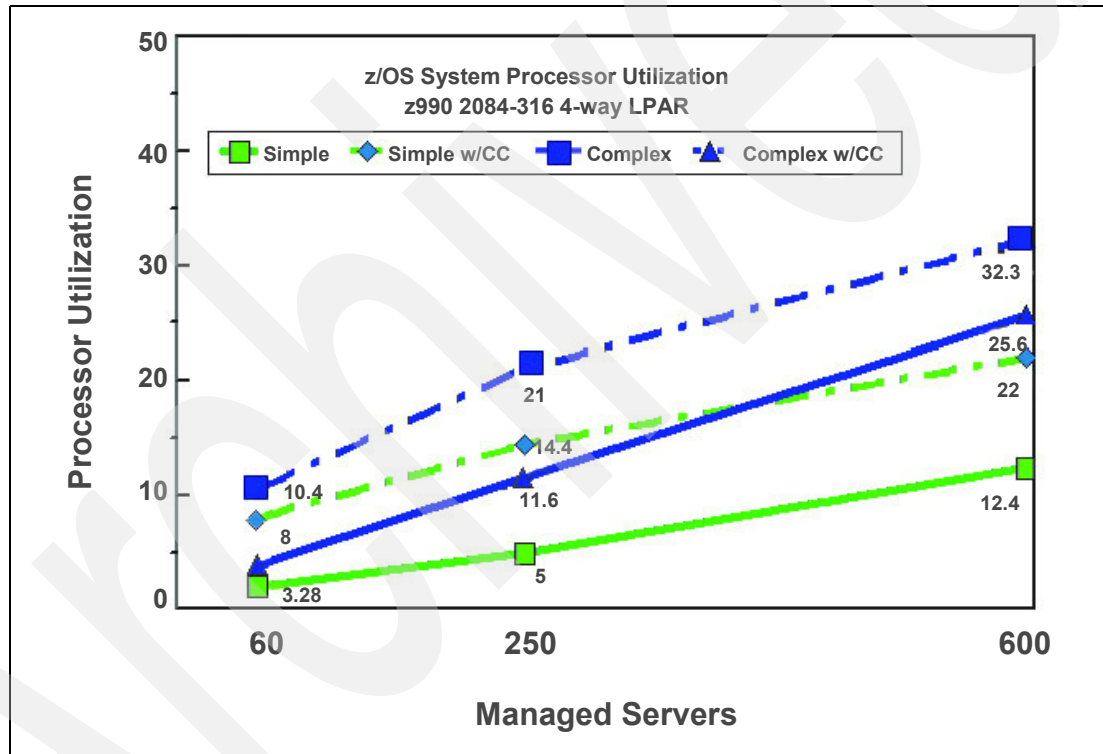


Figure 11-2 z/OS domain manager processor utilization

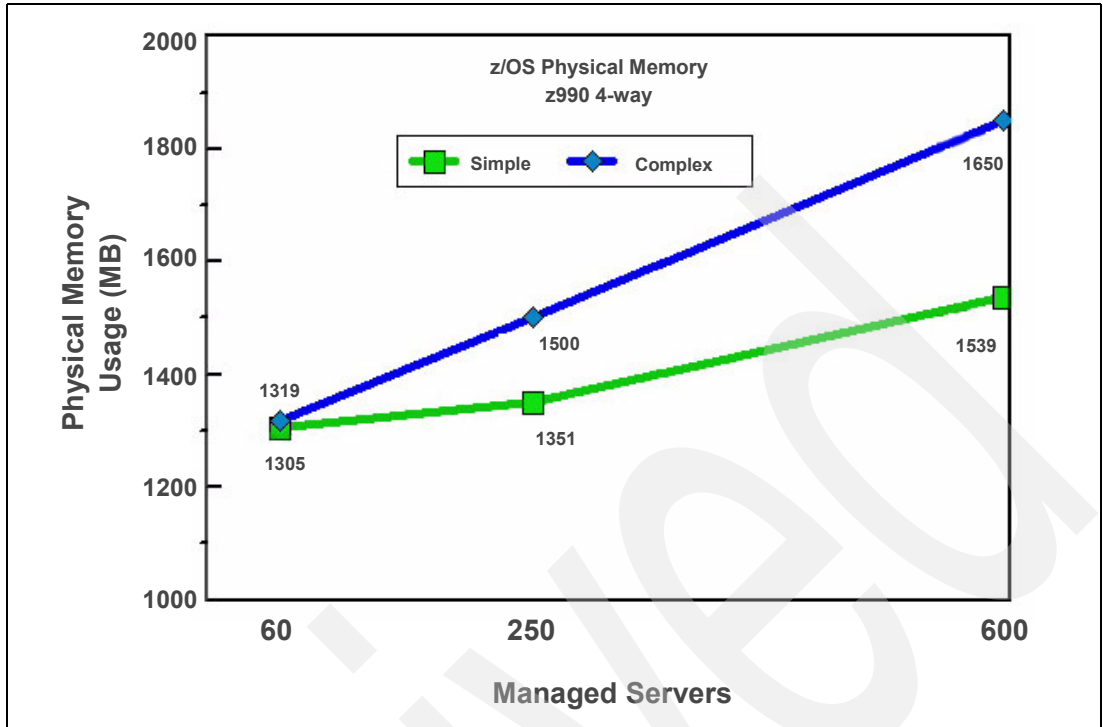


Figure 11-3 z/OS domain manager memory usage

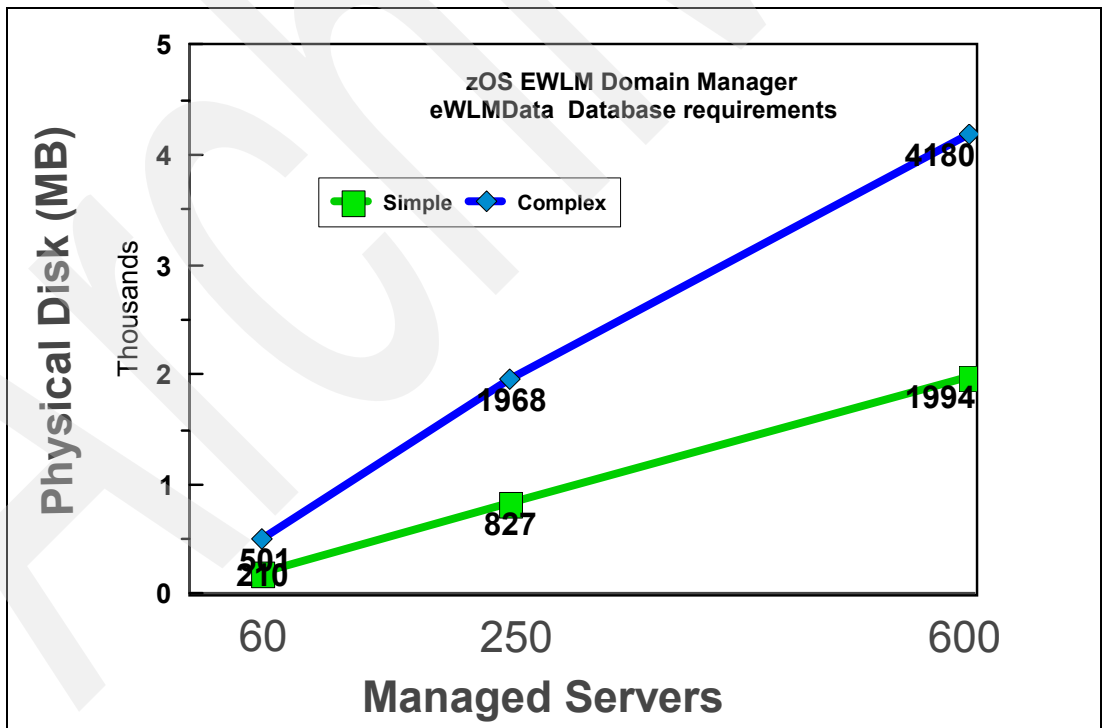


Figure 11-4 z/OS domain manager eWLM Database physical disk usage

11.2.4 z/OS test methodology

The following was the methodology utilized for the performance measurements:

- ▶ The Resource Monitor Facility (RMF) was utilized to capture most of the data in the detailed metrics tables.
- ▶ To determine the memory requirements for the domain manager process (in particular the java heap), we ran all of our tests with a large -maxHeap setting of 1160 MB, allowing the jvm heap to expand to a level it deemed appropriate for the given load. We left the -minHeap at the default of 128 MB. To determine the expected peak memory usage both Control Center Reporting and Policy activation was also conducted.
- ▶ Performance metrics were captured during steady state and with control center activity windows:
 - Steady state: The base domain manager processing of gathering and aggregating the managed servers statistics data. The Control Center was started but there was no end-user activity (no reporting or policy changes, and so on).
 - With control center reporting: This represented having one end-user actively running three real-time monitors (service class – goal achievement, transaction class – transaction rate, managed server - processor utilization) as well as actively displaying the monitor-managed servers and manage-interval reports for different service classes, transaction classes, and process classes. An approximate 5-second think time was utilized between a previous response and a new report request.
- ▶ An interval duration of 5 minutes was selected in the Preference panel to be able to capture most of the observed metrics. Generally, the processing characteristics over a 1-minute interval were consistent, though a 5-minute interval was used anyway.
- ▶ To determine the EWLM database size, the given configuration should run for 24 hours.

Important: Domain Manager Processes information:

When starting the z/OS domain manager, four address spaces will be started. One of these address spaces (typically EWLM5) is where all the key domain manager work gets done. Another one (typically EWLM3) is simply responsible for the initial configuration and actual Java launch of EWLM5. The other two are simply the USS shells responsible for starting the domain manager configuration. In this document, when referring to domain manager resource usage, we are essentially referring to the aggregate resource for all these processes.

Table 11-4 contains a summary of the metrics observation for the z/OS domain manager.

Table 11-4 Detailed metrics observations for z/OS domain manager

Policy	Servers	Processor utilization					Memory Usage				Disk	Network
		Total system processor utilization	Domain manager processor utilization	Virtualization Engine WebSphere processor utilization	Total EWLM memory usage (MB)	Domain manager memory usage (MB)	Virtualization Engine WebSphere memory usage (MB)	Domain manager Java heap size (MB)	eWLMData database size (MB)	Domain manager network activity (KB/s)		
Simple	60	2.0	8.0	1.1	1.7	5.1	1305	280	1025	124	210	72
Simple	250	5.0	14.4	3.9	5.6	7.4	1351	307	1044	137	827	276
Simple	600	12.4	22.0	10.7	11.5	8.5	1539	495	1044	310	1994	653
Complex	60	3.8	10.4	2.8	4.2	5.6	1319	270	1049	124	501	215
Complex	250	11.6	21.0	10.1	13.5	5.7	1500	454	1046	284	1968	850
Complex	600	25.6	32.3	23.8	23.7	7.5	1850	784	1066	598	4180	1906

Where:

- *Servers* is the number of the managed server in the EWLM Management Domain.
- *Total system processor utilization* is the average processor utilization for the entire z/OS image (the sum of the domain manager, Virtualization Engine WebSphere, and base operating system processor utilization).

For each scenario we list two values. The low value represents average processor utilization with no EWLM Control Center reporting. The high value represents average processor utilization with Control Center Reporting (see Methodology above).

- *Domain manager processor utilization* is the average processor utilization of the key domain manager address space (the other domain manager address spaces do not consume processing cycles other than at startup).

Again, we list two values. The low value represents average processor utilization with no EWLM Control Center reporting. The high value represents average processor utilization with Control Center Reporting (see 11.2.4, “z/OS test methodology” on page 277).

- *Virtualization Engine WebSphere processor utilization* is the average aggregate processor utilization for the three z/OS Virtualization Engine WebSphere address spaces (servant, controller, and daemon), plus the processor utilization of the WebSphere enclaves responsible for execution of requests to the Control Center application.

Only one value is listed here, which represents the scenario with EWLM Control Center reporting (see 11.2.4, “z/OS test methodology” on page 277.). When there was no reporting, then the processor utilization was essentially 0.

- *Total EWLM memory usage* is the sum of observed maximum memory usage of the domain manager and the Virtualization Engine WebSphere address spaces.

This, as well as all the memory usage metrics, represents memory usage captured during Control Center reporting and a domain policy switch.

- ▶ *Domain manager Memory usage* is the sum of the observed maximum memory usage of the domain manager address spaces.
- ▶ *Virtualization Engine WebSphere Memory usage* is the sum of the observed maximum memory usage of the Virtualization Engine WebSphere Application Server memory usage (servant, controller, and daemon).
- ▶ *Domain manager Java heap size* represents a subset of the domain manager memory usage.

It represents the maximum expanded Java heap for the key Domain Manager address space. We make note of this metric, for larger domains potentially require increasing the minimum/maximum heap specifications for the domain manager. The default minimum/maximum Java heap specifications are 128 MB and 512 MB. Reference IBM system information center if a change is required.

- ▶ *eWLMData database size*. This directory is located inside the EWLM ConfigID file. The domain manager utilizes an internal database to house key configuration/policy data as well as the one-day history of reporting data.

This field represents the combined disk requirements witnessed for these databases after a full day of processing. Also, an additional 20% buffer was added to account for additional disk space required during a policy switch.

Note: During a policy switch, reporting data history is kept for the current policy as well as the new policy being deployed.

- ▶ *Domain manager Network Activity (Kbytes/s)* is the aggregated rate of data inbound/outbound between the domain manager process and the EWLM Domain of managed servers.

This rate is predominantly driven by the managed server's incoming *server statistic* messages.

- ▶ Additional metric observed: *Control Center Reporting end-user response time*.

Using wall clock time we measured control center reporting response time for several of the measured environments. Disregarding the first-time execution of each report, the average end-user response time ranged from 2 seconds (for the 60 managed server, simple policy scenario) to 5 seconds (for the 600 managed server, complex policy scenario).

11.2.5 General sizing recommendations on z/OS

As discussed earlier, the permutations of the EWLM domain manager sizing factors are too many to create a simple sizing formula. Hence, this guidance is to be utilized for a rough estimate of initial resource requirements and a preliminary forecast. Once an initial management domain is in place, we advise utilizing the active resource utilizations in projecting/validating resource requirements for domain growth. For additional information see “Domain growth general recommendations” on page 281.

Processor requirements

As you can see from our results, there is a notable increase in processing requirements associated with a complex policy/workload. Additionally, during times of Control Center reporting activity, there will be an increase in processing requirements. Based on size of the expected domain and anticipated policy/workload topology complexity along with anticipated

frequency of Control Center activity, one can attempt to place themselves somewhere within our four processor utilization plots. If utilizing the zSeries LSPR (Large Systems Performance ITR ratios) for sizing, we suggest utilizing the CB-J workload ratios.

Policy switch considerations

An EWLM Domain policy switch (deployment of a new domain or service policy) requires a temporarily increased demand for processing, memory, and I/O resource utilization. With regard to memory and disk size requirements, our internal measurement results and resource requirement charts include the additional resource demand. On the other hand, since we anticipate policy switches to be an infrequent activity, we did not utilize the increased processor utilization in our average processor utilization metrics. For large (>100 managed servers), complex domains consider the temporarily increased demand for processing resources. In our policy switch testing, we observed temporary 2–3 times increases in the average processor utilization. For this reason, we recommend that planned large domain policy switches be made during off-peak times.

With regard to policy switch response time, we used wall clock time for measuring the elapsed time from the “Activation of a new policy (similar in complexity to the current policy)” to the “complete domain running with the new policy (as viewed by the Control Center)”. The policy switch response time ranged from 5 seconds (for the 60 managed server, simple policy scenario) to 45 seconds (for the 600 managed server, complex policy scenario). It is important to note that these policy switch results should be considered as best case, since they did not include delays associated with native processing (on the managed servers) of the new policy. Additionally, as mentioned above, a policy switch will temporarily demand additional processing cycles. Hence, to the degree that additional processing resource is available, the policy switch time would be elongated.

Memory guidance

The memory requirement metrics captured during our internal performance tests included some Control Center reporting (see 11.2.4, “z/OS test methodology” on page 277) and Domain policy deployments. The rationale behind this is that we believe it is best to size for peak memory requirements, especially to avoid the potential paging of the Java heap. In estimating memory requirements, based on anticipated Domain characteristics, one can choose a point on the memory usage plots.

The default EWLM domain manager `initHeap` and `maxHeap` (java) settings are 128 M and 512 M, respectively.

For small domains where growth is not anticipated (for example, test environments), one can reduce the required memory footprint by overriding the default domain manager `-initHeap` specification to 64 MB.

Note: As you can see in the memory requirements plot shown in Figure 11-3, the 60 managed servers/simple closely matches the 60 managed server/complex. The reason is that both of these scenarios actually required less than the default Java `initHeap` size. For large complex domains, such as our 600 managed server/complex scenario, the Domain Manager `-maxHeap` specification will need to be overridden. If necessary, these heap modifications would be made within the EWLM started procedure. Refer to the IBM system Software Information Center for details.

The default z/OS Virtualization Engine WebSphere servant region `minHeap` and `maxHeap` (Java) settings are 256 M and 512 M, respectively. For large complex domains, it may be necessary to increase the `maxHeap` setting. We did not require modifications for any of our test scenarios.

Domain eWLMData database size requirements

The amount of internal database disk required can be estimated utilizing our internal measurement plots, as shown in Figure 11-4 on page 276. Perhaps more important than the amount of disk is the placement of this internal database. On 1-minute intervals, the detailed aggregate domain statistics are hardened to the EWLM internal database. As the domain becomes larger and more complex, the amount of hardening I/O activity can be quite high and will potentially impact internal database requests associated with Control Center reporting. For this reason, we highly recommend that the internal database be placed on a high performance disk.

The internal database is created under the domain manager configID directory, which is under the <EWLM_DATA_ROOT>/servers path. The <EWLM_DATA_ROOT> path is set in the etc/ewlm_environment.conf as one of the first steps of Configuring EWLM (refer to 2.5.2, “Configuring the domain manager” on page 37). Hence, it is essential that the EWLM_DATA_ROOT directory be placed on a high performance disk.

The internal database is not the *only* disk requirement for EWLM domain manager, though it is the only one whose size is relative to the size/complexity of the domain.

Domain growth general recommendations

As you can see from our internal tests, our resource usage was generally linear as we scaled the number of Managed Servers, maintaining similar workload characteristics. Depending on how your domain grows, this fact can be used, along with your own domain manager resource usage metrics, as a basis for future planning. To assist in capacity planning for your Domain growth, once you have a reasonable domain and policy in place, we suggest monitoring:

- ▶ The average processor utilization of the domain manager/Virtualization Engine WebSphere address spaces and Virtualization Engine WebSphere transactions (CB enclaves). This could be easily accommodated with the creation of z/OS WLM service or reporting classes and monitoring of RMF.
- ▶ The average (or peak if available) memory usage associated with domain manager and Virtualization Engine WebSphere address spaces. In particular, the key working domain manager address space and the WebSphere servant address space. Again, this can be captured via use of WLM and RMF or other monitoring tools.

Note: We typically noticed a delta of 150–180 MB between the total memory usage of the DM address space and the DM java heap. Hence, ensure that your DM java maxHeap is larger than say your projected total DM memory usage - 200 MB. Otherwise consider increasing the DM maxHeap at the appropriate time.

- ▶ The amount of disk space utilized under the <EWLM_DATA_ROOT> directory as well as perhaps the average response times/device activity/utilization for the disk the EWLM_DATA_ROOT resides on.

Classifying the EWLM domain manager to z/OS WLM

Since the domain manager will be responsible for monitoring and managing resources, we recommend that the domain manager address spaces, Virtualization Engine WebSphere address spaces, and the Control Center requests (Virtualization Engine WebSphere transactions) be classified to SYSSTC or a high dispatch priority service class with high importance and aggressive goals.

For the domain manager address spaces (EWLMDM*), a classification rule should be added to both the OMVS and the STC subsystems. Three of the four EWLMDM* address spaces, including the key one, will get classified under OMVS.

For the Virtualization Engine WebSphere address spaces (VEWAS, VEWASD, VEWASS), all get classified under the STC subsystem.

The EWLM DM Control Center requests (Virtualization Engine WebSphere transactions) will be classified under the CB subsystem. Only EWLM Control Center requests would occur under the VEWAS server. Therefore we suggest utilizing a CN qualifier type with VEWAS as the qualifier name.

EWLM domain manager self-preservation

Starting with EWLM V2.1, EWLM automatically enforces a preservation mode if the domain manager is unable to process the managed server's incoming messages in a timely manner. If the domain manager is in preservation mode, the managed servers reduce the rate at which they send messages to the domain manager. During a preservation, EWLM does not lose any data. A preservation indicates that the managed servers reduce the rate at which they send the domain manager messages, but, eventually, all messages are sent to the domain manager. The duration of a domain manager in a preservation mode varies by the amount of resources available to the domain manager to process the incoming messages. If the domain manager is in preservation mode often or for a prolonged period of time, for example, more than 15 minutes, we suggest examining the system resources to determine which resource the domain manager is being starved of. Examine the system processor utilization and physical memory usage to see if either are constrained due to another application. If there is ample processing capacity and physical memory available, the domain manager jvm maxHeap specification potentially needs to be increased. One way of estimating is to examine the average physical memory usage of the domain manager process, using RMF or another performance monitor. The domain manager active jvm heap usage is typically 150–200 MB less than the total average domain manager process physical memory usage. If this subtraction yields a value close to the current jvm maxheap specification (default is 512 MB), then consider increasing the maxHeap and re-starting the domain manager at a convenient time.

zAAP eligibility

The domain manager processing can be zAAP-eligible. In fact, utilizing the “-Xifa:force” projection tool, while running a 250 complex managed server domain, we observed that approximately 97% of the processing necessary for domain management (domain manager, Virtualization Engine WebSphere) was zAAP eligible.

11.2.6 Domain manager on Windows

In this section are some charts that show the domain manager resources usage when running on Windows.

Figure 11-5 shows the CPU, the memory, and the disk usage for the domain manager internal database with the different model policies and the managed servers scaling factor.

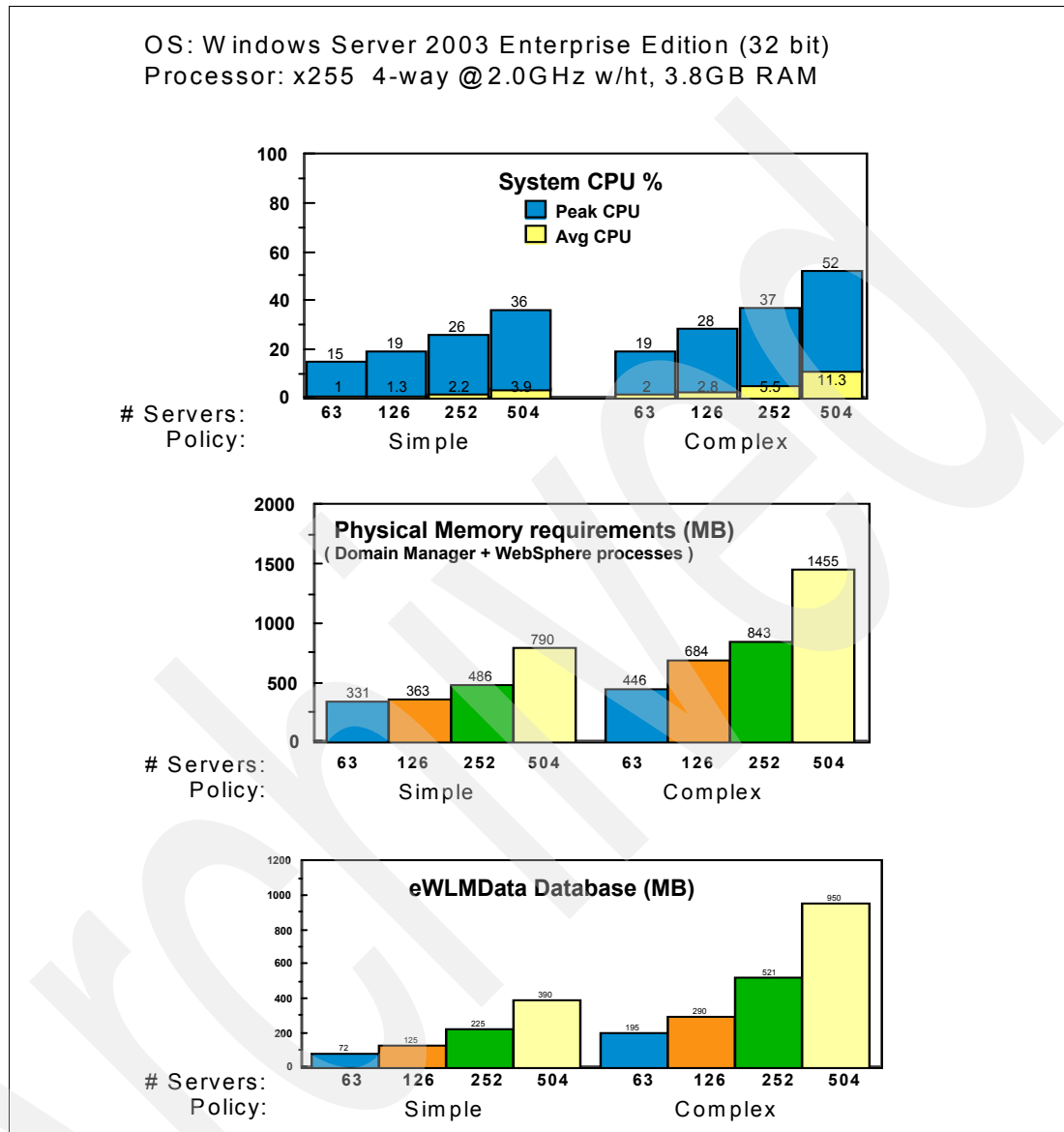


Figure 11-5 Resources usage on Windows domain manager

11.2.7 Windows test methodology

The following were our methodology considerations:

- ▶ The Windows performance monitor was used to capture most of the metrics in Table 11-5 on page 284 and Table 11-6 on page 284. The domain manager Java process was started with the gc option in order to capture the Java heap size. The overhead of this was measured and determined to be insignificant.
- ▶ To determine the memory requirements for the domain manager process (in particular the Java heap), we ran all of our tests with a large -maxHeap setting of 1800 (1800 MB), allowing the jvm heap to expand to a level it deemed appropriate for the given load (by default the Java heap will expand when it no longer can maintain at least 30 percent free space). We left the -minHeap at the EWLM default of 128 (128 MB).

- ▶ Performance metrics were captured during Steady State and with Control Center activity windows:
 - Steady state: The base domain manager processing of gathering and aggregating the managed servers statistics. The Control Center was started but there was no end-user activity (no reporting, policy changes, and so forth).
 - Control Center activity: Two concurrent end users actively displaying interval or real-time reports and the deployment of a new Domain Policy/service policy that was comparable in size to the current active policy.
- ▶ An interval duration of 10 minutes was selected in the Preference panel to be able to capture most of the Windows performance monitor metrics (for example, average system CP util Steady State, domain manager and Control Center working sets, and so forth). If we used an interval other than 10 minutes, the measurement interval is specified in the metric explanations that follow.
- ▶ We utilized default system settings. No specific I/O or system performance tuning was conducted.
- ▶ For each measurement point, a one-hour warm-up period was first run to ensure accurate sizing of the internal database.

Detailed results are displayed in Table 11-5 and Table 11-6.

Table 11-5 Metrics observations for simple policy

Servers	Avg system CP util Steady State	System CP util spikes with Control Center activity	DM working set peak with Control Center activity (MB)	DM Java heap size (MB)	ControlCenter Working set peak with Control Center activity (MB)	EWLM total working set peak (MB)	EWLM DB size (MB)	DM net KBytes Rcvd/s	NIC util	Elaps time: 60min interval report (sec)
63	1.0	15	178	134	153	331	72	38	<1%	1
126	1.3	19	210	154	153	363	125	75	<1%	1
252	2.2	26	296	250	190	486	225	151	<1%	2-3
504	3.9	36	493	434	297	790	390	304	<1%	3-4

Table 11-6 Metrics observations for complex policy

Servers	Avg system CP util Steady State	System CP util spikes with Control Center activity	DM working set peak with Control Center activity (MB)	DM Java heap size (MB)	ControlCenter Working set peak with Control Center activity (MB)	EWLM total working set peak (MB)	EWLM DB size (MB)	DM net KBytes Rcvd/s	NIC util	Elaps time: 60min interval report (sec)
63	2.0	19	248	198	198	446	195	146	<1%	2
126	2.8	28	393	334	291	684	290	290	<1%	2-3
252	5.5	37	483	422	360	843	521	574	<1%	4-5
504	11.3	52	1053	982	402	1455	950	1153	1%	8-12

Where:

- ▶ *Servers* is the number of the managed server in the EWLM Management Domain.
- ▶ *Average system CP utilization Steady State* is the average total processor utilization for the server. During steady state, the domain manager process consumes more than 98 percent of the total processor utilization.
- ▶ *System CP Utilization spikes with Control Center activity* is the highest average system processor utilization monitored (over a 15-second interval) upon requesting reporting data or conducting a policy switch or combination of both. The domain manager and WebSphere Application Server Control Center processes make up for more than 98 percent of the total System processor utilization.
- ▶ *DM Working Set Peak with Control Center activity* is the maximum size of the physical memory utilized for the domain manager process. This was the maximum witnessed during CC activity. Reporting and policy deployment can require significant additional resources beyond steady state, hence we provide this metric for sizing.
- ▶ *DM Java heap size* represents the maximum expanded domain manager Java heap.
The Java heap constitutes the majority of the domain manager working set. In all the tests, the total domain manager working set peak was approximately 50–70 MB greater than the Java heap. This value was captured via the "verbose:gc" Java start parameter.
- ▶ *Control Center Working Set Peak with CC activity* is the maximum size of the physical memory requirements for the WebSphere Application Server Control Center process during CC activity. Running reports or conducting policy changes will impact the physical memory requirements of the WebSphere Application Server CC process.
- ▶ *EWLM Total Working Set Peak* represents the total peak physical memory requirements for the EWLM domain manager. This is simply "domain manager Working Set Peak w/CC" + "CC Working Set Peak w/CC".
- ▶ *EWLM DB size*: The domain manager and WebSphere Application Server processes both have embedded internal database instances to house persistent data such as policies, server information, one-minute reporting data, and so forth. This field represents the combined disk requirements witnessed for these databases. In our tests, the largest disk consumer was DM's EWLM Data Reporting database (houses 1-minute reporting data utilized for CC interval reports). This database was responsible for more than 95 percent of the total database disk requirements.
- ▶ *DM Net KBytes Rcvd/s* is the aggregated rate of data inbound to the domain manager process from the Management Domain of managed servers. This rate is predominantly driven by the managed server's *server statistic* messages. These messages constitute each managed server's aggregated transaction/server data for the past 10 seconds and are sent every 10 seconds.
- ▶ *NIC Utilization* is the utilization of the network interface card. In our case this was a Broadcom NetXtreme Gigabit ethernet card.
- ▶ *Elapsed Time: 60min Interval Report* represents the elapsed time of requesting a 60-minute Monitor-Transaction Classes report from the EWLM Control Center. We think this command is the most data-intensive report request from the Control Center because it requires reading data for all transaction classes, so we utilized this metric as a measuring stick of Control Center responsiveness.

11.2.8 General sizing recommendations on Windows

Based on our internal testing, we offer the following general sizing guidance.

Minimum system requirements

A 2 GHz Intel®-based uniprocessor with 512 MB memory is the minimum system recommended for a domain manager that is handling between 1 and 50 managed servers and a policy of 40 transaction/process classes and 10 service classes.

General system requirements guidance

1 to 200 managed servers with a policy up to 80 transaction/process classes and 20 service classes should safely fit on an xSeries 2-way 2 Ghz with 2 GB RAM or comparable hardware. Default Java maxHeap settings of 512 MB for domain manager and 256 MB for the Control Center should be sufficient. Beyond product install disk requirements, an additional 1 GB of disk space should be sufficient for EWLM's database requirements.

200 to 500 managed servers with a policy up to 80 transaction/process classes and 20 service classes should safely fit on an xSeries 4-way 2 Ghz w/ 2 GB RAM or comparable hardware. We suggest monitoring memory and processor resource usage as the configuration grows to more than 100 servers to more accurately project your requirements for these larger configurations. Domain manager and WebSphere Application Server maxHeap settings potentially need to be increased to avoid an out-of-memory condition for Java heap. Beyond product install disk requirements, an additional 2 GB of disk space should be sufficient for EWLM's database requirements.

Memory guidance

Ensure sufficient physical memory to back the domain manager and WebSphere Application Server processes. We recommend monitoring the memory and the processor consumption of these processes upon any substantial changes to size or policy complexity of the Management Domain (for example, going from 10s to 100s of managed servers; 10 to 50 transaction classes). The default maxHeap settings for these are: Domain manager - 512 MB, WebSphere Application Server - 256 MB. The maxHeap for DM can be increased on the **startDM** script via the **-maxHeap** parameter. The maxHeap for WebSphere Control Center can be increased via the WAS Administrator Console.

The 504 managed server/simple policy and the 252/504 managed server/complex policy test scenarios all required increasing the default Java maximum heap for the domain manager process. Recall from our performance test methodology we had already increased the domain manager process **-maxHeap** to 1800 for all of our tests, so in our case, no modification was necessary. Additionally, the 504 managed server/complex policy test scenario required increasing the maximum heap for the WebSphere Application Server process. After first hitting an out-of-memory condition on the WebSphere Application Server, we re-started it with a maximum heap size of 320 MB.

11.2.9 Domain manager on AIX

Following are some charts that show the domain manager resources usage when running on AIX.

Figure 11-6 shows the CPU, the memory, and the disk usage for the domain manager internal database with the different model policies and the managed servers scaling factor.

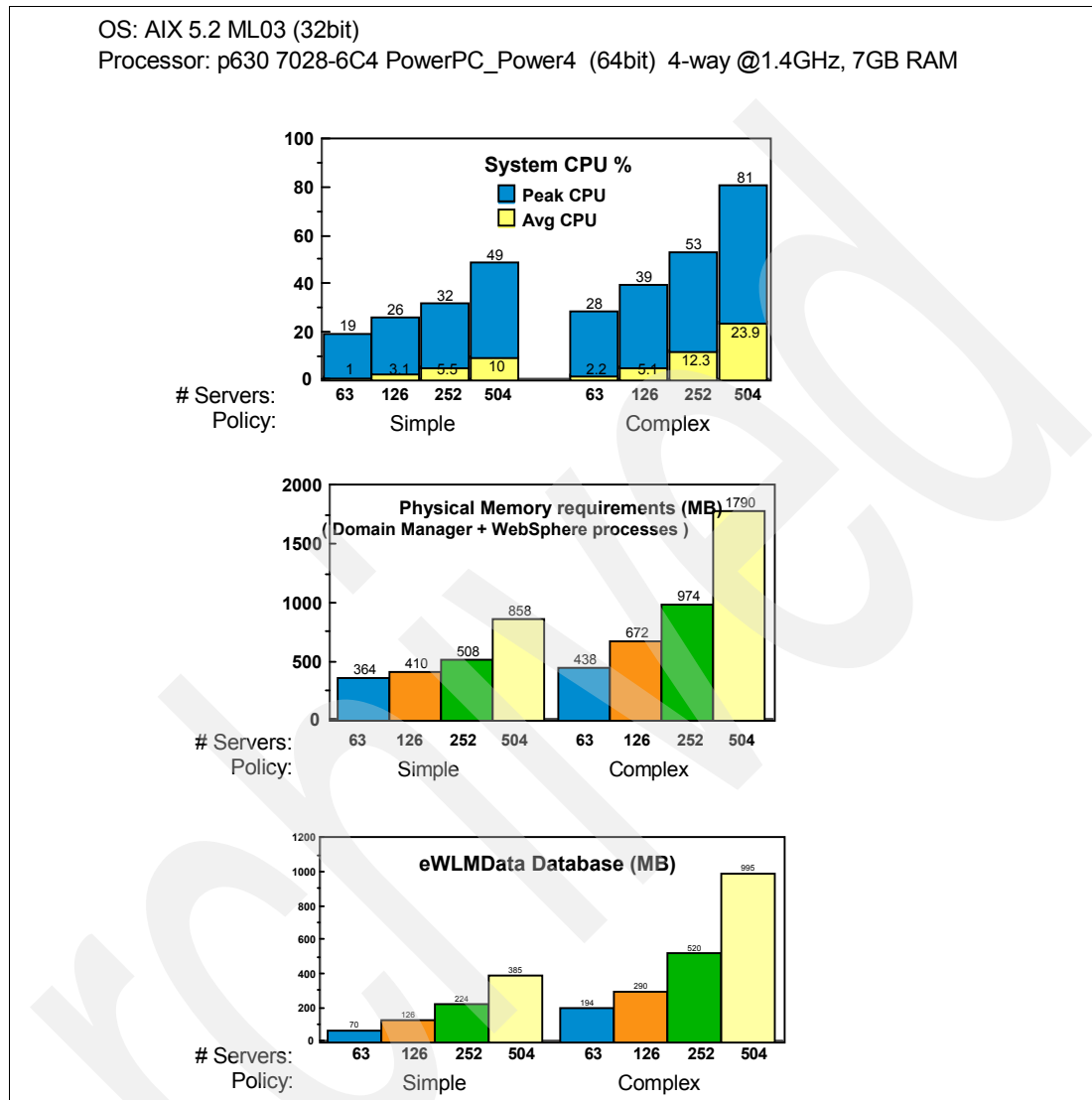


Figure 11-6 Resources usage on AIX domain manager

11.2.10 AIX test methodology

For AIX test methodology:

- ▶ We utilized the nmon performance monitor www.ibm.com/developerworks/eserver/articles/analyze_aix/index.html, in history mode, to gather most of the performance metrics in Table 11-7 on page 288 and Table 11-8 on page 289. The domain manager Java process was started with the gc option in order to capture the Java heap size. The SVMON command was used to capture memory utilization of the domain manager processes. The overhead of these was measured and determined to be insignificant.
- ▶ To determine the memory requirements for the domain manager process (in particular the Java heap), we ran all of our tests with a large -maxHeap setting of 1800 (1800 MB), allowing the jvm heap to expand to a level it deemed appropriate for the given load (by

default the Java heap will expand when it no longer can maintain at least 30 percent free space). We left the -minHeap at the EWLM default of 128 (128 MB).

- ▶ In order to set the -maxHeap to 1800 on AIX (32 bit), we first added the following to the domain manager **startDM** script (located in /opt/IBM/VE/EWLM/bin):

```
"export LDR_CNTRL=MAXDATA=0x20000000"
```

This would allow the possibility of a 2 GB maximum Java heap, though as we said, we set our maximum to 1800 MB.

- ▶ Performance metrics were captured during Steady State and with Control Center activity windows:
 - Steady State: The base domain manager processing of gathering and aggregating the managed servers statistics data. The Control Center was started but there was no end user activity (No reporting, no policy changes, and so forth).
 - Control Center activity: Two concurrent end users actively displaying interval or real-time reports and the deployment of a new Domain Policy/service policy, which was comparable in size to the current active policy.
- ▶ An interval duration of 10 minutes was selected in the Preference panel to be able to capture most of the nmon metrics (for example, Average system CP until Steady State, domain manager and Control Center working sets, and so forth). If we used an interval other than 10 minutes, the measurement interval is specified in the metric explanations that follow.
- ▶ We utilized default system settings. No specific I/O or system performance tuning was conducted.

Detailed results are displayed in Table 11-7 and Table 11-8 on page 289.

Table 11-7 Metrics observations for simple policy

Servers	Avg system CP util Steady State	System CP util spikes with Control Center activity	DM working set peak with Control Center activity (MB)	DM Java heap size (MB)	Control Center Working set peak with Control Center activity (MB)	EWLM total working set peak (MB)	EWLM DB size (MB)	DM net KBytes Rcvd/s	Elaps time: 60min interval report (sec)
63	1.0	19	204	136	160	364	70	38	1
126	3.1	26	208	136	202	410	126	76	1-2
252	5.5	32	298	220	210	508	224	150	2-3
504	10.0	49	528	427	330	858	385	303	4-6

Table 11-8 Metrics observations for complex policy

Servers	Avg system CP util Steady State	System CP util spikes with Control Center activity	DM working set peak with Control Center activity (MB)	DM Java heap size (MB)	Control Center Working set peak with Control Center activity (MB)	EWLM total working set peak (MB)	EWLM DB size (MB)	DM net KBytes Rcvd/s	Elaps time: 60min interval report (sec)
63	2.2	28	242	190	196	438	194	145	2.0
126	5.1	39	392	340	280	672	290	291	2-3
252	12.3	53	634	533	340	974	520	574	4-10
504	23.9	81	1387	1240	403	1790	995	1152	10-30

Where:

- ▶ *Servers* is the number of the managed server in the EWLM Management Domain.
- ▶ *Average system CP utilization Steady State* is the average total processor utilization for the server. During steady state, the domain manager process consumes more than 98% of the total processor utilization.
- ▶ *System CP Utilization spikes w/Control Center activity* is the highest average system processor utilization monitored (over a 15-second interval) upon requesting reporting data or conducting a policy switch or combination of both. The domain manager and WebSphere Application Server Control Center processes make up more than 98 percent of the total system processor utilization.
- ▶ *DM Working Set Peak w/Control Center activity* is the maximum size of the physical memory utilized for the domain manager process. This value was determined by running iterations of the following SVMON command and utilizing the maximum value witnessed.

SVMON -P pid -m (where pid is the DM process ID)

From the SVMON output we totalled the INUSE values for the following memory segments:

- 2:** Process private
- 3-4:** Native heap
- 5-c:** Java heap
- f:** Shared library data

Reporting and policy deployment can require significant additional resources beyond steady state; hence, we provide this metric for sizing.

- ▶ *DM Java heap size* represents the maximum expanded domain manager Java heap. The Java heap constitutes the majority of the domain manager working set. In all the tests, the total domain manager working set peak was approximately 50–70 MB greater than the Java heap. This value was captured via the "verbose:gc" Java start parameter, but also could be determined using the SVMON command.
- ▶ *CC Working Set Peak w/Control Center activity* is the maximum size of the physical memory utilized for the WebSphere Application Server Control Center process during CC activity. Similar to DM, we utilized SVMON to determine this value. In this case, since our java maximum heap was 1 GB or less, the memory segment mapping is a little different and segments 3–4 include both the native and Java heap. Reporting and policy

deployment can require significant additional resources beyond steady state; hence, we provide this metric for sizing.

- ▶ *EWLM Total Working Set Peak* represents the total peak physical memory requirements for the EWLM domain manager. This is simply "DM Working Set Peak w/CC" + "CC Working Set Peak w/CC".
- ▶ *EWLM DB size*: The DM and WebSphere Application Server processes both have embedded internal database instances to house persistent data such as policies, server information, one-minute reporting data, and so forth. This field represents the combined disk requirements witnessed for these databases. In our tests, the largest disk consumer was domain manager's EWLM Data Reporting database (houses 1-minute reporting data utilized for CC interval reports). This database was responsible for more than 95 percent of the total database disk requirements.
- ▶ *DM Net KBytes Rcvd/s* is the aggregated rate of data inbound to the domain manager process from the Management Domain of managed servers. This rate is predominantly driven by the managed server's *server statistic* messages. These messages constitute each managed server's aggregated transaction/server data for the past 10 seconds and are sent every 10 seconds.
- ▶ *Elapsed Time: 60min Interval Report* represents the elapsed time of requesting a 60-minute Monitor-Transaction Classes report from the EWLM Control Center. We think this command is the most data-intensive report request from the Control Center because it requires reading data for all transaction classes, so we utilized this metric as a measuring stick of Control Center responsiveness.

11.2.11 General sizing recommendations on AIX

Based on our internal testing, we offer the following general sizing guidance.

Minimum system requirements

A pSeries 1-way POWER4 1.4 GHz (or comparable) with 512 MB memory is the minimum system recommended for a domain manager that is handling between 1 and 50 managed servers and a policy of 40 transaction/process classes and 10 service classes.

General system requirements guidance

1 to 100 managed servers with a policy up to 80 transaction/process classes and 20 service classes should safely fit on a pSeries 2-way POWER4 1.4GHz (or comparable) with 1GB RAM. Default java -maxHeap settings of 512MB for domain manager and 256MB for the Control Center should be sufficient. Beyond product install disk requirements, an additional 1GB of disk space should be sufficient for EWLM's database requirements.

100 to 500 managed servers with a policy up to 80 transaction/process classes and 20 service classes should safely fit on a pSeries 4-way POWER4 1.4 GHz (or comparable) with 2.5 GB RAM. To more accurately project your requirements for these larger configurations, we suggest monitoring EWLM domain manager memory and processor resource usage as your domain grows close to 100 servers. Domain manager and WebSphere Application Server maxHeap settings potentially need to be increased to avoid out-of-memory conditions for Java heap. Beyond product install disk requirements, an additional 2 GB of disk space should be sufficient for EWLM's database requirements.

Memory guidance

Ensure sufficient physical memory to back the domain manager and WebSphere Application Server processes. We recommend monitoring the memory and processor consumption of these processes upon any substantial changes to size or policy complexity of the

Management Domain (for example, going from 10s to 100s of managed servers; 10 to 50 transaction classes). The default maxHeap settings for these are: Domain manager - 512 MB, WebSphere Application Server - 256 MB. The maxHeap for the domain manager can be increased on the **startDM** script via the **-maxHeap** parameter. The maxHeap for WebSphere Control Center can be increased via the WAS Administrator Console.

The 504 managed server/simple policy and the 252/504 managed server/complex policy test scenarios all required increasing the java maximum heap for the domain manager process. Additionally, the 504 managed server/complex policy test scenario required increasing the maximum heap for the WebSphere Application Server process. Recall we had already set the domain manager process **-maxHeap** to 1800 for all of our tests. After first hitting an out-of-memory condition on the WebSphere Application Server, we re-started it with a maximum heap size of 320 MB.

11.2.12 Domain manager on i5/OS

Following are some charts that show the domain manager resources usage when running on i5/OS.

Figure 11-7 shows the CPU, the memory, and the disk usage for the domain manager internal database with the different model policies and the managed servers scaling factor.

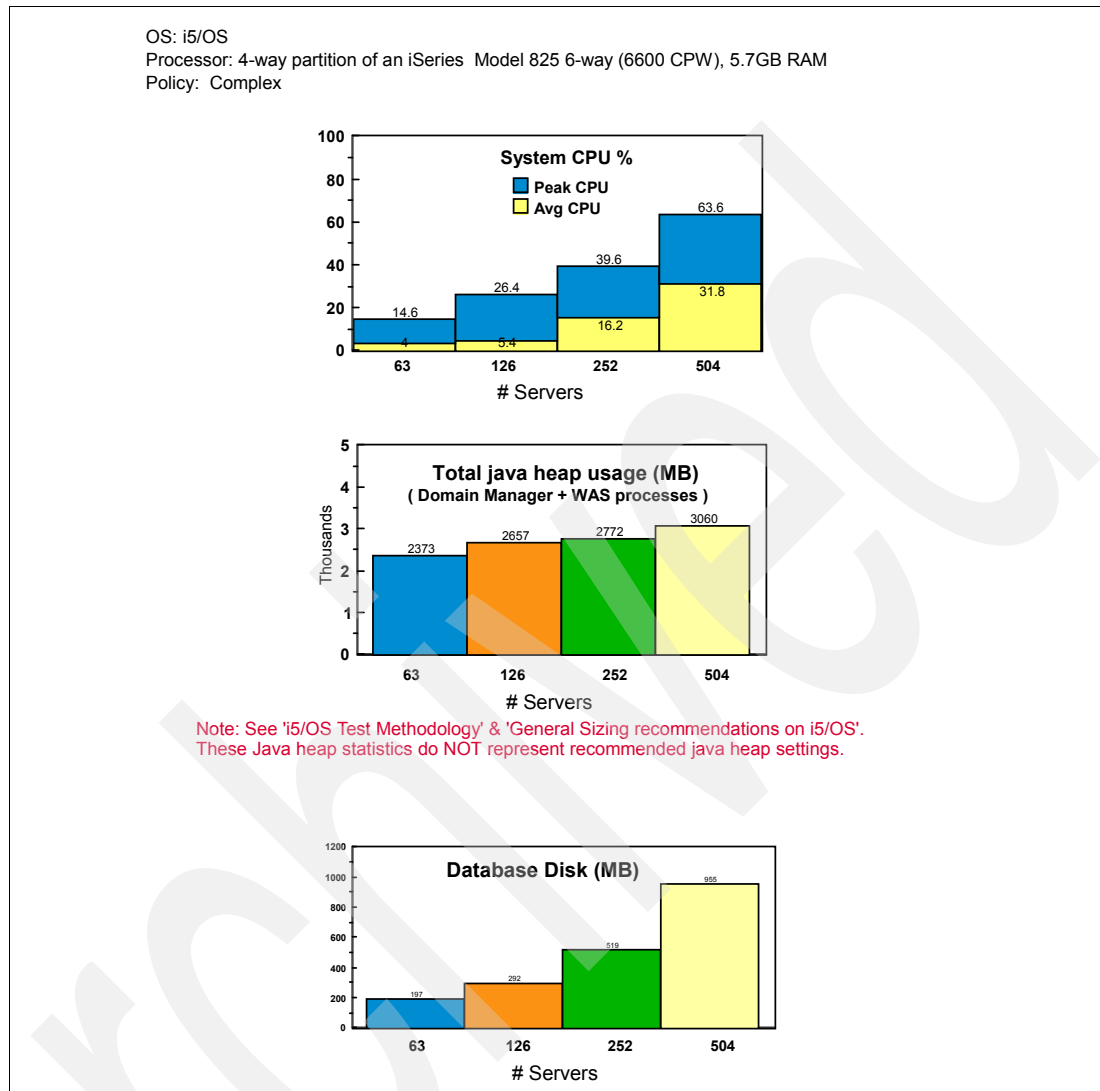


Figure 11-7 Resources usage on i5/OS domain manager

11.2.13 i5/OS test methodology

For the i5/OS test methodology:

- ▶ The Java heap in i5/OS is structured very differently than those on other platforms. The initial heap size is also the Garbage Collector (GC) allocation threshold. That number indicates how much the heap should be allowed to grow between GC cycles. After a GC cycle the heap shrinks to the size necessary to contain the objects that still have valid references. After several cycles, this value can be much larger or smaller than the initial heap. Setting the maximum heap is impractical on i5/OS because it causes the Garbage Collector to run in a synchronous manner, which is very degrading to performance.
- ▶ The initial heap setting has a very large impact on CPU and memory usage. Users will need to specify this value to achieve acceptable performance. The initial heap setting determines the balance of CPU versus memory usage. The initial heap is set by putting "os400.gc.heap.size.init=SOMEVALUE" in the /home/qwlm/SystemDefault.properties file, where SOMEVALUE is the initial heap size in kilobytes. This must be done before calling

STRWLM. If the `os400.gc.heap.size.init` option is omitted or set too low, the domain manager's CPU usage will rise drastically. This is because the Garbage Collector is working hard to keep the memory usage down. If it is set too high, then the memory footprint will grow.

- ▶ The i5/OS charts in this book represent processor and memory utilization with the initial heap size set to 1200 MB. In practice, domain managers managing fewer than 300 servers would most likely run with the initial heap size at 256 MB or less, greatly reducing the memory footprint of the domain manager.
- ▶ Performance metrics were captured during Steady State and with Control Center activity windows:
 - Steady State: The base domain manager processes that gather and aggregate the managed servers statistics. The Control Center was started but there was no end user activity (no reporting, no policy changes, and so forth). This measurement is a ten-minute average.
 - Control Center activity: A user requests a transaction class listing.
- ▶ We utilized default system settings. No specific I/O or system performance tuning was conducted.

Detailed results are shown in Table 11-9.

Table 11-9 Metrics observations for complex policy

Servers	Avg system CP util Steady State	System CP util spikes with Control Center activity	DM Java heap size with Control Center activity (MB)	Control Center WebSphere Application Server Java heap size with Control Center activity (MB)	Total Java heap usage (MB)	EWLM DB size (MB)	Elaps time: 60min interval report (sec)
63	2.2	28	190	196	438	194	2.0
126	5.1	39	340	280	672	290	2-3
252	12.3	53	533	340	974	520	4-10
504	23.9	81	1240	403	1790	995	10-30

Where:

- ▶ *Servers* is the number of the managed server in the EWLM Management Domain.
- ▶ *Average system CP utilization Steady State* is the average total processor utilization for the server. During steady state, the domain manager process consumes more than 98 percent of the total processor utilization.
- ▶ *System CP Utilization spikes w/Control Center activity* is the highest average system processor utilization monitored (over a 10-second interval) upon requesting reporting data or conducting a policy switch or combination of both. The domain manager and WebSphere Application Server Control Center processes make up more than 98 percent of the total System processor utilization.
- ▶ *DM Java heap size w/Control Center activity* represents the maximum expanded domain manager Java heap. The Java heap constitutes the majority of the domain manager process memory requirements.

- ▶ *CC Java heap size w/Control Center activity* represents the maximum expanded Control Center WebSphere Application Server Java heap. The Java heap constitutes the majority of the Control Center WebSphere Application Server process memory requirements.
- ▶ *Total Java heap usage*: This is simply adding the domain manager Java heap size to the Control Center Java heap size.
- ▶ *EWLM DB size*: The domain manager and WebSphere Application Server processes both have embedded internal database instances to house persistent data such as policies, server information, one-minute reporting data, and so forth. This field represents the combined disk requirements witnessed for these databases. In our tests, the largest disk consumer was domain manager's EWLM Data Reporting database (houses 1-minute reporting data utilized for CC interval reports). This database was responsible for more than 95 percent of the total database disk requirements.
- ▶ *Elapsed Time: 60min Interval Report* represents the elapsed time of requesting a 60-minute Monitor-Transaction Classes report from the EWLM Control Center. We think this command is the most data-intensive report request from the Control Center because it requires reading data for all transaction classes, so we utilized this metric as a measuring stick of Control Center responsiveness.

11.2.14 General sizing recommendations on i5/OS

Based on our internal testing, we offer the following general sizing guidance.

Minimum system requirements

A 700 CPW iSeries with 1.5 GB memory is the minimum system recommended for a domain manager that is handling 63 or fewer managed servers and a policy of 40 transaction/process classes and 10 service classes.

General system requirements guidance

Some general system requirements are:

- ▶ 100 to 300 managed servers with a policy up to 80 transaction/process classes and 20 service classes should safely fit on a 1100 CPW iSeries with 2 GB RAM.
- ▶ 300 to 500 managed servers with a policy up to 80 transaction/process classes and 20 service classes should safely fit on a 2200 CPW iSeries with 2.5 GB RAM.
- ▶ Domain manager initial heap settings need to be adjusted to fit the particular combination of available CPU and memory. Higher CPW configurations can use less memory by increasing the initial heap size. Configurations with more memory can use less CPU by decreasing the initial heap size. To reduce both memory and CPW requirements, first reduce the policy complexity, then reduce the number of managed servers.
- ▶ We suggest monitoring memory and processor resource usage as the configuration grows to more than 100 servers to more accurately project your requirements for these larger configurations.
- ▶ Beyond product install disk requirements, an additional 1 GB of disk space should be sufficient for EWLM's database requirements.

Memory guidance

Ensure sufficient physical memory to back the domain manager and WebSphere Application Server processes. We recommended that you monitor the memory and the processor consumption of these processes upon any substantial changes to size or policy complexity of the Management Domain, for example, going from 10s to 100s of managed servers or 10 to 50 transaction classes.

Recall that the memory footprint of the domain manager can be quickly reduced by using a less complex policy. Increased CPU utilization can be exchanged for a reduction in the memory footprint of the domain manager by reducing the initial heap size. If paging is observed and there is surplus CPW, consider adjusting the heap size first. If paging is observed and there is not enough spare CPW to reduce the heap size, reduce policy complexity, then reduce the number of managed servers as needed.

11.3 Additional considerations

Do not place the domain manager on a system where other applications can potentially saturate processor resources for prolonged periods of time.

For large Management Domains with complex policies consider placing the EWLM working directory on a higher performance disk. This is to improve access times of EWLM's internal database.

11.4 Resource requirements for the EWLM managed server

For EWLM to maintain information about the quality of service being delivered for applications on production machines, an investment in Application Response Metrics (ARM) transaction monitoring and performance data collection/aggregation is required. The relative amount of resource investment depends on the intensity of these transactions, the number of times that the transactions have transitions from one service or server to another, and the capabilities of the system.

Figure 11-8 exhibits a simple example of a two-system EWLM Management Domain, consisting of a Management LPAR and two managed servers (WebSphere and DB2 LPARs). Within the Management LPAR are the functional components necessary for an EWLM Domain Manager. This book focuses on the z/OS image on the right side of the diagram. Depicted, at a high level, is the flow of a *two-hop* (WebSphere to DB2) ARM-instrumented transactional workload and the synchronous/asynchronous interactions of the new functional components.

Within an EWLM managed server, there are three key functional areas of processing:

- ▶ The EWLM ARM and platform services. This is the operating system extension support that provides the actual ARM service routines for ARM-instrumented middleware and platform services for capturing process and system resource statistics. Data is passed to the EWLM managed server process asynchronous to the transactions themselves.
- ▶ The middleware ARM calls and associated processing. This is essentially the instrumentation hooks placed into middleware, which captures the transaction statistics (for example, active time, blocked time, queue time).
- ▶ The EWLM managed server process. This is a Java application that is common across all platforms. Its primary responsibility is to collect and aggregate transaction (ARM instrumented), process, and system resource data. Additionally, it is responsible for communicating this aggregated data to the EWLM domain manager on an interval basis, asynchronous to transactions themselves.

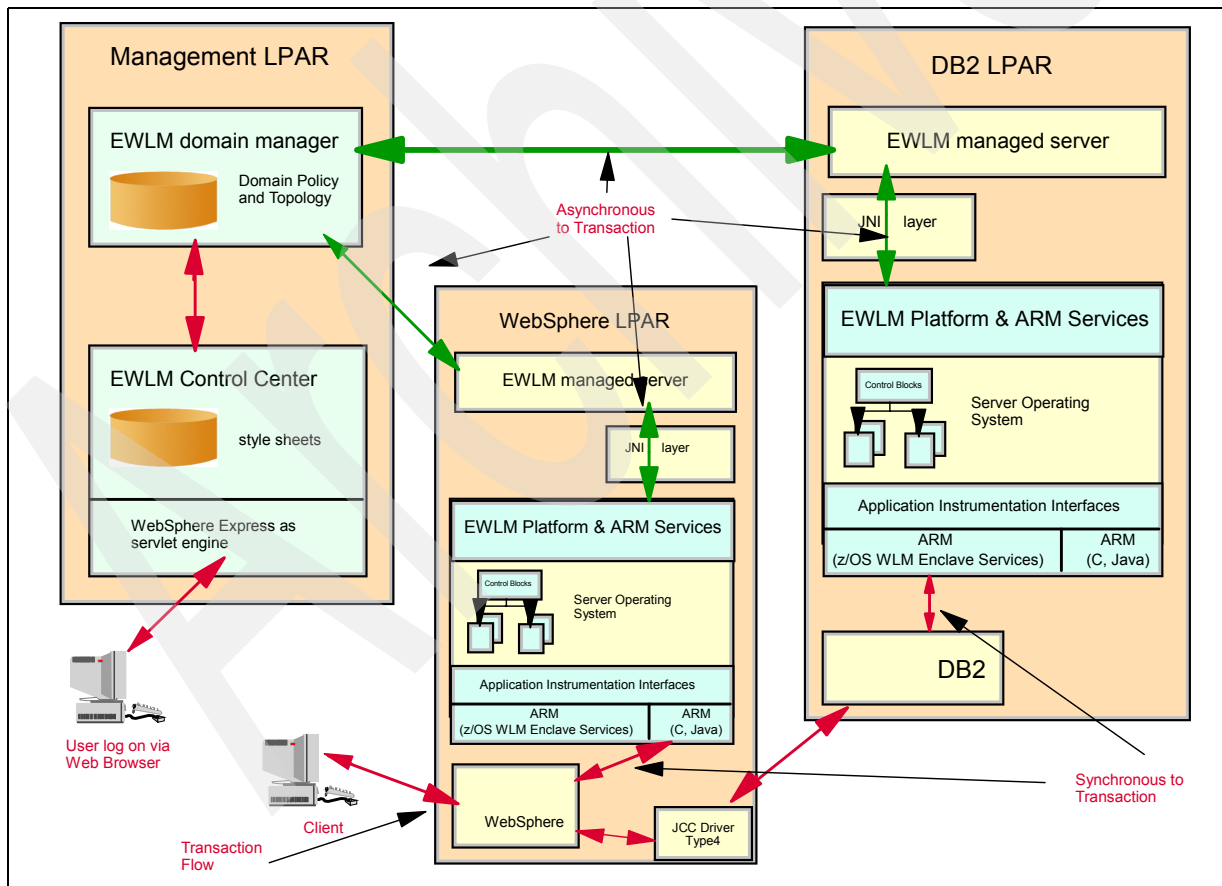


Figure 11-8 EWLM components flow

ARM processing

IBM middleware products, like DB2 and WebSphere, have the capability to collect transaction information without the need for direct changes to the application code to call the ARM APIs that help to collect transaction information. The relative investment of resources needed to support this information will depend on the number of ARM calls and EWLM weight of the transaction itself.

For example, a typical transaction flow is shown in Table 11-10.

Table 11-10 Typical transaction flow

WebSphere Application Server	DB2 Server
Start Transaction	
<i>Process information A.</i>	
Block transaction.	
	Start transaction.
	Run query 1.
	Stop transaction.
Unblock transaction.	
<i>Process information B.</i>	
Block transaction.	
	Start transaction.
	Run query 2.
	Stop transaction.
Unblock transaction.	
<i>Process information C</i>	
End transaction	

The start, stop, block, and unblock transactions are simple ARM calls to keep track of the state of the transaction as it flows between the WebSphere Application Server and the DB2 server. Generally, the time spent in the WebSphere application (beginning with *process information*) and the time spent in DB2 (beginning with *run query*) is much more than the time spent in the ARM calls, resulting in a fairly low resource investment to enable EWLM on the system.

However, one can tell that if the queries are extremely simple or if the processing in between DB2 calls in WebSphere is not very substantial, there is a possibility that the ARM overhead will be more noticeable.

11.4.1 Considerations for the z/OS managed server

Measurements were conducted to evaluate the resource requirements of EWLM managing an ARM-instrumented DB2 DDF transactional workload and an ARM-instrumented WebSphere transactional workload. In each evaluation, the focus was the resource requirements for the given z/OS hop. For both evaluations, a *hop* represented one ARM-instrumented middleware component on one operating system image (as shown in Figure 11-8 on page 296).

Additionally, in this section we provide expectations on zAAP-eligibility of the new processing requirements and suggestions about the z/OS WLM policy changes to manage the new EWLM managed server java application.

DB2 DDF ARM-instrumented workload exploiting EWLM

Prior to discussing the measurement results, it is important to appreciate the differences between an EWLM ARM instrumented DB2 transaction and the current z/OS WLM definition of a DB2 transaction.

For details and a description of DB2 transaction versus EWLM transactions, refer to 1.3.1, “Workload management on zSeries” on page 15.

z/OS DB2 DDF measurements

The following environment has been used for the measurements:

Workloads In an attempt to represent some workload variations, two independent workloads were measured:

- TRADE - Simulates an online stock trading application

- IRWW (IBM Relational Warehouse Workload (IRWW)) - Simulates a typical customer workload against a warehouse database.

And one was projected: IRWW Stored Procedure. Same workload as IRWW, but converted to use a Stored Procedure, minimizing the amount of DRDA message flows.

Measurements were executed in slightly different environments, with the following common description:

Processor Two to three dedicated processor LPARs of a z990 2084-3xx model.
Memory: 6 GB.

Software EWLM V2.1
z/OS1.6 + APAR OA12005
DB2 V8 + DB2 V8 PTF UK08986

JCC driver 2.6.81 or later was utilized on the client (WebSphere).

Note: The WebSphere frontend was running on another platform. A Type 4 JDBC connector was utilized for accessing DB2.

Methodology For all three workloads, a base scenario was measured. The base scenario consisted of running the workload without an EWLM managed server and ensured that the transactions were not ARM-instrumented. For the latter, we ensured that WebSphere (the frontend) was not passing in a corrector to DB2 DDF. For IRWW and TRADE, an EWLM-managed scenario was measured. For IRWW Stored Procedures, the EWLM-managed scenario was projected.

The results are:

► Disk requirements

Captured for TRADE workload. Assumed to be the same for the rest. The additional disk required was approximately 25 MB, which was for the EWLM-managed server executable and one configID directory.

We document 128 MB in the IBM System Software Information Center documentation, which includes a buffer for potential EWLM diagnostics dumps/traces and maintenance back-ups.

► Memory requirements

Captured for TRADE workload. In our measurements we witnessed an approximate 125 MB growth in real memory usage (approximately 110 MB associated with the EWLM managed server address spaces and approximately 15 MB growth in the z/OS WLM address space). The actual working set size could potentially be less. No tuning or memory constraint tests were performed.

► Impact to transaction response time

We noticed a negligible increase in response time when running with EWLM. The average transaction response time went from 12 to 13 milliseconds.

► Processor requirements

As mentioned earlier, the key factors determining the relative investment are the size of the transactions (the average transaction processing cycles) and the number of times that the transactions have transitions from one service or server to another. (In the case of DB2 DIST this is the number of DRDA messages per DB2 transaction - Commit).

Figure 11-9 depicts the relative increase in processing required for the three workloads.

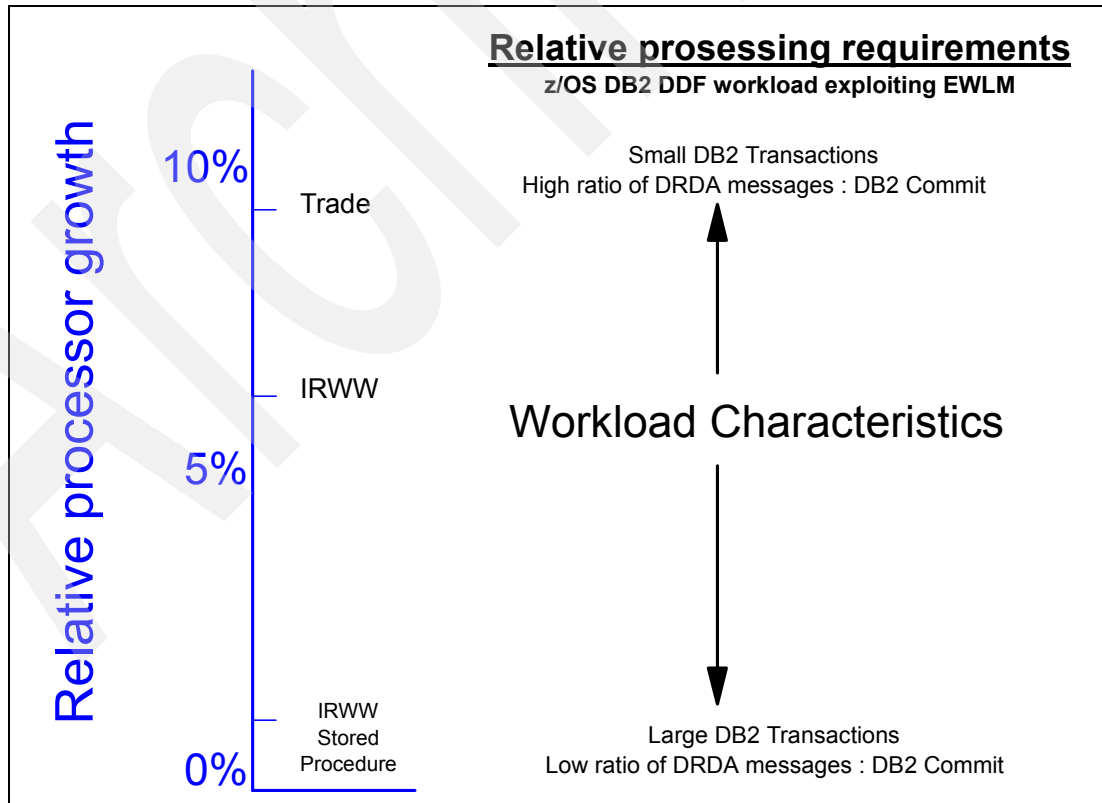


Figure 11-9 DB2 workload processing requirements

► Workload characteristics

The DB2 TRADE transactions are generally considered to have light-weight processing requirements. On average, the number of DRDA messages per DB2 Commit (or z/OS WLM transaction) was 8.

The DB2 IRWW transactions are generally considered to have medium-weight processing requirements. In our measurements, the IRWW were approximately 2.4X larger than the TRADE transactions. On average, the number of DRDA messages per DB2 Commit (or zWLM transaction) was 19.

The DB2 IRWW Stored Procedures transactions are generally considered to have medium-processing. On average, the number of DRDA messages per DB2 Commit (or z/OS WLM transaction) was 2.

Summary

The higher processing requirements are predominantly associated with the necessary EWLM transaction processing within the EWLM Managed server Java application. Essentially, the more DRDA messages, the more EWLM transactions. Additionally, the smaller the transaction, the larger the relative impact. Conversely, the larger the transaction, the smaller the relative impact.

WebSphere ARM-instrumented workload exploiting EWLM

The underlying ARM-instrumentation implementation for WebSphere is different than that of z/OS DB2 DDF. While the DB2 DDF instrumentation is built on top of the existing z/OS WLM enclave support, the WebSphere implementation utilizes the new JAVA ARM APIs.

z/OS WebSphere Application Server measurements

The following environment has been used for the measurements:

Workload	TRADE.
Processor	Four dedicated processor LPARs of a z990 2084-3xx model. Memory: 6 GB.
Software	EWLM V2.1. z/OS1.6. WebSphere Application Server V6.0.1. DB2 V8.

Important: For this evaluation, the DB2 subsystem was residing on the same LPAR as WebSphere. A Type2 RRS (local) JDBC connection was utilized. This local connection typifies the majority of z/OS installations running both a z/OS WebSphere and z/OS DB2. Recall that DB2 only ARM-instruments distributed transactions. Hence, the DB2 in this scenario is not considered a second hop to EWLM. To EWLM, this is a single-hop WebSphere transaction and DB2 itself is not ARM-instrumented.

Methodology The base scenario consisted of running the workload without an EWLM Managed Server and ensured that the transactions were not ARM-instrumented.

The results are:

► Disk requirements

Captured for TRADE workload. These are assumed to be the same for the rest. The additional disk required was approximately 25 MB, which was for the EWLM Managed server executables and one config-id directory.

We document 128 MB in the IBM System Software Information Center documentation, which includes a buffer for potential EWLM diagnostics dumps/traces and maintenance back-ups.

► Memory requirements

Captured for TRADE workload. In our measurements we witnessed an approximate 115-MB growth in real memory usage. Approximately 90 MB (10 MB heap, 80 MB Native heap), associated with the EWLM Managed Server address spaces and approximately 15-MB growth in the z/OS WLM address space. The actual working set size could potentially be less. No tuning or memory constraint tests were performed.

► Impact to transaction response time

We noticed a negligible increase in response time when running with EWLM. The average transaction response time went from 21 to 23 milliseconds.

► Processor requirements

As mentioned earlier, the key factors determining the relative investment are the intensity of the transactions (the average transaction processing cycles) and the number of times that the transactions have transitions from one service or server to another. (In the case of WebSphere this is the number of calls to DB2 associated with each WebSphere transaction.) Figure 11-10 depicts the relative increase in processing required for the three workloads.

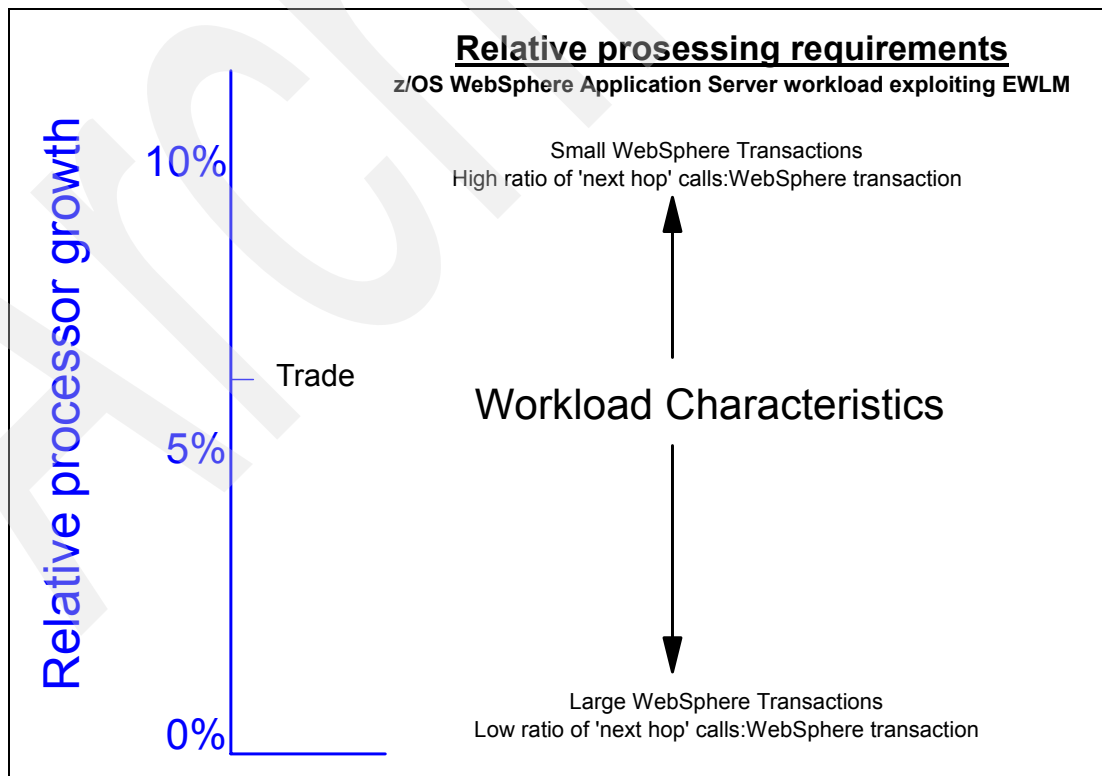


Figure 11-10 WebSphere workload processing requirements

- ▶ Workload characteristics

The WebSphere TRADE transactions are generally considered to have light-weight processing requirements. On average, the number of DB2 calls per WebSphere transaction was 7. Some of the WebSphere TRADE transactions do not have any DB2 transactions associated with them.

Summary

The predominant portion of the processing requirements is due to the WebSphere ARM Block and UnBlock calls associated with multiple DB2 calls for each TRADE WebSphere transaction.

11.4.2 Classifying the EWLM managed server to z/OS WLM

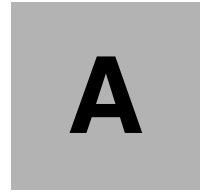
Since the managed server is a key component of monitoring and managing resources, we suggest that its address spaces be classified to either SYSSTC or a high dispatch priority service class with high importance and aggressive goals.

For these managed server address spaces (EWLMMS*), a classification rule should be added to both the OMVS and STC sub-systems. Three of the four EWLMMS* address spaces, including the key one, will get classified under OMVS.

Additionally, a report class might be suggested for the same, if monitoring the resource usage of the managed server Java agent itself, is important.

11.4.3 zAAP eligibility

The managed server processing can be zAAP-eligible. In fact, for either DB2 DDF or WebSphere EWLM exploitation scenarios, we would anticipate roughly 10–20% of the total additional processing cycles, eligible for zAAP. This is based on zAAP projection (utilizing –Xifa:force) experiments conducted with the TRADE workload.



EWLM domain policies

This appendix describes the domain policies provided with the EWLM product.

The default and sample policies

EWLM provides a default Domain Policy that is deployed automatically until a user-defined policy is deployed in the installation. When you are running under the default Domain Policy, it is important to understand that all work is assigned to a discretionary goal. Once a Domain Policy is deployed, the default Domain Policy cannot be activated again, but its components (like transaction classes and so forth) are available to be included in new domain policies.

In addition to the default Domain Policy, EWLM provides two sample domain policies: The Sample Domain Policy and the Sample Banking Domain Policy. Most of the time these policies should represent a good starting point to build your own policy. So, an easy way to start is to analyze the content of the sample policies, determine which one is closer to your installation objectives, make a copy of the policy, and alter it to reflect the specific needs of your installation. Now, let us have a look at the sample policies.

Sample Domain Policy characteristics

The Sample Domain Policy with its service policy is designed for an installation with Apache WebSphere transactions using the UDB workload mainly running on Windows platforms.

The following characteristics apply to this policy:

- ▶ One workload: EWLM Workload, which represents all work and is the EWLM default
- ▶ One application: IBM Webserving Plugin
- ▶ Five transaction classes are defined to this policy:
 - Web Low; Filter=EWLM Application Instance - stringmatch on IISA
Service Class: Web Low
 - Web Hot; Filter=EWLM Application Instance - stringmatch on IISB
Service Class: Web Hot
 - Web Medium; Filter=EWLM Application Instance - stringmatch on IIS*
Service Class: Web Medium
 - IBM Webserving Plugin (Default for Policy); Filter=\`*stringmatch*` (this is the catch all for work not classified out of the web serving plug-in)
Service Class: EWLM Service Class (EWLM default)
 - EWLM Default Application Transaction Class (EWLM Default) (catch all for any other)
Service Class: EWLM Service Class (this is the EWLM default)
- ▶ One platform: Windows 2000/2003
- ▶ One process: Web browser
 - Filter = ExecutablePath - stringmatch on
C:\Program\Files\Netscape\Communicator\Program\netscape.exe
Service Class: Web browsers
- ▶ Five service classes:
 - Web Low
 - Goal type: Response Time Percentile at 75%
 - Goal Importance: Low
 - Goal: 10 second response time
 - Web Medium
 - Goal type: Response Time Percentile at 80%

- Goal Importance: Medium
- Goal: 5 second response time
- Web High
 - Goal type: Response Time Percentile at 85%
 - Goal Importance: High
 - Goal: 3 second response time
- Web browser
 - Goal type: Velocity
 - Goal Importance: Low
 - Goal: moderate
- EWLM Service Class
 - Goal type: Discretionary

Sample Banking Domain Policy characteristics

The Sample Banking Domain Policy has the objective of representing a Web banking application environment through its Banking 2004 Service Policy.

The Web banking applications include funds transfer, an account query, a mortgage rate advisor, brokerage stock buying, the static Web page content (graphics and text), and a WebSphere Application Server EJB application, which processes bank statements. All are transaction-based work. All are on ARM 4.0 instrumented middleware applications, except for the mortgage rate advisor application, which is on AIX. It passes requests to DB2 UDB, which is ARM 4.0 instrumented. The uninstrumented Web application server can be included in the end-to-end Domain Policy only as process-oriented work, since the transaction-level performance data is not available through the instrumentation. But because the uninstrumented Web application server passes data to DB2 UDB, which is instrumented, the UDB portion of the transactions can be monitored and managed by EWLM and therefore can be included in the Domain Policy.

The Web banking application is considered to be running on Microsoft IIS on Windows 2003 servers, IBM WebSphere Application Server 5.1 on Solaris 8, with a DB2 Universal Database (UDB) backend on AIX. The mortgage application runs on a non-instrumented Web application server on AIX 5L Version 5.2 ML03.

Table A-1 describes the service classes contained in the Sample Banking Domain Service Policy.

Table A-1 Sample Banking Domain Policy service classes

Service class	Importance	Performance goal
WebSphere EJB	Highest	85% complete in 4 seconds
Account Queries	High	80% complete in 3 seconds
Brokerage Buyers	Highest	80% complete in 1 seconds
Stock Quotes	Medium	80% complete in 10 seconds
Static Web Serving	Low	90% complete in 20 seconds
Mortgage Rate	Medium	1 minute avg response time
Non instrumented Web Appl	NA	Discretionary
Web Browsers	Medium	Moderate velocity

Service class	Importance	Performance goal
EWLM Service class	NA	Discretionary

For the *mortgage rate* application, consider the end-to-end response time and the time the transactions normally spend in the database query. Since the Web application server is not instrumented, the performance objectives defined in the Domain Policy represent only the portion of time the transactions spend in DB2 UDB. The application is not quite real-time. The portion of the time spent in DB2 for the query is estimated to be about 20 seconds, so the response time objective is set for 1 minute.

For the mortgage rate application, EWLM can manage and monitor the uninstrumented Web application server regions. A service class is defined for the non-instrumented Web application server region with a velocity goal. The administrator defines a general service class name. Velocity is specified as a category Fastest, Fast, Moderate, Slow, or Slowest. The administrator would also like to manage the processes running the Web browsers in the banks. The browsers, Internet Explorer and Netscape Navigator, are processes, and are also assigned a velocity goal.

For work not otherwise categorized, EWLM provides a service class in every Domain Policy called EWLM service class. It is assigned a discretionary goal, which specifies that work is to be completed when resources are available. You can assign a discretionary goal type to a service class representing low priority work that does not require a particular performance goal. EWLM processes discretionary work using resources not required to meet the goals of other service classes.

Based on the service classes, the following workloads are defined for reporting purposes:

- ▶ WAS applications: WebSphere Application Server EJB application that processes the bank statements.
- ▶ DB2 applications: The mortgage advisor transactions originating from the non-ARM instrumented application running in DB2 UDB.
- ▶ Non-instrumented applications: The Web application server process and the Web browser processes.
- ▶ Web site applications: The funds transfer, account query, mortgage rate advisor, and brokerage buyer transactions, and the static Web serving work.
- ▶ EWLM provides a workload, EWLM Workload, which is assigned to the EWLM service class.

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246785>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246785.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
SG246785.zip	Zipped Code Samples, readme file

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder. Open the readme file: this file contains the description of the other files and the instructions about how to upload and use the files.

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 310. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *DB2 UDB V8 and WebSphere V5 Performance Tuning and Operations Guide*, SG24-7068

Other publications

These publications are also relevant as further information sources:

- ▶ *WebSphere Studio Workload Simulator User's Guide*, SC31-6307
- ▶ *WebSphere Studio Workload Simulator Programming Reference*, SC31-6308
- ▶ *InfoCenter EWLM*
<http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/ewlminfo/eicaa.pdf>

Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ Information Center for IBM @server
http://publib.boulder.ibm.com/eserver/v1r2m0f/en_US/info/ewlminfo/kickoff.htm
- ▶ IBM Virtualization Engine
http://publib.boulder.ibm.com/eserver/cur/v1r2m0f/en_US/index.htm?info/veicinfo/eicarkickoff.htm
- ▶ The WebSphere Application Server Version 5.1 Information Center
<http://publib.boulder.ibm.com/infocenter/ws51help/>
- ▶ The WebSphere Application Server Version 6.0 Information Center
<http://publib.boulder.ibm.com/infocenter/ws60help/>
- ▶ The DB2 Information Center
<http://publib.boulder.ibm.com/infocenter/db2help/>
- ▶ The IBM WebSphere Studio Workload Simulator for z/OS and OS/390 home page
<http://www.ibm.com/software/awdtools/studioworkloadsimulator/>
- ▶ IBM Systems Software Information Center (Enterprise Workload Manager)
<http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/ewlminfo/eicaakickoff.htm>
- ▶ IBM Systems Software Information Center (Virtualization Engine)
<http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/eicay/eicayintrove.htm>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

- activate a policy 155
- AIX Workload Manager 17–18
- algorithm trace 258
- Application Environment 129
- application flow 172
- applications
 - security 102
- ARM 7, 104
 - arm_block_transaction 12
 - arm_register_application 12
 - arm_start_tran() 13
 - arm_start_transaction 12
 - arm_stop_application 12
 - arm_stop_tran() 13
 - arm_stop_transaction 12
 - arm_unblock_transaction 12
 - correlator 14
 - description 11
 - sample API calls 13
 - verification 124
- ARM enablement on z/OS 16
- arm_stop_application 12
- assessing the environment 212

B

- balancing
 - application-level 200
 - system-level 200

C

- CBPDO 58
- certificate, trusting 93
- changeCC, use to enable security 92
- Client/Server SSL authentication 89
- Client/Server SSL security 99
- common components 29
- communication 6
- compressed files 42
- configuration ID 38
- Configuring the domain manager 37
- configWAS.sh 61
- Control Center 5–6
 - administrator 55
 - description 8
 - log in 8
 - modes of operation 9
 - monitor 55
 - monitoring tasks 9
 - operator 55
 - reporting 10
 - reports 250
 - role 55

- roles 127, 191
- security 90
- users 54
- correlator 14
- createDM.sh 65
- createWAS.sh 64

D

- data collection 4
- data gathering 9
- DB2
 - AIX variables 107
 - instrumenting 106
 - instrumenting on AIX 107
 - instrumenting on Solaris 110
 - transaction 17
 - variables 107
 - verify variables 109
- DB2 PE 239
- DB2 UDB Enterprise Server Edition 29
- DB2 Universal Database Version 8.2 on z/OS
 - maintenance 108
- DDF 16
- DDF transactions 237
- domain manager 4
 - backup 6
 - definition 4
 - description 5
 - directory structure 40
 - performance on AIX 286, 291
 - performance on i5/OS 291
 - performance on Windows 282
 - policy activation 6
 - security 95
 - size 5
- domain policy 7, 132
 - application 137
 - building a policy 144
 - default 304
 - definition 4
 - deploying 143, 146
 - import function 141
 - platform 138
 - using the wizard 143
 - workload 133

E

- edge server 214
- entc 199
- Enterprise Workload Manager 2
- Entitled Capacity 193
- entitled capacity 196
- Error logs 257
- EWLM

- concepts 3
- domain manager 5
- policy-based 3
- scalability 6
- EWLM domain manager
 - configuration 34
- EWLM for z/OS 58
 - prerequisite 59
- EWLM procedures 67
- EWLMs partition management
 - sample 224

F

- firewall 78
 - EWLM support 79
 - packet filter 78
 - proxy 78
 - stateful inspection 79
- firewall broker
 - handling 86

G

- graphical reporting 177

H

- Hardware Management Console 193
- HMC 193
- hop 14
- Hop0 173
- HTTP
 - independent plug-in 122
 - instrumenting 119
 - WebSphere HTTP plug-in 119–120

I

- IBM HTTP plug-in 122
- IBM SDK for z/OS 60
- IBM Tivoli Director Server 29
- IBM Tivoli Directory Server 30
- IBM Virtualization Engine Suite 2
- IIS Plug-in 122
- importance 129, 134
- install EWLM domain manager 35
- installation
 - ITSO environment 29
 - verification 50
- installation and configuration 22
- installation log 252
- installEWLM.sh 63
- Instrumentation
 - verification 124
- instrumentation 11
- ITDS 29

J

- Java Virtual Machine
 - properties 112, 115

JVM

- custom properties 112

K

- keystore 92, 97
 - creating 92
 - installing 93

L

- LDAP repository 35
- LDAP server 30
- Load Balancer 200
 - AIX support 201
 - algorithms 200
 - domain manager 202
 - SASP 200
 - Solaris support 201
 - Windows support 201
- Load balancer
 - managed server support 201
- load balancers
 - installation 201
- load balancing 5
- log files 252
- log in
 - security 91

M

- manage
 - activate a policy 155
 - managed server 155
- Managed 7
- managed server 7
 - configuration 46
 - definition 4
 - description 7
 - Detailed Statistics 163
 - directories configuration 49
 - disabling 157
 - properties 157
 - security 96
 - status 156
 - z/OS installation 67
- managed servers
 - load balancers 201
- management domain 3–4, 8
- memory 286, 290, 294
- Micro-Partitioning 192
- Monitor
 - Preferences 150

N

- non-ARM instrumented environment 230

O

- on demand business 3
- onfiguration Wizard log 256

operating system extensions 7

P

Packaging 22
packaging 22
Packet filter firewall
 advantages 78
 disadvantages 78
Partition Load Manager 17–18
partition management 5, 192
partition workload group 194
partitions setting 218
Percentile Response Time 133
percentile response time 130
performance
 memory 286, 290, 294
performance goal 130
performance goals 133
Performance Index 160, 177
performance index 161, 225
physc 199
Physical Processor 192
PMI 111, 113
policies activation 6
Policy 5
Preferences 150
process class 138
 rules 136
processes
 security 101
Processing Capacity 192
processor utilization 198
Proxy firewall
 advantages 78
 disadvantages 78
 using 82
Proxy server 83

R

Redbooks Web site 310
 Contact us xiii
Report
 application topology 172
 exceptions 160
 managed server 163–164
 Managed Server details 168, 172
 process class details 165
 process classes 162
 server topology 172
 service class details 165
 service classes 161
 transaction class details 166, 168
 transactions classes 162
response time 130
RMF 239
RMF Monitor I 240

S

Sample Banking Domain Policy 304
Sample Banking Domain policy
 definitions 305
Sample Domain Policy 304
Sample Domain policy
 definitions 304
SASP 200
security
 authentication 89
Server SSL authentication 89
Server SSL security 97
ServerPac 58
ServerSSL security 97
service class 129, 133, 137
 Average Response Time 134
 Discretionary 134
 Importance 134
 Percentile Response Time 133
 Velocity 134
service classes
 recommandations 134
service level agreements 11
Service Policy 10
service policy 133
 adding 148
shifting capacity 233
SOCKS server 83
 configuration samples 83
SRM transaction 17
Stateful Inspection firewall
 using 81
Stateful inspection firewall
 advantages 79
 disadvantages 79
STRWLM 101
SVG viewer 24
system resources
 contention 129

T

transaction
 definition 12
transaction class 137
 rules 136
transaction classes
 defining 219

V

VE Common Components 26
Virtual Processor 192
Virtualization Engine Common Components 29

W

WebSphere Application Server 4, 29
 instrumenting 111, 113
Weight 193
WLM 3

- concepts example 128
- constructs 129
- WLM policy 15
- work classification 128
- work requests
 - identification 128
- Workload Manager 3
 - concepts 3

Z

- z/OS Workload Manager 3



IBM Enterprise Workload Manager V2.1

Archived



IBM Enterprise Workload Manager V2.1



How to configure operating systems and enable middleware for EWLM

Sample scenario for building a real-world Domain Policy

Load balancing and partition management exploitation

This IBM Redbook provides an introduction to the Enterprise Workload Manager (EWLM) Version 2 Release 1. In addition to describing the overall product concept and functionality, it presents a detailed discussion of the elements that comprise an EWLM solution.

Step-by-step instructions take you through the installation of EWLM code on multiple platforms, for both the domain manager and managed servers, and also show how to enable the instrumentation of the middleware for a 3-tier Web application. The features for administering EWLM are described, along with the monitoring, managing, and reporting capabilities.

Sample scenarios implemented in the ITSO environment are used to guide you through the process of classifying workload and defining and deploying your own EWLM policy. These scenarios are then used to demonstrate the business goal based partition management capabilities of EWLM.

Techniques for securing your EWLM domain as well as the load balancing capabilities of EWLM are described. Troubleshooting hints and tips are provided, along with some basic performance considerations to help you design the optimum EWLM solution.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**

SG24-6785-01

ISBN 0738494445