IBM Enterprise Workload Manager

How to configure operating systems and enable middleware for EWLM

Sample scenario for building a real-world domain policy

Integrate with security, load balancers, and firewalls

Paola Bari
Christopher Covill
Klaus Majewski
Cathrin Perzl
Margaret Radford
Kaoru Satoh
Dino Tonelli
Lynn Winkelbauer

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM Enterprise Workload Manager

March 2005

SG24-6350-00

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

First Edition (March 2005)

This edition applies to the EWLM product within the Version 1, Release 1, Modification 0 of IBM Virtualization Engine Management Servers (product number 5724-i71).

Contents

Notices

Trademarks

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

eserver®	DB2®	Redbooks™
eServer™	i5/OS™	Tivoli®
ibm.com®	iSeries™	Virtualization Engine™
IBM®	OS/390®	WebSphere®
AIX 5L™	OS/400®	xSeries®
AIX®	POWER4™	z/OS®
Cloudscape™	pSeries®	zSeries®
DB2 Universal Database™	Redbooks (logo) ™	

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Intel Inside (logos) are trademarks of Intel Corporation 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). In addition to describing the overall product concept and functionality, it presents a detailed discussion of the elements that are part of the 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 and reporting capabilities.

A sample scenario implemented in the ITSO environment is used to guide you through the process of classifying workload, and defining and deploying your own EWLM policy. This scenario is then used to demonstrate techniques for securing your implementation.

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.

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.

Christopher Covill is a Certified IT Specialist with 28 years of experience in systems programming (Network, CICS, DB2®, VM, MVS, VSE), performance, as well as systems and applications Assembly Language programming work. He is currently the team lead for the IBM Global Services Performance Services Team in Boulder, Colorado providing service to the distributed as well as mainframe platforms. Performance, a key issue during the IBM HONE application rightsizing effort from the mainframe to AIX®, was managed and considered one of Chris's major accomplishments on the distributed platform.

Klaus Majewski is the Technical Director for Stonesoft in the USA. He holds an M.Sc. in Computer Science from the Helsinki University of Technology. Before joining Stonesoft in 1999 Mr. Majewski worked as a Security Consultant for IBM Finland. For Stonesoft, Klaus started in Security Consulting and R&D. During the past 3 years he has worked with certifications for Stonesoft Firewall and VPN products (Common Criteria EAL4+, FIPS 140-2, and ICSA). He is responsible for technical issues pertaining to Stonesoft products on IBM platforms in the US. He has CISSP and CISA certifications.

Cathrin Perzl is an IT Specialist from IBM Munich, Germany. She is a member of the pSeries® Technical Sales Team of the Systems and Technology Group in EMEA Central Region. Her areas of expertise and main responsibilities are pSeries, AIX, and Workload Manager support to customers. 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.

Margaret Radford is a Certified IT Specialist in ADM/Application Integration. She joined IBM in 1986 with a BS degree in Computer Science and has held various development, management, and consulting positions. She recently moved from IBM Global Services - Global AMS to the IBM Design Center for e-business on demand in Poughkeepsie, where she is specializing in Grid and Autonomic solutions.

Kaoru Satoh is an IT specialist from IBM Japan. He works in the Integrated Technology Services, IBM Global Services and has three years of experience in Linux® Services. He is a Red Hat Certified Engineer. His areas of expertise include Linux, WebSphere Application Server, and Linux High Performance Computing Clusters. He holds a Master of Engineering degree.

Dino Tonelli is a systems performance analyst for IBM, Poughkeepsie. He has 15 years of experience in zSeries® performance, working on Business Intelligence, zWLM, Parallel Sysplex and LSPR projects. Dino is currently conducting performance evaluations of IBM's Virtualization Engine™ EWLM product. He holds an MS degree in Computer Science from Polytechnic University.

Lynn Winkelbauer is a System Engineer for IBM, Poughkeepsie. She has 20 years of experience in the zSeries area, working in Performance, Parallel Sysplex, and e-business on demand, specializing in the Linux area. She holds a Bachelor of Science degree in Computer Science and Business Administration. Her areas of expertise include Linux for zSeries, CICS Transaction Server, WebSphere, and performance analysis.

The EWLM Redbook team, from left to right: Kaoru Satoh, Margaret Radford, Cathrin Perzl, Klaus Majewski, Paola Bari, Lynn Winkelbauer, Christopher Covill

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

Richard Conway, Robert Haimowitz, Tamas Vilaghy, Alison Chandler, Ella Buslovich
International Technical Support Organization, Poughkeepsie Center

Jeff Aman, John Arwe, Dave Bostjancic, Michael Buzzetti, CheKim Chhuor, Geoffrey Lubold, Kirsten Brunner McDonald, Peggy McKeon, Dave Manners, Bogdan Radulescu, Hiren Shah, Diane Smith, Sherry Whitney, Peter Yocom
IBM Poughkeepsie

Alan Bivens, Mark Hulber, Mathew S. Thoennes
IBM Watson Lab

George DeStefano, Eric Kobrin, Linda Kinnunen, Robert Mahlik, William R Menoyo, Tim O'Keefe, Tom Schuler, Richard Sit, Robb Wiedrich
IBM Rochester

Yuksel Gunal
IBM Middletown

Michael Mall
IBM Austin

Brian You
IBM Seattle

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. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

IBM Enterprise Workload Manager Overview

This chapter provides an introduction to the IBM Enterprise Workload Manager (EWLM). It explains how ELWM fits into the overall on demand environment and describes the functionality of the elements that make up this solution.

1.1 EWLM strategy

Enterprise Workload Manager (EWLM) is a product in the IBM Virtualization Engine Suite for Servers that can dynamically monitor and manage distributed heterogeneous workloads to achieve user-defined business goals. Before considering its detailed functionality, let's look at how EWLM fits into the IBM Virtualization Engine and how it supports an on demand operating environment.

An "on demand" operating environment *is not* a single product, brand, platform, or architecture. An on demand operating environment is a set of capabilities that enable business flexibility and IT simplification. When these capabilities are implemented, they enable a company to evolve into an on demand business.

Creating an on demand operating environment is not a matter of throwing away your existing infrastructure, it's a matter of optimizing it. Optimization is focused on two key areas:

- Integration: Increasing business flexibility through software capabilities designed to simplify integration; integrating people, processes, and information in a way that allows companies to become more flexible and responsive to dynamic markets, customers, and competitors.
- Infrastructure management: Creating an infrastructure that's easier to provision and manage through automation; creating a single, consolidated, logical view of and access to all the available resources in the network.

Infrastructure management is what we focus on here in the context of the IBM Virtualization Engine and EWLM. The optimization of infrastructure management is about enabling access to and simplifying the management of IT resources by creating a consolidated logical view of resources across a processor complex, cluster, or distributed network through automation and virtualization. This is called *resource virtualization*, that is, hiding the physical from the logical implementation.

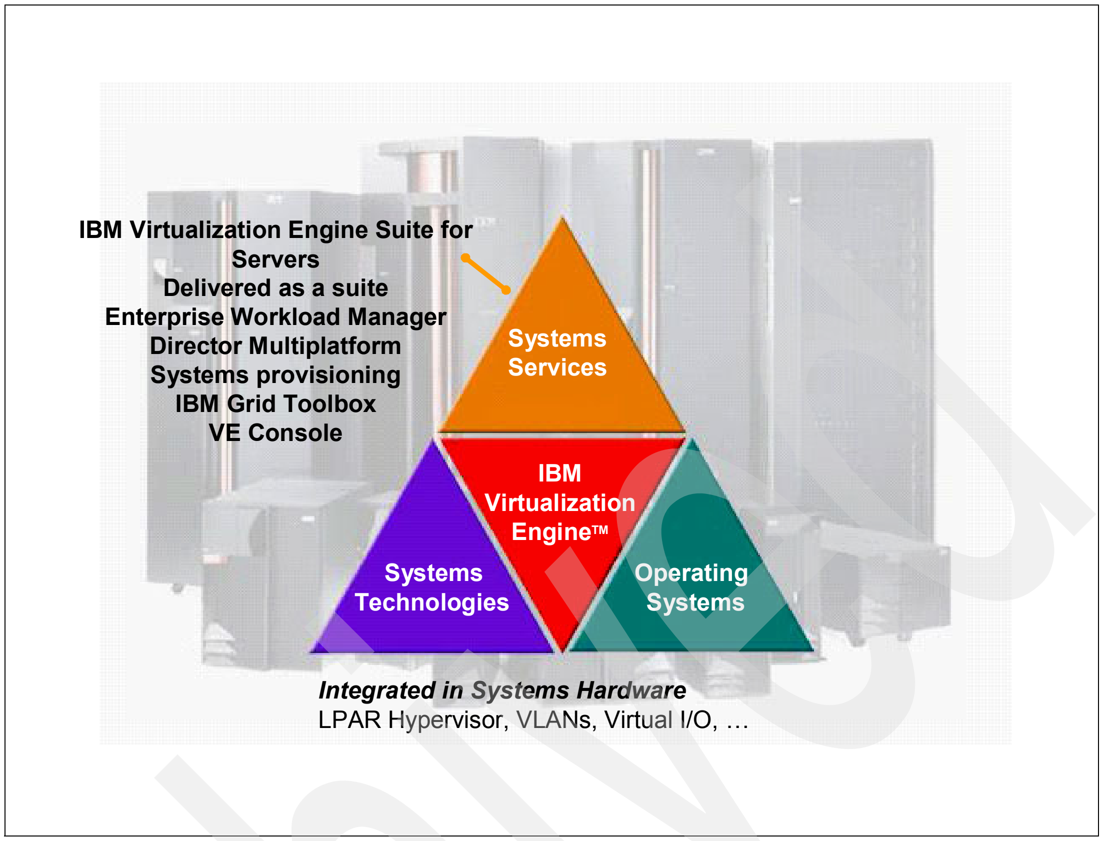

IBM Virtualization Engine Suite for Servers
Delivered as a suite
Enterprise Workload Manager
Director Multiplatform
Systems provisioning
IBM Grid Toolbox
VE Console
Systems Services
IBM Virtualization Engine™
Systems Technologies
Operating Systems
Integrated in Systems Hardware
LPAR Hypervisor, VLANs, Virtual I/O, …

Figure 1-1 IBM Virtualization Engine

The IBM Virtualization Engine encompasses System Technologies, Operating Systems, and System Services.

- System Technologies typically are delivered within the underlying hardware architecture, most often with the processor itself, for example, Logical Partitions (LPAR).
- Operating Systems are often hardware specific.
- System Services span processor architecture and operating systems and typically are delivered to manage a heterogeneous environment, for example EWLM.

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 work flow as shown in Figure 1-2.

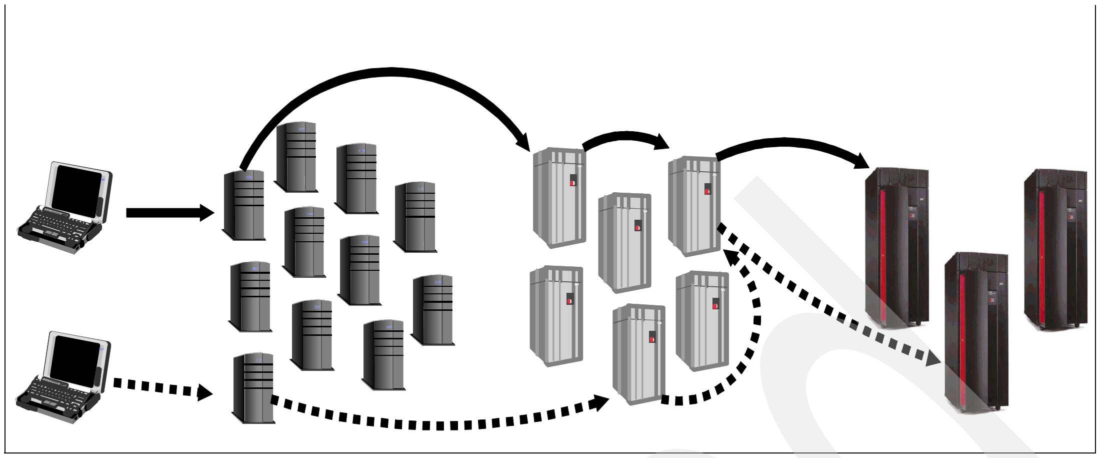


Figure 1-2 e-business environment

The server configuration shown in Figure 1-2, 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.

At the current time, 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 which 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 was the case with the z/OS Workload Manager, the EWLM product introduces one level of management in the first release to get you started. Over time EWLM support will expand to internally automate management of server and application performance. Adaptive heuristic management is a necessary element of being "on demand": avoiding problems, repairing problems, and helping you to raise server utilization to respectable levels.

1.2 EWLM concepts

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 have 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.

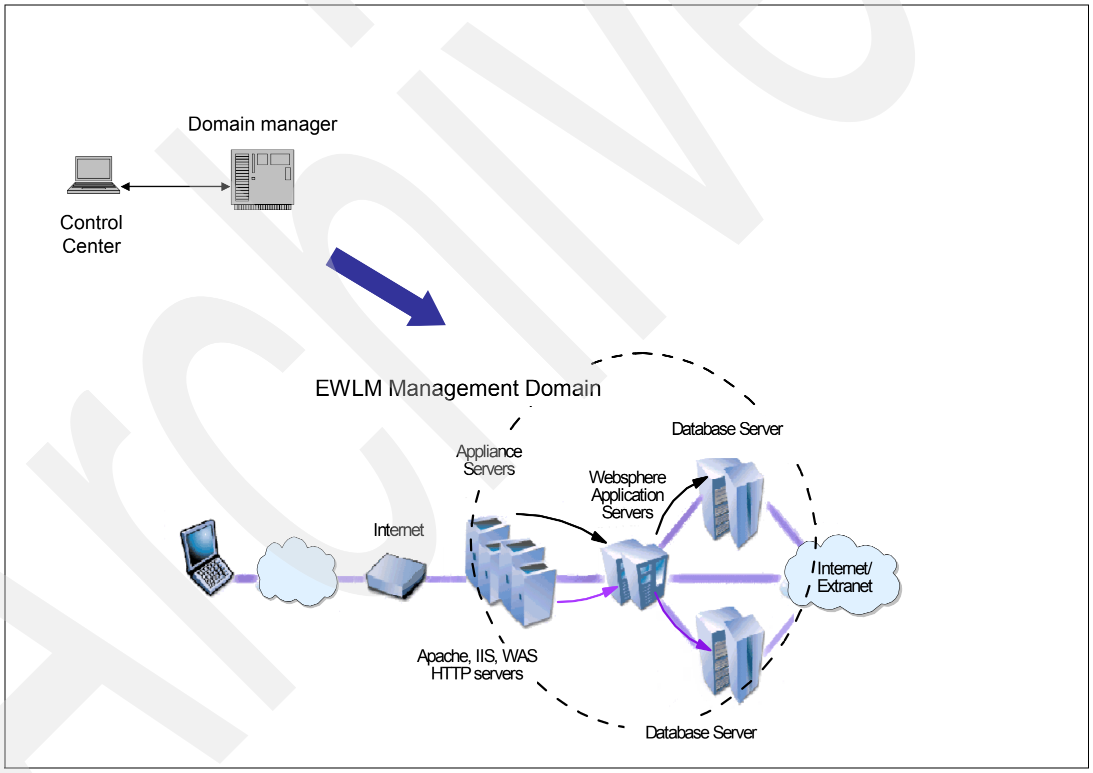


Figure 1-3 EWLM architecture

There is a management focal point for each EWLM management domain, called the EWLM *domain manager*. As shown in Figure 1-3, 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 which 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.

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 size of the server that you install the domain manager on is dependent on the number of managed servers and the Domain Policy complexity. See “Initial performance considerations” on page 219 for more details.

The domain manager has a complete view of what’s going on in the domain. It keeps recent historical data collected from the managed servers. The data can be displayed for as little as one minute or up to an hour interval.

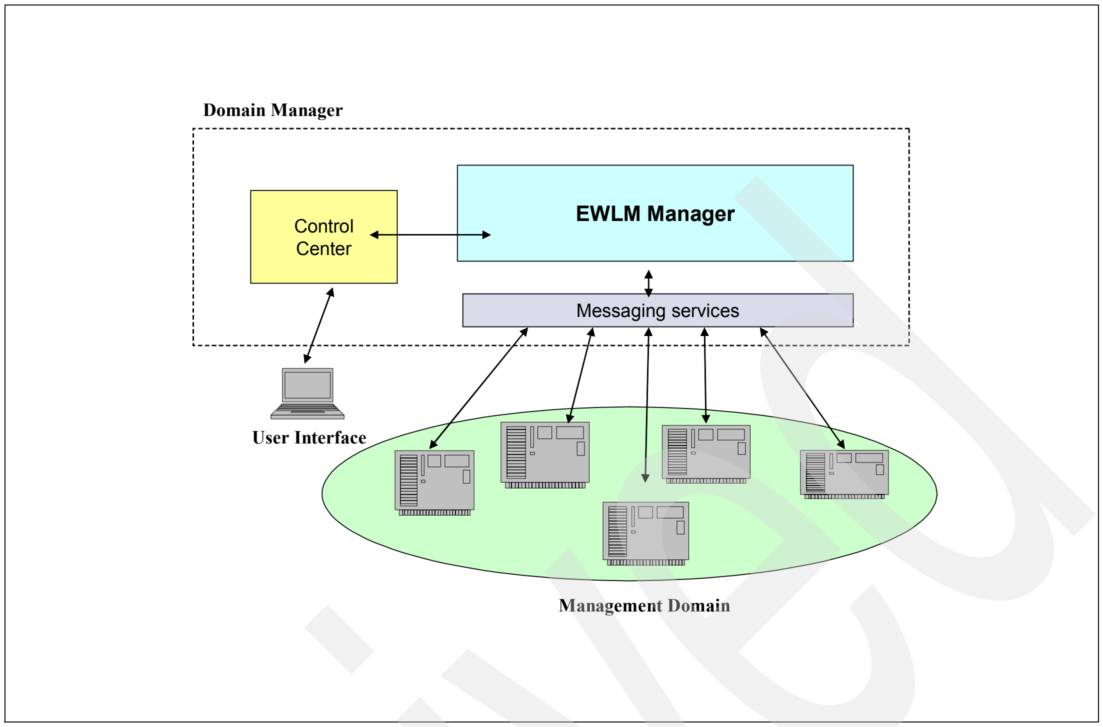


Figure 1-4 Management domain

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 DMZs, this communication can be secured using SSL. Security and firewalls are discussed in detail in Chapter 6, "Using a firewall and securing EWLM" on page 155.

After the first connection from a managed server, the domain manager remembers that this server has been part of the management domain for at least and it will keep it as part of the configuration for at least seven days: so, even if the managed server is not always active, the domain manager will show the managed server though the Control Center in a state of communication error. Only after sever days that the managed server has not been connected to the domain manager, it will be removed from the management domain.

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 how well the business goals are being met and to understand the topology of transactions, applications, and servers in the management domain. This data is the basis for the future autonomic management actions.

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% are successful. Otherwise, the change will be backed out.

The Control Center process in Figure 1-4 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 domain 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 hostname 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 data* for non-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 pushes the policy information to the managed server. This is initiated when an EWLM administrator activates a policy from the EWLM Control Center. The managed server keeps this policy information in an internal database.

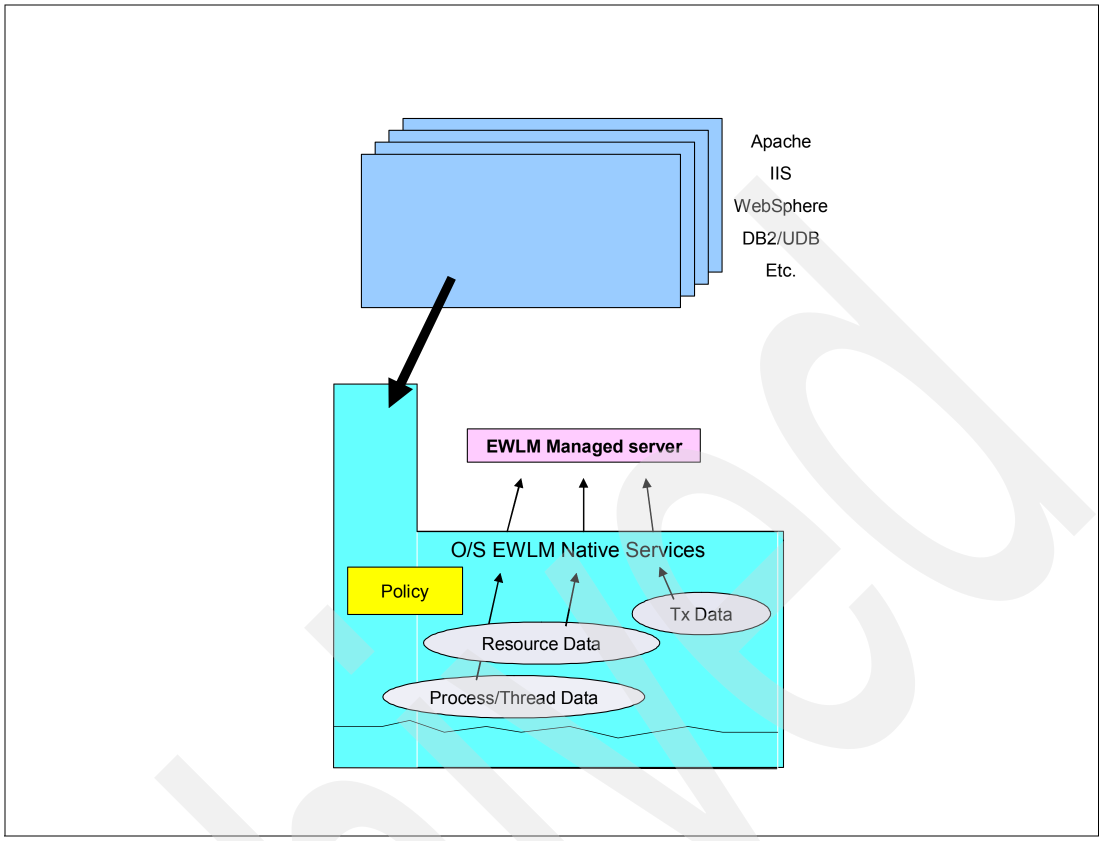


Figure 1-5 EWLM managed server policy and data

Figure 1-5 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 an administrator or operator to log in and uses role-based authentication. User IDs must be created at the operating system level first and then the userid can be added to the Control Center using configuration commands that identify the role of Administrator, Monitor, or Operator as described later in "Setup function of the Control Center" on page 82.

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 “Browser to Control Center” on page 172 for further details.

Note: The Control Center can be launched from the Virtualization Engine (VE) Console. As the Virtualization Engine Suite for Servers becomes more tightly integrated over time, the Control Center will become part of the VE Console dashboard.

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

The most 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 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 Count 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

1.2.4 Domain policy

An EWLM Domain Policy provides a solid context to associate relevance of all other statistics collected. It is a contract which prescribes how EWLM should treat work within a management domain. This includes how the work can be classified, prioritized and monitored. Management of work, meaning taking action based on performance statistics, in this release is only available in conjunction with Cisco Systems Inc. load balancer.

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 1-6, 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 which 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.

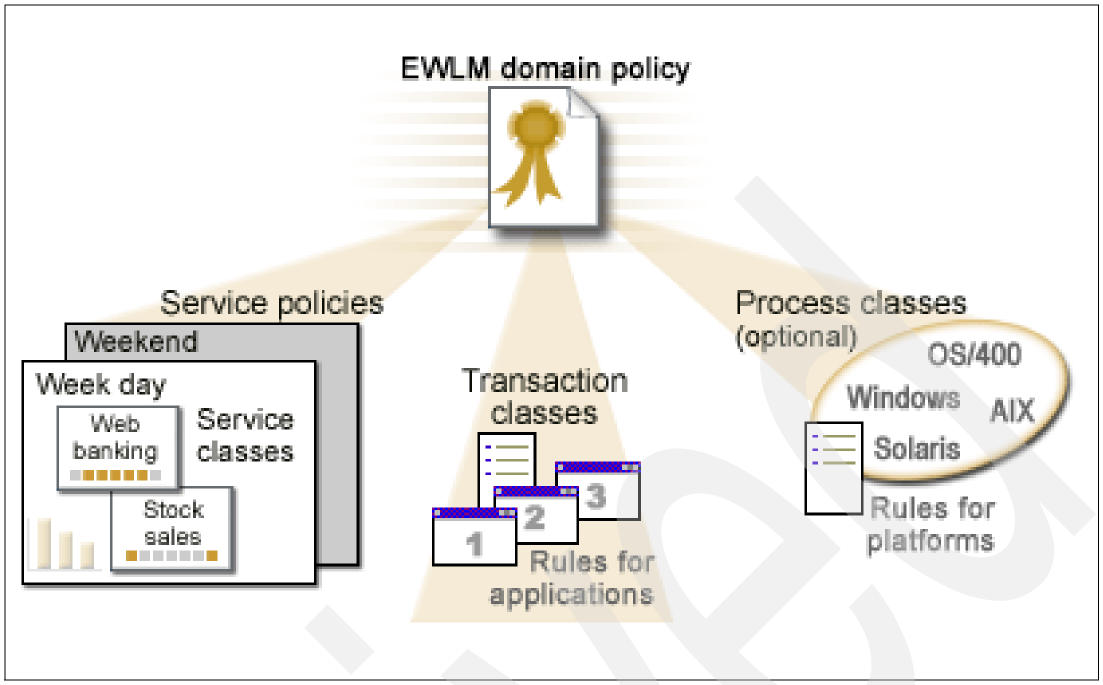


Figure 1-6 EWLM Domain policy elements

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 work flow 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 looked at and 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. There are four types of EWLM performance goals:

- Average response time
- Percentile response time
- Velocity
- Discretionary

These EWLM goal definitions are server and application agnostic. This means that hardware and application upgrades can occur without requiring changes to the goals defined.

To prepare for developing a Domain Policy, the administrator should:

- Create a list of the instrumented applications to be managed in the EWLM Management Domain. See "EWLM supported platforms and applications" on page 16 to identify whether your application is supported in this release.

  Note: Customer-developed applications can be instrumented so that EWLM can manage them. This requires using ARM 4.0, which is described in the next section, "Instrumentation."

- Create a list of all the transaction types to be managed for each business application. Assign a business goal and importance for each type of transaction based on your existing SLA.
- Create a list of operating system platforms to be managed.
- Create a list of various processes on each of the managed platforms that will be processing the transactional or non-transactional work and assign business goals and importance.

  Note: For applications that are not instrumented, EWLM provides process-based management instead of transaction-based management.

- Group the transactions and processes based on the business goals and importance. This will be the basis for the service class.

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 4.0 (ARM) 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. Currently there are three types of applications instrumented with ARM 4.0: WebSphere Application Server, DB2 UDB, and Web servers. ARM 4.0 supported middleware includes the following:

- DB2 UDB V8.2
- WebSphere V5.1.1
- WebSphere Plug-ins delivered with 5.1.1

  For the correct plug-in releases, refer to the Web Servers section of the table at:
  `www.ibm.com/software/webservers/appserv/doc/v51/prereqs/was_v511.htm`

- Independent IHS/Apache Web server ARM plug-ins
  - Configures into IHS 2.0.42
    Placed into the http serve/modules directory (minimum release tested)

  - Available from the Apache Web site
- Independent IIS Web server ARM plug-in
  - Configures into IIS 5.0 and later
  - ARM 4.0 support ships with IBM Virtualization Engine Suite for Servers

Let's look at how ARM, EWLM, and instrumented applications work together to provide end-to-end response times and application/transaction topology using correlation. Figure 1-7 has three EWLM managed servers and a domain manager in a 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()`.

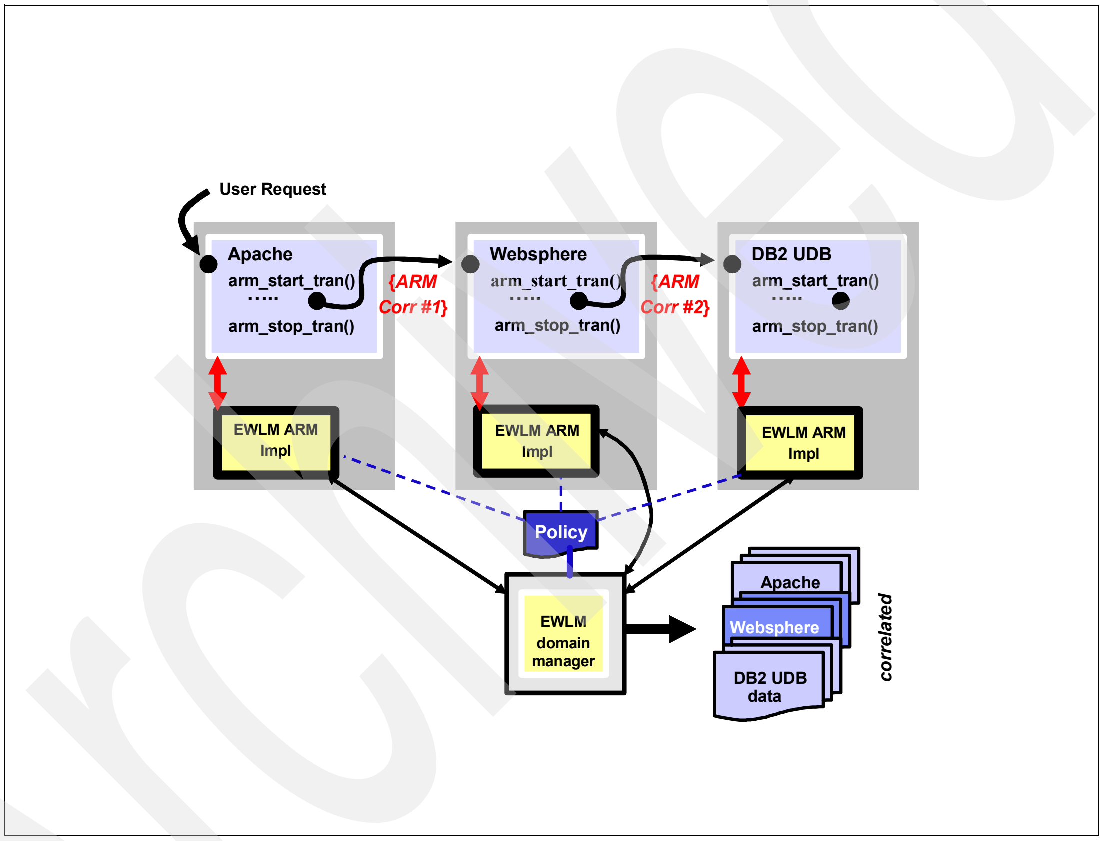


Figure 1-7 Transaction Correlation using EWLM, ARM, and Middleware

The `arm_start_tran()` call at each middleware layer indicates to the EWLM ARM implementation the start of a transaction (or sub-transaction) instance, and therefore that EWLM needs to record the transaction start time for the middleware service. The arm_start_tran() call specifies ARM properties (also called filters) for the transaction instance that the EWLM ARM implementation compares against service policy information to classify the transaction instance to a particular transaction class and, consequently, a particular service class. The service class, in turn, associates the transaction instance with a business goal and relative importance to other work occurring in the EWLM domain.

The `arm_stop_tran()` call at each middleware layer indicates to the EWLM ARM implementation the completion of a transaction (or sub-transaction) instance, and therefore that EWLM needs to compute the response time for the transaction instance. The arm_stop_tran() call also specifies a transaction completion status, such as succeeded or failed, that the EWLM ARM implementation will include in its transaction completion data capture.

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

1. Description of the Apache server as the first middleware hop in this transaction instance processing.
2. 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 EWLM 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:

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

Sometime following its 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:

1. Description of the DB2 server as the third middleware hop in this transaction instance processing.

2. 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 also 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 EWLM supported platforms and applications

EWLM supports the following operating systems for the domain manager and managed server for the current release of EWLM. It is expected that support will be provided for additional operating systems in the near future.

- Supported operating systems for the domain manager:
  - Red Hat Enterprise Linux Advanced Server 2.1 (RHEL AS 2.1)
  - SUSE Linux Enterprise Server 8 (SLES 8)
  - AIX 5L™ Version 5.2 ML03
  - i5/OS™ Version 5 Release 3 with the most current maintenance level

    (Check that SI12723, SI12725, MF32801, MF32814, MF32677 and SI13547 are installed.)
  - Windows 2000 Server Family (the latest service pack is recommended)
  - Windows 2003 Server Family (the latest service pack is recommended)
- Supported operating systems for managed servers:
  - AIX 5L Version 5.2 ML03
  - i5/OS Version 5 Release 3 with the most current maintenance level

    (Check that SI12723, SI12725, MF32801, MF32814, MF32677 and SI13547 are installed.)
  - Windows 2000 Server Family (the latest service pack is recommended)
  - Windows 2003 Server Family (the latest service pack is recommended)
  - Solaris 8 and 9 supporting 32 and 64 bit user programs

- Supported browser for user interface:
  - Internet Explorer V6 SP1

EWLM is first installed at the domain manager. The domain manager installation also creates install images on the domain manager operating system instance for the target managed server operating systems that you select. These managed server install images must be copied to the target operating systems and an install executed there for the managed server.

Table 1-1 identifies the space requirements for installation. Column 4 is the amount of space required at the domain manager for the target operating system's managed server install image. Column 5 is the space required at the managed server target operating system. The values reported in the table are a guideline for the required space based on the maintenance level of code we used in our installation, but you might use them for initial planning.

Table 1-1 EWLM space requirements for installation

Operating system	**Installation directory**	**Domain manager space and VE Install**	**Managed server installation space at the domain manager**	**Managed server space**	**Domain manager working directory***
AIX	/opt/IBM/VE	1.3GB	41MB	60MB	100MB
Linux	/opt/IBM/VE	1.3GB	n/a	n/a	110MB
Solaris	/opt/IBM/VE	n/a	39MB	191MB	n/a
i5/OS	/QIBM/ProdData/VE	20MB (only DM)	22MB	20MB	100MB
Windows	C:\Program Files\IBM\VE	1.6GB	44MB	53MB	100MB

* This size is required for the installation. More space is required when data is collected. Refer to "Domain manager resource description" on page 220 for a more accurate evaluation of the space required by the internal database.

2

Installing and configuring EWLM

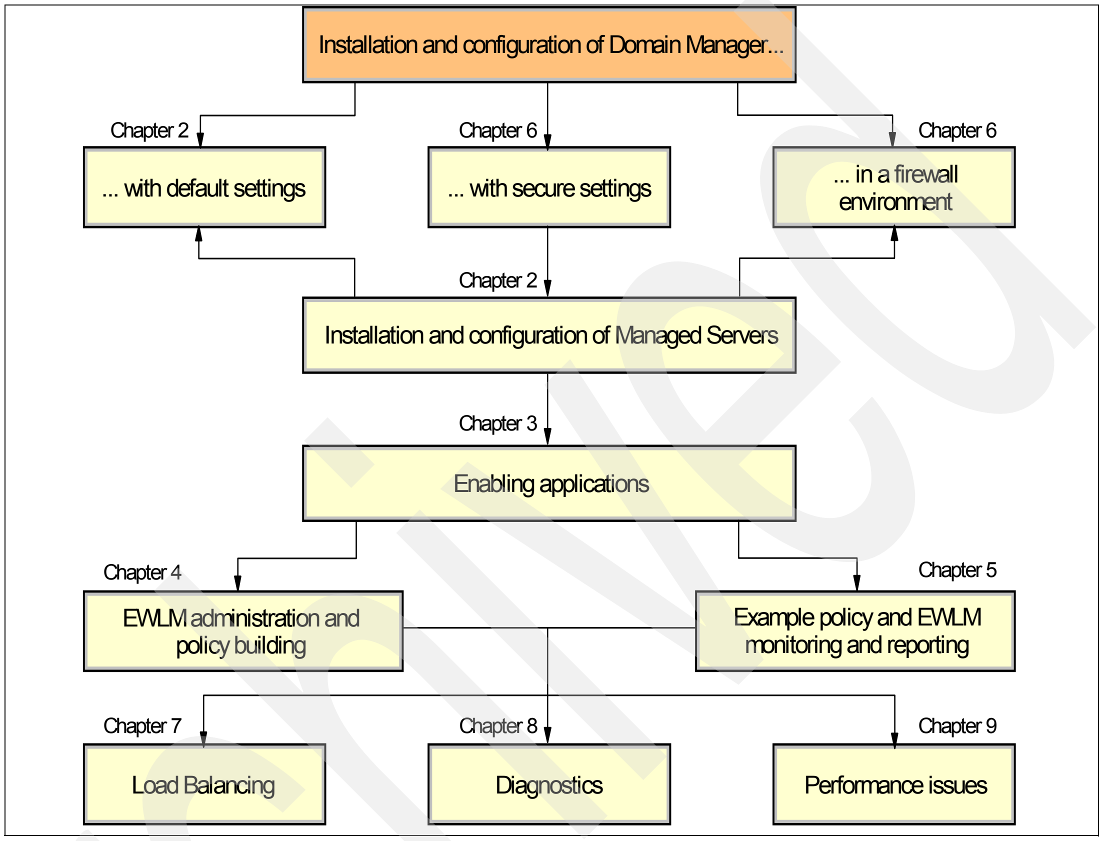
In this chapter we describe how to:

- Install EWLM code on the domain manager and managed servers
- Configure the domain manager and managed servers for
  - Operating systems in our ITSO environment
  - Other supported operating systems

We introduce you to our ITSO test environment (platforms, operating systems, and applications) in Section 2.2 on page 21. All of the examples and descriptions used throughout the book are based on this environment.

2.1 Overview

Figure 2-1 illustrates how the installation and configuration topics are organized in the rest of this book. This flow chart also indicates the correct order of steps when setting up EWLM in your environment.



Installation and configuration of Domain Manager...
Chapter 2
... with default settings
Chapter 6
... with secure settings
Chapter 6
... in a firewall environment
Chapter 2
Installation and configuration of Managed Servers
Chapter 3
Enabling applications
Chapter 4
EWLM administration and policy building
Chapter 5
Example policy and EWLM monitoring and reporting
Chapter 7
Load Balancing
Chapter 8
Diagnostics
Chapter 9
Performance issues

Figure 2-1 Installation and customization tasks

The first step in setting up Enterprise Workload Manager is to install the EWLM code. We describe this procedure in 2.3, “Virtualization Engine suite” . We introduce the Virtualization Engine suite, its connection to EWLM and guide you through the installation steps of EWLM. We then discuss the configuration of the domain manager and the managed servers. In this chapter this is done with default settings only. If you are interested in security or firewall settings, refer to Chapter 6, “Using a firewall and securing EWLM” on page 155.

The next step is to instrument applications to make them Application Response Measurement (ARM) enabled. The instrumentation of middleware is necessary to make EWLM aware of the application topology and the workflow in your environment. This is described in detail in Chapter 3, “Enabling middleware for EWLM” on page 57. In Chapter 4, “Administering EWLM” on page 77 we present a general overview of EWLM operation and administration. We provide information on terminology and include a discussion on sample policies.

In Chapter 5, “The ITSO EWLM scenario” on page 119 we guide you through an example of the planning and preparation steps needed to integrate EWLM in your environment. We analyze our ITSO environment, our applications and workflow. We explain in detail the implementation steps we used for building our domain policy and discuss monitoring and reporting functionality. The final three chapters provide further information on diagnostics, performance considerations, and some load balancing issues.

2.2 The ITSO test environment

EWLM supports dynamic business application workflow technology for multiple tier distributed heterogeneous operating system environments. The environment we used in the lab at ITSO is illustrated in Figure 2-2. It consists of a Web Server tier, a two server WebSphere cluster as the application tier, and a database tier. These four servers are configured to participate in one EWLM management domain as managed servers.

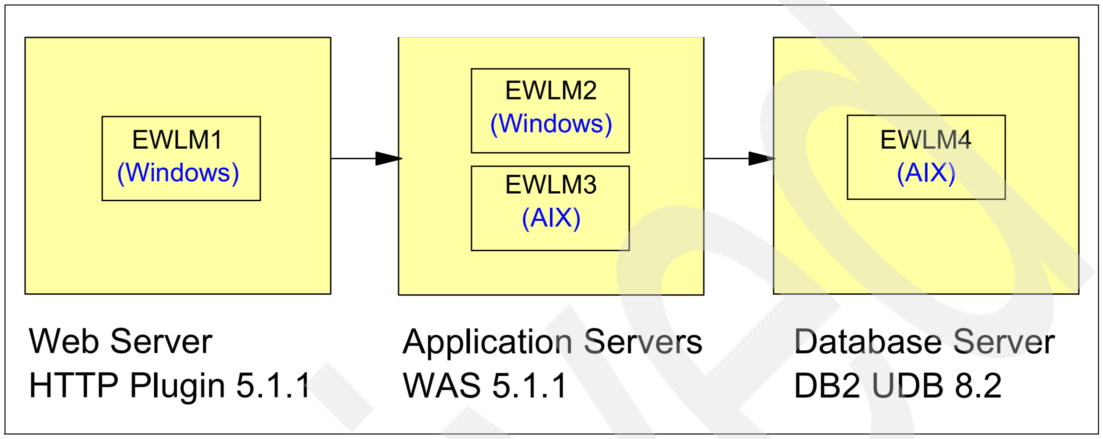


Figure 2-2 Three tier architecture

Our entry application is the HTTP plug-in running on EWLM1. The WebSphere cluster consists of the two servers EWLM2 and EWLM3 running WebSphere Application Server 5.1.1. Our third tier is DB2 Universal Database (UDB) Version 8.2 running on the AIX server EWLM4. The two servers EWLM1 and EWLM2 run on Windows version 2003. The AIX servers EWLM3 and EWLM4 are deskside models of IBM @server pSeries 630 running on maintenance level three of AIX 5L Version 5.2. The domain manager EWLMDM1 in our environment is running on Red Hat Enterprise Linux AS 2.1. This is where we run the EWLM Control Center. Our configuration is shown in Figure 2-3 on page 22.

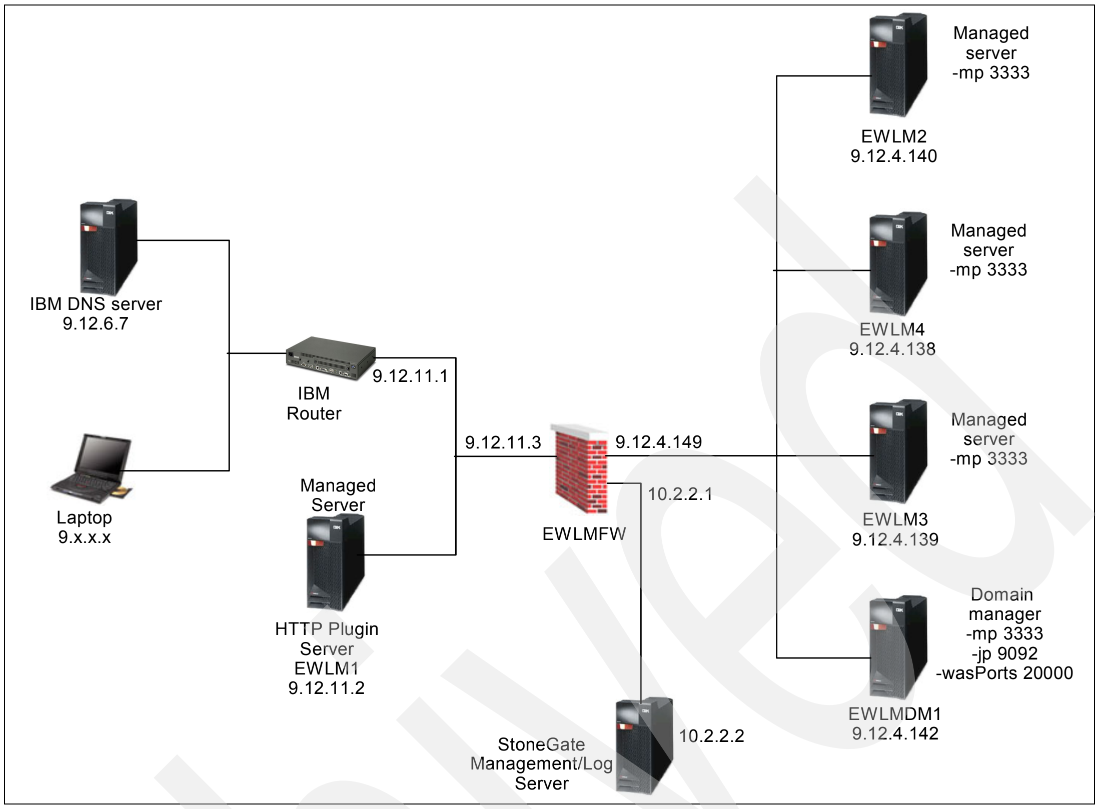


Figure 2-3 ITSO environment

As you can see, we have a firewall integrated in our ITSO environment. In this chapter, however, we describe the setup of our domain manager and the managed servers using default settings only. We integrated security and firewall settings at a later time, after we had our configuration set up. The changes we made to the environment for security purposes are described in Chapter 6, “Using a firewall and securing EWLM” on page 155.

2.3 Virtualization Engine suite

For this release, Enterprise Workload Manager is an optionally installable system service of the IBM Virtualization Engine. You can install EWLM domain manager, EWLM managed servers for IBM and non-IBM platforms, Single-System EWLM, or some combination of these. The available options are explained in this section and depend on the package you have purchased and on the operating systems in your environment. Single-System EWLM is only available on AIX and i5/OS. The Single-System EWLM option was not explored as part of the redbook project. The EWLM Control Center is installed as part of the domain manager installation. The managed server installation includes a firewall broker installation image.

We describe the EWLM installation in this section. After the installation is complete, you must run a set of configuration scripts to get the domain manager, EWLM Control Center, and managed servers up and running. Our discussion of the configuration process is divided into two parts:

- First, we describe the installation and configuration of the domain manager and managed servers in our ITSO environment. This includes:
  - Domain manager on Linux
  - Managed servers on AIX and Windows

- Second, we concentrate on the differences when installing and configuring EWLM code on the domain manager and managed servers on all other supported operating systems. This is explained in Section 2.5, “Installing the domain manager on other operating systems” and 2.6, “Configuring managed servers on other operating systems” and includes:
  - Domain manager on Windows, AIX and i5/OS
  - Managed servers on i5/OS and Solaris

Table 2-1 provides an overview of user roles required on each operating system in order to install and configure EWLM, operate as WebSphere Application Server administrator, and for mapping users to the EWLM Control Center. Details for each operating system are provided in the appropriate sections in this chapter.

Table 2-1 Operating system users

OS name	User role	User name	User requirements
Domain manager requirements			
Linux	EWLM installation and configuration	root	Root authority
	WAS administration	root	Root authority
	EWLM Control Center roles	ewlmadm ewlmops ewlmmon	Administrator Operator Monitor
AIX	EWLM installation and configuration	root	Root authority
	WAS administration	root	Root authority
	EWLM Control Center roles	ewlmadm ewlmops ewlmmon	Administrator Operator Monitor
Windows	EWLM installation and configuration	Administrator ibmewlm	- Member of Administrators group - Act as part of OS - Log on as a service
	WAS administration	Administrator ibmewlm	- Member of Administrators group - Act as part of OS - Log on as a service
	EWLM Control Center roles	ewlmadm ewlmops ewlmmon	Administrator Operator Monitor
i5/OS	EWLM installation and configuration	qsecofr ibmewlm	Requires *ALLOBJ, *SECADM, *JOBCTL, *IOSYSCFG special authority
	WAS administration	Administrator ibmewlm	No special requirements
	EWLM Control Center roles	ewlmadm ewlmops ewlmmon	Administrator Operator Monitor

OS name	User role	User name	User requirements
Managed server requirements			
AIX	Managed server installation and configuration	root testuser	- root authority or with capabilities=CAP_EWLM_AGENT,CAP_PROPAGATE - root authority
Windows	EWLM installation and configuration	testuser	Requires user to belong to Administrators group
i5/OS	EWLM installation and configuration	testuser	Requires managed server to run under QWLM user profile which has the necessary access rights
Solaris	EWLM installation and configuration	root or testuser	Root authority or with effective user identification (EUID) of zero

Installation of EWLM code

The installation process for EWLM is part of the Virtualization Engine installation wizard. There are two phases in this installation process. The first one involves transferring the media, during which the installation wizard copies images of the CDs to whichever destination you select. The second part is the installation of the Virtualization Engine (VE) code and the VE services. The installation wizard allows you to select which VE system services to install and performs the actual installation.

In this redbook we are interested in the installation of IBM Virtualization Engine system services for EWLM only. This installs the domain manager and EWLM Control Center as well as the required runtime environment. In addition, the managed server installation images are copied to our server. These must be distributed and installed to the appropriate platforms as a separate step.

The EWLM installation requires you to go through the following steps:

- Ensure that your system meets the minimum hardware and software requirements.
- Complete a planning checklist for your platform. The checklist is included in the installation process as part of the installation assistant.

To start the installation process you need the Virtualization Engine installation set of CDs for the operating system of your domain manager - called the *stack copy*. For our domain manager running on Linux we inserted the first CD, labeled “VE Console & Runtime for Linux on xSeries® (1 of 3).” The VE installation wizard automatically appears on the screen. If if does not start automatically, mount the CD device and launch autorun as follows:

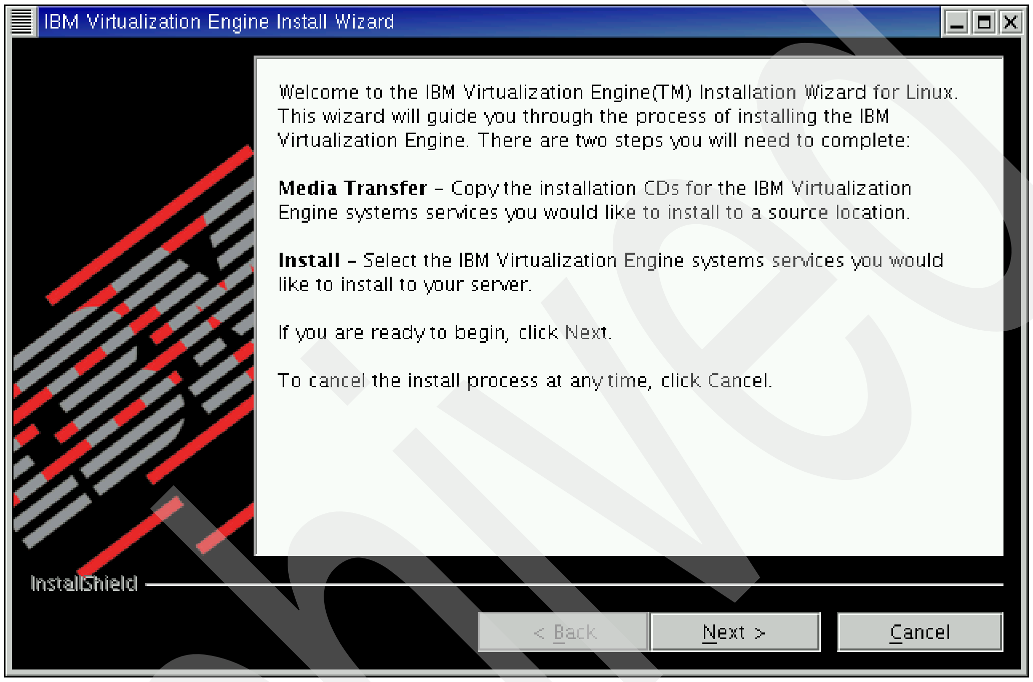
```
mount CDROM file system
# mount /dev/cdrom
launch installation program using mount point /mnt/cdrom
# /mnt/cdrom/autorun
```

The VE Installation Wizard requires the following CDs to be copied to a local directory during this first phase:

- VE Console & Runtime for Linux on xSeries (1 of 3)
- VE Console & Runtime for Linux on xSeries (2 of 3)
- VE Console & Runtime for Linux on xSeries (3 of 3)
- EWLM Domain Manager for Linux on xSeries
- EWLM Managed Servers for AIX 5L and i5/OS - Linux on xSeries Stack copy
- EWLM Managed Servers for Windows and Solaris - Linux on xSeries Stack copy

A similar set is available for each platform that supports a domain manager.

Before starting the installation process, make sure that the target directory does not exist, otherwise the wizard will not be able to operate. If you are using a GUI, you can click the autorun folder to launch the installation program. If your domain manager runs on other operating systems, we provide information on how to launch the installation wizard in 2.5, "Installing the domain manager on other operating systems" . The welcome screen of the installation wizard is shown in Figure 2-4.



Welcome to the IBM Virtualization Engine(TM) Installation Wizard for Linux. This wizard will guide you through the process of installing the IBM Virtualization Engine. There are two steps you will need to complete:

Media Transfer - Copy the installation CDs for the IBM Virtualization Engine systems services you would like to install to a source location.

Install - Select the IBM Virtualization Engine systems services you would like to install to your server.

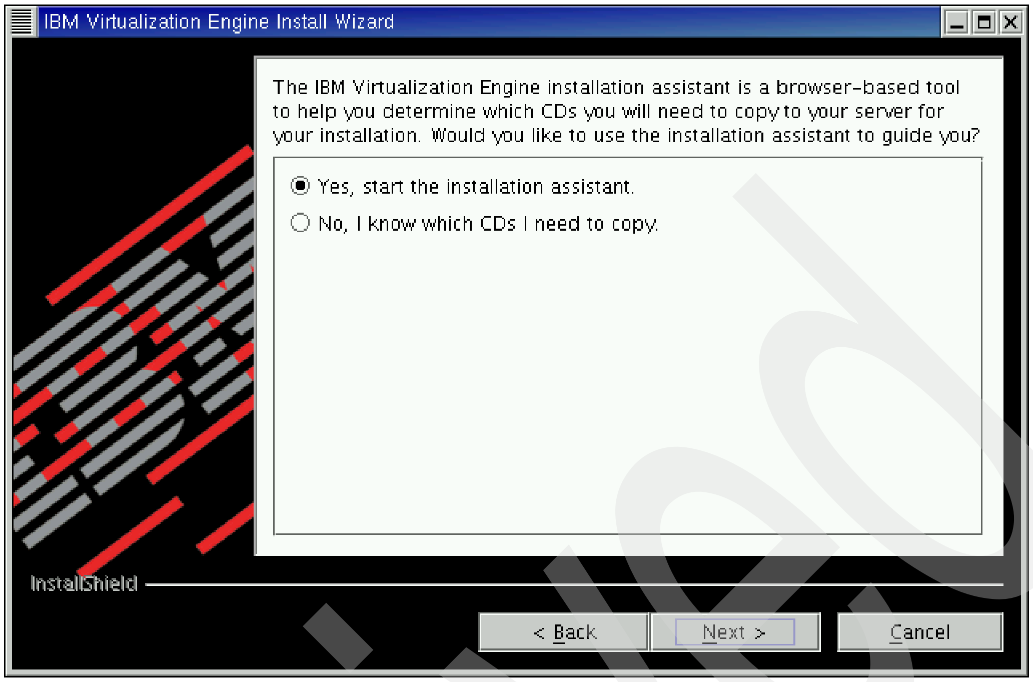
If you are ready to begin, click Next.

To cancel the install process at any time, click Cancel.

Figure 2-4 InstallShield Wizard for IBM Enterprise Workload Manager

We begin with the media transfer; we need to specify a location to which we want to copy the Virtualization Engine CD images. We accepted the default location /opt/IBM/VE/Install and clicked **Next**.

You can start the Installation Assistant concurrently in a different window by selecting "YES" on the installation wizard before starting the copying of the CDs, as you can see in Figure 2-5 on page 26.



The IBM Virtualization Engine installation assistant is a browser-based tool to help you determine which CDs you will need to copy to your server for your installation. Would you like to use the installation assistant to guide you?

[✓ Yes, start the installation assistant.]

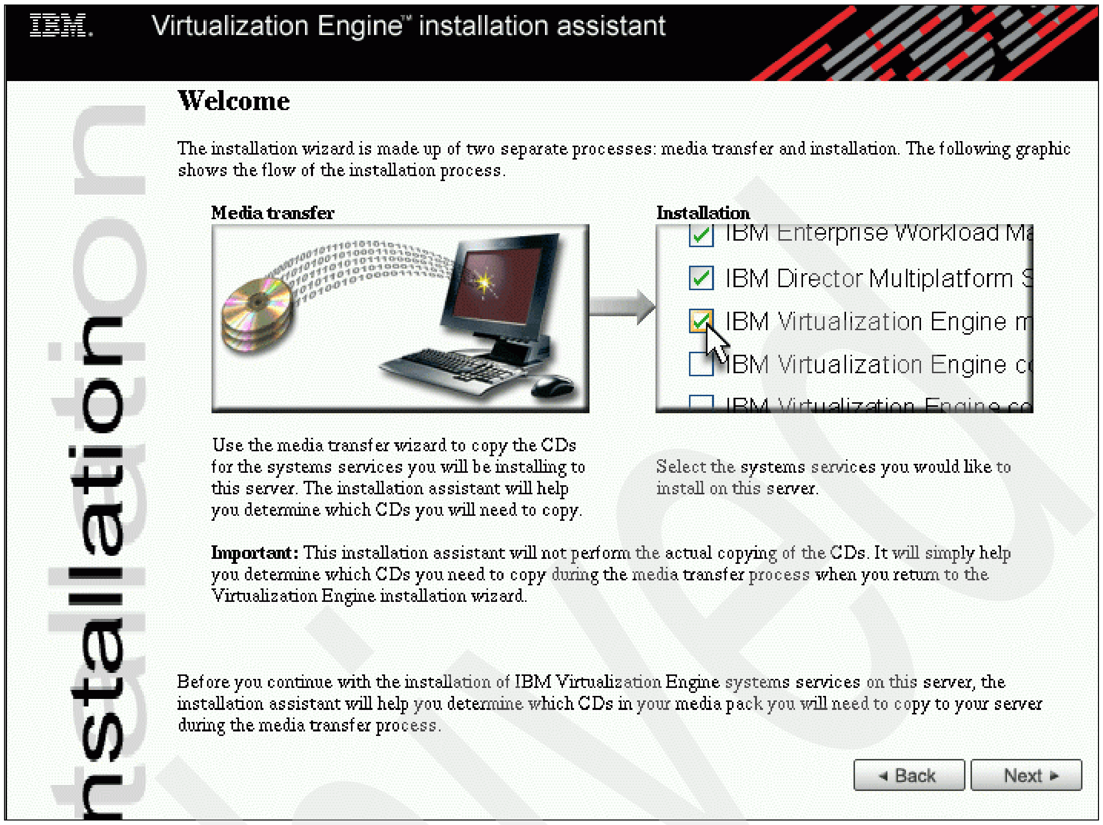
No, I know which CDs I need to copy.

Figure 2-5 Installation Assistant start up

The Installation Assistant helps determine which CDs you need to copy from your media package. It begins by indicating the requirements for the Installation Assistant itself to run.

If you are familiar with the VE code installation, you can select No in the panel shown in Figure 2-5 to skip the Installation Assistant. However, we assume that this is your first installation of EWLM and advise you to follow the installation assistant.

Going through the installation Assistant ensures that your system meets all necessary requirements, and following the planning steps will help you organize the installation and configuration of the Virtualization Engine and selected components like EWLM.


Virtualization Engine™ installation assistant

Welcome

The installation wizard is made up of two separate processes: media transfer and installation. The following graphic shows the flow of the installation process.

Media transfer

Installation

- ✓ IBM Enterprise Workload Ma
- ✓ IBM Director Multiplatform S
- ✓ IBM Virtualization Engine m
- IBM Virtualization Engine c
- IBM Virtualization Engine co

Use the media transfer wizard to copy the CDs for the systems services you will be installing to this server. The installation assistant will help you determine which CDs you will need to copy.

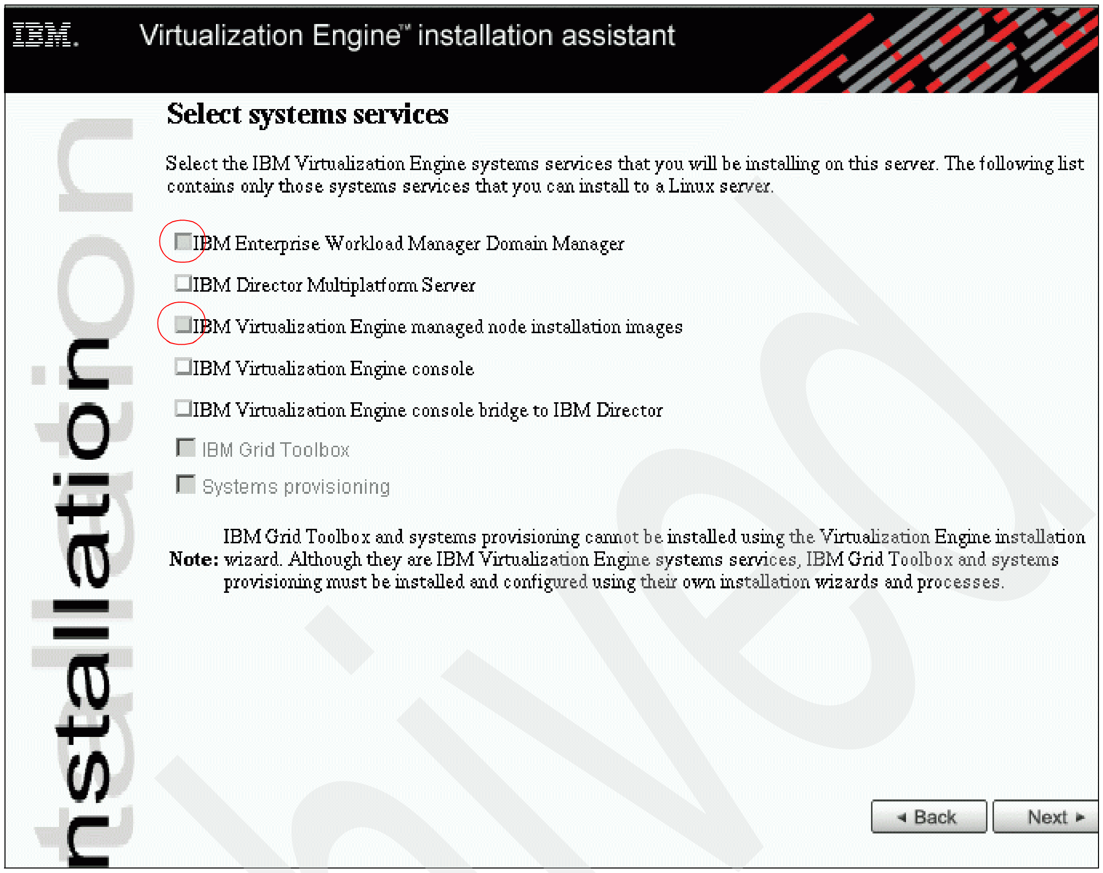
Select the systems services you would like to install on this server.

Important: This installation assistant will not perform the actual copying of the CDs. It will simply help you determine which CDs you need to copy during the media transfer process when you return to the Virtualization Engine installation wizard.

Before you continue with the installation of IBM Virtualization Engine systems services on this server, the installation assistant will help you determine which CDs in your media pack you will need to copy to your server during the media transfer process.

Figure 2-6 IBM Virtualization Engine installation assistant

When you click **Next**, you are presented with a series of informational screens and then you are prompted to select which Virtualization Engine services are to be installed. This is shown in Figure 2-7. The Installation Assistant might look a little different depending on the operating system of your domain manager and contains only the systems services which you can install on each platform.



Select systems services

Select the IBM Virtualization Engine systems services that you will be installing on this server. The following list contains only those systems services that you can install to a Linux server.

- ☐ IBM Enterprise Workload Manager Domain Manager
- ☐ IBM Director Multiplatform Server
- ☐ IBM Virtualization Engine managed node installation images
- ☐ IBM Virtualization Engine console
- ☐ IBM Virtualization Engine console bridge to IBM Director
- ☐ IBM Grid Toolbox
- ☐ Systems provisioning

Note: IBM Grid Toolbox and systems provisioning cannot be installed using the Virtualization Engine installation wizard. Although they are IBM Virtualization Engine systems services, IBM Grid Toolbox and systems provisioning must be installed and configured using their own installation wizards and processes.

Figure 2-7 Selection of IBM Virtualization Engine systems services

We are interested in installing IBM Enterprise Workload Manager and chose to install the following components from the list:

- IBM Enterprise Workload Manager domain manager
- IBM Virtualization Engine managed node installation images

For information on other services on the list, refer to the IBM eServer information center at the following URL:

`http://publib.boulder.ibm.com/eserver/`

The installation assistant uses your input to calculate the required space and provides a list of CDs which you need to copy.

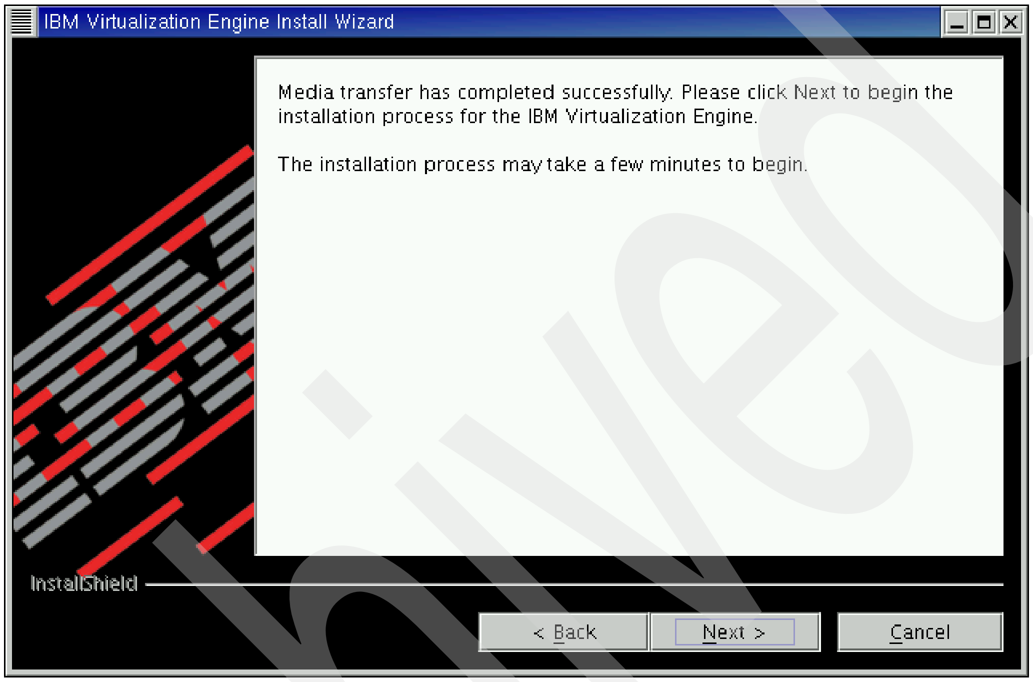
Note: Remember to print the list of CDs which you need to copy for the installation as provided by the installation assistant.

The wizard will copy the CDs you insert, even if they do not match the list provided by the installation assistant. However, if you have not inserted the required CDs the actual installation process will fail.

If you need further information on the Virtualization Engine and its components, the installation assistant can provide it. This is a good source when you are unsure what services or components to select.

When you have completed the installation assistant, return to the Virtualization Engine installation wizard and begin to copy the CD images. After each CD is copied, you are asked if you want to copy another CD. When you have finished copying the required CD images to your domain manager you can start the installation process.

The wizard displays a message saying it assumes that you have already copied the media and the installation process begins (Figure 2-8).

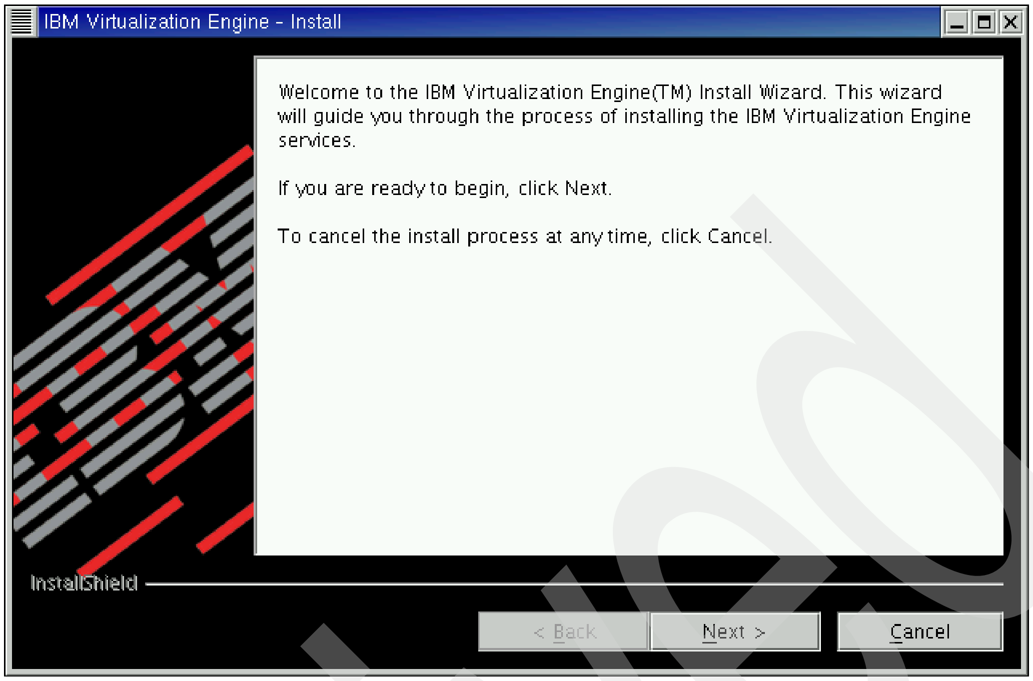


Media transfer has completed successfully. Please click Next to begin the installation process for the IBM Virtualization Engine.

The installation process may take a few minutes to begin.

Figure 2-8 Installation part of Virtualization Engine installation wizard

This ends the first phase of the installation process. Next, the wizard will guide you through the installation of the VE services (beginning with the panel shown in Figure 2-9). If you need to restart the Virtualization Engine Install Wizard for any reason (for example, if you hit the “Cancel” function or if the Wizard has a problem), you can restart the from this point—without repeating the CD copying phase—by executing **`./installVELinux.bin`** in /opt/IBM/VE/Install.



Welcome to the IBM Virtualization Engine(TM) Install Wizard. This wizard will guide you through the process of installing the IBM Virtualization Engine services.

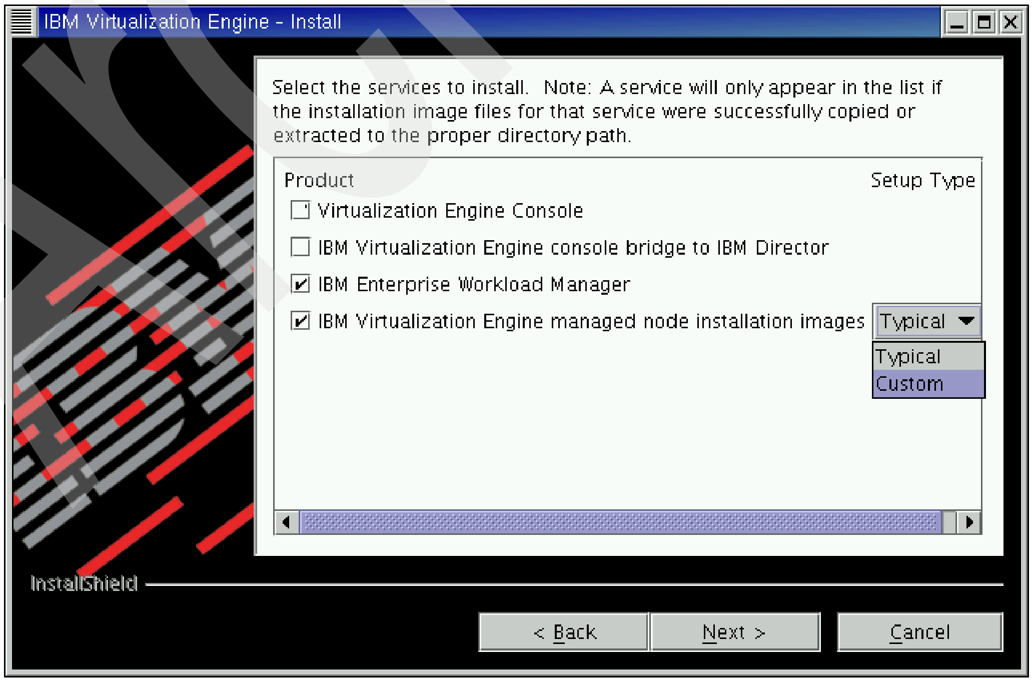
If you are ready to begin, click Next.

To cancel the install process at any time, click Cancel.

Figure 2-9 Installation of the VE services

After accepting the license agreement, which can be viewed in several different languages, the first task is to specify a source location for installing the EWLM code. For our domain manager running on Linux we use the default directory /opt/IBM/VE.

The next installation wizard menu is shown in Figure 2-10. It lists the available components of the Virtualization Engine. For the installation of EWLM we are interested in the last two checked boxes only (IBM Enterprise Workload Manager and IBM Virtualization Engine managed node installation images). If you have purchased other components of the VE package you can select to install these products as well.



Select the services to install. Note: A service will only appear in the list if the installation image files for that service were successfully copied or extracted to the proper directory path.

Product	Setup Type
☐ Virtualization Engine Console	
☐ IBM Virtualization Engine console bridge to IBM Director	
☑ IBM Enterprise Workload Manager	
☑ IBM Virtualization Engine managed node installation images	Typical

Figure 2-10 Selection of services to install

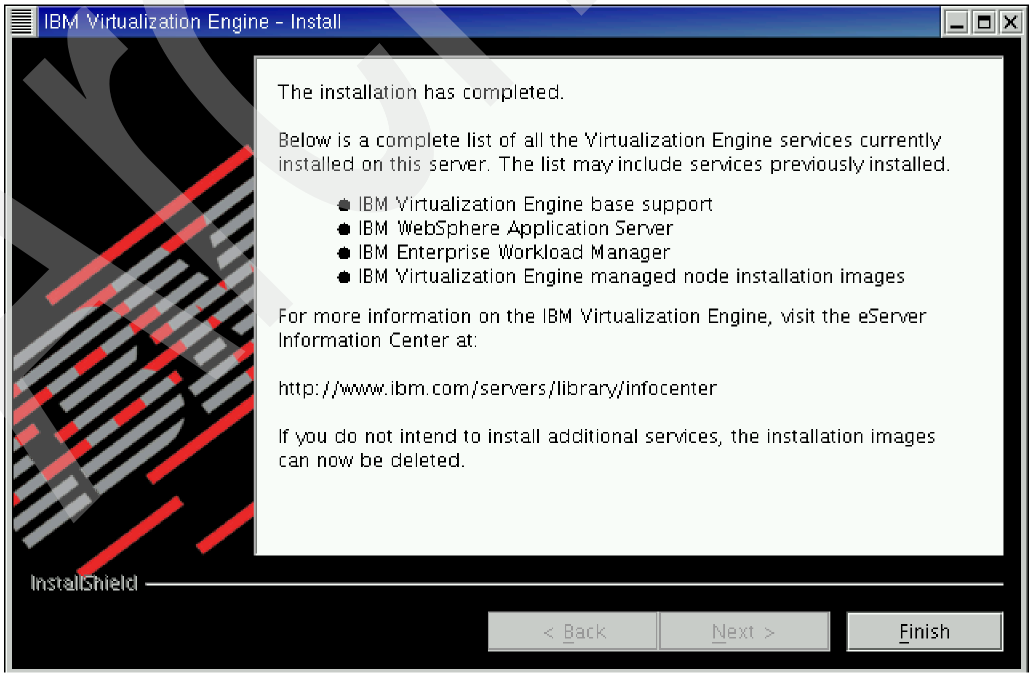
At this point you need to decide whether you want a typical or a custom installation. If you select the “IBM Enterprise Workload Manager” option only, the wizard installs the domain manager only. If you select “IBM Enterprise Workload Manager” and “IBM Virtualization Engine managed node installation images,” the wizard installs both the domain manager and managed node. If you select “Typical” in the pull-down menu, the install copies all of the managed server images which you have purchased; if you select “Custom,” the install allows you to deselect specific managed server images. If this is the first time you are installing EWLM, we advise you to follow the typical installation, as we did for our ITSO environment.

After accepting another license agreement for each of the selected services, we are presented with an installation summary. This provides information on the installation directory and lists the features which will be installed. For our domain manager we have the minimum required package for EWLM, which includes the IBM Virtualization Engine base support, IBM WebSphere Application Server and IBM Enterprise Workload Manager. Make sure you have enough space in your installation directory, otherwise you will have to restart the wizard after increasing the directory size. A small amount of space is also required in /tmp.

When you install Virtualization Engine systems services, like EWLM, that require WebSphere Application Server (WAS), be aware that the VE installation wizard will install WebSphere Application Server 5.1 in the VE installation root directory. The default location on Linux is /opt/IBM/VE. If you already have installed WebSphere Application Server 5.0 or 5.1 you need to understand coexistence considerations. For information on these issues, refer to the IBM information center at:

```
https://publib.boulder.ibm.com/eserver/cur/v1r2m0f/en_US/index.htm?info/veicinfo/eicarkicko
ff.htm
```

Next, we need to specify a node name and an IP address or host name. The wizard detects those names automatically but allows you to change them. The node name for our domain manager is specified as `ewlmdm1` with hostname `ewlmdm1.itso.ibm.com`. The installation process begins and takes only a couple of minutes on our domain manager running on Linux. After a successful installation of EWLM, the last screen you see is shown in Figure 2-11 on page 31. It includes a summary of the installed features.



The installation has completed.

Below is a complete list of all the Virtualization Engine services currently installed on this server. The list may include services previously installed.

- IBM Virtualization Engine base support
- IBM WebSphere Application Server
- IBM Enterprise Workload Manager
- IBM Virtualization Engine managed node installation images

For more information on the IBM Virtualization Engine, visit the eServer Information Center at:

http://www.ibm.com/servers/library/infocenter

If you do not intend to install additional services, the installation images can now be deleted.

Figure 2-11 Successful installation summary

The wizard suggests that if you do not need the installation wizard any longer, you might remove the Install directory to save space.

You are now ready to configure the domain manager and associated WebSphere Application Server instance.

2.4 EWLM configuration in our ITSO environment

After you finish installing the domain manager and managed servers you need to perform additional configuration tasks. These include configuring the domain manager, the managed servers and firewall brokers, starting WebSphere Application Server, and starting the EWLM elements.

In this section, we first go through the configuration steps for our domain manager running on Linux. After we can log on successfully to the EWLM Control Center, we transfer the EWLM code to our managed servers running on AIX and Windows and configure each of these. This is also the order you should follow when setting up your Enterprise Workload Manager environment.

In Section 2.5, “Installing the domain manager on other operating systems” we guide you through the differences if your domain manager is running on AIX, Windows or i5/OS. In Section 2.6, “Configuring managed servers on other operating systems” we provide information on managed server issues on other operating systems including i5/OS and Solaris.

We need to complete the following steps to configure EWLM in our environment:

- Configure the domain manager.
- Create users for administrating EWLM.
- Start the WebSphere Application Server instance on the domain manager.
- Configure the managed server.
- Enable ARM services on the operating system.

2.4.1 Configuring domain manager on Linux

Our domain manager runs on Red Hat Enterprise Linux AS 2.1. As part of the installation process of the Virtualization Engine install, directory /opt/IBM/VE is created. Its size is approximately 1.3 GB. The EWLM installation creates a subdirectory EWLM; Example 2-1 shows the structure of the subdirectory /opt/IBM/VE/EWLM/bin where EWLM configuration scripts and associated log files are located. These log files are named after their creation scripts and provide useful information for debugging events. We go through a detailed explanation of many of these scripts when we explain the configuration steps for our domain manager and managed servers.

Example 2-1 Output of EWLM configuration commands on domain manager

```
# cd /opt/IBM/VE/EWLM/bin
# ls
addusers.jacl      config.jar         displayDM.sh           startDM.sh
admin.jacl         configWizardDM.sh  displayusers.jacl      startWAS.log
application.jacl   createDM.log       displayWASPorts.jacl   startWAS.sh
build.xml          createDM.sh        removeusers.jacl       stopWAS.sh
changeCCapp.sh     db2j.log           security.jacl          trace.jacl
changeCC.log       dbconfig.jacl      setupapps.sql          variable.jacl
changeCC.sh        definition.sql     setupproperties.sql    variables.sh
```

```
changeDM.sh          deleteDM.sh        sslSecurityUpdate.jacl
changeWASPorts.jacl  displayCC.sh       startDM.log
```

All of these configuration scripts require the EWLM working directory as a parameter. The working directory contains information related to EWLM components. For domain manager, it also contains information associated with the EWLM Control Center and the WebSphere Application Server instance. If you run more than one instance of a domain manager on your system you must specify a different working directory for each instance.

If you want to run the WebSphere Application Server instance associated with the domain manager and the domain manager itself as `root` user you can skip the next step. If you want to assign an administration user some of these tasks you need to follow the steps we outline in the next paragraphs. In this way you can use a `non-root` user to login to the WebSphere Application Server administration console. However, you still need to start the WebSphere Application Server instance and the domain manager as `root` user. This is due to WebSphere Application Server requirements. They specify that if you have global security enabled and if you are using local OS registry, the WebSphere Application Server instance must be started as `root` user.

In our ITSO environment we created a user `ibmewlm` as WAS administration user. Example 2-2 shows the command sequence we used at our domain manager to create user `ibmewlm` belonging to the administration group.

Example 2-2 Creation of administrative user on Red Hat Linux

```
Create user ibmewlm
# adduser ibmewlm

Assign password to user ibmewlm
# passwd
Changing password for user ibmewlm.
New password:

Retype new password:
passwd: all authentication tokens updated successfully.

Assign user ibmewlm to administration group
# usermod -g root -G root ibmewlm

Check if ibmewlm is associated with administration group
# su - ibmewlm
# id
uid=500(ibmewlm) gid=0(root) groups=0(root)
```

After successful creation of user `ibmewlm` we can start configuring our domain manager in the EWLM environment. We configure the domain manager in our test scenario with the minimum steps needed, and taking default values only. We assume that no SSL security is desired and that we have no firewalls implemented in our ITSO environment at this point. (We present a detailed discussion of security and firewall settings in Chapter 6, “Using a firewall and securing EWLM” on page 155.)

The first step in setting up the domain manager is to issue the **createDM** command. The syntax of this command looks as follows:

```
createDM <workingDir> -adminUser <userid> -adminPW <password> -wasPorts <port> -jp <port>
-ma <ipaddress> -mp <port> -dn <domainname> -auth [None | ServerSSL | ClientSSL] -sslks
<path> -sslpw <password>
```

In this section we provide information on the most relevant script parameters. For a detailed explanation of each parameter of the script refer to the IBM eServer information center at:

`http://publib.boulder.ibm.com/eserver/`

At this point you have the choice between creating the domain manager using a configuration wizard or the command line. In this section we provide detailed information on both methods, starting with command line.

Configuring domain manager using command line

We created the domain manager in our ITSO environment using the commands shown in Example 2-3 on page 34. The command takes a few minutes to complete. You can follow the process completion on your screen; it will look like the output shown in the second part of the example.

The **createDM** command is used to create an instance of the domain manager. Multiple instances of domain manager, each managing a different EWLM domain, may be consolidated on a single server. The working directory should not exist before creating the domain manager. It will be created when running the **createDM** command. It contains all information related to EWLM components as well as information associated with the EWLM Control Center. The working directory must be provided as an absolute path. If you still run into problems using the **createDM** command, we suggest you use a subdirectory name rather than a directory name as your working directory. This is not stated as such in the command documentation but was helpful in our environment.

Associated with each instance of the domain manager there is a WebSphere Application Server instance. This instance uses executables of the VE-installed WebSphere but provides its own directory for configuration files within the EWLM working directory. The `-adminUser` parameter specifies the user `ibmewlm` we created first on our operating system. Alternatively you can specify the `root` user. This userid is used as the WebSphere Administrator and hence must conform with the requirements WebSphere Application Server sets for a WebSphere Administrator. With the **changeCC** command you can change the administrator associated with the WebSphere Application Server instance for an EWLM domain manager or Single System EWLM.

There are three port numbers you specify when creating your domain manager. Verify that the ports are not being used beforehand. The `-wasPort` parameter assigns WebSphere Application Server ports on the domain manager. The number you specify is the start of a range of 15 contiguous ports used by the WebSphere Application Server instance. The `-jp` port parameter indicates the port used for any communication between the EWLM Control Center and the domain manager. The `-mp` parameter indicates the communication port between the domain manager and the managed servers. In addition, you need to specify the IP address or hostname of your domain manager and assign a name for the EWLM domain. This can be any name of your choice and is not to be confused with the domain resolution of your domain manager. In this setup we use a basic configuration for security encryption, which means using `-auth` parameter with value *`None`*.

Example 2-3 Creating the domain manager in our ITSO environment

```
# cd /opt/IBM/VE/EWLM/bin
# ./createDM.sh /opt/EWLMDM -adminUser ibmewlm -adminPW 111111 -wasPorts 20000 -jp 9092 -ma
9.12.4.142 -mp 3333 -dn itsoewlm -auth None

Processing createDM request. Please be patient as this may take a while...
Configuring EWLM Control Center
...processing 20% complete
...processing 30% complete
...processing 40% complete
```

```
...processing 50% complete
...processing 70% complete
...processing 80% complete
...Configuration of EWLM Control Center complete
...Ports assigned to EWLM Control Center:
- HTTP 20003
- HTTPS 20004
...Use -changeCC -controlCenterPorts to change these ports if desired.
...Ports assigned to WebSphere Admin Console
- HTTP 20001
- HTTPS 20000
...Use -changeCC -adminConsolePorts to change these ports if desired.
...Port assigned to WebSphere Admin: 20009
...Use -changeCC -adminPort to change this port if desired.
...Configuring EWLM Domain Manager
...processing 90% complete
PROCESSING COMPLETE
```

As a result of the **createDM** command, the domain manager working directory is created. The domain manager stores some information in the local file system of the server that it runs on, as shown in Figure 2-12.

```
EWLM Working Directory (created at Configuration time)
  -  EWLMData
     ·  PolicyDB ---> contains the deployed domain policies
     ·  PropertiesDB ---> contains the customization settings
     ·  ReportingDB ---> contains temporary performance statistics
     ·  ServerDB ---> contains the management domain configuration
  -  Diagnostics ---> contains logs, dumps and traces
  -  Interfaces ---> contains control information for programming interfaces
```

Figure 2-12 Domain manager local file system

Even though we have created the domain manager successfully, at this stage the EWLM Control Center is not working yet. We need to first add users to the operating system and the EWLM Control Center, then start the WebSphere Application Server, and then we can start the domain manager. First we want to view the properties of the domain manager; this can be done by issuing the **displayDM** command as shown in Example 2-4 on page 35. Your output should look similar to the one we accumulated on our domain manager running on Red Hat Enterprise Linux. If there is anything set up incorrectly on your domain manager you can run the **changeDM** or even the **deleteDM** command and reconfigure your domain manager at this stage.

Example 2-4 Output of the displayDM command to view domain manager properties

```
# cd /opt/IBM/VE/EWLM/bin
# ./displayDM.sh /opt/EWLMDM
Processing displayDM request.  Please be patient as this may take a while...
WLMConfig - configurable property settings:
   ViaProxyPort/vp(null)
   TracePlugin/tlog(Off)
   InterBrokerPort/dp(null)
   InterBrokerAddress/da(null)
   JmxPort/jp(9092)
   FirewallBrokerList/fb(null)
   ReportingTrace/rt(250)
   ViaProxyHost/va(null)
   DomainName/dn(itsoewlm)
   JniTrace/jt(250)
   SSLKeystore/sslks(null)
   ComponentTrace/ct(250)
   DomainManagerPort/mp(3333)
   MessageLog/ml(250)
   CommunicationTrace/nt(250)
   TraceDistHubBroker/tcomm(0)
   SocksPort/sp(null)
   FirewallBrokerPort/fp(null)
   FailureLimit/fl(50)
   SSLKeystorePassword/sslpw(password suppressed)
   DomainManagerAddress/ma(9.12.4.142)
   AuthorityLevel/auth(None)
   ProcessMode/mode(DomainManager)
   LBPublicPort/lbp(Off)
   LBSecurePort/lbs(Off)
   EventTrace/et(250)
   TraceLevel/tl(Min)
   TestComponent/t(null)
   DumpRetentionQuantity/dpn(25)
   DumpRetentionAge/dpa(30)
   SocksHost/sa(null)
WLMConfig - non-configurable property settings:
   ManagedServerFailureTime(null)
   ManagedServerIdentity(b05b63dad0404b3a5f460829852701c6)
   ManagedServerId(-1)
   StatisticsInterval(10)
   DomainManagerIdentity(null)
PROCESSING COMPLETE
```

Creating users for administrating EWLM

The next step is to map users or groups to corresponding EWLM application roles. These are needed to sign in to the EWLM Control Center with different identification and correspond to the three valid roles *Administrator*, *Operator* and *Monitor*. In our environment we created a user called `ewlmadm` as the Control Center administrator, `ewlmops` for operator tasks, and `ewlmmon` as the Control Center monitor. But before doing this we needed to create those three users in our operating system and assign them passwords. On our Linux domain manager we issued the **`adduser`** and **`passwd`** commands as before when creating the `ibmewlm` user.

To map these users to specific EWLM Control Center roles we ran the **`changeCC`** command as shown in Example 2-5. The working directory /opt/EWLMDM already exists; it was created with the **`createDM`** command. After a successful creation and mapping of these users we received the `PROCESSING COMPLETE` message.

Example 2-5 Creation and mapping of users for EWLM Control Center

```
# cd /opt/IBM/VE/EWLM/bin
# ./changeCC.sh -addUser /opt/EWLMDM -adminUser ibmewlm -adminPW 111111 -roleUser ewlmadm
-role Administrator
# ./changeCC.sh -addUser /opt/EWLMDM -adminUser ibmewlm -adminPW 111111 -roleUser ewlmops
-role Operator
# ./changeCC.sh -addUser /opt/EWLMDM -adminUser ibmewlm -adminPW 111111 -roleUser ewlmmon
-role Monitor
```

If you have a large number of users which you want to assign to the EWLM Control Center roles you can map multiple users at once to the EWLM Control Center with the following command:

```
# ./changeCC.sh -addUser /opt/EWLMDM -adminUser ibmewlm -adminPW 111111 -roleUser
"ewlmadm2|ewlmadm3|ewlmadm4" -role Administrator
```

When you look in the command directory /opt/IBM/VE/EWLM/bin you see that similar commands exist to remove users or to add and remove groups. In addition, you can display the role and existence of users for your EWLM Control Center as we show for our environment in Example 2-6. The syntax is similar to the commands used before, where `/opt/EWLMDM` is our working directory and `ibmewlm` is the WebSphere Application Server administrator. This command takes a while to run and has the following output:

Example 2-6 Display mapping of users in EWLM Control Center

```
# cd /opt/IBM/VE/EWLM/bin
# ./displayCC.sh -users /opt/EWLMDM -adminUser ibmewlm -adminPW 111111
Processing displayCC -users request.  Please be patient as this may take a while...
...processing 33% complete
...processing 66% complete
Role:  Administrator
Mapped Users:  ewlmadm
Role:  Operator
Mapped Users:  ewlmops
Role:  Monitor
Mapped Users:  ewlmmon
PROCESSING COMPLETE
```

In addition to displaying the users and groups of the EWLM Control Center, you can also display information on the assigned ports. Like all display commands, it takes a while to complete and the output looks as shown in Example 2-7.

Example 2-7 Display of assigned ports on domain manager

```
# cd /opt/IBM/VE/EWLM/bin
# ./displayCC.sh -ports /opt/EWLMDM -adminUser ibmewlm -adminPW 111111
Processing displayCC -ports request.  Please be patient as this may take a while...
...processing 33% complete
...processing 66% complete
...Ports assigned to EWLM Control Center:
      - HTTP 20003
      - HTTPS 20004
...Use -changeCC -controlCenterPorts to change these ports if desired.

...Ports assigned to WebSphere Admin Console
      - HTTP 20001
      - HTTPS 20000
...Use -changeCC -adminConsolePorts to change these ports if desired.

...Port assigned to WebSphere Admin: 20009
...Use -changeCC -adminPort to change this port if desired.
PROCESSING COMPLETE
```

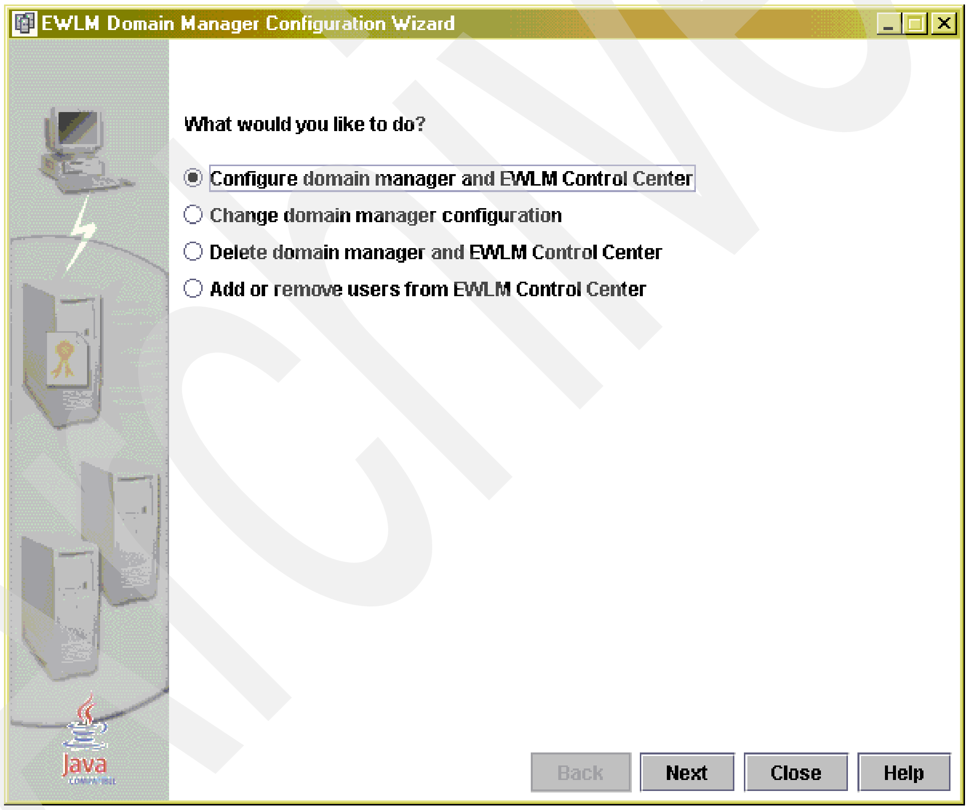
The setup seems correct. When we created the domain manager we specified the WebSphere Application Server port as 20000. By definition, this is the start of the range of the next 15 ports available for WebSphere Application Server use. We advise you to remember the ports assigned to the EWLM Control Center and to the WebSphere Application Server Admin Console for future logon, or you can save the URL in your Web browser. An alternative is that the Launchpad in the VE console shows the URL for the domain manager once it is configured. If you need to change ports there is a **`changeCC`** command which serves this purpose. For a detailed description of available options just type `./changeCC.sh [option]` and the valid options will be displayed. At this stage you are ready to start WebSphere Application Server and the domain manager. If you have configured the domain manager using command line you can continue with the section, “Starting WebSphere Application Server and the domain manager” on page 43. Alternatively you can configure the domain manager and map users to the EWLM Control Center using the EWLM Domain Manager Configuration Wizard described.

Configuring domain manager using the wizard

To start the domain manager configuration wizard on Linux run the following commands:

```
# cd /opt/IBM/VE/EWLM/bin
# ./configWizardDM.sh
```

The screen shown in Figure 2-13 is displayed. As you can see from the available options, you can use this wizard to create your domain manager and also at a later time to change or delete it. We first create the domain manager and then map users to the EWLM Control Center.



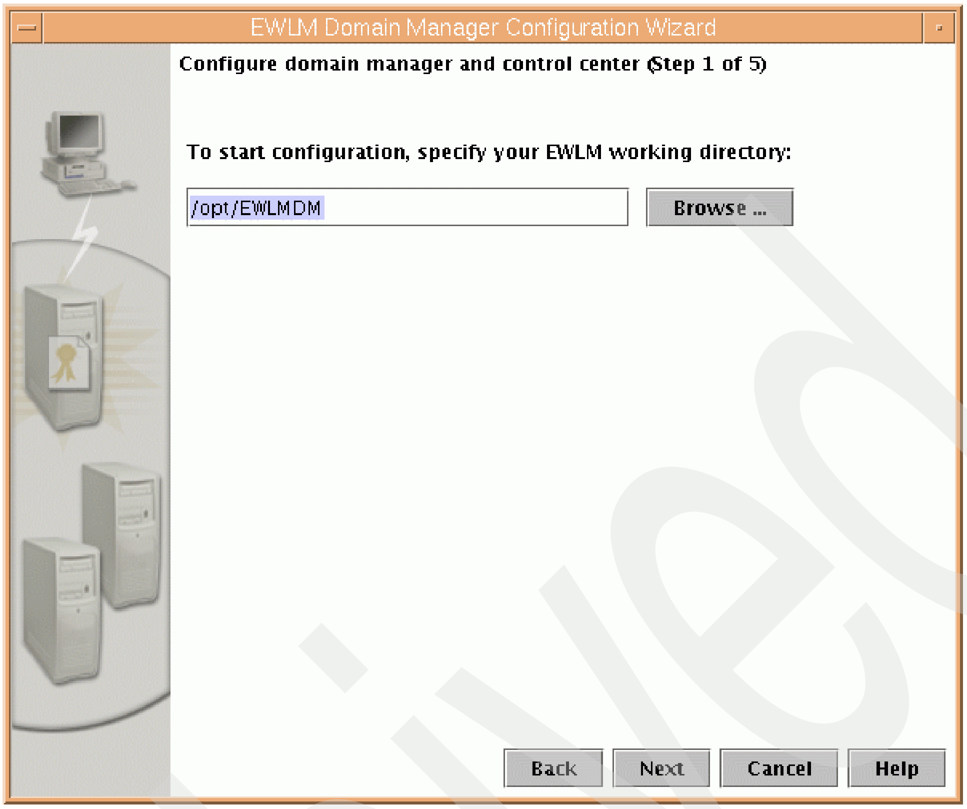
EWLM Domain Manager Configuration Wizard

What would you like to do?

- [✓ Configure domain manager and EWLM Control Center]
- Change domain manager configuration
- Delete domain manager and EWLM Control Center
- Add or remove users from EWLM Control Center

Figure 2-13 EWLM domain manager Configuration Wizard

Next, as shown in Figure 2-14, you are asked to define the domain manager working directory. For the domain manager on Linux and AIX, this working directory cannot be defined directly off of /. The working directory needs to be a subdirectory, for example /opt/ewlmdm.


EWLM Domain Manager Configuration Wizard

Configure domain manager and control center (Step 1 of 5)

To start configuration, specify your EWLM working directory:

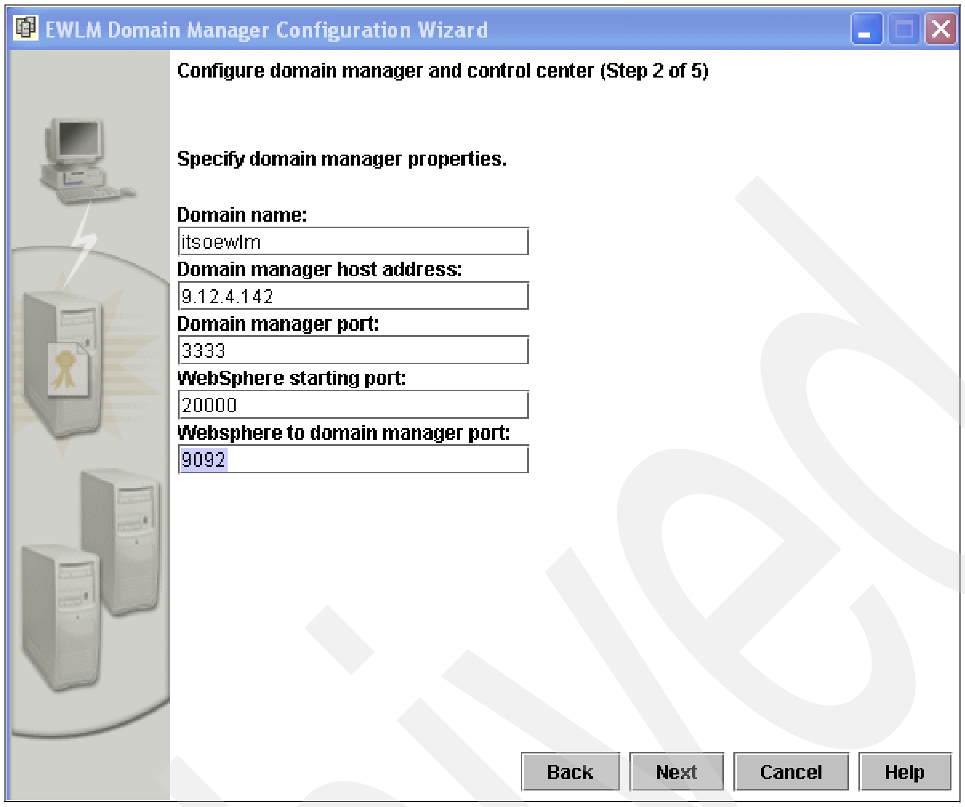
/opt/EWLMDM

Figure 2-14 Installation wizard: Installation directory

In Figure 2-15, the next screen of the wizard asks you to provide the domain name, the host address and port of the domain manager, as well as the WebSphere starting port and the WebSphere to domain manager port. We use the following values:

- Domain name: `itsoewlm`
- Domain manager host address: `9.12.4.142`
- Domain manager port: `3333`
- WebSphere starting port: `20000`
- WebSphere to domain manager port: `9092`

The *domain manager port* specifies the port of the domain manager used by the managed server or firewall broker to connect to the domain manager and should match the `-mp` parameter on the createMS command for the managed server. The *WebSphere to domain manager port* indicates the port that the EWLM Control Center uses to communicate with the domain manager.



Configure domain manager and control center (Step 2 of 5)

Specify domain manager properties.

Domain name: itsoewlm

Domain manager host address: 9.12.4.142

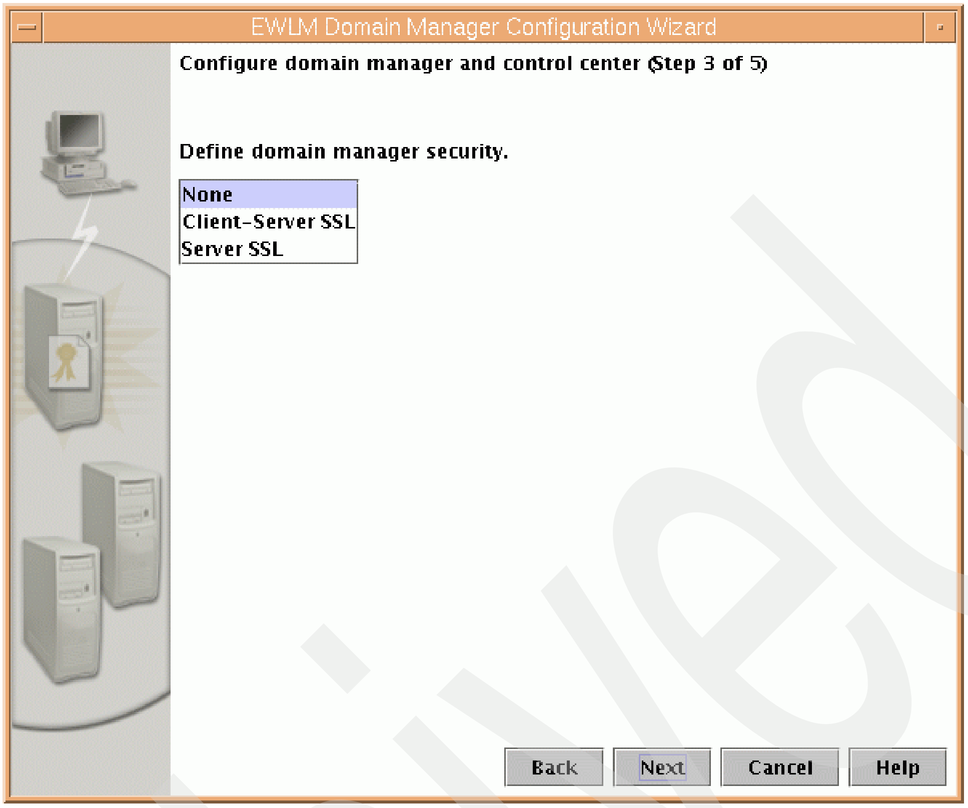
Domain manager port: 3333

WebSphere starting port: 20000

Websphere to domain manager port: 9092

Figure 2-15 Installation wizard: Domain manager properties

The next panel in the wizard, shown in Figure 2-16, asks for the security level. In our configuration, we started with `None` and later on we implemented SSL.


EWLM Domain Manager Configuration Wizard

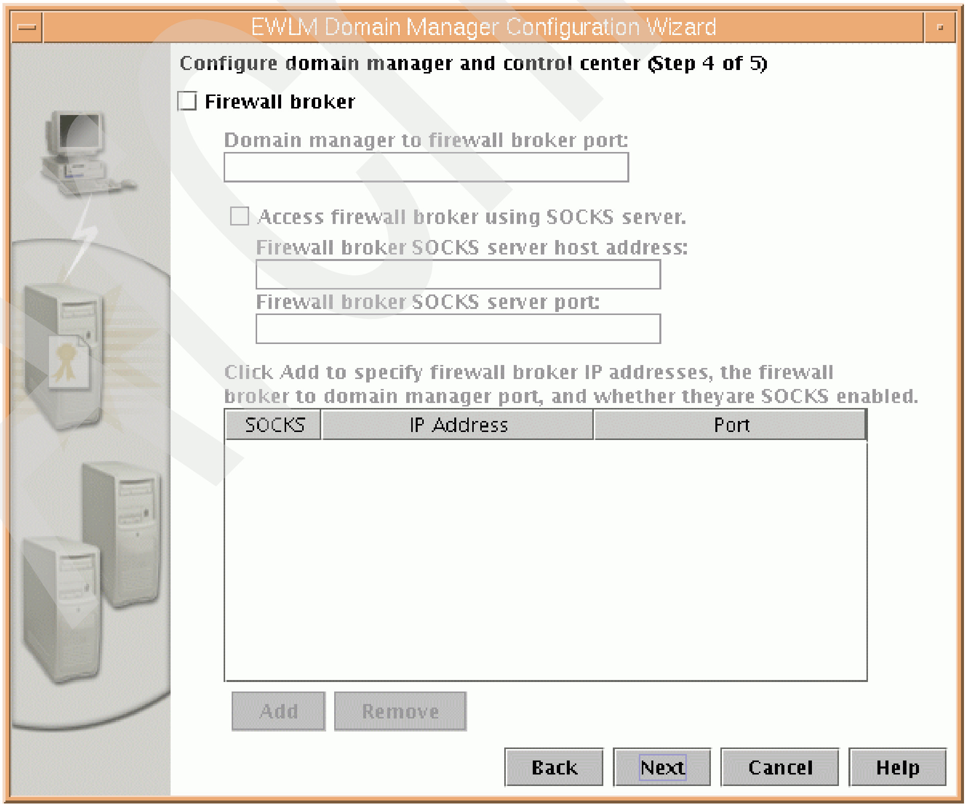
Configure domain manager and control center (Step 3 of 5)

Define domain manager security.

- None
- Client-Server SSL
- Server SSL

Figure 2-16 Installation wizard: Security

The next screen (Figure 2-17) asks if the firewall broker needs to be install or not.


EWLM Domain Manager Configuration Wizard

Configure domain manager and control center (Step 4 of 5)

☐ Firewall broker

Domain manager to firewall broker port:

☐ Access firewall broker using SOCKS server.

Firewall broker SOCKS server host address:

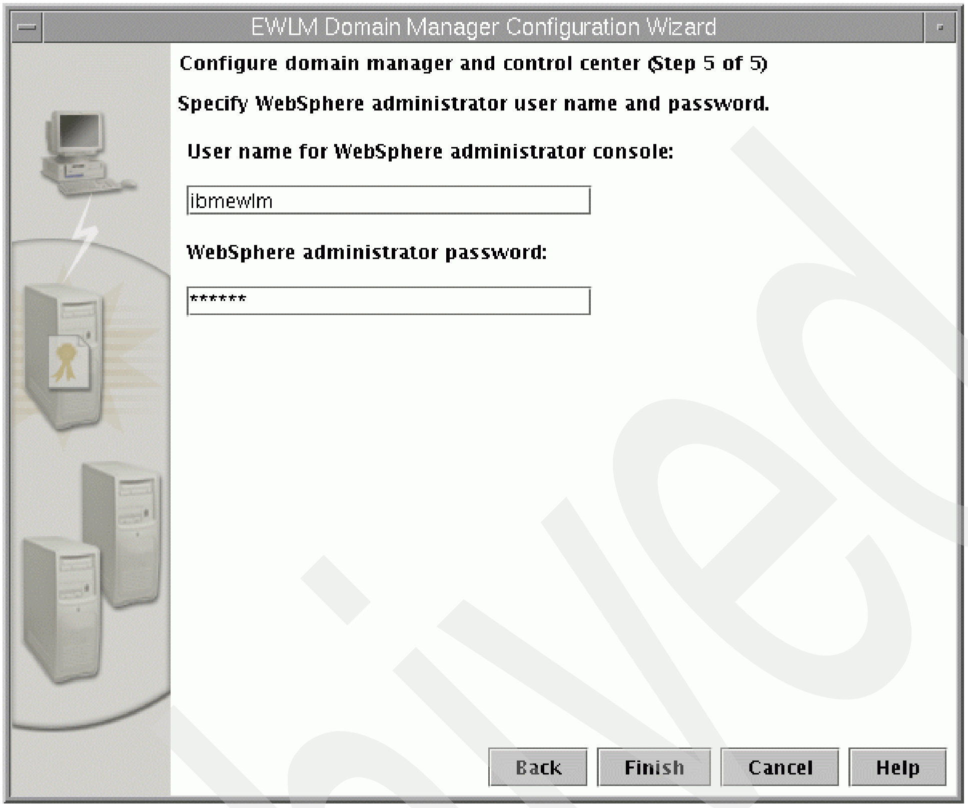
Firewall broker SOCKS server port:

Click Add to specify firewall broker IP addresses, the firewall broker to domain manager port, and whether theyare SOCKS enabled.

SOCKS	IP Address	Port

Figure 2-17 Installation wizard: Firewall broker specifications

Figure 2-18 asks for the WebSphere userid and password.



Configure domain manager and control center (Step 5 of 5)

Specify WebSphere administrator user name and password.

User name for WebSphere administrator console:

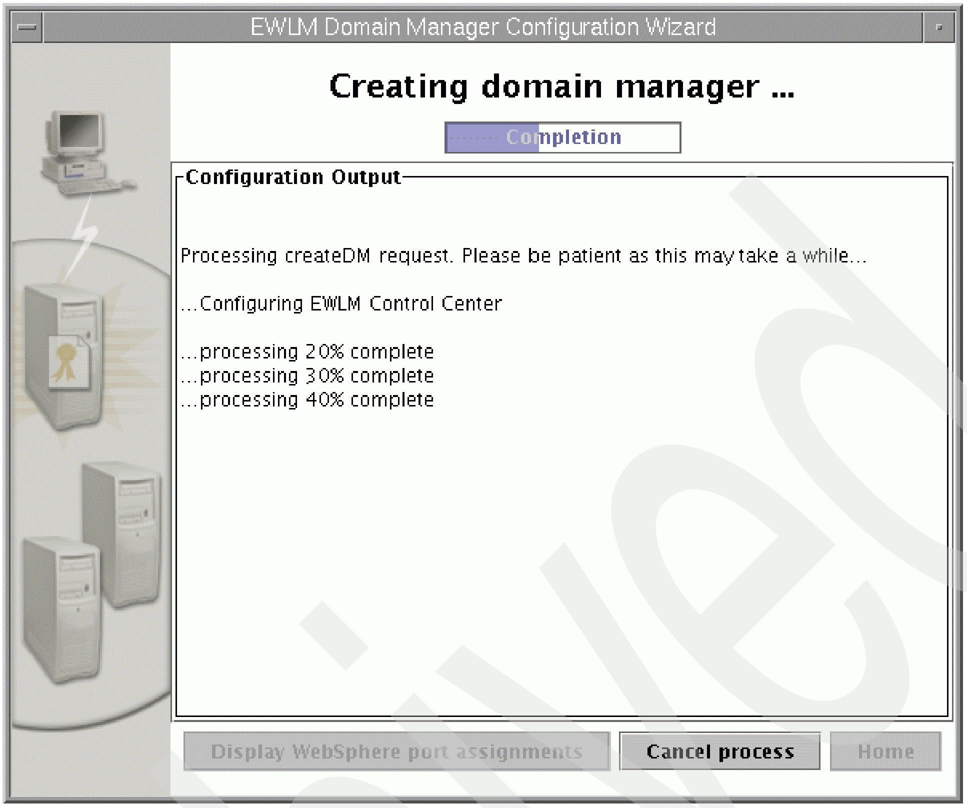
ibmewlm

WebSphere administrator password:

******

Figure 2-18 Installation wizard: WebSphere Administration user and password

When you click **Finish**, this starts of the creation of the domain manager. This looks similar to the output we received with command line creation of domain manager and is shown in Figure 2-19.


EWLM Domain Manager Configuration Wizard
Creating domain manager ...
Completion
Configuration Output
Display WebSphere port assignments
Cancel process
Home

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

...Configuring EWLM Control Center

...processing 20% complete
...processing 30% complete
...processing 40% complete
```

Figure 2-19 Configuration output when creating domain manager

After the process completes successfully we can go back to the home page of the wizard by clicking the Home button on the bottom right. From there we can map operating system users to the EWLM Control Center. We map users `ewlmadm`, `ewlmops` and `ewlmmon` to the valid roles Administrator, Operator, and Monitor.

Starting WebSphere Application Server and the domain manager

We are now ready to start WebSphere Application Server and the domain manager. At the time of this writing, using `root` user is the only working option.

To start WebSphere Application Server on the domain manager run the following commands:

```
# cd /opt/IBM/VE/EWLM/bin
# ./startWAS.sh /opt/EWLMDM
```

This starts the WebSphere Application Server instance associated with the domain manager and makes the EWLM Control Center and WebSphere Application Server administration console available for use. As with many other commands from this directory there is a log file associated. This is displayed in Example 2-8.

Example 2-8 Starting WebSphere Application Server on domain manager

```
# cat startWAS.log
startWAS command started at:  Fri May 21 15:55:41 EDT 2004 Platform =  Linux
STEP 1: Access EWLM working directory
EWLM working directory: /opt/EWLMDM/
STEP 2: Start WebSphere
ADMU0116I: Tool information is being logged in file
           /opt/EWLMDM/WAS/logs/server1/startServer.log
ADMU3100I: Reading configuration for server: server1
```

```
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 3003
Server started successfully
PROCESSING COMPLETE
```

Within this output file there is a link to another log file called startServer.log, which in our test case is located at /opt/EWLMDM/WAS/logs/server1. If WebSphere Application Server fails, you may find some indication of what caused the failure in this log. However, as part of the command sequence, the directory and all subdirectories will be cleaned up if WebSphere Application Server cannot start. In this case, you should look at the log file during the startup of WebSphere Application Server to see at which step the process runs into errors.

If you need to stop WebSphere Application Server to clean the process and restart it, you can stop it using the **`stopWAS`** command, where you need to specify the working directory, the `-adminUser` and `-adminPW` parameters.

If you were able to start the WebSphere Application Server successfully you can now start the domain manager using the **`startDM`** command as shown:

```
# cd /opt/IBM/VE/EWLM/bin
# ./startDM.sh /opt/EWLMDM
```

On the **`startDM`** command, you need to specify the working directory that was created using the **`createDM`** command. In addition, you can specify the initial and maximum allocated heap to the EWLM domain manager. If you do not specify any values the default values of 128 MB for initial heap size and 512 MB for maximum heap size apply. We advise you to run with default numbers and change them only if you run into any problems which are related to this. Refer to “Additional considerations” on page 235 for further details.

It is important that you start this command in the foreground and that you leave it there. At the time of writing the only recommended way to stop it is to use Ctrl-C. There is no **`stopDM`** command available. If you try to use the Linux **`kill`** command you might not be able to start the domain manager again because there are still some processes left which did not get killed automatically. You need to find and kill those first before you can restart your domain manager. You need to kill only one parent java process which forks into many java threads, it is displayed when you run the **`startDM`** command. The domain manager process includes the string `com.ibm.wlm.rte.WLM` and can also be found as shown in Example 2-9.

Example 2-9 Stopping domain manager

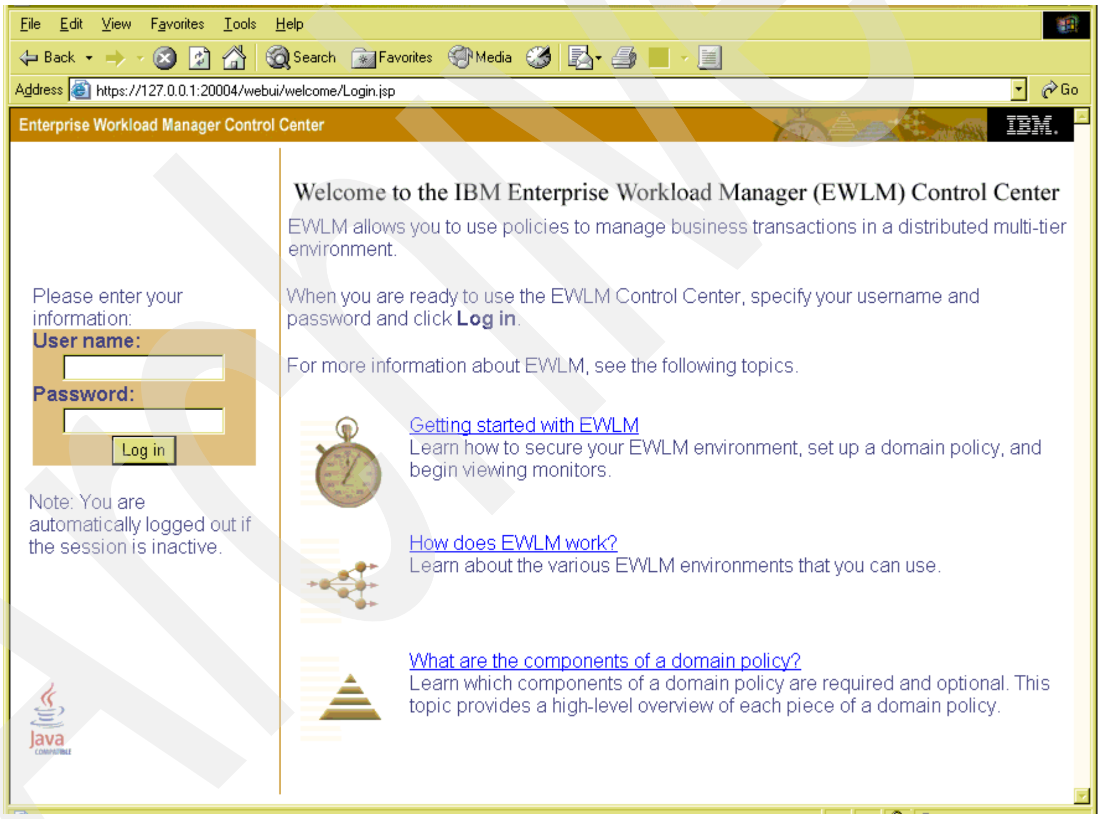
```
# ps -efww | grep com.ibm.wlm.rte.WLM |head -1

root     10973 10969  0 17:22 pts/0    00:00:04 /opt/IBM/VE/jre141/bin/java
-Djava.library.path=/opt/IBM/VE/EWLM:/usr/lib -classpath
/opt/IBM/VE/EWLM/classes/ewlm_updates.jar:/opt/IBM/VE/EWLM/classes/dm/Domain_Manager.jar:/o
pt/IBM/VE/EWLM/classes/ewlm_interfaces.jar:/opt/IBM/VE/EWLM/classes/db2j.jar:/opt/IBM/VE/EW
LM/classes/beepcore.jar:/opt/IBM/VE/EWLM/classes/beepsasl.jar:/opt/IBM/VE/EWLM/classes/beep
tls-jsse.jar:/opt/IBM/VE/EWLM/classes/commons-logging.jar:/opt/IBM/VE/EWLM/classes/tmx4jc.j
ar:/opt/IBM/VE/EWLM/classes/tmx4ji.jar:/opt/IBM/VE/EWLM/classes/concurrent.jar:/opt/IBM/VE/
EWLM/classes/AuthDatabaseSPI.jar:/opt/IBM/VE/EWLM/classes/rmiuuidspi.jar:/opt/IBM/VE/EWLM/c
lasses/log4j.jar:/opt/IBM/VE/EWLM/classes/xercesImpl.jar:/opt/IBM/VE/EWLM/classes/xml-apis.
jar:/opt/IBM/VE/EWLM/classes/dhbbroker.jar:/opt/IBM/VE/EWLM/classes/dhbcore.jar:/opt/IBM/VE
/EWLM/classes/jms.jar:/opt/IBM/VE/EWLM/classes/samplebroker.jar:/opt/IBM/VE/EWLM/classes:
com.ibm.wlm.rte.WLM /opt/EWLMDM/

# kill -KILL 10973
```

If for some reason the Ctrl-C sequence does not work, you can also run the **killall java** command, although it might be dangerous in a production environment because this command kills *all* java processes including WebSphere Application Server. If you want to kill the domain manager processes only you can use the sequence shown in the previous example.

Once you have issued the command to start the domain manager you can now open a browser. The EWLM Control Center has to be on the domain manager, but you can connect to the Control Center from any browser running either on the domain manager itself or on other machines. We opened a browser with the following URL:

```
http://ewlmdm1.itso.ibm.com:20003/webui
```

Instead of using the name `ewlmdm1` of the domain manager you can also use the IP address. You can substitute the EWLM Control Center HTTP port from the `createDM` command output as shown in Example 2-7 on page 37. This opens the EWLM Control Center welcome page shown in Figure 2-20. When you look at the URL and compare it with the one we entered you can see that we automatically got rerouted from an http to a secure https site. For more information on this refer to Chapter 6, “Using a firewall and securing EWLM” on page 155.



https://127.0.0.1:20004/webui/welcome/Login.jsp

Welcome to the IBM Enterprise Workload Manager (EWLM) Control Center

EWLM allows you to use policies to manage business transactions in a distributed multi-tier environment.

When you are ready to use the EWLM Control Center, specify your username and password and click **Log in**.

For more information about EWLM, see the following topics.

Getting started with EWLM
Learn how to secure your EWLM environment, set up a domain policy, and begin viewing monitors.

How does EWLM work?
Learn about the various EWLM environments that you can use.

What are the components of a domain policy?
Learn which components of a domain policy are required and optional. This topic provides a high-level overview of each piece of a domain policy.

Please enter your information:
User name:
Password:

Note: You are automatically logged out if the session is inactive.

Figure 2-20 EWLM Control Center welcome page

We are ready to log in with one of the users we mapped to the EWLM Control Center role. There will be different options available depending on whether you log in as user `ewlmadm`, `ewlmops`, or `ewlmmon`. We give a detailed description of these roles in “EWLM Control Center overview” on page 78. There are, in addition, commands to turn on tracing for the EWLM Control Center; these can be useful when you need to debug the system. More information on debugging options is in 8.6, “Problems using the EWLM Control Center” . Another useful source of information is provided by the WebSphere server logs. They are located on our

Linux domain manager at /opt/EWLMDM/WAS/logs/server1 and are called `SystemOut.log` and `SystemErr.log`.

When we log in to the EWLM Control Center now and look at the managed servers in the monitoring section on the left we do not see any servers. The domain manager does not detect any managed servers because there are no managed servers that have joined this domain yet.

2.4.2 Configuring managed servers on AIX and Windows

Once our domain manager is running we are ready to install and configure the managed servers that belong to the EWLM management domain. In our ITSO environment, the EWLM Management Domain consists of two systems running AIX 5L Version 5.2 ML03, and two systems running Windows Server Version 2003. For information on the installation and configuration of managed servers running other supported operating systems refer to 2.6.1, “Managed server on i5/OS” and to 2.6.2, “Managed server on Solaris” . In this section we describe the setup of the managed servers of our ITSO environment only. On Windows we do not configure and run the managed server as Administrator but as user `tested`. On AIX we show the configuration as `root` user even though it is not required. The managed server must be able to call the EWLM APIs, which require either `root` authority or that you use a `non-root` user like `ibmewlm` and grant authority by typing the following command:

```
chuser capabilities=CAP_EWLM_AGENT,CAP_PROPAGATE ibmewlm
```

When you install the EWLM code on the domain manager you are also given the option to install the images for the managed servers. This means that the code for the managed servers is copied to your domain manager but it does not get pushed out to the managed servers automatically. You need to manually transfer the EWLM managed server code from the domain manager to each managed server. This distribution of managed server images can be done either by file transfer protocol (FTP) using binary mode on AIX and Solaris, or by mapping a drive and copying the files to the file system, or by any other distribution mechanism that works in your environment. Be aware that FTP does not maintain the executable bit on the installation image file. You need to run **`chmod 700`** on the installation image to turn on the executable bit.

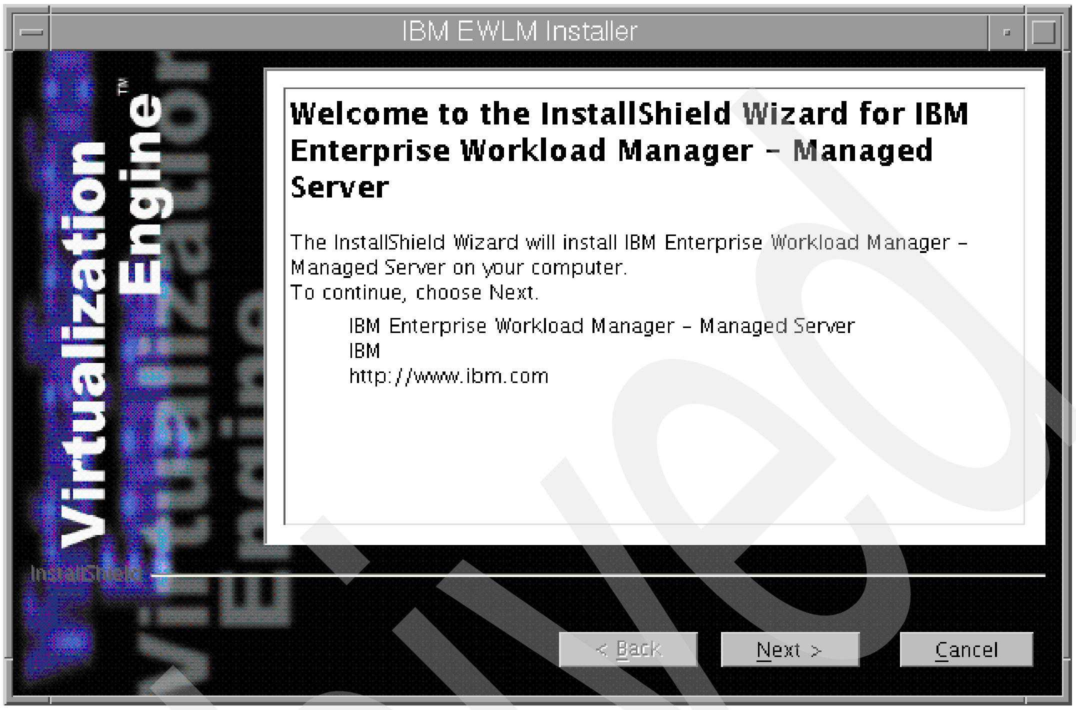
The managed server code location on the domain manager is a subdirectory of your installation directory; on our domain manager it is at /opt/IBM/VE/ManagedNodes. For the current release of EWLM, the IBM installation image includes AIX and i5/OS. The non-IBM installation image includes images for Windows and Solaris. Which image will be in your domain manager directory depends on what you have purchased. You can identify which image goes with which platform by the names of the installable files, which are as follows:

- AIX - EWLMAixMS.bin
- Windows - EWLMWinMS.exe
- Solaris - EWLMSolarisMS.bin
- i5/OS - EWLMOS400MS.jar and Q5733VE1110.savf

We explain in 2.6.1, “Managed server on i5/OS” why you need two files for a managed server installation on i5/OS.

Installing managed server code

In our ITSO environment, the managed servers are Windows and AIX machines which means we need the files called `EWLMWinMS.exe` and `EWLMAixMS.bin`. We transfer these files to our managed servers into the /software/EWLM directory, which can be a location of your choice. Make sure the rights are set so the files can be executed. We start the installation wizard by double-clicking the `EWLMWinMS.exe` file on Windows and by issuing the **`./EWLMAixMS.bin`** command on AIX. This brings up the installation window shown in Figure 2-21. This installation shield looks similar for all operating systems.



Welcome to the InstallShield Wizard for IBM Enterprise Workload Manager – Managed Server

The InstallShield Wizard will install IBM Enterprise Workload Manager – Managed Server on your computer.
To continue, choose Next.

IBM Enterprise Workload Manager – Managed Server
IBM
http://www.ibm.com

Figure 2-21 Installation wizard to configure the managed servers

In the next step you are asked to provide an installation directory for the managed server or you can use the default directory. This is the /opt/IBM/VE/EWLMMS directory on AIX or C:\Program Files\IBM\VE\EWLMMS on Windows. When proceeding further with the installation you are asked which features you would like to install. The choice is to install the managed servers with or without Firewall Broker; we chose the managed servers only. For information on firewall issues in this server topology refer to 6.1, “Firewalls” . You are then provided with an installation summary before the installation process starts. With the choice of installing the managed servers only you need about 60 MB of space in the installation directory you provided. The installation process takes only a few minutes including some post installation tasks. The final summary includes the following message:

```
The InstallShield Wizard has successfully installed IBM Enterprise Workload Manager -
managed server. Choose Finish to exit the wizard.
```

Looking then at the directory we chose in the installation process we see a variety of available commands and associated log files for the managed servers. This looks similar to the command directory for the domain manager but with fewer commands available. An output taken from one of our AIX machines is shown in Example 2-10. It looks similar for Windows except that for Windows the directory is at C:\Program Files\IBM\VE\EWLMMS\bin.

Example 2-10 AIX command directory

```
ls /opt/IBM/VE/EWLMMS/bin
# ls
ConfigurationWizard.log  createFB.sh          displayMS.log
changeFB.sh              createMS.log         displayMS.sh
changeMS.sh              createMS.sh          startFB.sh
config.jar               deleteFB.sh          startMS.log
configWizardMS.sh        deleteMS.sh          startMS.sh
```

```
createFB.log               displayFB.sh               variables.sh
```

Note: Alternatively, you can install the manage server from a command line or batch file. From the directory where you copied the installation images run the following command:

```
EWLM_MSfilename -silent -P managedServerBean.active=[true | false]
              -P firewallBrokerBean.active=[true | false]
              -P productBean.installLocation=mydirectory
```

We are now ready to run the **createMS** command to configure the managed servers. We divide this configuration section into separate sections for Windows and AIX, but first we need to say a few words about syntax of the **createMS** command shown here:

```
createMS <working dir> -ma <address> -mp <port> -auth [None |ServerSSL | ClientSSL] -sslks
<path> -sslpw <password>
```

As with the creation of the domain manager, the working directory should not exist. It is created by the **createMS** command and must be provided as an absolute path. With the `-ma` flag we specify the IP Address or hostname of the domain manager. With the `-mp` flag we specify the port used for communication between the domain manager and the managed servers. The `-auth` flag is provided for the specification of authority level. The `-ssl` values are needed if you want to have security settings active. Similar to what we described for the domain manager, you can also run an installation wizard to go through the **createMS** process. Double-click **configWizardMS.bat** in Windows or call the **./configWizardMS.sh** script on AIX and fill in the required values.

Creation of managed server on Windows

We start running the **createMS** command on our Windows managed servers with the following command sequence:

```
> cd "C:\Program Files\IBM\VE\EWLMMS\bin"
> createMS C:\ewlmMS -ma 9.12.4.142 -mp 3333 -auth None
```

We can then use the **startMS** command to start our managed server as shown:

```
> cd "C:\Program Files\IBM\VE\EWLMMS\bin"
> startMS C:\ewlmMS
   Starting EWLM Managed Server....
   A message is displayed when the EWLM Managed Server completes activation.
   WARNING: Closing this message window ends the EWLM Managed Server.
<10:22:16.2216> EWLM successfully started
```

This command runs in a command window. To stop the managed server use the Ctrl-C keys. In addition, there are commands like **displayMS** and **changeMS** to view or change the properties of the managed servers, as well as a **deleteMS** command. The output of the managed servers properties looks very similar to the one shown in Example 2-4 on page 35.

Once you are comfortable with the **startMS** command, we recommend running this process as a service by using the **startMSService.bat** command. This is also the recommended way to run it in a production environment.

Creation of managed server on AIX

Setting up the managed server on AIX works in a very similar way but some additional steps are necessary. The first thing to notice is that with ML03 of AIX 5L Version 5.2 there are some SMIT panels available to manage the EWLM environment. You can verify this by typing the **smitty ewlm** command. The two available options are to change or show the EWLM configuration and the status of the EWLM services. Alternatively, you can run the **ewlmcfg** command with several options. With the **-q** flag, for example, you query the status of EWLM.

Note: You can use the command **ewlmcfg -c** to enable and **ewlmcfg -u** to disable the EWLM/ARM on AIX. If you disable the EWLM environment using the **ewlmcfg -u** command after EWLM is up, the domain manager shows a server state indicating that ARM is disabled. Also you need to stop and start a managed server after enabling the EWLM environment. If you don't restart the managed server, it does not communicate with domain manager.

This step provides interfaces that are required by the EWLM code. With any default installation you will not see any filesets when searching for EWLM using the **lslpp** command. After you run the managed server installation wizard described in 2.4.2, "Configuring managed servers on AIX and Windows" the **lslpp** command output looks as follows:

```
# lslpp -l |grep ewlm
ewlm.common               1.1.0.0  COMMITTED  EWLM Common AIX Support
```

This fileset comes with the EWLM code. In addition, when setting up your AIX system as a managed server you need to install one additional fileset called bos.net.ewlm.rte from the standard AIX CDs. It is part of ML03 of AIX 5L Version 5.2 and can also be downloaded from the following Web site:

```
http://techsupport.services.ibm.com
```

When installing from CD, insert the CD into the CDROM and issue the following command:

```
# installp -acgXd /dev/cd0 bos.net.ewlm.rte
```

When this fileset is installed successfully the output of the **lslpp** command should look as follows:

```
# lslpp -l |grep ewlm
bos.net.ewlm.rte          5.2.0.0  COMMITTED  netWLM
ewlm.common               1.1.0.0  COMMITTED  EWLM Common AIX Support
```

At this point we are ready to run the **createMS** and the **startMS** commands similarly to what we just did on Windows. We show the command sequence for AIX in Example 2-11. Optionally you can also run the **displayMS** command to view the properties of this managed server or use the **changeMS** or **deleteMS** command to modify your setup.

Example 2-11 Configuring AIX as managed server

```
# cd /opt/IBM/VE/EWLMMS/bin
# ./createMS.sh /opt/ewlmMS -ma 9.12.4.142 -mp 3333 -auth None
# ./startMS.sh /opt/ewlmMS
   Starting EWLM Managed Server....
   A message is displayed when the EWLM Managed Server completes activation.
   WARNING: Closing this message window ends the EWLM Managed Server.
<10:22:16.2216> EWLM successfully started
```

As mentioned previously, there are SMIT panels available for EWLM managed server configuration, including the permanent or temporary enablement of EWLM services. When you look at the AIX ARM/EWLM implementation you see that there are a number of configurable values. Most of the values indicate the maximum number of specified items that may be created. For example, *registered applications* specifies the maximum number of applications that can be concurrently registered in the system. If an application calls one of the ARM services that would exceed the limit, the application will get a failure return code indicating the limit was exceeded. Another example is the *transactions per process* limit. This limit is there to catch applications that have a *transaction leak* flaw in their instrumentation.

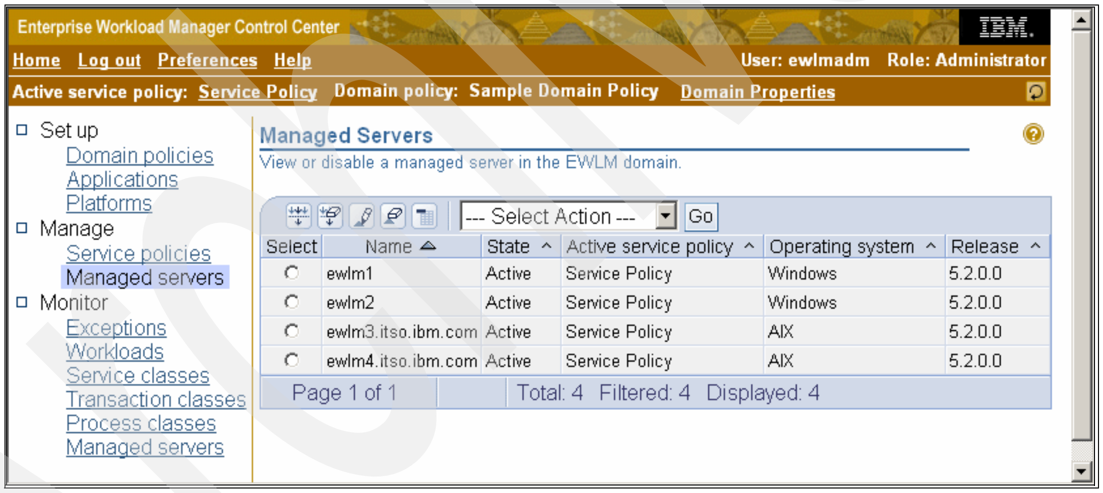
These numbers can be changed if needed. However, if limits are made too small, applications may start getting failure return codes. If you make them too large, you run the risk of the system failing by running out of heap space. If you make the completion buffer size too small, the EWLM agent may not see all completed transactions. If you change the priority of the managed server to a high number, the managed server may not run often enough. If you choose this number to be too small this may prevent critical system daemons from running. We therefore advise you to work with default numbers to start with and to change them only if you run into problems.

Verify managed servers status from Control Center

The configuration of our domain manager running on Linux is complete, as is the configuration of all managed servers in our environment running on AIX and Windows. When we go back to the EWLM Control Center and look at the status of the managed servers we should be able to see all four machines now. We open a browser with the following URL:

```
http://ewlmdm1.itso.ibm.com®:20003/webui
```

We log in to the EWLM Control Center with any of the mapped users, which in our setup are `ewlmadm`, `ewlmops` and `ewlmmon`. The available options will be different for these user roles but all three can do monitoring tasks. When we click on managed servers on the left menu bar we see the output shown in Figure 2-22. This shows that all four systems are active. It provides information on the operating system and its version, where Windows release 5.2 corresponds to Windows 2003 and Windows release 5.0 corresponds to Windows 2000.



Enterprise Workload Manager Control Center
IBM
Home Log out Preferences Help
User: ewlmadm Role: Administrator
Active service policy: Service Policy Domain policy: Sample Domain Policy Domain Properties
Set up
Domain policies
Applications
Platforms
Manage
Service policies
Managed servers
Monitor
Exceptions
Workloads
Service classes
Transaction classes
Process classes
Managed servers
--- Select Action ---
Go

Managed Servers

View or disable a managed server in the EWLM domain.

Select	Name	State	Active service policy	Operating system	Release
○	ewlm1	Active	Service Policy	Windows	5.2.0.0
○	ewlm2	Active	Service Policy	Windows	5.2.0.0
○	ewlm3.itso.ibm.com	Active	Service Policy	AIX	5.2.0.0
○	ewlm4.itso.ibm.com	Active	Service Policy	AIX	5.2.0.0

Page 1 of 1   Total: 4 Filtered: 4 Displayed: 4

Figure 2-22 Monitoring status of managed servers after initial configuration

We are now ready to start instrumenting our applications. This is described in Chapter 3, "Enabling middleware for EWLM" on page 57. If your domain manager or managed servers run on supported operating systems other than the ones we described for our test setup, refer the Section 2.5, "Installing the domain manager on other operating systems" and 2.6, "Configuring managed servers on other operating systems" on page 54. We will not go through the installation and configuration steps in detail but outline the procedure highlighting any differences you may see.

2.5 Installing the domain manager on other operating systems

In the previous section we guided you through the installation of the domain manager on a Linux platform. These steps are valid in general. Nonetheless, there are some differences and further issues if your domain manager is on other supported operating systems. This is the focus of the following sections.

2.5.1 Domain manager on AIX

The configuration of the domain manager on AIX is very similar to the Linux configuration. EWLM. On AIX you need to create a CDROM file system first using the **`smitty crcdrfs`** command. Insert the first CD, labeled "VE Console & Runtime for AIX 5L (1 of 3)" and mount the CDROM file system. The mount point in our example is /cdrom. Change to this directory and look at the **`ls`** command output. The sequence of steps looks as follows:

```
# mount /cdrom
# cd /cdrom
# ls
Common/             VEInstall.ico         installVECore.jar*
EWLM/               installVE.jar*        log.txt
VEC/                installVEAgents.jar*  uninstallVEAix.sh*
VECoreLog.txt       installVEAix.sh*
```

The following command executes the **`installVEAix`** script and the Virtualization Engine wizard appears on your screen:

```
# ./installVEAix.sh
```

Alternatively, you can mount the CDROM and launch autorun to start the installation. Use the **`mount -v cdrfs -oro /dev/cd0 /mnt`** command to mount the CD. Launch the installation program by typing the **`/mnt/autorun`** command. Autorun creates temporary directories, copies the setup file from the CD to the temporary directory and launches the **`installVEAix.sh`** script.

The space required for the installation directory /opt/IBM/VE is about 500 MB. The required space for the installation directory for all operating systems is shown in Table 1-1 on page 17. At the time of writing you must be `root` user to install EWLM code and to run WebSphere Application Server and the domain manager. This is due to WebSphere requirements which we explained in the first part of this chapter.

2.5.2 Domain manager on Windows

When you set up your domain manager on Windows you start by copying the installation images to your server as we described in 2.3, "Virtualization Engine suite" . The CDs you need for running domain manager on Windows are labeled "VE Console & Runtime for Windows (*N* of 3)."

The configuration commands and associated log files are located at C:\Program Files\IBM\VE\EWLM\bin. To create the domain manager on Windows you can either issue the command sequence described in "Configuring domain manager using command line" on page 34 or you can call a configuration wizard as shown in "Configuring domain manager using the wizard" on page 38.

If you want to run the WebSphere Application Server and the domain manager as user `Administrator` you can immediately start the configuration. If you want to create another user to run these commands, you need to make sure that this user has the right operating system privileges. For example, create a user `ibmewlm` as a member of the administrative group with the following privileges:

- Act as part of the operating system
- Log on as a service

We log into our Windows machine as user `Administrator` and follow this procedure:

- Click **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Local Security Policy** → **Local Policies** → **User Rights Assignments**
- Click **Add** and assign these two privileges to `ibmewlm`:
  - `Act as part of the operating system`
  - `Log on as a service`

These steps are necessary to ensure that this userid conforms with the requirements WebSphere sets for a WebSphere Administrator. In addition you need to create users to map to the EWLM Control Center roles. On Windows we create users `ewlmadm`, `ewlmops`, and `ewlmmon` as `Standard Users`.

At this point you have two choices, just as we described for the configuration of domain manager on Linux. Creating the domain manager using the configuration wizard is very similar to what we discussed in 2.4.1, “Configuring domain manager on Linux” . We therefore concentrate on the important command sequence only. In Example 2-12 we outline the command line steps you need to follow when you want to configure your domain manager on Windows without the wizard.

Example 2-12 Outline of configuration of domain manager on Windows

```
Logon as Administrator:
> cd "C:\Program Files\IBM\VE\EWLMMS\bin"
> createDM C:\IBM\EWLMDM -ma 9.12.6.143 -mp 3333 -adminUser ibmewlm -adminPW 111111
-wasPorts 20000 -jp 9092 -dn itsoewlm -auth None
# changeCC -addUser C:\IBM\EWLMDM -adminUser ibmewlm -adminPW 111111 -roleUser ewlmadm
-role Administrator
> changeCC -addUser C:\IBM\EWLMDM -adminUser ibmewlm -adminPW 111111 -roleUser ewlmops
-role Operator
> changeCC -addUser C:\IBM\EWLMDM -adminUser ibmewlm -adminPW 111111 -roleUser ewlmmon
-role Monitor

Logon as ibmewlm:
> cd "C:\Program Files\IBM\VE\EWLMMS\bin"
> startWAS C:\IBM\EWLMDM
> startDM C:\IBM\EWLMDM
```

Once you are comfortable with the **`startDM`** command, we recommend running this process as a service by using the **`startDMService.bat`** command. This is also the recommended way to run it into a production environment.

If you want to map several users to the `-role Administrator` flag, for example, the command syntax is slightly different than what we showed for Linux. You can specify multiple users by separating the users with the colon character and surrounding the list with double quotes as follows:

```
>cd "C:\Program Files\IBM\VE\EWLMMS\bin"
>changeCC -addUser C:\IBM\EWLMDM -adminUser ibmewlm -adminPW 111111 -roleUser
"ewlmadm1:ewlmadm2:ewlmadm3" -role Administrator
```

Log files associated with these creation commands are in the same directory. Log files associated with the WebSphere Application Server are at C:\IBM\ewlmDM\WAS\logs\server1 as well as at C:\Program Files\IBM\VE\WebSphere\AppServer\logs\server1.

If all commands completed successfully you are now ready to start your browser and log on to the EWLM Control Center.

2.5.3 Domain manager on i5/OS

Installing the EWLM code when your domain manager runs on i5/OS cannot be done directly. For this you need to use the iSeries™ Access family of products to administer and access your iSeries server. In our ITSO environment we use IBM eServer iSeries Access for Windows. This means, the installation of the Virtualization Engine base support and EWLM code works as on Windows. However, you need to insert the CD labelled “VE Console & Runtime for OS/400® (1 of 3)” into your Windows PC; this automatically launches the installation wizard on the Windows PC. For the first screen you need to provide the i5/OS server name and login information. The installation of WebSphere Application Server takes a lot longer than on the other operating systems we discussed in this chapter. You may find that it takes up to two hours.

Once the EWLM code and WebSphere Application Server are installed on the i5/OS we need to log on to the system. We run the **STRWLM CL** command with the following options:

```
# STRWLM CFGID(ewlmadm) MODE(*DMNMGR) DMNMGRNAME("9.56.204.66") DOMAIN(itsoewlm)
DMNMGRPORT(9092)
```

All configuration scripts require the EWLM configuration ID as a parameter. This allows all information associated with a specific instance of an EWLM component to be easily found. The ID is created by running the **STRWLM CL** command, which creates a directory under /qibm/userdata/wlm/servers/<configID>. It contains all EWLM-related information. In addition, it is used to name the WAS instance and causes the WebSphere data to be placed in a directory called /qibm/userdata/webas51/base/<configID>.

The WebSphere Application Server instance is created using the **createCC** command. For this to run we issue the **qsh** command to be able to run shell commands. We run the command sequence as shown:

```
# cd /qibm/proddata/wlm/bin
# createCC ewlmadm -adminUser qsecofr -adminPW 111111 -wasPorts 20000 -jp 9092
```

For the configID parameter we take `ewlmadm`. It uniquely identifies the domain manager server job and must be the value you passed on the **STRWLM CL** command. The adminUser is used as WebSphere administrator and must conform with the requirements WebSphere sets. However, for i5/OS there are no special requirements for WebSphere Application Server. To install and configure EWLM the system administrator needs to have *ALLOBJ, *SECADM, *JOBCTL, and *IOSYSCFG special authority. In our test setup we used the administration user `qsecofr`.

The next step is to map users to the specific EWLM roles. These users must be created first on the operating system. We can then issue the **changeCC** command with the arguments shown in Example 2-13. Run the **startWAS** command and open a browser to the EWLM Control Center as described for our ITSO environment. You should be able to log on as one of the users specified here.

Example 2-13 Configure EWLM on i5/OS domain manager

```
# cd /qibm/proddata/wlm/bin
# changeCC -addUser ewlmadm -adminUser qsecofr -adminPW 111111 -roleUser ewlmadm -role
Administrator
# changeCC -addUser ewlmadm -adminUser qsecofr -adminPW 111111 -roleUser ewlmops -role
Operator
# changeCC -addUser ewlmadm -adminUser qsecofr -adminPW 111111 -roleUser ewlmmon -role
Monitor
# ./startWAS.sh ewlmadm
```

2.6 Configuring managed servers on other operating systems

For our ITSO environment we configured managed servers on AIX and Windows. In this release of the EWLM product, the two other supported operating systems for managed servers are i5/OS and Solaris, which we outline in this section.

2.6.1 Managed server on i5/OS

The i5/OS managed servers can only be installed using a remote installer, as we did with the i5/OS domain manager. You first need to copy the installation images for i5/OS to your Windows PC. These are located at /opt/IBM/VE/ManagedNodes on your domain manager running on Linux or AIX, or at C:\Program Files\IBM\VE\ManagedNodes. Run the following command from the directory on the Windows system where you copied both files `EWLMOS400MS.jar` and `Q5733VE1110.savf`:

```
# java -jar EWLMOS400MS.jar -os400 <target_system_name> <userid> <password>
```

The `target_system_name` is the hostname of the target i5/OS. You also need to specify the user ID and password to access this system. For testing purposes we use user `qsecofr`. In general, this user needs to have authority to install programs on i5/OS. In addition, this operating system does not allow you to redirect the installation target path. This means, the managed server and firewall broker, if applicable, are installed in /QIBM/ProdData/WLM. This procedure installs the managed server on i5/OS.

Once the managed server is installed it needs to be configured. You need to log in to the i5/OS system and run the **`STRWLM CL`** command to start the managed server job under the QWLM subsystem. This configures and starts the managed server, as takes place with the **`createMS`** and **`startMS`** command on other platforms. The **`STRWLM CL`** command runs under the QWLM profile with job name QWLMSVR. You must have *JOBCTL special authority to run this command. In our test example we use the administration user `qsecofr`. The **`STRWLM CL`** command you need to issue looks as follows:

```
QWLM/STRWLM CFGID(ewlmms) MODE(*MGDSVR) DMNMGRPORT(3333) DMNMGRNAME(ewlmdm1.itso.ibm.com)
AUTLVL(*DFT)
```

2.6.2 Managed server on Solaris

Even though we do not have a Solaris server in our ITSO environment, we have included a short discussion for completeness. The steps are very similar to the ones we discussed when running managed server on AIX.

To install the managed server on Solaris you need to transfer the EWLMSolarisMS.bin file from your domain manager into a temporary directory. Set the execution bit on with the **`chmod 700`** command. The **`createMS`** and **`startMS`** commands require either root authority or the user must have an effective user identification (EUID) of zero.

For application programs to be able to access the ARM APIs the user must be authorized. You can authorize up to 1024 users in the /etc/EWLM/auth file. To add users to the authorization list you need to open the /etc/EWLM/auth file. Add the user names to the line application_auth and separate by commas if you specify more than one user. In addition, the file /etc/EWLM/limits specifies limits for ARM operations to prevent misbehaving programs from negatively affecting the system. Care should be taken in changing the value of the limits since it is possible for ARM API calls to fail due to incorrect values specified for the limits. For more information on Solaris as managed server refer to the IBM information center at:

```
http://publib.boulder.ibm.com/eserver/
```

2.7 Uninstalling

This section describes the procedure for uninstalling the product. When you copied the EWLM components to your system using the IBM Virtualization Engine, an uninstaller suite was copied to the directory where the VE was installed and launched. You can use the uninstall wizard to remove the domain manager, the EWLM Control Center, the managed server images and the required runtime components from your systems running on AIX, Linux, and Windows. By using this uninstaller, all the VE components will be removed—not just the EWLM code.

Uninstalling EWLM on Linux, AIX, and Windows

Verify that your domain manager and EWLM Control Center is stopped. To uninstall EWLM, run the command sequence on your Linux domain manager as shown in Example 2-14. For domain manager on other operating systems go to the installation directory you have chosen. The default directory for AIX is as shown for Linux. For Windows go to C:\Program Files\IBM\VE and run **`uninstallVEWin.exe`**. We discuss i5/OS separately because it does not have the uninstall wizard.

Example 2-14 Uninstalling EWLM on Linux

```
# cd /opt/IBM/VE
# ./uninstallVELinux.bin
```

This command starts the uninstall wizard, which allows you to select the components you want to uninstall. Verify that the directory /opt/IBM/VE is deleted from your system. If not, you can do it manually after the uninstall wizard has finished.

Uninstalling EWLM on i5/OS

To uninstall the EWLM systems services from an i5/OS server you need to perform the following steps:

- Log in to i5/OS domain manager
- Enter **`DSPSFWRSC`** to display all software resources which you have installed. Look for resource ID 5733VE1.
  - If the display indicates that only *BASE, 40 (EWLM base support), 41 (domain manager) or 43 (single-system) are installed, run the following command to uninstall the managed servers and firewall brokers:
    `DLTLICPGM LICPGM(5733VE1) OPTION(*ALL)`
  - If the display indicates that other options are installed, run the following commands to uninstall domain manager or single-system EWLM:
    `DLTLICPGM LICPGM(5733VE1) OPTION(41) or DLTLICPGM LICPGM(5733VE1) OPTION(43)`
  - If the display indicates that option 42 (managed server) or option 44 (firewall broker) are installed, you must run the following command to remove option 40 (EWLM base support):
    `DLTLICPGM LICPGM(5733VE1) OPTION(40)`

Uninstalling managed servers and firewall brokers

The uninstall wizard running on the domain manager on AIX, Linux, and Windows does not uninstall the managed servers or firewall brokers. Log on to your managed server and verify that the managed server and firewall broker are stopped. Go to directory /opt/IBM/VE/EWLM/_uninst or C:\Program Files\IBM\VE\EWLM_uninst and run the uninstaller. For AIX and Solaris it is called **`uninstall.bin`**; for Windows it is called **`uninstall.exe`**. Follow the steps of the uninstall wizard to remove the managed server and firewall broker from the system. Repeat this step for each managed server you want to uninstall.

3

Enabling middleware for EWLM

In this chapter we describe procedures to enable the supported IBM middleware, thereby providing an end-to-end view of transaction flow at the EWLM Control Center. We also describe how to configure the J2EE application Trade3 to work with EWLM. (In Chapter 5 this application is used to demonstrate how the EWLM domain manager works with a workload.)

The specific middleware covered in this chapter is:

- DB2 Universal Database V8.2
- WebSphere Application Server V5.1.1
- IBM HTTP Server V1.3.28, V2.0.47
- Trade3 application

3.1 Overview

To construct data that can be displayed by the EWLM domain manager, each middleware and application that you want EWLM to monitor must be instrumented according to the ARM 4.0 standards. An instrumented middleware 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:

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

The middleware 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 3-1 provides the default variables that we are using so you can compare them with the values used in your installation if you are not using the default application installation directories.

Table 3-1 Environment variables

Variable	Component	Platform	Value
<WAS_HOME>	WebSphere Application Server	AIX	/usr/WebSphere/AppServer
		i5/OS	/qibm/proddata/webas51
		Windows	C:\Program Files\WebSphere\AppServer
		Solaris	/opt/WebSphere/AppServer
<EWLMMS_HOME>	EWLM managed server	AIX	/opt/IBM/VE/EWLMMS
		i5/OS	/qibm/proddata/wlm
		Windows	C:\Program Files\IBM\VE\EWLMMS
		Solaris	/opt/IBM/VE/EWLMMS
<EWLMMS_LIBPATH>	EWLM managed server ARM implementation libraries	AIX	/usr/lpp/ewlm/java/lib
		i5/OS	/qibm/proddata/wlm/classes/ms
		Windows	<EWLMMS_HOME>\classes\ms
		Solaris	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

Variable	Component	Platform	Value
<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

3.2 Pre-instrumentation tasks

Before enabling any middleware, the EWLM managed server function needs to be installed on each server operating system. In addition, the domain manager function needs to be installed on one server in the EWLM domain that will serve as the EWLM control point.

Table 3-2 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 redbook, but explain how to enable the instrumentation provided by these middleware products.

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

`http://publib.boulder.ibm.com/infocenter/ws51help/`

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

`http://publib.boulder.ibm.com/infocenter/db2help/`

Table 3-2 Pre-instrumentation tasks

Host name	ewlmdm1	ewlm1	ewlm2, ewlm3	ewlm4
Role	**Domain manager**	**Web server**	**Application server**	**Database server**
Install and configure EWLM domain manager	X[1]			
Configure EWLM Control Center	X[1]			
Create users and assign roles	X[1]			
Install and configure IBM HTTP Server V1.3.28 of V2.0.47		X		
Install and configure web server plug-in		X		
Install and configure WebSphere Application Server V5.1.1			X	
Install and configure DB2 Universal Database V8.2 or higher				X
Install EWLM managed server		X[2]	X[2]	X[2]
Create new EWLM managed server		X[2]	X[2]	X[2]
Instrument operating system		X[2]	X[2]	X[2]

Host name	ewlmdm1	ewlm1	ewlm2, ewlm3	ewlm4
Role	Domain manager	Web server	Application server	Database server
Verify operating system instrumentation		X[2]	X[2]	X[2]
Verify communication between domain manager and managed server	X[2]	X[2]	X[2]	X[2]

[1] Refer to 2.4, "EWLM configuration in our ITSO environment" on page 32, 2.5, "Installing the domain manager on other operating systems" on page 50 and 4.1, "EWLM Control Center overview" on page 78 for details.
[2] Refer to 2.4, "EWLM configuration in our ITSO environment" on page 32 and 2.6, "Configuring managed servers on other operating systems" on page 54 for details.

3.3 Enabling DB2 Universal Database for ARM

Instrumenting DB2 Universal Database on each operating system platform is done in a very similar way, but there are some differences. Therefore, we describe the steps for each platforms separately.

AIX

Instrumenting DB2 Universal Database on AIX includes these steps:

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

Example 3-1 Add ARM related capabilities to user db2inst1

```
root@ewlm4:/> chuser “capabilities = CAP_ARM_APPLICATION,CAP_PROPAGATE” db2inst1
root@ewlm4:/> lsuser db2inst1
db2inst1 id=106 pgrp=db2grp1 groups=db2grp1,staff,dasadm1 home=/home/db2inst1 sh
ell=/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 pwdwarntime=0 account_lo
cked=false minage=0 maxage=0 maxexpired=-1 minalpha=0 minother=0 mindiff=0 maxre
peats=8 minlen=0 histexpire=0 histsize=0 pwdchecks= dictionlist= capabilities=CA
P_ARM_APPLICATION,CAP_PROPAGATE fsize=-1 cpu=-1 data=491519 stack
=32767 core=-1 rss=-1 nofiles=2000 time_last_login=1086381962 tty_last_login= ho
st_last_login=ewlm4 unsuccessful_login_count=0 roles=
```

4. Verify the user’s capabilities at the /etc/security/user file as shown in Example 3-2.

Example 3-2 Verify user's capabilities

```
root@ewlm4:/> grep -p db2inst1 /etc/security/user
db2inst1:
        admin = false
        umask = 22
        capabilities = CAP_ARM_APPLICATION,CAP_PROPAGATE
```

Set DB2 variable

In order for a DB2 instance to call the ARM instrumented APIs, set DB2 variable:

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

   ```
   root@ewlm4:/> su - db2inst1
   ```

2. Set the DB2LIBPATH variable and verify it using the **`db2set`** command as shown in Example 3-3.

Example 3-3 Set the DB2LIBPATH and verify the DB2 variables on AIX

```
$ db2set DB2LIBPATH=/usr/lib
$ db2set
DB2_RR_TO_RS=YES
DB2LIBPATH=/usr/lib
DB2COMM=tcpip
DB2AUTOSTART=YES
```

3. Restart the DB2 instance as shown in Example 3-4.

Example 3-4 Restart DB2 instance on AIX

```
$ db2 force applications 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
06-07-2004 14:14:03     0   0   SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.
$ db2start
06-07-2004 14:14:07     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.

Windows

Instrumenting DB2 Universal Database on Windows includes these steps:

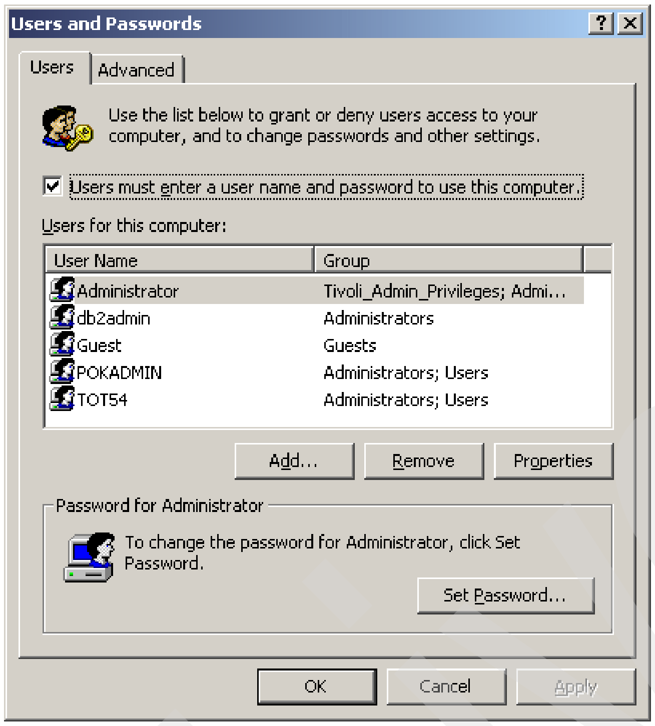
1. Verify the DB2 instance user group.
2. Set the DB2 variable.

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 DB2 installer automatically, but you have to check it using following sequence:

1. Click **Start** → **Setting** → **Control Panel** → **Users and Password**.

2. Verify `db2admin` belongs to the `Administrators` group as shown in Figure 3-1.



Use the list below to grant or deny users access to your computer, and to change passwords and other settings.

☑ Users must enter a user name and password to use this computer.

Users for this computer:

User Name	Group
Administrator	Tivoli_Admin_Privileges; Admi...
db2admin	Administrators
Guest	Guests
POKADMIN	Administrators; Users
TOT54	Administrators; Users

Password for Administrator

To change the password for Administrator, click Set Password.

Figure 3-1 db2admin group membership

Verify DB2 variables

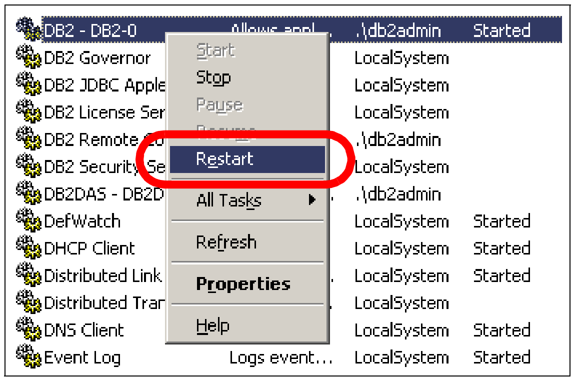
ARM is enabled on DB2 UDB by default, so the DB2 instance should be able to register to ARM. You might still want to verify that the following DB2 variables are set properly and if not, then they need to be set to the correct values:

1. Log in as the DB2 instance owner `db2admin`.
2. Verify using the **`db2set`** command. Click **Start** → **Run**, type `db2cmd`, click **OK,** and follow the sequence shown in Example 3-5.

Example 3-5 Set and verify DB2 variables on Windows Server 2003

```
C:\Documents and Settings\db2admin>db2set
DB2SATELLITEID=ITSOTOT54
DB2ACCOUNTNAME=TOT54\db2admin
DB2INSTOWNER=TOT54
DB2PORTRANGE=60000:60003
DB2_GRP_LOOKUP=LOCAL
DB2INSTPROF=C:\PROGRA~1\IBM\SQLLIB
DB2COMM=TCPIP
```

3. If you change any of the variables, you then need to restart the DB2 instance as illustrated in Figure 3-2.



DB2 - DB2-0	Allows appl...	.\db2admin	Started
DB2 Governor		LocalSystem	
DB2 JDBC Apple		LocalSystem	
DB2 License Ser		LocalSystem	
DB2 Remote		.\db2admin	
DB2 Security Se		LocalSystem	
DB2DAS - DB2D		.\db2admin	
DefWatch		LocalSystem	Started
DHCP Client		LocalSystem	Started
Distributed Link		LocalSystem	Started
Distributed Trar		LocalSystem	
DNS Client		LocalSystem	Started
Event Log	Logs event...	LocalSystem	Started

Figure 3-2 Restart DB2 instance on Windows Server 2003

Solaris

To instrument 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 username 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.

3.4 Enabling WebSphere Application Server for ARM

Use the following steps to instrument WebSphere Application Server:

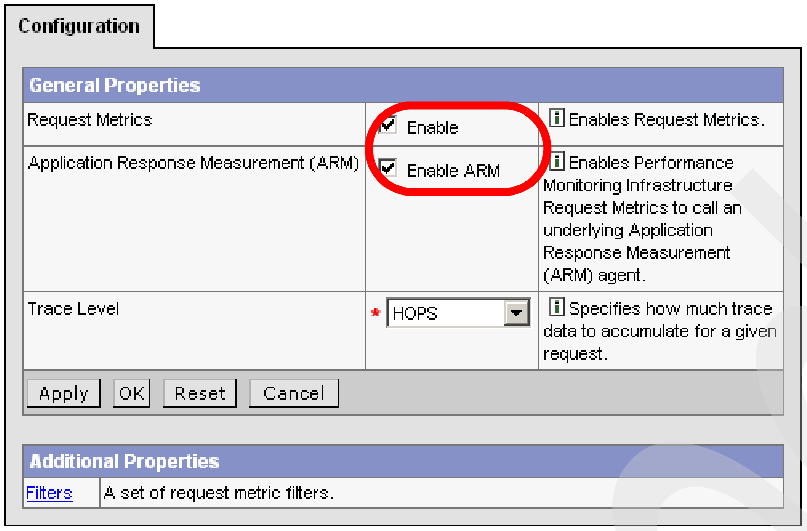
1. Enable PMI Request Metrics.
2. Create Java™ Virtual Machine custom properties.

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 3-3. Leave the `Trace Level` set to `HOPS`.


Configuration

General Properties

Request Metrics	✓ Enable	Enables Request Metrics.
Application Response Measurement (ARM)	✓ Enable ARM	Enables Performance Monitoring Infrastructure Request Metrics to call an underlying Application Response Measurement (ARM) agent.
Trace Level	* HOPS	Specifies how much trace data to accumulate for a given request.

Additional Properties

Filters	A set of request metric filters.

Figure 3-3 Enable Request Metrics and Application Response Management

4. Click **OK**.

Create Java™ Virtual Machine custom properties

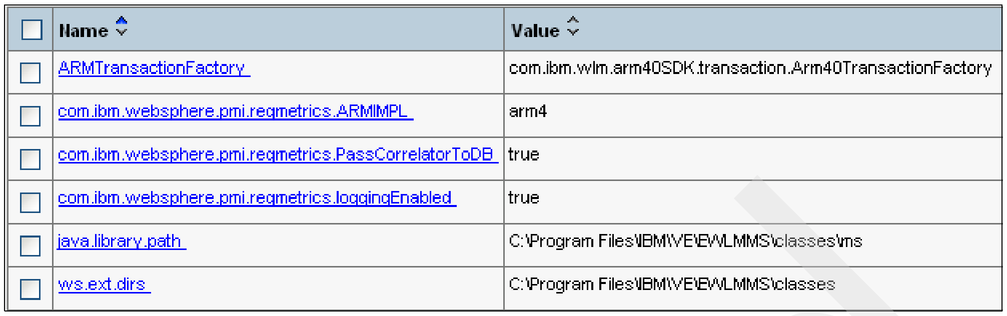
Once you enable PMI Request Metrics, you need to set the proper JVM custom properties for the type of ARM agent and the ARM transaction factory. You have to create the custom properties for all application servers you want EWLM to monitor. Also, these settings affect the WebSphere HTTP plug-in configuration.

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 3-3 are not present in the list, create them as shown in Figure 3-4.

Table 3-3 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	<EWLMMS_LIBPATH>
ws.ext.dirs	<EWLMMS_HOME>/classes



	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	C:\Program Files\IBM\VE\EWLMMS\classes\ms
☐	ws.ext.dirs	C:\Program Files\IBM\VE\EWLMMS\classes

Figure 3-4 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 3-6. Note that if this property is not created, the correlator information is not logged.

Example 3-6 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 detai
l=select  q1."ADDRESS",  q1."PASSWORD",  q1."USERID",  q1."EMAIL",  q1."CREDITCA
RD",  q1."FULLNAME" from ACCOUNTPROFILEEJB q1 where  ( q1."USERID" = ?)  for upd
ate of "ADDRESS" elapsed=6
```

4. Click **Save**.

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

5. Check **Synchronize changes with nodes** and click **Save** to save all your changes.
6. Restart the application server.

Note: If a WebSphere Application Server Network Deployment is present in your installation, set these properties for the deployment manager as well or you may see two problems:

1. The HTTP plug-in configuration regeneration cannot set the correct `loggingEnable` value in the plug-in configuration file (`plugin-cfg.xml`).
2. Future releases may not be able to pick up the property.

On the other hand, if you do not want to instrument the Network Deployment, you can edit the plugin-cfg.xml for "`loggingEnable=false`".

3.5 Enabling IBM HTTP Server for ARM

There are two instrumented plug-ins for IBM HTTP Server:

- The *WebSphere HTTP plug-in* that is provided by WebSphere installation

- The *independent plug-in* that is provided for nonWebSphere Application Server application usage

Also, there are two versions of IBM HTTP Server 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.

Instrumenting IBM HTTP Server depends on which version of IBM HTTP Server you are currently running and what type of plug-in you will use.

3.5.1 WebSphere HTTP plug-in

The WebSphere HTTP plug-in receives a user's HTTP request and determines whether the request should be handled by the web server or an application server using the plug-in configuration file, `plugin-cfg.xml`. If a request for an application server is received, the plug-in dispatches it to the appropriate web application with ARM correlator information.

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 WebSphere Application Server instrumentation is successfully completed. Perform the following steps to instrument WebSphere HTTP plug-in:

1. Install the WebSphere HTTP plug-in to IBM HTTP Server using the install wizard.
2. Apply WebSphere Application Server fix pack 1, if needed.
3. Verify the IBM HTTP Server configuration file.

   For IBM HTTP Server 1.3.28, add 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 IBM HTTP Server 2.0.47, add 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.
   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. 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. If the `loggingEnabled` parameter is `true` and `traceLevel` is set to other than `NONE`, you will see the correlator information in HTTP plug-in log file as shown in Example 3-7.

Example 3-7 The correlator information at http_plugin.log

```
[Wed May 26 19:12:48 2004] 00000658 000008b4 - PLUGIN:
parent:ver=1,ip=9.12.11.2,time=1085613001328,pid=1624,reqid=49,event=1 -
```

```
current:ver=1,ip=9.12.11.2,time=1085613001328,pid=1624,reqid=49,event=1 type=HTTP
detail=/PlantsByWebSphere/servlet/ShoppingServlet elapsed=16 bytesIn=0 bytesOut=12520
[Wed May 26 19:12:48 2004] 00000658 000008c0 - PLUGIN:
parent:ver=1,ip=9.12.11.2,time=1085613001328,pid=1624,reqid=50,event=1 -
current:ver=1,ip=9.12.11.2,time=1085613001328,pid=1624,reqid=50,event=1 type=HTTP
detail=/trade/app elapsed=0 bytesIn=0 bytesOut=3057
[Wed May 26 19:12:48 2004] 00000658 000008bc - PLUGIN:
parent:ver=1,ip=9.12.11.2,time=1085613001328,pid=1624,reqid=51,event=1 -
current:ver=1,ip=9.12.11.2,time=1085613001328,pid=1624,reqid=51,event=1 type=HTTP
detail=/PlantsByWebSphere/servlet/ShoppingServlet elapsed=16 bytesIn=0 bytesOut=12520
```

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"
```

Copy the file to this path on the IBM HTTP Server machine.

5. Restart the IBM HTTP Server.

3.5.2 Independent HTTP plug-ins

IBM HTTP Server plug-in

The independent plug-in is only available on IBM HTTP Server V2.0.47 at this point, and is a plug-in to capture requests that do not go through WebSphere Application Server HTTP plug-in. If you have Tomcat, JBoss, or CGI, you can use the independent plug-in to monitor their performance. 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 in your application flow.

We do not describe how to use the independent plug-in in this redbook.

Perform the following steps to instrument the independent IBM HTTP Server plug-in:

1. Copy the plug-in module mod_arm4_ap20.so to <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 IIS plug-in for EWLM in 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, the initial 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: EWLMIISFilter.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 use 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 TTPS traffic.

If you are running IIS 6.0, the following are additional steps to allow authentication:

9. 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**.
10. Right-click on the newly created **Application Pool** → **Properties** option. Select the **Identity** tab.
11. 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**.
12. 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.

13. Enable *anonymous* access and enter the built-in account for anonymous access: (IUSR_%COMPUTER_NAME%). Click **OK**.
14. Select the **Home Directory** tab on the Properties sheet. At the bottom, select the Application pool that you just created, and click **OK**.
15. Select **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Computer Management** → **Local Users and Groups** → **Groups** → **eWLMArm4Users** → **Add** → **Advanced** → **Find Now** → **SERVICE** → **OK**.
16. Restart the IIS plug-in after the updates have been made.

3.6 Verifying application enablement

Each platform has its own verification command.

AIX

You can verify that instrumentation has been enabled using the **`lsarm`** command:

```
$ lsarm -a
```

The command may print the following 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 job in question. If it has ARM calls, it will have QPMWARM4 activation group active.

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 3-8.

Example 3-8 ewlmWinAdTool output

```
cd /d "%EWLMMS_HOME%\classes\ms"
C:\Program Files\IBM\VE\EWLMMS\classes\ms>ewlmWinAdTool ARM4PsInfo

Windows Platform Administrative and Diagnostic Tool for EWLM
      Ver. 1.00.0130.4230 *
      EWLM V1R1.0 (C) Copyright IBM Corporation, 2004.

>>> ARM4PsInfo: 3 process(es) is doing ARM instrumentation...

       pid +|-           gupid(hex)         #ARs #TRs #AIs  #TIs    #TBs
  -------- --- --------------------------   ---- ---- ----  ------- -------
      1600 [+] 8C096030.01C44D9B.00000640 : 1/64 1/64 1/256 0/16384 0/16384
        AR[00]: E39D49C700000000h (IBM Webserving Plugin)
        TR[00]: E39D4A0F00000000h (WebRequest)
          .APP: (IBM Webserving Plugin) E39D49C700000000h
        AI[00]: D794770C00000000h (aiid=D794770Ch, aeid=BFE900C800000000h)
          .GRP: (IBM_HTTP_Server/2.0.47 Apache/2.0.47 (Win32))
          .INS: (TOT54/PID=      1600)
          .APP: (IBM Webserving Plugin) E39D49C700000000h
```

```
     pid +|-          gupid(hex)         #ARs #TRs #AIs  #TIs    #TBs
-------- --- --------------------------   ---- ---- ----  ------- -------
    1316 [+] E22AC5A0.01C44DA1.00000524 : 2/64 2/64 2/256 0/16384 0/16384
      AR[00]: E2D1578000000000h (IBM DB2 Universal Database)
      AR[01]: 0000000000000000h (N/A)
      TR[00]: E2D157C800000000h (SQL)
        .APP: (IBM DB2 Universal Database) E2D1578000000000h
      TR[01]: E2D1578100000001h (IBM DB2 Universal Database)
        .APP: (N/A) 0000000000000000h
      AI[00]: D794770E00000000h (aiid=D794770Eh, aeid=BFE900CA00000002h)
        .GRP: (DB2)
        .INS: (5)
        .APP: (IBM DB2 Universal Database) E2D1578000000000h
      AI[01]: D794770F00000001h (aiid=D794770Fh, aeid=BFE900CA00000002h)
        .GRP: (DB2)
        .INS: (6)
        .APP: (IBM DB2 Universal Database) E2D1578100000001h

     pid +|-          gupid(hex)         #ARs #TRs #AIs  #TIs    #TBs
-------- --- --------------------------   ---- ---- ----  ------- -------
    3176 [+] 5D0958A0.01C44DA7.00000C68 : 1/64 2/64 1/256 0/16384 0/16384
      AR[00]: E3644F5C00000000h (WebSphere)
      TR[00]: E3644FA400000000h (EJB)
        .APP: (WebSphere) E3644F5C00000000h
      TR[01]: E3644F5C00000000h (WebSphere)
        .APP: (IBM DB2 Universal Database) E2D1578100000001h
      AI[00]: D794771000000000h (aiid=D7947710h, aeid=BFE900CB00000003h)
        .GRP: (TOT54)
        .INS: (TOT54.server1)
        .APP: (WebSphere) E3644F5C00000000h

     pid +|-          gupid(hex)         #ARs #TRs #AIs  #TIs    #TBs
-------- --- --------------------------   ---- ---- ----  ------- -------
    1640 [+] 85E07D60.01C44E53.00000668 : 1/64 1/64 1/256 0/16384 0/16384
      AR[00]: E3A292AE00000000h (Apache HTTP Server v2.0.42 test)
      TR[00]: E3A292F600000000h (HTTP Request)
        .APP: (Apache HTTP Server v2.0.42) E3A292AE00000000h
      AI[00]: DC20EA9400000000h (aiid=DC20EA94h, aeid=BFE32AA200000003h)
        .GRP: (IBM_HTTP_Server/2.0.47 Apache/2.0.47 (Win32))
        .INS: (TOT54/PID=1640)
        .APP: (Apache HTTP Server v2.0.42) E3A292AE00000000h
```

Solaris

You can verify instrumentation using the **lsarm** command:

```
$ lsarm -a
```

The command may print the following ARM registered applications:

```
APPL: IBM DB2 Universal Database
APPL: WebSphere
APPL: IBM Webserving Plugin
APPL: Apache HTTP Server
```

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 following sequence of commands:

```
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 something similar to Example 3-9.

Example 3-9   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 knows about ARM functions.

3.7 Running Trade3 and Plants in the ITSO environment

Trade3 is the third generation of WebSphere end-to-end benchmark and performance sample application. The new Trade3 benchmark models an online stock brokerage application and has been re-designed and developed to cover WebSphere's significantly expanding programming model. This simulates a real-world workload driving WebSphere's implementation of J2EE 1.3, Web Services including key WebSphere performance components and features.

You can download the Trade3 application from:

`http://www.ibm.com/software/webservers/appserv/benchmark3.html`

Refer to *DB2 UDB V8 and WebSphere V5 Performance Tuning and Operation Guide*, SG24-7068 for more information on deploying Trade3 to WebSphere Application Server. Using the instructions contained in the Trade3 Readme.html file, we deployed the Trade3 application into the ITSO environment. Note that you have to install the following features of WebSphere Application Server to configure and deploy the Trade3 application:

- Administration
  - Scripted Administration
- Embedded Messaging
- Ant and Deployment Tools
  - Deploy Tool
  - Ant Utilities

The *Plants by WebSphere* application demonstrates several J2EE functions using an online store that specializes in plant and garden tool sales, and is included with WebSphere

Application Server. It uses EJB, Servlet, JSP, and the Cloudscape™ database. You have to install the following feature to run the Plants by WebSphere application on your application server:

- Application Server
  - Application Server samples

Note: The Trade3 and other application server samples are for demonstration purposes only. The code provided is not intended to run in a secure production environment.

3.7.1 Configure Trade3 JDBC driver

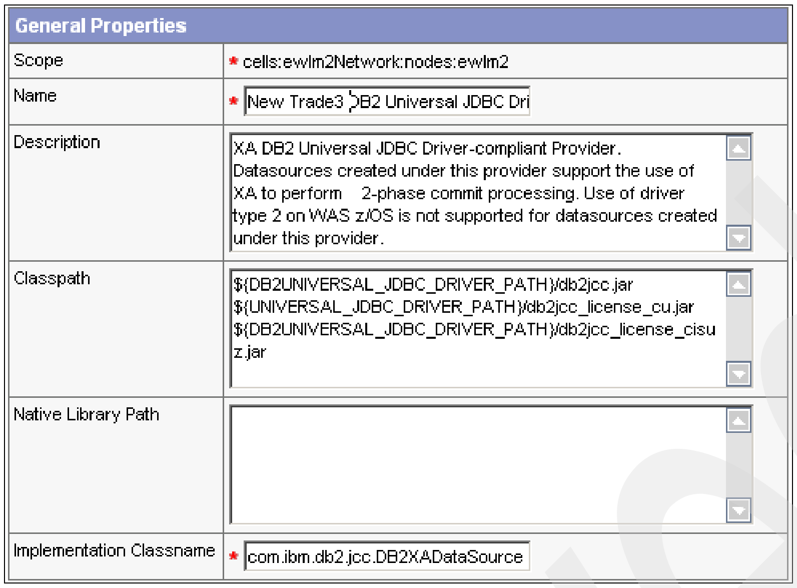
Trade3 uses the DB2 JDBC driver, db2java.zip, by default. This driver does not propagate the ARM correlator to the next hop; therefore, we modify Trade3 to use the JCC driver which pass the ARM correlator to the database server. The JCC driver, db2jcc.jar, is the DB2 Universal JDBC driver which supports both JDBC type 2 and type 4 mode and is shipped as part of DB2 Universal Database V8.2.

You have to install JCC driver V2.3.42 or higher and configure the data source in order for an application using db2java.zip as its JDBC driver to work with EWLM. For your reference, we provide the procedure required to modify the Trade3 data source configuration to use the JCC driver:

1. Change the current data source JNDI name.
   a. Log in to the WebSphere Administrative Console.
   b. Click **Resources** → **JDBC Providers**.
   c. Select **Node** and click **Apply**.
   d. Select **DB2 Universal JDBC Driver Provider (XA)** → **Data Sources** → **DB2 Universal JDBC Driver XA DataSource**.
   e. Change the current JNDI name from jdbc/TradeDataSource to another to prevent Trade3 uses this old data source. For example, change the name to `jdbc/dummy`.
   f. Click **Apply** or **OK**.
2. Create a new JDBC provider and data source.
   a. Click **Resources** → **JDBC Providers** again.
   b. Select **Scope** and click **Apply**.
   c. Click **New** to create a new JDBC provider.
   d. Select **DB2 Universal JDBC Driver Provider (XA)** from the drop-down list and click **OK**.
   e. The settings page for the new JDBC provider appears. Update the Classpath properties with the actual path of your JCC driver jar files, or alternatively, set the value of those variables as shown in Table 3-4 and Figure 3-5, and click **OK**. Leave the other properties as they are.

Table 3-4 JDBC Provider properties for Trade3

Property name	Value
Name	New Trade3 DB2 Universal JDBC Driver Provider
Classpath	${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar ${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar



General Properties	
Scope	* cells:ewlm2Network:nodes:ewlm2
Name	* New Trade3 DB2 Universal JDBC Dri
Description	XA DB2 Universal JDBC Driver-compliant Provider. Datasources created under this provider support the use of XA to perform 2-phase commit processing. Use of driver type 2 on WAS z/OS is not supported for datasources created under this provider.
Classpath	${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar ${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar
Native Library Path	
Implementation Classname	* com.ibm.db2.jcc.DB2XADataSource

Figure 3-5 Create JDBC provider

f. Click **Apply**.

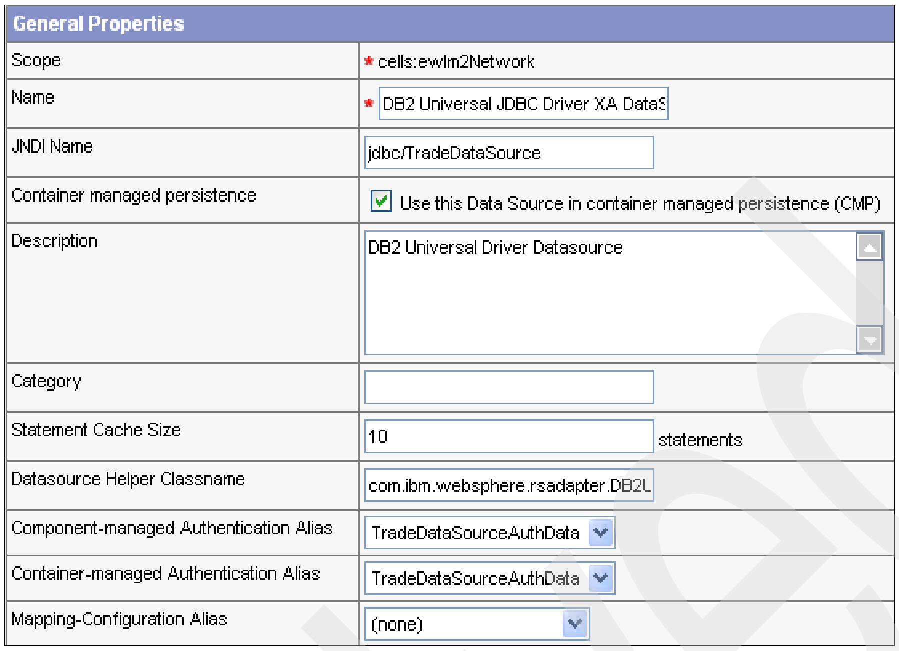
g. Select **Data Sources**.

h. Click **New** to create a new data source within the JDBC provider.

i. Enter properties as shown in Table 3-5 and Figure 3-6, and click **Apply**. Leave other properties as is.

Table 3-5 Data source properties for Trade3

Property name	Value
Name	TradeDataSource
JNDI name	jdbc/TradeDataSource
Description	Trade3 Datasource
Component-managed Authentication Alias	TradeDataSourceAuthData
Container-managed Authentication Alias	TradeDataSourceAuthData



General Properties	
Scope	* cells:ewlm2Network
Name	* DB2 Universal JDBC Driver XA DataS
JNDI Name	jdbc/TradeDataSource
Container managed persistence	✓ Use this Data Source in container managed persistence (CMP)
Description	DB2 Universal Driver Datasource
Category	
Statement Cache Size	10 statements
Datasource Helper Classname	com.ibm.websphere.rsadapter.DB2L
Component-managed Authentication Alias	TradeDataSourceAuthData
Container-managed Authentication Alias	TradeDataSourceAuthData
Mapping-Configuration Alias	(none)

Figure 3-6 Create data source

j. Select **Custom Properties**.

k. Change custom properties as shown in Table 3-6.

Table 3-6 TradeDataSource custom properties

Property name	Value
databaseName	trade3db
driverType	4
serverName	ewlm4.itso.ibm.com
portNumber	50000

l. Click **Save**.

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

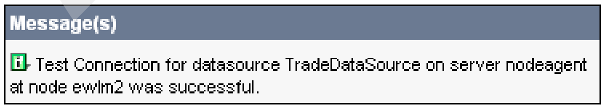
3. Test the data source connection.

a. Click **Resources** → **JDBC Providers**.

b. Select **Scope** and click **Apply**.

c. Select **New Trade3 DB2 Universal JDBC Driver Provider** → **Data Sources**.

d. Check **TradeDataSource** and click **Test Connection** to ensure connection for the datasource is successful.



Message(s)

Test Connection for datasource TradeDataSource on server nodeagent at node ewlm2 was successful.

Figure 3-7 Connection test message

e. Click **Save**.

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

4. Restart the application server.
5. Log in to Trade3. Now you can see the correlation information in the application server's SystemOut.log and IBM HTTP Server's access.log.

   If you set `com.ibm.websphere.pmi.reqmetrics.loggingEnabled` to `true` at "Enable PMI Request Metrics" on page 63, you will see the timing information in the server's SystemOut.log as shown in Example 3-6 on page 65 and Example 3-7 on page 66.

4

Administering EWLM

This chapter describes how to use the Control Center to administer EWLM. This includes:

- Overview of Control Center roles, navigation, and functions
- Defining and deploying the policies that influence EWLM operation
- EWLM monitoring and reporting capabilities

4.1 EWLM Control Center overview

The EWLM Control Center is accessed using one of three roles. These roles are Administrator, Operator, and Monitor. Administrator possess the greatest authority, followed by Operator and then Monitor. We recommend that you grant authorities using the guidelines summarized in Table 4-1.

Table 4-1 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 too. The authorities of both operator and monitor are included.

You can use the **changeCC** script to create an administrator ID and the **displayCC** script to verify the list of userids and their properties. This is shown in Example 4-1. These are domain manager commands that are keyed and entered to the OS hosting the domain manager before you can use them to log on to the Control Center.

Example 4-1 Administrator userid

```
[root@ewlmdm1 bin] - delete
# cd /opt/.../bin - include
# ./changeCC.sh -addUser /opt/EWLMDM -adminUser ibmewlm -adminPW 111111 -roleUser ewlmadm
-role Administrator

Processing changeCC -addUser request.  Please be patient as this may take a while...
...processing 33% complete
...processing 66% complete
ADDUSER THE END


[root@ewlmdm1 bin] - delete
# cd /opt/.../bin - include
# ./displayCC.sh -users /opt/EWLMDM/ -adminUser ibmewlm -adminPW 111111

Processing displayCC -users request.  Please be patient as this may take a while...
...processing 33% complete
...processing 66% complete
Role:  Administrator
Mapped Users:  ewlmadm
```

```
Role:  Operator
Mapped Users:  ewlmops
Role:  Monitor
Mapped Users:  ewlmmon
PROCESSING COMPLETE
```

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.

Use the **`changeCC`** script with `-role` field of Operator to create an operator ID.

Monitor

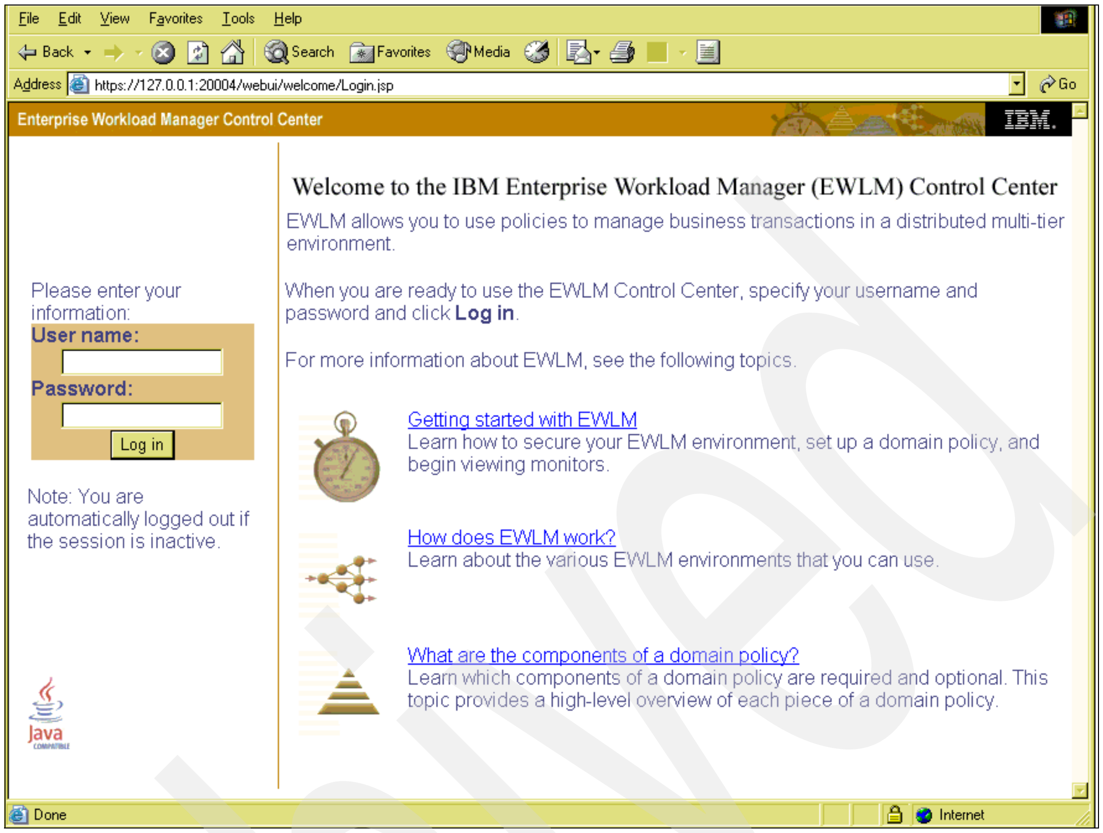
The monitor function within EWLM has read-only capability and can display most of the components within the domain represented by the domain manager.

Use the **`changeCC`** script with `-role` field of Monitor to create a monitor ID.

Once you have defined your userid, you can log in to the Control Center by specifying the following URL on your browser:

```
//<control center_ip:port>/webui
```

The Control Center can be specified by IP address or by hostname and the port is the HTTP port assigned to the Control Center during the domain manager creation. You should receive the log-in screen shown in Figure 4-1.


File Edit View Favorites Tools Help

Address https://127.0.0.1:20004/webui/welcome/Login.jsp

Enterprise Workload Manager Control Center

Please enter your information:

User name:

Password:

Log in

Note: You are automatically logged out if the session is inactive.

Welcome to the IBM Enterprise Workload Manager (EWLM) Control Center

EWLM allows you to use policies to manage business transactions in a distributed multi-tier environment.

When you are ready to use the EWLM Control Center, specify your username and password and click **Log in**.

For more information about EWLM, see the following topics.

Getting started with EWLM
Learn how to secure your EWLM environment, set up a domain policy, and begin viewing monitors.

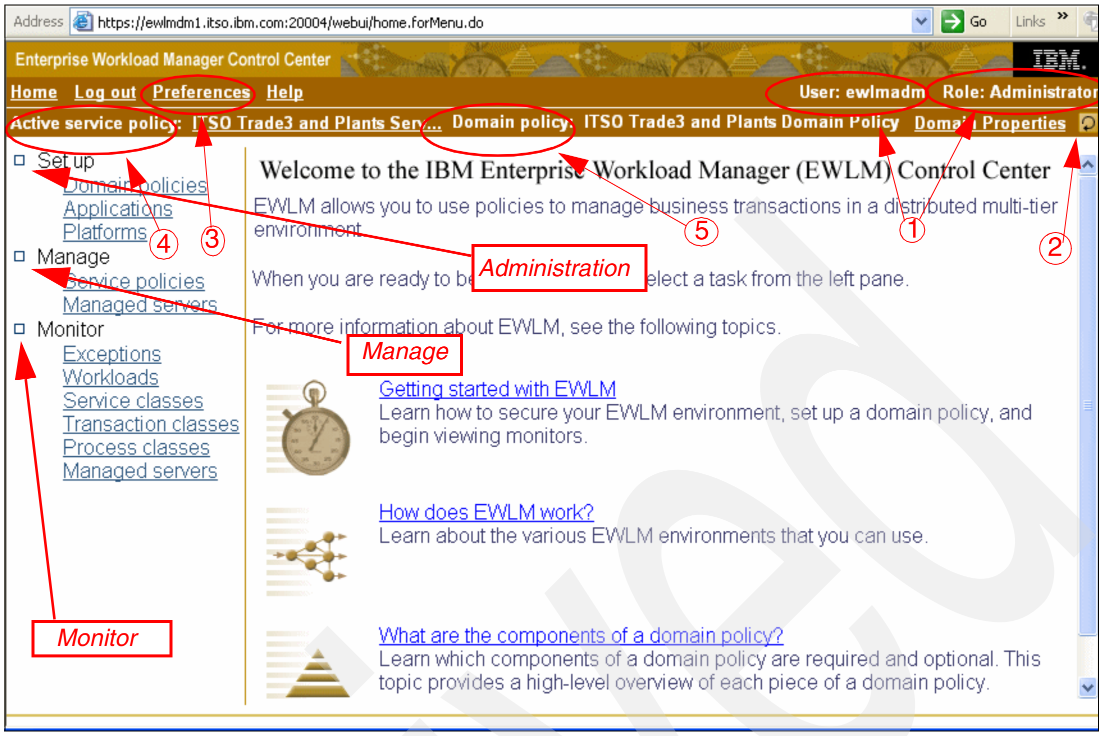
How does EWLM work?
Learn about the various EWLM environments that you can use.

What are the components of a domain policy?
Learn which components of a domain policy are required and optional. This topic provides a high-level overview of each piece of a domain policy.

Done

Figure 4-1 Log-in screen of EWLM Control Center

You can now enter the EWLM Control Center with one of the userids you created as administrator, operator, or monitor. To have full authority you need to enter as an administrative user, which in our case is user `ewlmadm`. Once you enter your userid and password, the Control Center completes the login and shows you the Welcome screen as in Figure 4-2.


Address https://ewlmdm1.itso.ibm.com:20004/webui/home.forMenu.do

Enterprise Workload Manager Control Center

Home Log out Preferences Help User: ewlmadm Role: Administrator

Active service policy: ITSO Trade3 and Plants Ser... Domain policy: ITSO Trade3 and Plants Domain Policy Domain Properties

Welcome to the IBM Enterprise Workload Manager (EWLM) Control Center

EWLM allows you to use policies to manage business transactions in a distributed multi-tier environment.

When you are ready to begin, select a task from the left pane.

For more information about EWLM, see the following topics.

Getting started with EWLM
Learn how to secure your EWLM environment, set up a domain policy, and begin viewing monitors.

How does EWLM work?
Learn about the various EWLM environments that you can use.

What are the components of a domain policy?
Learn which components of a domain policy are required and optional. This topic provides a high-level overview of each piece of a domain policy.

Figure 4-2 Welcome panel for administrative userid

The EWLM Welcome screen illustrates the capability of each function: Set up, Manage, and Monitor. These three main functions are listed on the left-hand panel. *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.

Some information that is displayed on the Welcome screen is carried forward to other screens throughout the Control Center, including the following items identified in Figure 4-2:

1. The logged-in userid (User) and the associated authority (Role).
2. The curved arrow button. When clicked, this refreshes the content of the screen.
3. Preferences - Used to change the reporting interval of the monitoring. Interval options are 1, 5, 10, 15, 30, 45 or 60 minutes.
4. The Active Service Policy Name that was activated last and is currently running.
5. The Domain Policy Name that was last deployed and is currently in use.

Note: EWLM Control Center doesn't support the use of the browser back button, and if used, it can cause unpredictable results.

The following sections provide details for the setup, management, and monitoring functions.

4.2 Setup function of the Control Center

This section describes how to set up the domain policy that defines the performance goals for the workload running in your management domain. Work running in a multi-tiered heterogeneous environment can be categorized into *transactions* and *processes*. For both transaction- and process-oriented work, EWLM provides reporting based on performance goals defined in a service policy, which is similar to a service level agreement. Before going into the details of how to set up your policies, let's get familiar with some terminology first.

You are already familiar with the concept of a *management domain*, a heterogeneous environment comprised of 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 in 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.

Domain policy

You can think of a *domain policy* as a contract which 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, service classes, and optionally platforms and process classes for an EWLM management domain. The structural view of the components is shown in Figure 4-3. In the following sections we look at each component in turn and describe the relation of these components to each other.

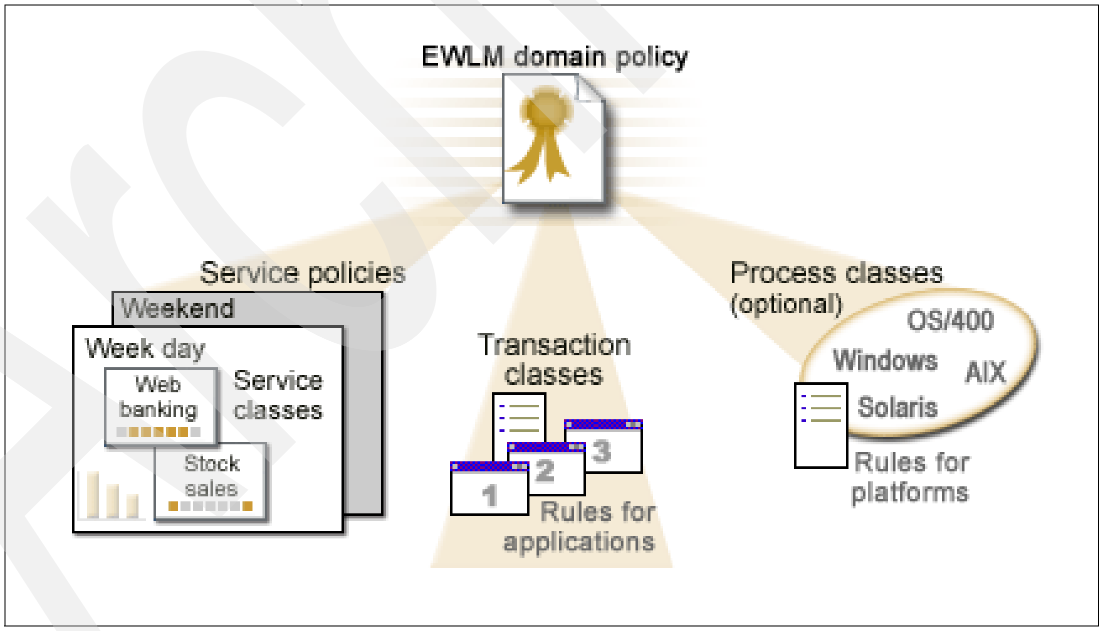


Figure 4-3 Structural view of domain policy

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. You can have several service policies that identify different performance objectives intended for different times. For example, you might have one service policy for weekday processing and another one for month-end processing. Similar to a service level agreement contract, a service policy is defined by a name and consists of service classes, transaction classes, process classes, and optionally, defined workload.

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 many cases a single service policy is sufficient.

Generally, all transactions running in the management domain will be assigned to a service class, whether a default service class for an application environment or an EWLM service class. There is only one set of transaction classes and process classes in a Domain Policy. You can assign these transaction classes and process classes to the multiple service policies within the Domain Policy.

As shown in Figure 4-4, 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.

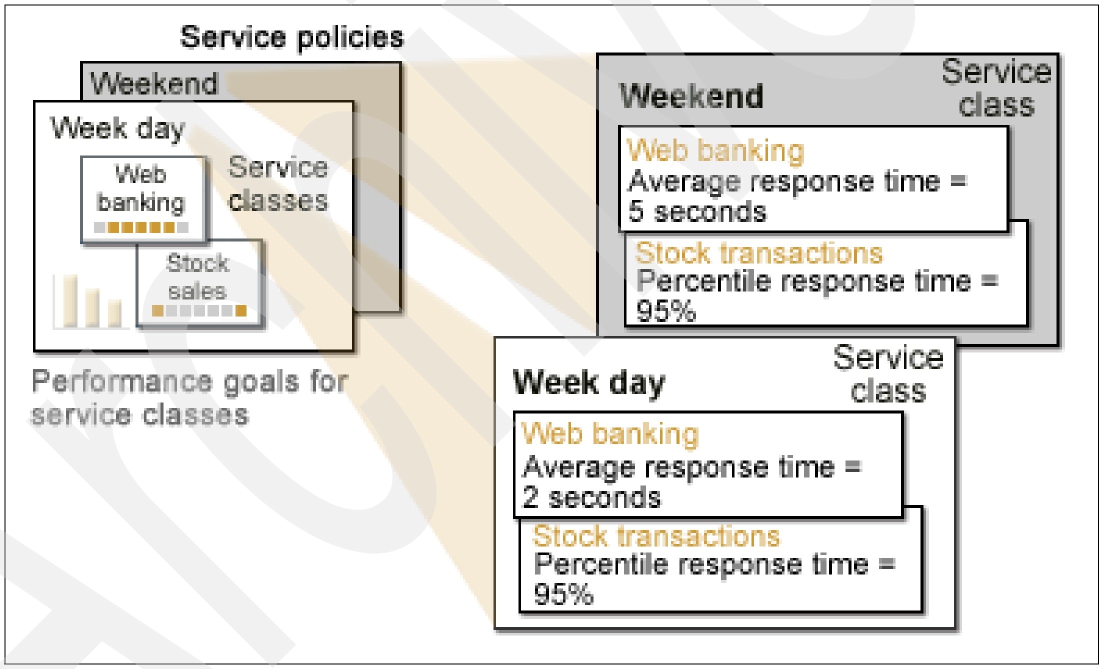


Figure 4-4 Service policy

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. This will be good positioning for later, when EWLM introduces the management functions. It is hard to be precise at this point in time, but we suggest that a good rule of thumb is to limit your service classes to around 30 for ARM transactional work. If you have hundreds of service classes in mind, you might be trying to micro-manage the environment.

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 it important from a performance point of view is the number of transaction classes active on each server. If there are lots of 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, if there are lots of transaction classes with every transaction class active on every server there might be a performance impact.

Start with simple policy and grow as required.

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 which 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.

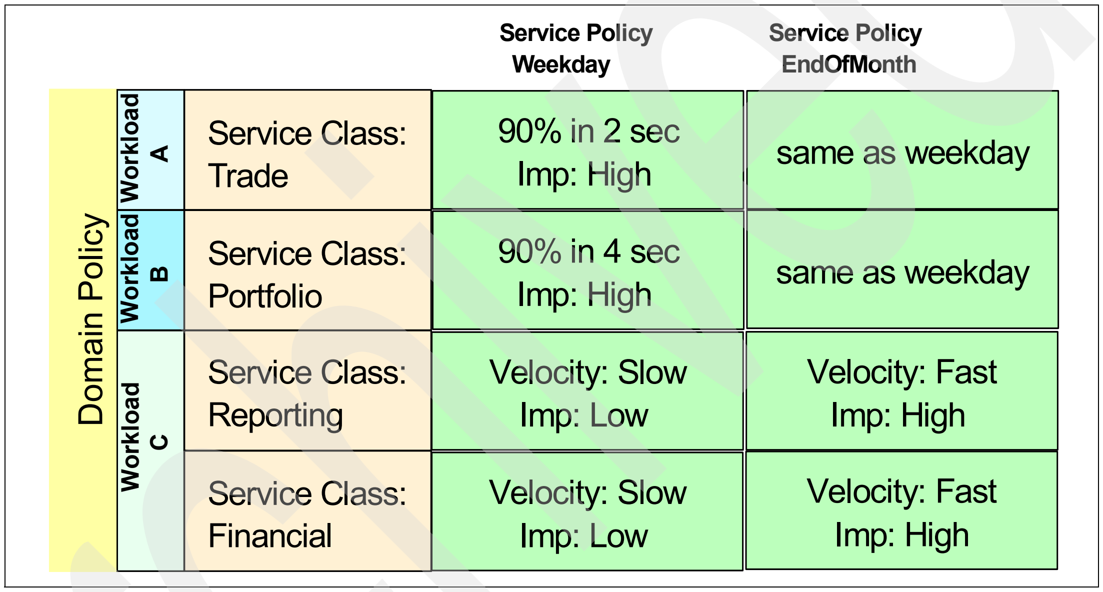
In addition, you must also specify how important it is to your business that the assigned goal is met through the *Importance* factor. Importance plays a role only when a service class is not achieving its goal. For instance, resources may be taken away from a less important service class in order for a more important service class to achieve its goal. You can specify five levels of importance settings:

- Highest
- High
- Medium
- Low
- Lowest

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.

Before continuing with further descriptions, let's consider Figure 4-5, which shows an example of how all these pieces are related to each other.



			Service Policy Weekday	Service Policy EndOfMonth
Domain Policy	Workload A	Service Class: Trade	90% in 2 sec Imp: High	same as weekday
	Workload B	Service Class: Portfolio	90% in 4 sec Imp: High	same as weekday
	Workload C	Service Class: Reporting	Velocity: Slow Imp: Low	Velocity: Fast Imp: High
		Service Class: Financial	Velocity: Slow Imp: Low	Velocity: Fast Imp: High

Figure 4-5 Relationship between the main elements and the domain policy

Given the illustrated business objective, how do we associate an individual transaction with the correct business goal (service class)? Classification describes how work is assigned to service classes. You have one set of classification definitions for a management domain. Classification consists of transaction classes and process classes. Transaction classes are grouped according to the application, while process classes are grouped by the platform.

Transaction class

Transaction classes are used to group transactions which have the same goal or which 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 which are ARM-instrumented. In future releases of EWLM there will be many more applications that are ARM-enabled. For the first release of EWLM all applications which are not ARM-instrumented can still be monitored – but as process classes rather than transaction 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. A one-to-one mapping of transaction classes to service classes is possible, as is mapping multiple transaction classes to one service class.

Process class

A *process class* is similar to a transaction class and 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, whereas 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 and a transaction 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.

Note: Non-ARM instrumented transactional and application servers have to be monitored as part of a process class.

Rules

Rules provide a way to identify and to classify the transaction 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.

The key is to configure the filtering granular enough to reflect the business goals in the classification of the transactions. This involves defining the filters at a granular enough level within both the transaction classes and process classes. The level of filtering depends on your environment. Table 4-2 gives a summary of the filter types you can use to classify process workloads for all supported operating systems of your managed servers. Following that is a list of the filter types you can use to classify transaction workloads.

Table 4-2 Process filters

Filter Name	AIX	Windows	OS400	Solaris
EWLM: ClusterName	Yes	Yes	Yes	Yes
EWLM: Hostname	Yes	Yes	Yes	Yes
EWLM: Management Domain Name	Yes	Yes	Yes	Yes
EWLM: OS Level	Yes	Yes	Yes	Yes
EWLM: OS Platform	Yes	Yes	Yes	Yes
EWLM: System Name	Yes	Yes	Yes	Yes
Executable Path	Yes	Yes	No	Yes
Username	Yes	Yes	No	Yes
Groupname	Yes	No	No	Yes

Filter Name	AIX	Windows	OS400	Solaris
WLMTag	Yes	No	No	No
Command Line Arguments	No	Yes	No	Yes
DLLList	No	Yes	No	No
Job Name	No	No	Yes	No
Job User Name	No	No	Yes	No
Effective User Profile	No	No	Yes	No
Effective Group Profile	No	No	Yes	No

Transaction filters

- EWLM: Cluster Name
- EWLM: Hostname
- EWLM: Management Domain Name
- EWLM: OS Level
- EWLM: OS Platform
- EWLM: System Name
- EWLM: Application Instance
- EWLM: Transaction Name
- EWLM: URI
- EWLM: User name
- PluginType
- ServerVersion
- Scheme
- HostInfo
- Port
- RemoteAddress
- Protocol
- QueryString
- RemoteUser
- ServerName

Application

An *application* must be ARM-instrumented in order for it to be monitored by EWLM as an application and it is the starting point for classification of transactions into transaction classes. For example, WebSphere Application Server, HTTP Server, and DB2 are application environments.

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.

Platform

The *platform* identifies the environments where process classes are supported. In this release of EWLM, AIX, Windows, and Solaris are 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.

4.2.1 Classifying your workload

To help with understanding the process of classifying work, think of it as a series of funnels that increasingly categorize the work each step of the way to the desired goal within EWLM. Figure 4-6 depicts this concept. EWLM, of course, rather than using funnels, uses filters to do the classification. As a result of the classification, the work is assigned to a service class. It is through this mechanism that EWLM can monitor the actual achievement with respect to desired performance goals.

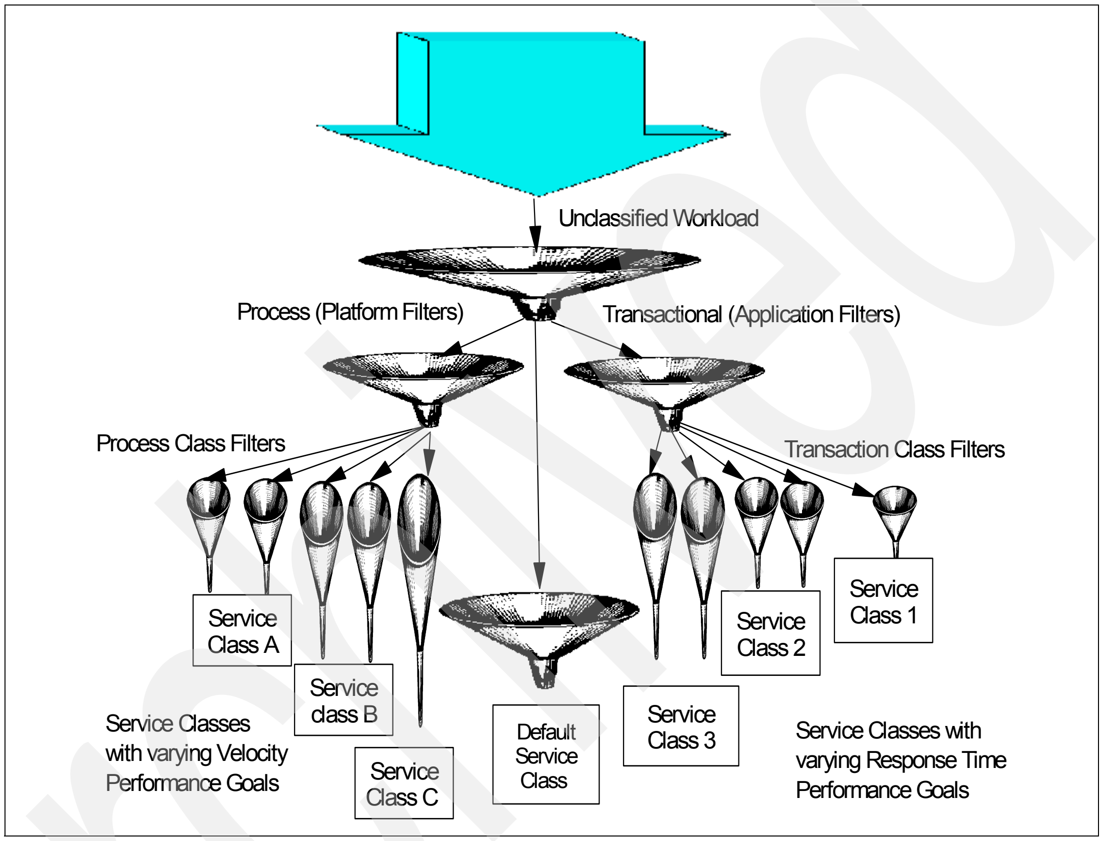

Unclassified Workload
Process (Platform Filters)
Transactional (Application Filters)
Process Class Filters
Transaction Class Filters
Service Class A
Service class B
Service Class C
Default Service Class
Service Class 3
Service Class 2
Service Class 1
Service Classes with varying Velocity Performance Goals
Service Classes with varying Response Time Performance Goals

Figure 4-6 Classification of workload

The key to classification is to configure the filtering well enough to get all workflow through the funnels at the lower level as shown in Figure 4-6. This involves defining the filters at a granular enough level within both the transaction classes and process classes. The narrower the funnel, the more classified the work has been made. How narrow the funnel is (or how granular the level of filtering), depends on the requirements of your installation. Likewise, the performance goals that are defined within the service classes should be directly related to the business needs or goals of your installation.

The funnel in the lower center tied to the default service class represents workload that goes unclassified. This is usually by accident and occurs when new workload enters the domain without being defined to EWLM, or the workload was never defined in any of the transaction or process classes originally. We recommend that you avoid this circumstance by defining within the service policy two default service classes: one for unclassified transactions and one for unclassified processes. Define each of these service classes with a name like *unknown*, *unclassified*, or *network*. Usually these service classes should not have workload assigned; so, if you notice that work has been reported within them, it means that more classification is needed. Performance goals in both of these default service classes should be set to discretionary.

Before building a domain policy, developing an accurate understanding of your workload is a fundamental step. We highly recommend that you plan time to analyze your workload: this helps ensure that the best filtering and classification can be applied and the work managed to the utmost efficiency according to the business needs within your installation. The following is a sample methodology you can follow to identify and classify your workload.

Assessment

Start the classification of your workload by identifying all the business functions within the management domain. List them in business terms rather than technical ones. Interview your business function analyst with the objective of building the list of your applications with their business goals and their level of importance to your business. If you have a formal service level agreement, that would be a good starting point. If not, the application administrators should have an idea of what is “good,” what is “bad,” and the relative importance of the business functions.

Once you build the business application list, the next step is to be able to identify each application to EWLM so that the classification of work and consequently the assignment of the performance goals can be performed. To complete this task, you will probably need to interview the application developer or the application manager to understand how the application flows and its starting point.

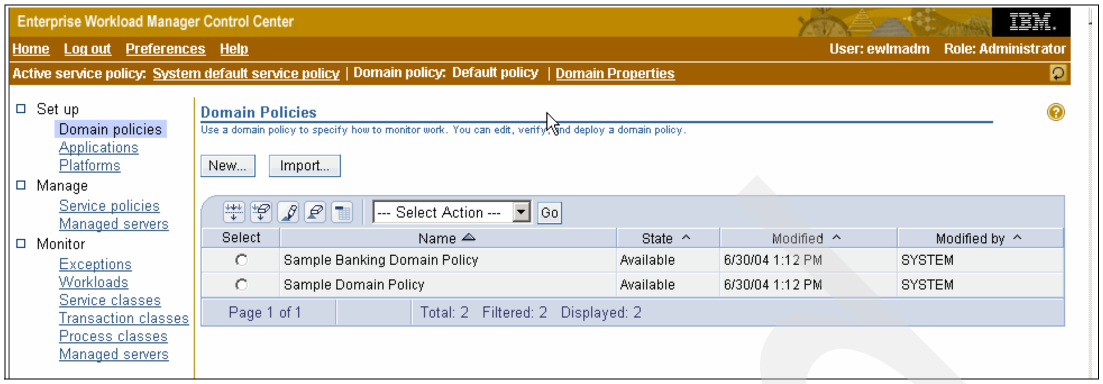
For transactions, you need to identify what the edge server is. The *edge server* is the entry point of the application into the management domain, meaning the first hop, where the classification takes place. So, for example, the Trade application consists of a set of HTTP requests that will arrive at the Web server: in this case the Web server represents the edge server where the EWLM classification takes place. To be able to recognize these HTTP requests as part of the Trade application, you need to provide some transactions filters that EWLM will use to match the incoming requests. If a match occurs, then the HTTP request belongs to the Trade application. So, once you identify your transactions through the filtering mechanism, you can associate them to a service class that implicitly means a performance goal. This goal should represent the business goal provided by your SLA.

Beside transactions, your application probably has some processes that require classification to EWLM, for example, utilities. Processes are platform specific, so for each of them you need to identify on which platform they run and how to classify them through the filtering provided by the platform. Once you have this information, with the help of the system programmer, you can assign a service class and the related performance goal to the process.

Now that you collected all the information about your applications, you are ready to define your service policy. There are different ways to proceed. We recommend that you take advantage of the sample policies EWLM provides. In the following sections, we describe the Default policy and the Sample policy. We consider these policies a good starting point, and if they do not differ a lot from your target environment, you might consider copying and adapting one of these policies to your needs. This should limit the amount of effort required to build your initial policy.

4.2.2 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. Figure 4-7 shows the default setup when you complete EWLM 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.


Enterprise Workload Manager Control Center

Home Log out Preferences Help User: ewlmadm Role: Administrator

Active service policy: System default service policy | Domain policy: Default policy | Domain Properties

Domain Policies

Use a domain policy to specify how to monitor work. You can edit, verify and deploy a domain policy.

Select	Name	State	Modified	Modified by
○	Sample Banking Domain Policy	Available	6/30/04 1:12 PM	SYSTEM
○	Sample Domain Policy	Available	6/30/04 1:12 PM	SYSTEM

Page 1 of 1 Total: 2 Filtered: 2 Displayed: 2

Figure 4-7 Default Domain policy

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’s 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 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 catchall 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) (catchall 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 though 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 4-3 describes the service classes contained in the Sample Banking Domain Service Policy.

Table 4-3 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

Service class	Importance	Performance goal
Brokerage Buyers	Highest	80% complete in 1 seconds
Stock Quotes	Medium	80% complete in 10 seconds
Static Web Serving	Low	80% complete in 20 seconds
Mortgage Rate	Medium	1 minute avg response time
Non instrumented Web Appl	High	Fast Velocity
Web Browsers	Medium	Moderate Velocity

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 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.

4.2.3 Building a domain policy

Once you have a clear idea of how to classify your workload and how to assign performance goals to each type, you are ready to build the domain policy. We describe the fundamentals in this section but provide you with an example of our own policy building in “The ITSO EWLM scenario” on page 119. This includes the planning, assessing and designing phase. We recommend that you go through both “Administering EWLM” on page 77 and “The ITSO EWLM scenario” on page 119 before starting to create your own domain policy.

There are several ways to build your policy. 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 then **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 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 in the current database.

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 **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 associated: 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, for example, `ITSO_Redbook_NormPol1`.
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 or not 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 **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 **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 **ITSO_Redbook_NormPol1**.

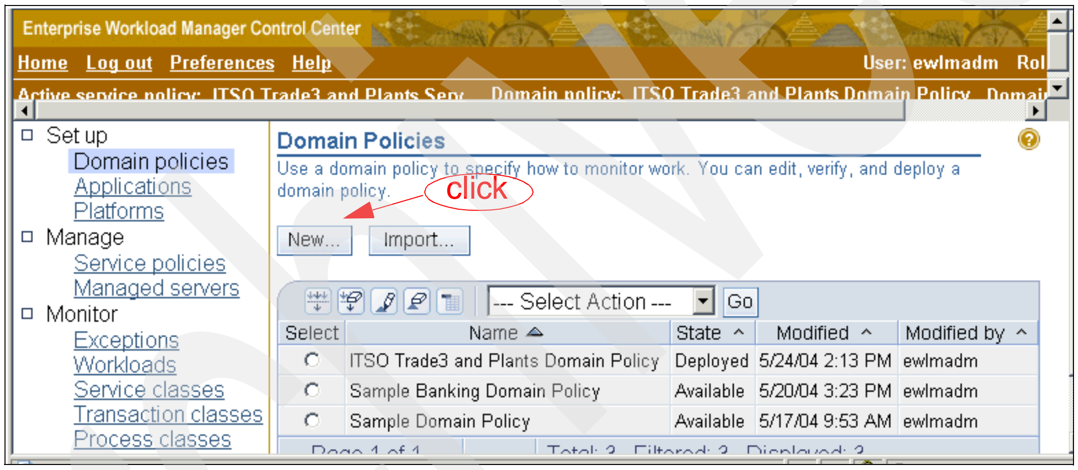
8. EWLM displays the Deploy Domain Policy and Activate Service Policy Status page. When you get to this point, you have defined, verified, and deployed a domain policy, and activated a service policy for your installation.

Deploying a domain policy can take quite a few minutes. Be prepared to wait until all servers have come up as active. Alternatively, after you make changes to your policy, you may find that it is easier 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

This section guides you through the navigation of the Control Center panels that are used to set up a new policy. We recommend that you get familiar with the navigation by reading through this section prior to your first attempt at building your own policy.

The domain policy is the anchor level defined within the domain. Therefore, the domain policy is the first item that should be 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 4-8.

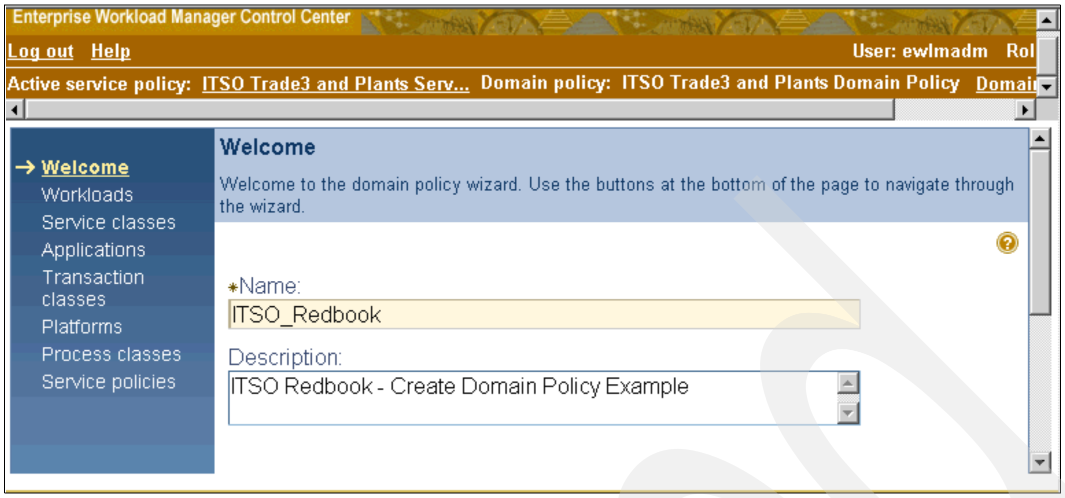


Use a domain policy to specify how to monitor work. You can edit, verify, and deploy a domain policy.

Select	Name	State	Modified	Modified by
○	ITSO Trade3 and Plants Domain Policy	Deployed	5/24/04 2:13 PM	ewlmadm
○	Sample Banking Domain Policy	Available	5/20/04 3:23 PM	ewlmadm
○	Sample Domain Policy	Available	5/17/04 9:53 AM	ewlmadm

Figure 4-8 Initiating a new Domain Policy

The Domain Policies screen lists all the policies currently defined to EWLM. Once the remaining steps here are complete, the result will be an additional name on this list. To define a new policy, click **New**. The Domain policy wizard welcome screen shown in Figure 4-9 is presented.


Enterprise Workload Manager Control Center

Log out Help User: ewlmadm Rol

Active service policy: ITSO Trade3 and Plants Serv... Domain policy: ITSO Trade3 and Plants Domain Policy Domai

Welcome

Welcome to the domain policy wizard. Use the buttons at the bottom of the page to navigate through the wizard.

*Name:
ITSO_Redbook

Description:
ITSO Redbook - Create Domain Policy Example

Figure 4-9 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 simply assigned the name `ITSO_Redbook` to the domain policy being created. Click **Next** (not shown in the figure) at the bottom of the screen to continue.

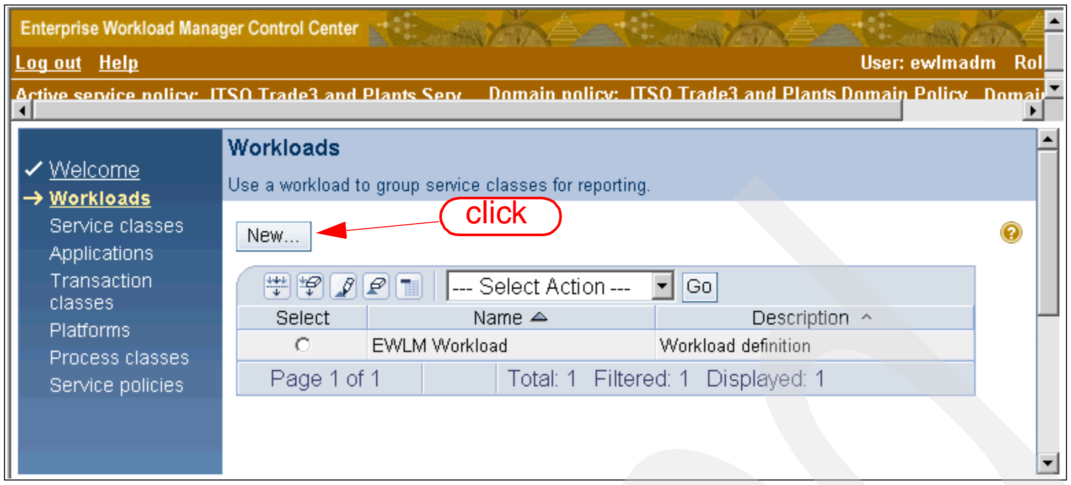
Any time you click **Next**, the wizard proceeds to the next resource to be defined until it completes the list in the left panel. For this discussion, we go through the steps to produce just the workload definition, then we skip to the end of the wizard without going through all the other resource panels.

Figure 4-10 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 you do proper planning before starting to build your policy. This includes preparing a list of service classes, mapping transaction and process classes to service classes, as well as preparing a list of platforms, applications, and workload. Once these elements are defined, the order you type them into your configuration is not important.

Defining a workload means assigning a group of service classes to it 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 4-10; the panel in Figure 4-11 is presented.


Enterprise Workload Manager Control Center

Log out Help User: ewlmadm Rol

Active service policy: ITSO Trade3 and Plants Serv Domain policy: ITSO Trade3 and Plants Domain Policy Domai

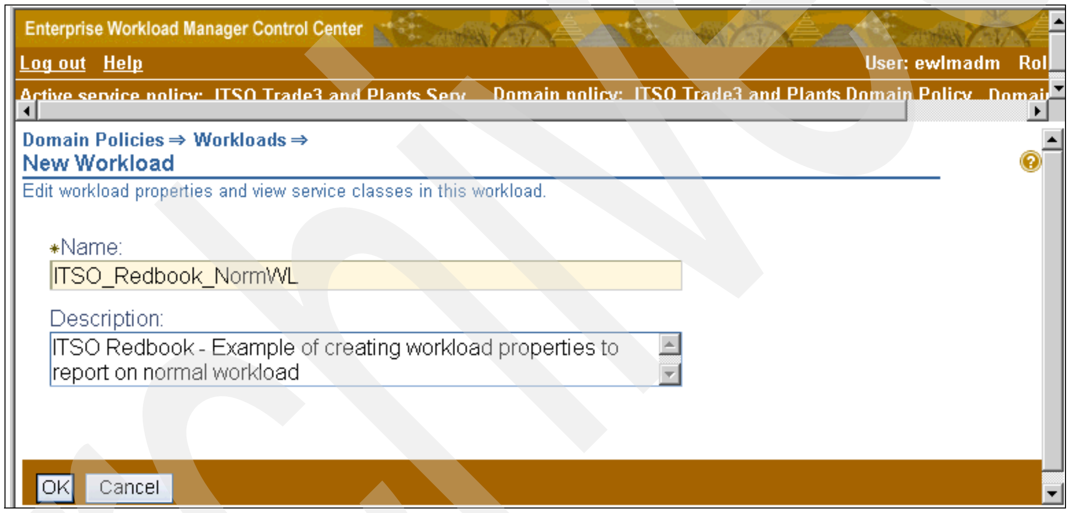
Workloads

Use a workload to group service classes for reporting.

Select	Name	Description
○	EWLM Workload	Workload definition

Page 1 of 1 Total: 1 Filtered: 1 Displayed: 1

Figure 4-10 Workload Definition


Enterprise Workload Manager Control Center

Log out Help User: ewlmadm Rol

Active service policy: ITSO Trade3 and Plants Serv Domain policy: ITSO Trade3 and Plants Domain Policy Domai

Domain Policies ⇒ Workloads ⇒

New Workload

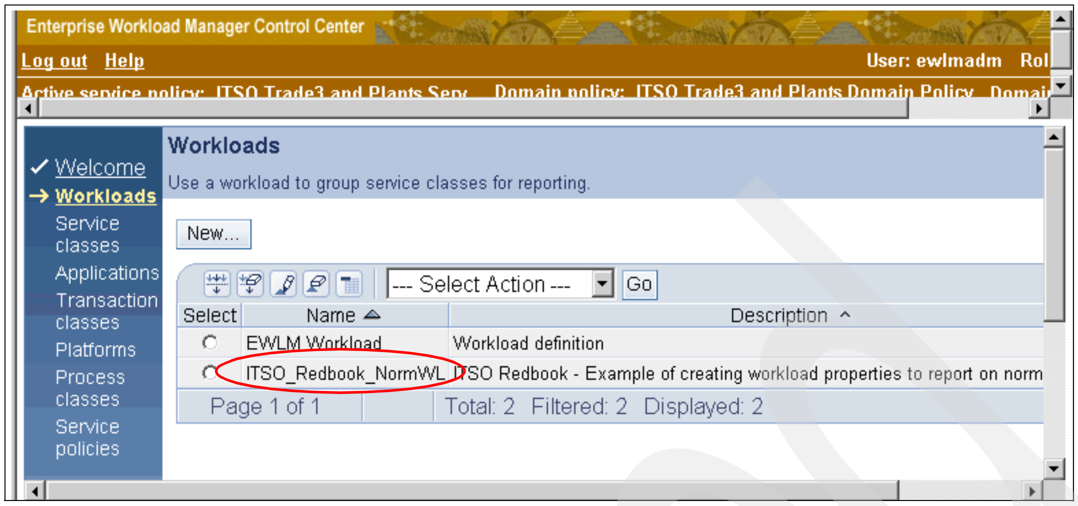
Edit workload properties and view service classes in this workload.

*Name:
ITSO_Redbook_NormWL

Description:
ITSO Redbook - Example of creating workload properties to report on normal workload

Figure 4-11 Define new workload panel

In Figure 4-11, the New Workload screen, enter the name and description to assign to this grouping of service classes. In our example, the name given to this workload was `ITSO_Redbook_NormWL`. When one or more service classes are created, they will be associated to a workload name to create the link between them. When done, click **OK** to continue.


Enterprise Workload Manager Control Center

Log out Help User: ewlmadm Rol

Active service policy: ITSO Trade3 and Plants Serv Domain policy: ITSO Trade3 and Plants Domain Policy Domai

✓ Welcome
→ Workloads
Service classes
Applications
Transaction classes
Platforms
Process classes
Service policies

Workloads

Use a workload to group service classes for reporting.

New...

--- Select Action --- Go

Select	Name	Description
○	EWLM Workload	Workload definition
○	ITSO_Redbook_NormWL	ITSO Redbook - Example of creating workload properties to report on norm

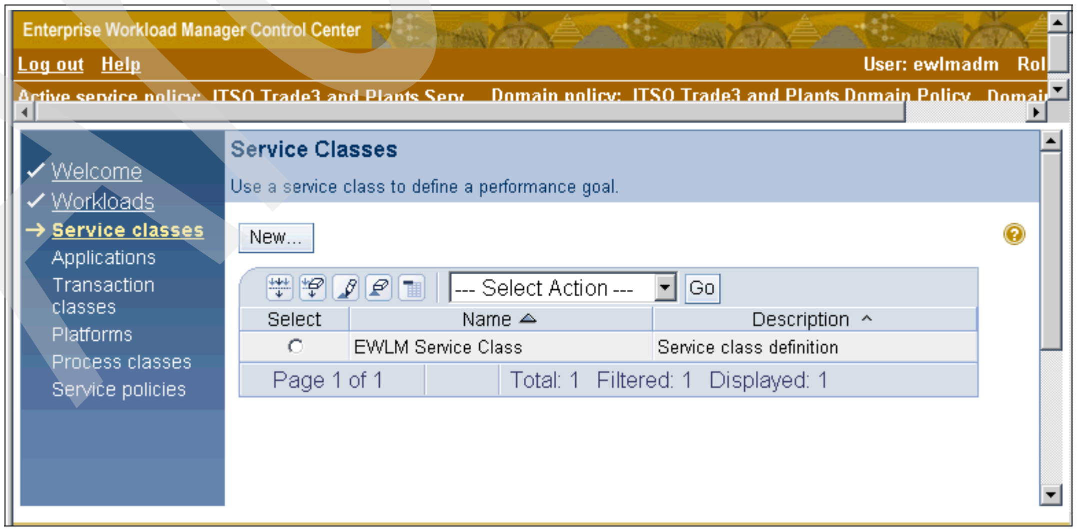
Page 1 of 1 Total: 2 Filtered: 2 Displayed: 2

Figure 4-12 Workload list

As illustrated in Figure 4-12, every time a definition is completed, the wizard shows the new definition, in this case the new workload just created. Click the **Next** button (not shown) at the bottom of the screen to continue with the next definition.

Tip: You can keep on going through the setup, defining service classes next, and so forth, but during this process your information is not saved. When using the domain policy wizard, or any other part of the Control Center for that matter, if you step away from it for awhile, a logon time-out could occur and the updates are lost. The same loss will occur if you experience a browser problem. 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 creation of 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.


Enterprise Workload Manager Control Center

Log out Help User: ewlmadm Rol

Active service policy: ITSO Trade3 and Plants Serv Domain policy: ITSO Trade3 and Plants Domain Policy Domai

✓ Welcome
✓ Workloads
→ Service classes
Applications
Transaction classes
Platforms
Process classes
Service policies

Service Classes

Use a service class to define a performance goal.

New...

--- Select Action --- Go

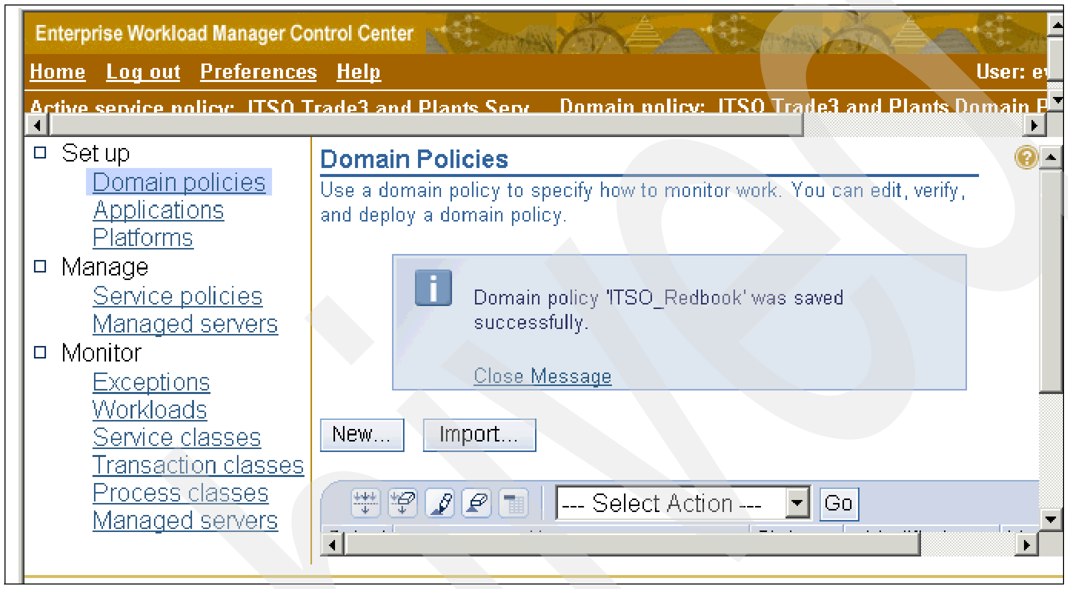
Select	Name	Description
○	EWLM Service Class	Service class definition

Page 1 of 1 Total: 1 Filtered: 1 Displayed: 1

Figure 4-13 Service classes

The screen in Figure 4-13 gives you a place to create service classes. As you can see, as you proceed in the wizard the elements that have been defined have a check mark and they are underlined. This means that you can actually select any one of them and edit the individual definition. 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.



Domain Policies

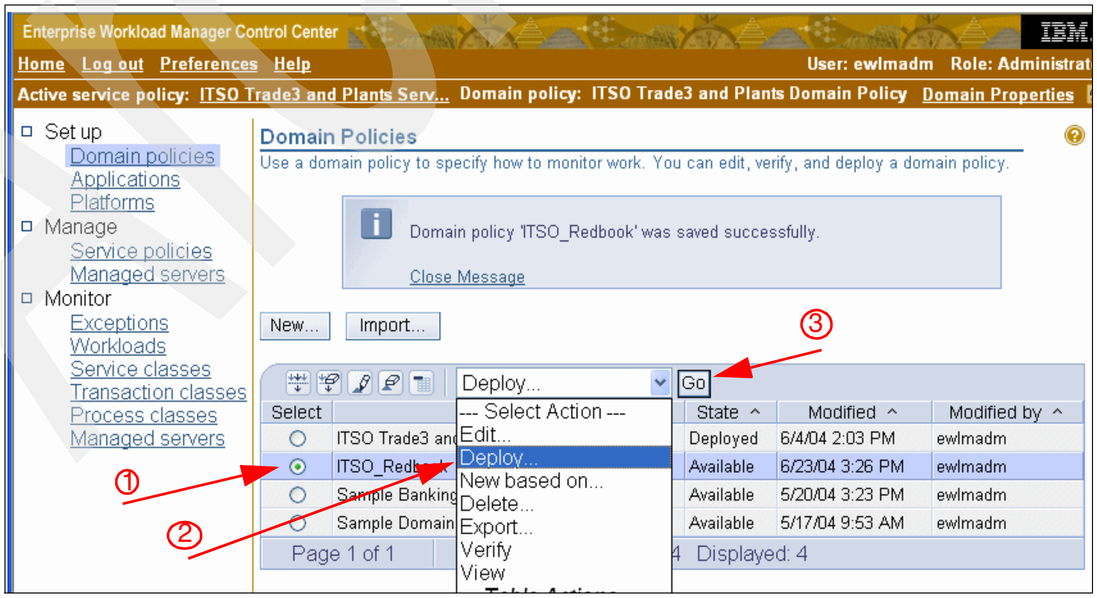
Use a domain policy to specify how to monitor work. You can edit, verify, and deploy a domain policy.

Domain policy 'ITSO_Redbook' was saved successfully.

Close Message

Figure 4-14 Saving the Domain Policy

You should receive the message displayed in Figure 4-14 as confirmation that the policy has been successfully saved. Now you are ready to deploy and activate it.



Domain Policies

Use a domain policy to specify how to monitor work. You can edit, verify, and deploy a domain policy.

Domain policy 'ITSO_Redbook' was saved successfully.

Close Message

Select		State	Modified	Modified by
○	ITSO Trade3 an	Deployed	6/4/04 2:03 PM	ewlmadm
●	ITSO_Redb	Available	6/23/04 3:26 PM	ewlmadm
○	Sample Banking	Available	5/20/04 3:23 PM	ewlmadm
○	Sample Domain	Available	5/17/04 9:53 AM	ewlmadm

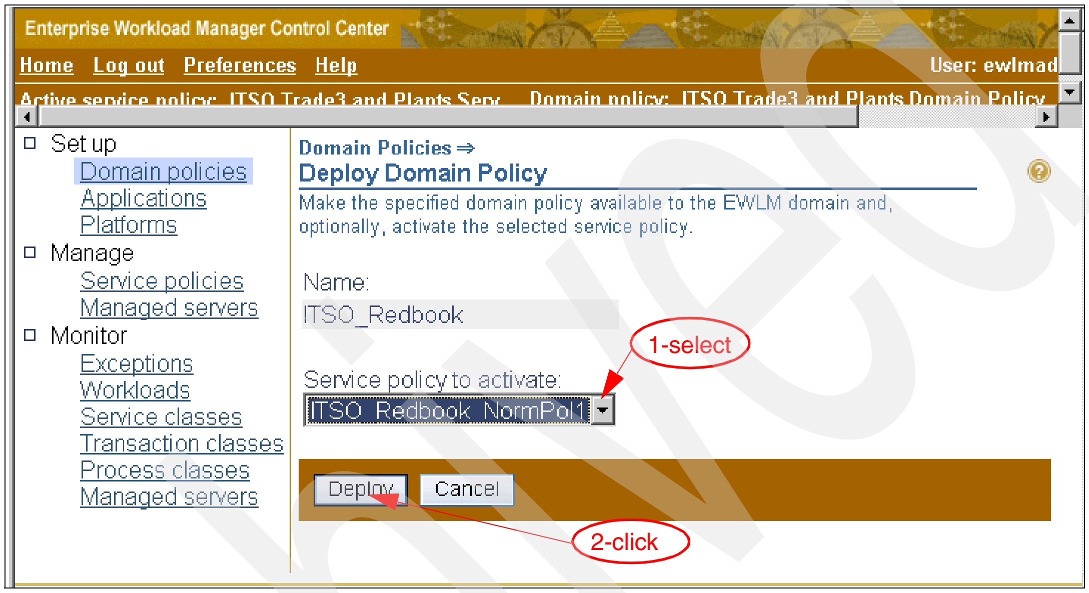
Page 1 of 1 4 Displayed: 4

Figure 4-15 Domain Policy deployment

Before EWLM can do anything based on 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 like the domain policy can be active at any one time. Figure 4-15 shows the steps to deploy a domain policy:

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 this example, the domain policy **ITSO_Redbook** was selected for deployment.



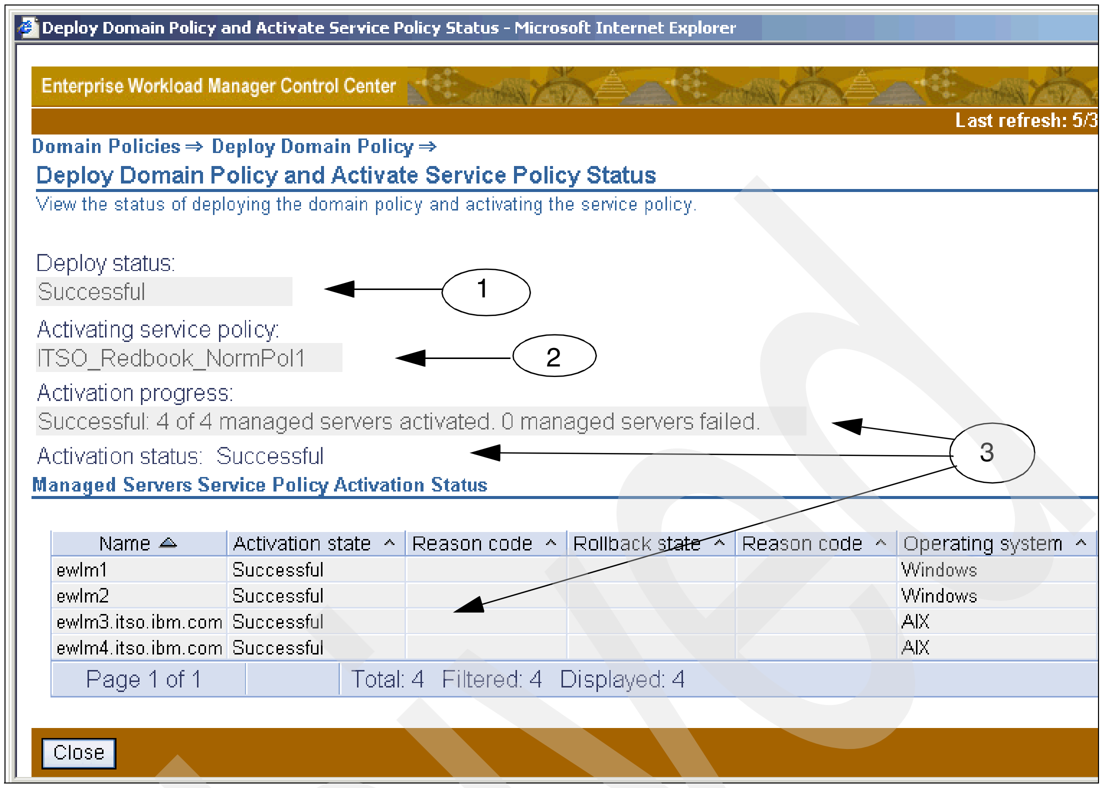
Make the specified domain policy available to the EWLM domain and, optionally, activate the selected service policy.

Name:
ITSO_Redbook

Service policy to activate:
ITSO_Redbook_NormPol1

Figure 4-16 Service policy selection

The screen in Figure 4-16 is displayed next. The drop-down menu displays all the service policies available for activation within the domain policy. In addition, it shows the option of **None**. When None is selected, activation of the service policy is done as a second step. Here, the ITSO_Redbook_NormPol1 service policy is selected to be activated once the domain policy is deployed. Click **Deploy** to deploy the domain policy and activate the selected service policy on all the managed servers.


Deploy Domain Policy and Activate Service Policy Status - Microsoft Internet Explorer

Enterprise Workload Manager Control Center

Last refresh: 5/3

Domain Policies ⇒ Deploy Domain Policy ⇒

Deploy Domain Policy and Activate Service Policy Status

View the status of deploying the domain policy and activating the service policy.

Deploy status:
Successful

Activating service policy:
ITSO_Redbook_NormPol1

Activation progress:
Successful: 4 of 4 managed servers activated. 0 managed servers failed.

Activation status: Successful

Managed Servers Service Policy Activation Status

Name	Activation state	Reason code	Rollback state	Reason code	Operating system
ewlm1	Successful				Windows
ewlm2	Successful				Windows
ewlm3.itso.ibm.com	Successful				AIX
ewlm4.itso.ibm.com	Successful				AIX

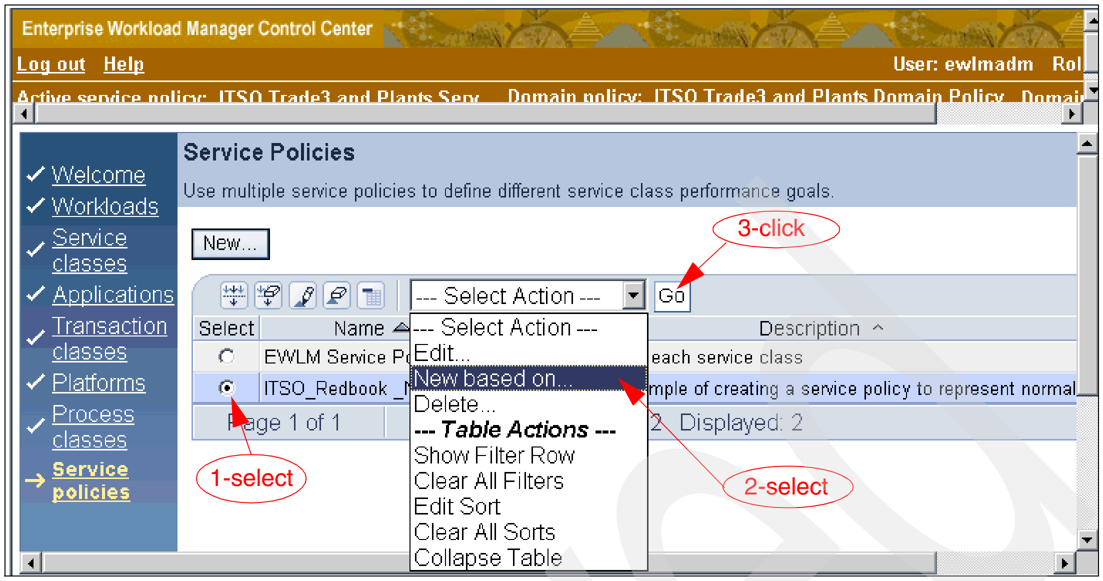
Page 1 of 1 Total: 4 Filtered: 4 Displayed: 4

Figure 4-17 Policy activation

Following deployment and activation, the screen in Figure 4-17 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 (4 of 4).

Adding a 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 classes, transaction 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 for this service policy. If you were to use the “New” function instead, the goals for all service classes would be discretionary. The method using the New based on function is shown Figure 4-18.



Enterprise Workload Manager Control Center

Log out Help User: ewlmadm Rol

Active service policy: ITSO Trade3 and Plants Serv Domain policy: ITSO Trade3 and Plants Domain Policy Domai

Service Policies

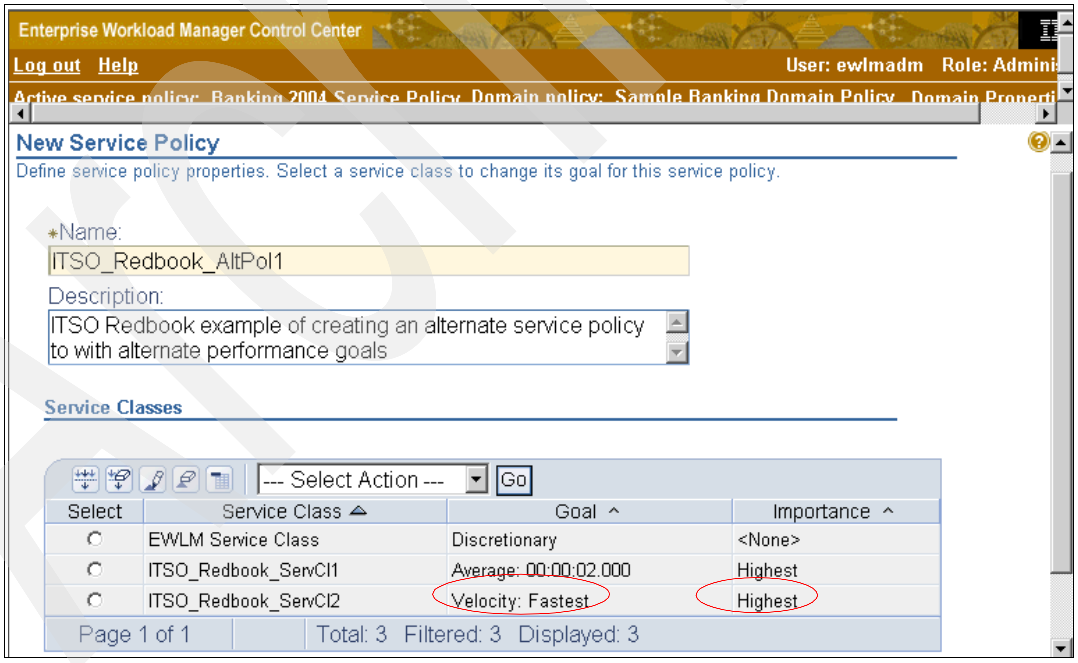
Use multiple service policies to define different service class performance goals.

Select	Name	Description
○	EWLM Service P	each service class
●	ITSO_Redbook _	mple of creating a service policy to represent normal

Page 1 of 1 2 Displayed: 2

Figure 4-18 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 policies with the exact definitions of the source one. After the new policy has been defined, 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.



Enterprise Workload Manager Control Center

Log out Help User: ewlmadm Role: Admini

Active service policy: Banking 2004 Service Policy Domain policy: Sample Banking Domain Policy Domain Properti

New Service Policy

Define service policy properties. Select a service class to change its goal for this service policy.

*Name: ITSO_Redbook_AltPol1

Description: ITSO Redbook example of creating an alternate service policy to with alternate performance goals

Service Classes

Select	Service Class	Goal	Importance
○	EWLM Service Class	Discretionary	<None>
○	ITSO_Redbook_ServCl1	Average: 00:00:02.000	Highest
○	ITSO_Redbook_ServCl2	Velocity: Fastest	Highest

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 4-19 Defining a Service policy

Figure 4-19 shows that all the service classes created in the new service policy have the same goals as the original 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.

4.3 Manage

The initial EWLM release has just a few operational functions, but this list will greatly expand in the future.

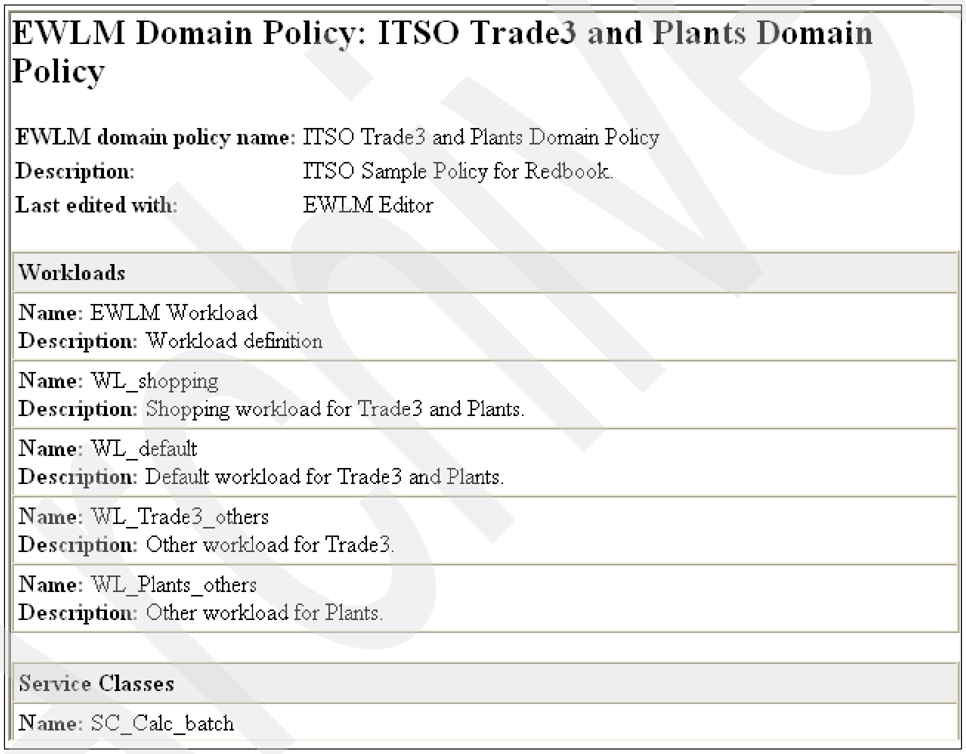
Manage is one of the functions a user with the operator role can accomplish. The Manage function provides the ability to manage the following:

- Service policies
- Managed servers

4.3.1 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 4-20 is displayed.



EWLM Domain Policy: ITSO Trade3 and Plants Domain Policy

EWLM domain policy name:	ITSO Trade3 and Plants Domain Policy
Description:	ITSO Sample Policy for Redbook.
Last edited with:	EWLM Editor

Workloads
Name: EWLM Workload **Description:** Workload definition
Name: WL_shopping **Description:** Shopping workload for Trade3 and Plants.
Name: WL_default **Description:** Default workload for Trade3 and Plants.
Name: WL_Trade3_others **Description:** Other workload for Trade3.
Name: WL_Plants_others **Description:** Other workload for Plants.

Service Classes
Name: SC_Calc_batch

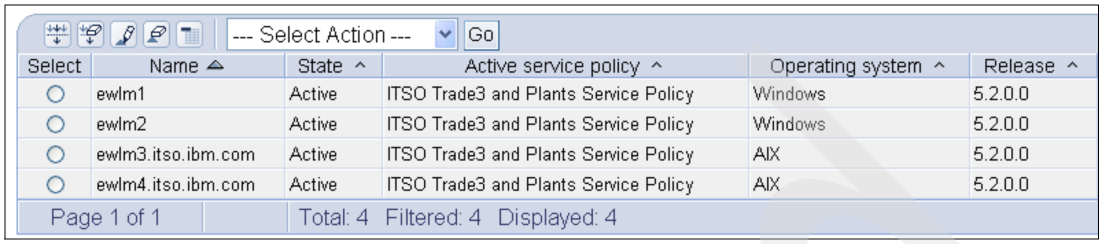
Figure 4-20 View Service policy

As you scroll through the panel all the definitions belonging to the service policy are shown: Workloads, Service Classes, Application, Transaction Classes (with their filters), Platforms and Process Classes (with filters).

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.

4.3.2 Managed servers

When you select the Managed Servers options, a summary of all the connected managed servers is shown, as in Figure 4-21.



Select	Name	State	Active service policy	Operating system	Release
○	ewlm1	Active	ITSO Trade3 and Plants Service Policy	Windows	5.2.0.0
○	ewlm2	Active	ITSO Trade3 and Plants Service Policy	Windows	5.2.0.0
○	ewlm3.itso.ibm.com	Active	ITSO Trade3 and Plants Service Policy	AIX	5.2.0.0
○	ewlm4.itso.ibm.com	Active	ITSO Trade3 and Plants Service Policy	AIX	5.2.0.0

Page 1 of 1 Total: 4 Filtered: 4 Displayed: 4

Figure 4-21 Managed servers view

From this summary, you can see the hostname 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 domain manager**: The managed server requested to join the EWLM domain. The request is initiated by the managed server, not the EWLM Control Center.
- **Join pending**: The EWLM domain manager has accepted the join request from the EWLM managed server and is waiting for acknowledgement from the managed server.
- **Active**: The server has joined the EWLM domain successfully. If there is not an active service policy, EWLM uses the default service policy.
- **Resynchronizing with domain manager**: The managed server is currently resynchronizing with domain manager to ensure that it is running with the current EWLM domain policy.
- **EWLM ARM services not enabled**: The server has joined the EWLM domain successfully but ARM services is not enabled on the managed server's platform. Therefore, the server is started but the platform's operating system is not enabled with ARM services. ARM services is required for the EWLM managed server to communicate with the server's platform. The managed server is not providing any reporting data to the domain manager.
- **Policy activation pending**: The domain manager requested that the managed server activate an EWLM service policy. The domain manager is waiting for acknowledgement from the managed server. After the domain manager receives acknowledgment, the state changes to policy activation in progress.
- **Policy activation in progress**: The managed server is activating an EWLM service policy. After the activation completes, the state changes to active.
- **Leaving domain**: The managed server requested to leave the EWLM domain. The request initiated from the managed server, not the EWLM Control Center.
- **Left domain**: The managed server is not joined to the EWLM domain.
- **Communication error**: Communication between the domain manager and managed server is disrupted. The domain manager has made several attempts to obtain the status of the server but no response has been received.
- **Incorrect policy**: The managed server is using a domain policy that differs from the deployed domain policy that is active on the EWLM domain. The reporting data from the managed server is ignored. This state occurs if a managed server fails to activate a

service policy; therefore, it continues to use the previously activated policy that belongs to another domain policy that is not deployed. If you encounter this state, contact an IBM service representative for further assistance.

- **EWLM disabling**: A user has requested to disable the managed server using the EWLM Control Center. The managed server is in the process of ending EWLM on the server. ARM services continues to collect performance data but it is not sent to the EWLM domain manager.
- **EWLM disabled**: The managed server is not running the EWLM managed server function or the server has not attempted to join the EWLM domain. To start EWLM on this server, you need to manually start EWLM on the server. The server remains in the list of managed servers for up to seven days. Then, the managed server is removed from the list.

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, if LPAR is enabled, 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.

4.4 Monitor

A main feature of the initial release of EWLM is the reporting capability. When a policy has been activated, you can monitor how it is running in the EWLM management domain using:

- First level:
  - Exceptions
  - Goal versus actuals for:
    - Service classes
    - Transaction classes
    - Process classes
  - Managed servers
- Details reports
- Topology reports – application and server
- Real time performance monitors
  - Goal achievement
  - Processor utilization
  - Transaction count and rate

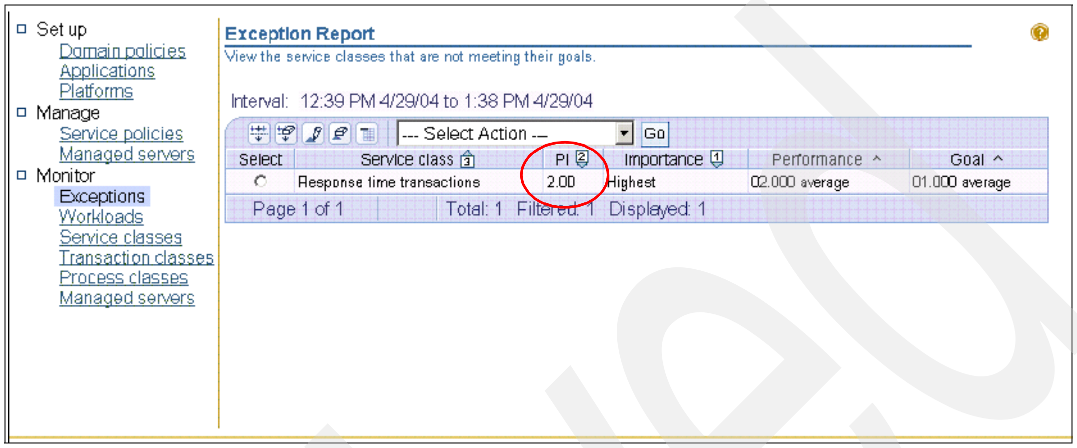
As briefly discussed in Figure 4-2 on page 81, the reporting interval can be changed by clicking the Preferences tab. The interval can be set to 1, 5, 10, 15, 30, 45 or 60 minutes. The reporting interval is displayed on all of the reporting panels, usually mid screen.

4.4.1 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.

Exceptions report

You can start with the Exceptions report. It shows a list of services 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.



Exception Report

View the service classes that are not meeting their goals.

Interval: 12:39 PM 4/29/04 to 1:38 PM 4/29/04

--- Select Action --- Go

Select	Service class	PI	Importance	Performance	Goal
○	Response time transactions	2.00	Highest	02.000 average	01.000 average

Page 1 of 1 Total: 1 Filtered: 1 Displayed: 1

Figure 4-22 Exceptions report

Figure 4-22 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.

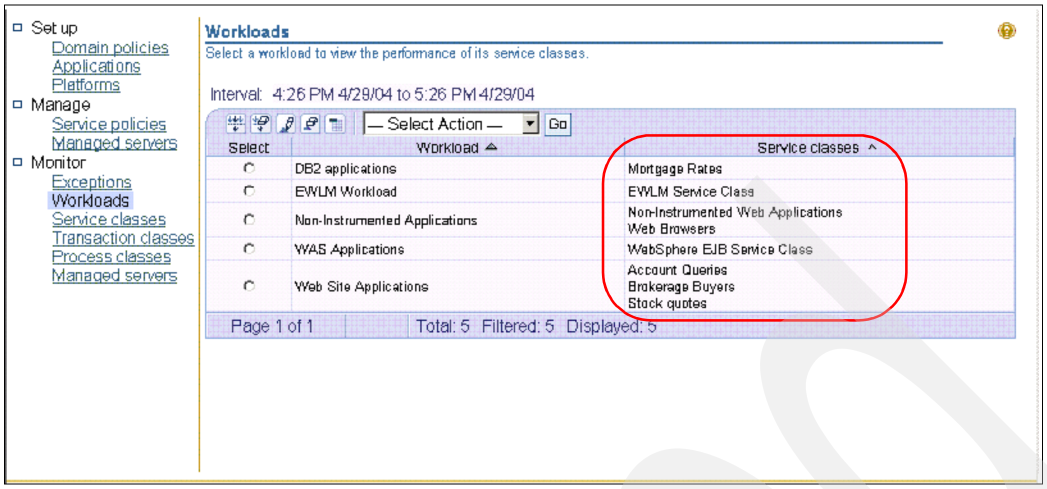
Table 4-4 Performance Index (PI)

Situation	PI value
Service Class is doing better than the goal defined for it	Less than (<) 1
Service Class is attaining the goal defined for it	1
Service Class is missing the goal defined for it	Greater than (>) 1

On the screen, you can immediately see the PI, which can be the starting point for understanding what caused the problem.

Workloads report

This report provides information organized at the workload level, including the service classes that belong to the workload definition. A sample Workloads report is shown in Figure 4-23. After selecting a workload, you can drill down to see each individual service classes. This level of reporting is useful to identify the performance characteristics of a specified workload.



Workloads

Select a workload to view the performance of its service classes.

Interval: 4:26 PM 4/29/04 to 5:26 PM 4/29/04

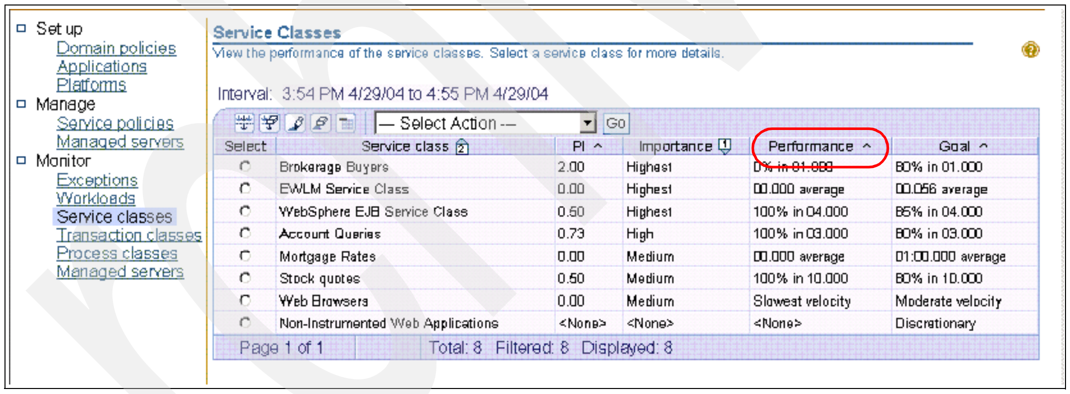
Select	Workload	Service classes
○	DB2 applications	Mortgage Rates
○	EWLM Workload	EWLM Service Class
○	Non-Instrumented Applications	Non-Instrumented Web Applications Web Browsers
○	WAS Applications	WebSphere EJB Service Class
○	Web Site Applications	Account Queries Brokerage Buyers Stock quotes

Page 1 of 1 Total: 5 Filtered: 5 Displayed: 5

Figure 4-23 Workload report

Service Classes report

This report provides detailed information on service classes both at the individual service class level and at the level of combined service classes known to the domain policy. The service class report shown in Figure 4-24 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 broad level, how all service classes are performing.



Service Classes

View the performance of the service classes. Select a service class for more details.

Interval: 3:54 PM 4/29/04 to 4:55 PM 4/29/04

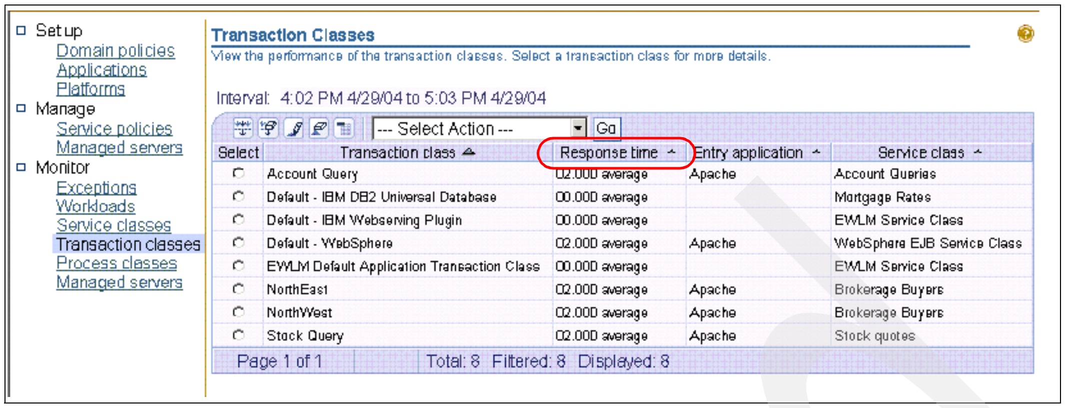
Select	Service class	PI	Importance	Performance	Goal
○	Brokerage Buyers	2.00	Highest	0% in 01.000	80% in 01.000
○	EWLM Service Class	0.00	Highest	00.000 average	00.056 average
○	WebSphere EJB Service Class	0.50	Highest	100% in 04.000	85% in 04.000
○	Account Queries	0.73	High	100% in 03.000	80% in 03.000
○	Mortgage Rates	0.00	Medium	00.000 average	01:00.000 average
○	Stock quotes	0.50	Medium	100% in 10.000	80% in 10.000
○	Web Browsers	0.00	Medium	Slowest velocity	Moderate velocity
○	Non-Instrumented Web Applications	<None>	<None>	<None>	Discretionary

Page 1 of 1 Total: 8 Filtered: 8 Displayed: 8

Figure 4-24 Service Classes report

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 4-25, 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 “Transaction class details” on page 109.



Transaction Classes

View the performance of the transaction classes. Select a transaction class for more details.

Interval: 4:02 PM 4/29/04 to 5:03 PM 4/29/04

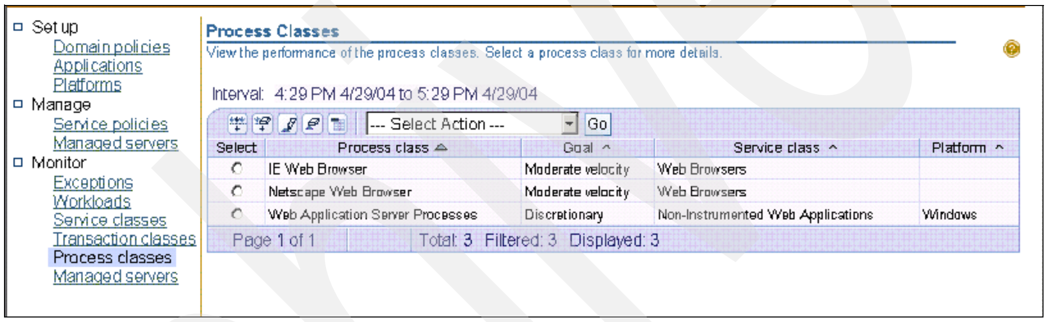
Select	Transaction class	Response time	Entry application	Service class
○	Account Query	02.000 average	Apache	Account Queries
○	Default - IBM DB2 Universal Database	00.000 average		Mortgage Rates
○	Default - IBM Webserving Plugin	00.000 average		EWLM Service Class
○	Default - WebSphere	02.000 average	Apache	WebSphere EJB Service Class
○	EWLM Default Application Transaction Class	00.000 average		EWLM Service Class
○	NorthEast	02.000 average	Apache	Brokerage Buyers
○	NorthWest	02.000 average	Apache	Brokerage Buyers
○	Stock Query	02.000 average	Apache	Stock quotes

Page 1 of 1 Total: 8 Filtered: 8 Displayed: 8

Figure 4-25 Transaction classes report

Process Classes report

Process class reporting provides attribute and statistical performance information on process classes. This includes processor usage and delay as well as storage and I/O delays.



Process Classes

View the performance of the process classes. Select a process class for more details.

Interval: 4:29 PM 4/29/04 to 5:29 PM 4/29/04

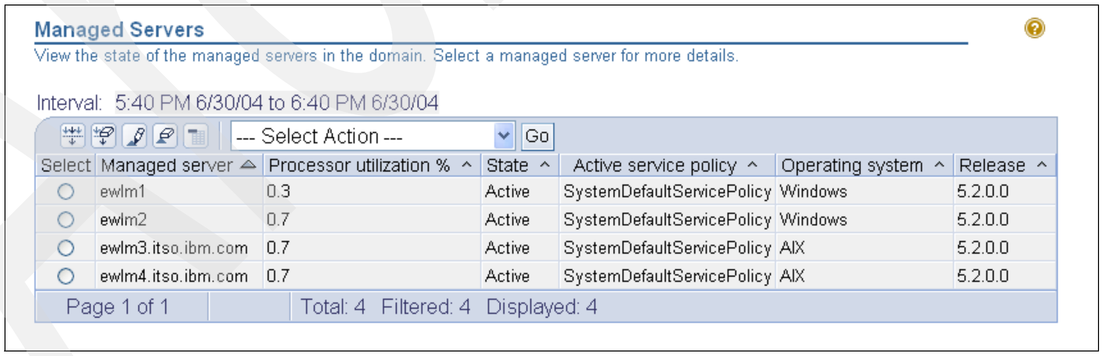
Select	Process class	Goal	Service class	Platform
○	IE Web Browser	Moderate velocity	Web Browsers	
○	Netscape Web Browser	Moderate velocity	Web Browsers	
○	Web Application Server Processes	Discretionary	Non-Instrumented Web Applications	Windows

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 4-26 Process Classes report

Managed Servers report

The Managed server report is shown in Figure 4-27.



Managed Servers

View the state of the managed servers in the domain. Select a managed server for more details.

Interval: 5:40 PM 6/30/04 to 6:40 PM 6/30/04

Select	Managed server	Processor utilization %	State	Active service policy	Operating system	Release
○	ewlm1	0.3	Active	SystemDefaultServicePolicy	Windows	5.2.0.0
○	ewlm2	0.7	Active	SystemDefaultServicePolicy	Windows	5.2.0.0
○	ewlm3.itso.ibm.com	0.7	Active	SystemDefaultServicePolicy	AIX	5.2.0.0
○	ewlm4.itso.ibm.com	0.7	Active	SystemDefaultServicePolicy	AIX	5.2.0.0

Page 1 of 1 Total: 4 Filtered: 4 Displayed: 4

Figure 4-27 Managed Servers report

For each managed server, you can display the managed server detailed statistics (more information is in “Managed server details” on page 110) and the Processor Utilization Monitor by selecting the desired task from the pull-down menu.

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 4.3.2, “Managed servers” on page 103.

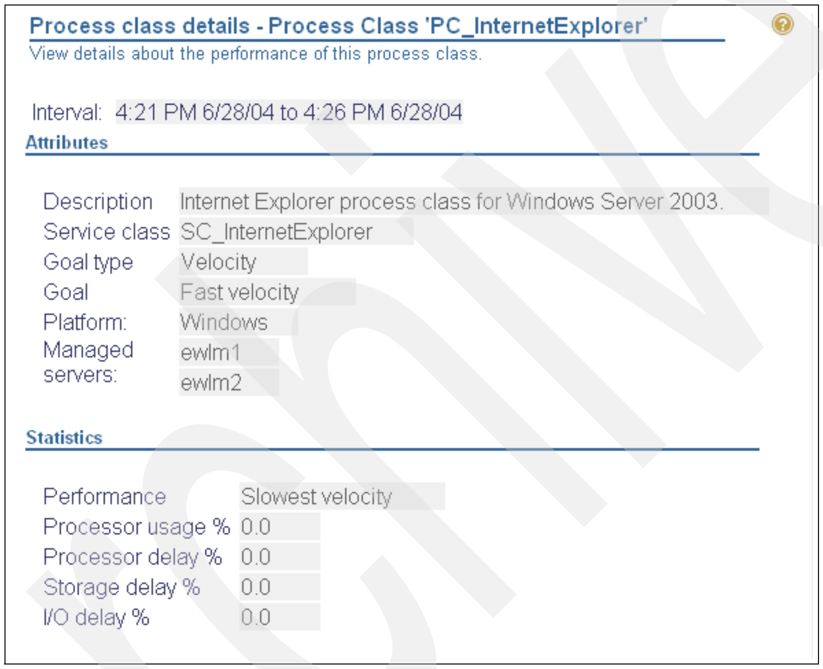
4.4.2 Details reports

EWLM provides detail 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.

Process class details

You obtain the Process class details report (Figure 4-28) 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, storage delay, and I/O delay


Process class details - Process Class 'PC_InternetExplorer'

View details about the performance of this process class.

Interval: 4:21 PM 6/28/04 to 4:26 PM 6/28/04

Attributes

Description	Internet Explorer process class for Windows Server 2003.
Service class	SC_InternetExplorer
Goal type	Velocity
Goal	Fast velocity
Platform:	Windows
Managed servers:	ewlm1
	ewlm2

Statistics

Performance	Slowest velocity
Processor usage %	0.0
Processor delay %	0.0
Storage delay %	0.0
I/O delay %	0.0

Figure 4-28 Process class detail report

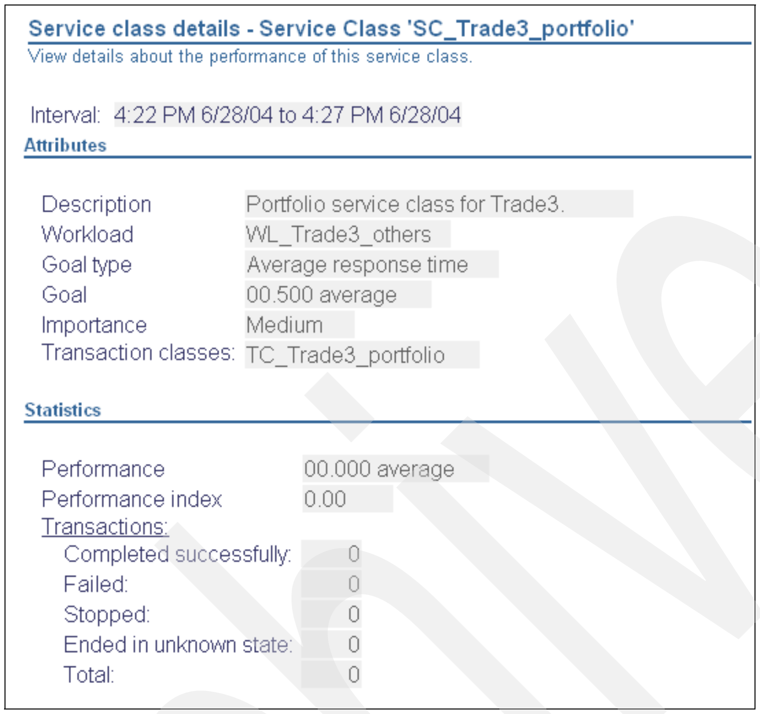
Service class details

You obtain the Service class details report (Figure 4-29) by selecting Service class details on the pull-down menu. The following information is returned:

- Goal type and the goal
- Importance level
- Transaction classes associated to this service class
- Statistics calculated in the displayed interval for:
  - Performance Index
  - Transactions (total, successfully completed, failed, stopped, ended unknown):
    Completed successfully: The number of transactions associated with this service class that finished processing.
    Failed: The number of transactions associated with this service class that failed to complete successfully.
    Stopped: The number of transactions associated with this service class that stopped being processed before they completed or failed.

Ended in unknown state: The number of transactions associated with this service class that did not end in a completed, failed, or aborted state.
Total: The total number of transactions that are associated with this service class.

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.



Service class details - Service Class 'SC_Trade3_portfolio'
View details about the performance of this service class.

Interval: 4:22 PM 6/28/04 to 4:27 PM 6/28/04

Attributes

Description	Portfolio service class for Trade3.
Workload	WL_Trade3_others
Goal type	Average response time
Goal	00.500 average
Importance	Medium
Transaction classes:	TC_Trade3_portfolio

Statistics

Performance	00.000 average
Performance index	0.00
Transactions:	
Completed successfully:	0
Failed:	0
Stopped:	0
Ended in unknown state:	0
Total:	0

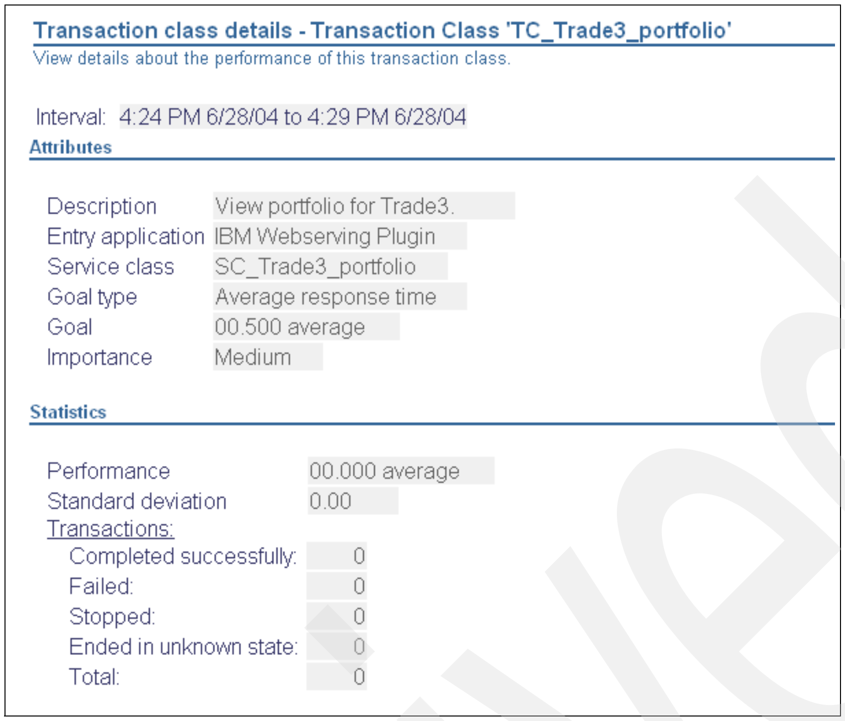
Figure 4-29 Service class details report

Transaction class details

You obtain the Transaction class details report (Figure 4-30) by selecting Transaction class details on the pull-down menu. The following information is returned:

- Service class associated with the transaction class
- Goal type and the goal
- 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.


Transaction class details - Transaction Class 'TC_Trade3_portfolio'

View details about the performance of this transaction class.

Interval: 4:24 PM 6/28/04 to 4:29 PM 6/28/04

Attributes

Description	View portfolio for Trade3.
Entry application	IBM Webserving Plugin
Service class	SC_Trade3_portfolio
Goal type	Average response time
Goal	00.500 average
Importance	Medium

Statistics

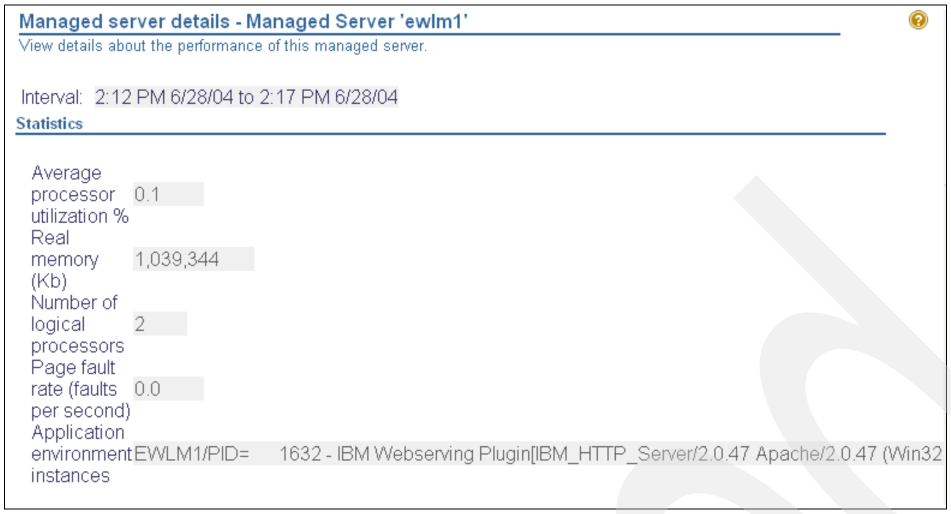
Performance	00.000 average
Standard deviation	0.00
Transactions:	
Completed successfully:	0
Failed:	0
Stopped:	0
Ended in unknown state:	0
Total:	0

Figure 4-30 Transaction class details report

Managed server details

You obtain the Managed server details report (Figure 4-31 and Figure 4-32) by selecting the Managed server detailed statistics on the pull-down menu. The following information is returned:

- Average processor utilization %
- Real memory (Kb)
- Number of logical processors
- Page fault rate
- Application environment - you can see the ARM enabled middleware running on this server (application name and group name):
  - For the Web server you can see an entry similar to EWLM1/PID= 1632 - IBM Webserving Plugin[IBM_HTTP_Server/2.0.47 Apache/2.0.47 (Win32)] where EWLM1 is the hostname and PID is the process ID for the HTTP.
  - For the WebSphere Application Server you can see an entry similar to ewlm2.TradeClusterServer1 - WebSphere[ewlm2Network] where ewlm2.TradeClusterServer1 is the application server name and ewlm2Network is the EWLM naming for the WebSphere Application Server cluster.
  - For the DB2 you can see IBM DB2 Universal Database[db2inst1].
- List of the Service classes active on this managed server with the following details:
  - Service class name
  - Processor usage %
  - Processor delay %
  - Storage delay %
  - I/O delay %


Managed server details - Managed Server 'ewlm1'

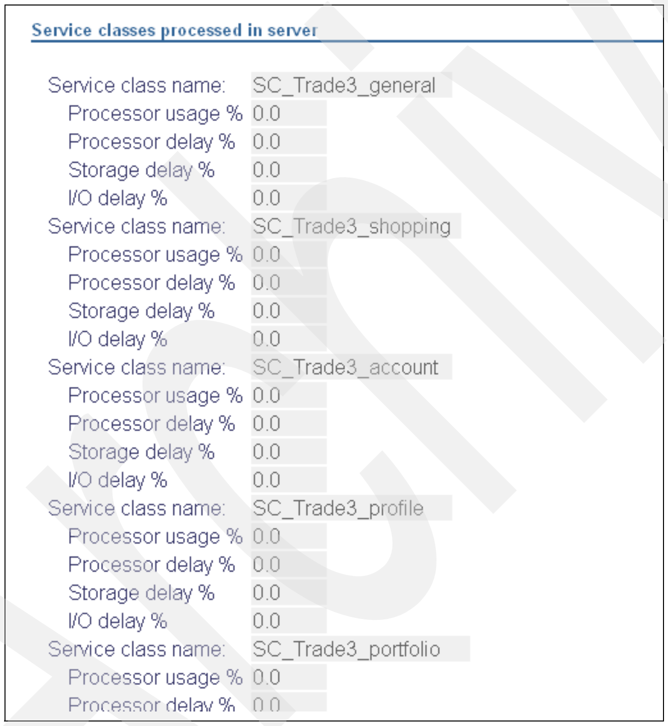
View details about the performance of this managed server.

Interval: 2:12 PM 6/28/04 to 2:17 PM 6/28/04

Statistics

Average processor utilization %	0.1
Real memory (Kb)	1,039,344
Number of logical processors	2
Page fault rate (faults per second)	0.0
Application environment instances	EWLM1/PID= 1632 - IBM Webserving Plugin[IBM_HTTP_Server/2.0.47 Apache/2.0.47 (Win32

Figure 4-31 Managed server details report


Service classes processed in server

Service class name:	SC_Trade3_general
Processor usage %	0.0
Processor delay %	0.0
Storage delay %	0.0
I/O delay %	0.0
Service class name:	SC_Trade3_shopping
Processor usage %	0.0
Processor delay %	0.0
Storage delay %	0.0
I/O delay %	0.0
Service class name:	SC_Trade3_account
Processor usage %	0.0
Processor delay %	0.0
Storage delay %	0.0
I/O delay %	0.0
Service class name:	SC_Trade3_profile
Processor usage %	0.0
Processor delay %	0.0
Storage delay %	0.0
I/O delay %	0.0
Service class name:	SC_Trade3_portfolio
Processor usage %	0.0
Processor delay %	0.0

Figure 4-32 Managed server details

4.4.3 Application and server topology reports

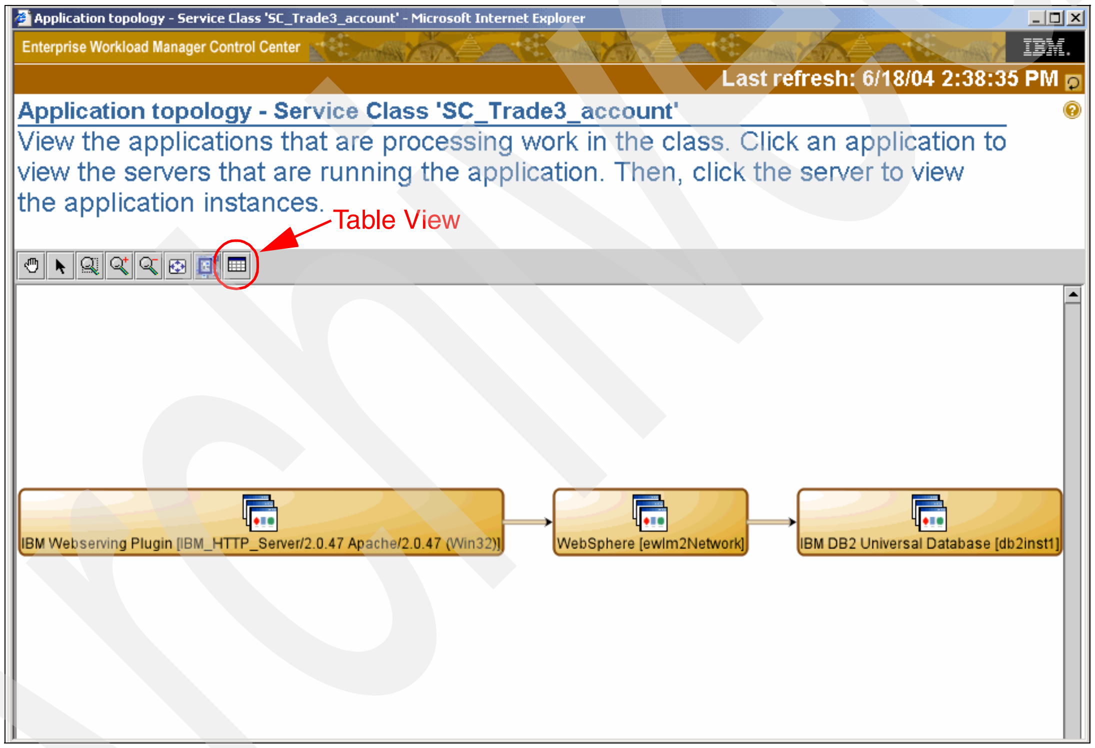
The application and server topology reports are built by EWLM dynamically following the flow of your transaction or your process. The *application topology* is the logical representation of the application flow while the *server topology* is the physical representation of which server you application 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 4-33 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 the place 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 on the matrix icon as shown in Figure 4-33.


Application topology - Service Class 'SC_Trade3_account' - Microsoft Internet Explorer

Enterprise Workload Manager Control Center

Last refresh: 6/18/04 2:38:35 PM

Application topology - Service Class 'SC_Trade3_account'

View the applications that are processing work in the class. Click an application to view the servers that are running the application. Then, click the server to view the application instances.

Table View
IBM Webserving Plugin [IBM_HTTP_Server/2.0.47 Apache/2.0.47 (Win32)]
WebSphere [ewlm2Network]
IBM DB2 Universal Database [db2inst1]

Figure 4-33 Application topology report

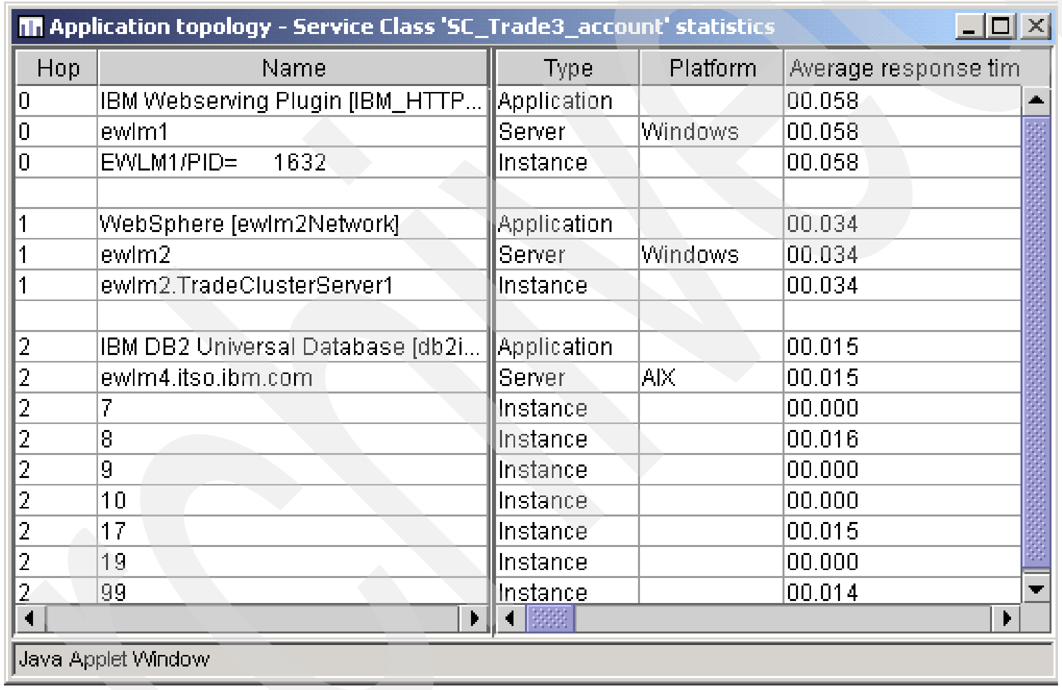
The example in Figure 4-34 is a partial view of the Topology tableview. The first column lists the hops (0,1 and 2): these represent the individual segments where the transactions are processed. Hop0 identifies the edge server that is the starting and end point for the transactions, where the EWLM classification 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 one hour.

The following information can be seen in the Tableview by scrolling to the right:

- Average response time
- Active time (seconds)

- Processor time (microseconds)
- Processor delay %
- Processor using %
- Stopped transaction
- Failed transactions
- Successful transactions
- I/O delay %
- Page delay %
- Idle %
- Other %
- Topology uncertainly count
- Reporting gap count
- Hard page faults
- Queue time (milliseconds)
- Standard deviation


Application topology - Service Class 'SC_Trade3_account' statistics

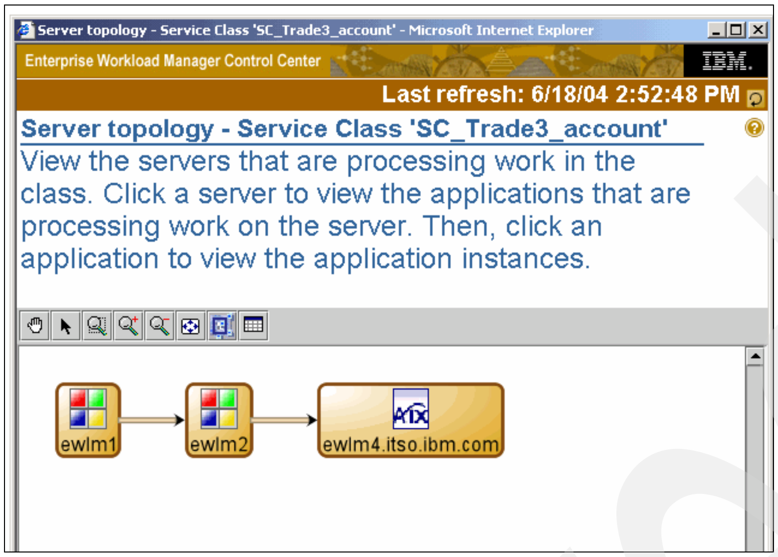
Hop	Name	Type	Platform	Average response tim
0	IBM Webserving Plugin [IBM_HTTP...	Application		00.058
0	ewlm1	Server	Windows	00.058
0	EWLM1/PID= 1632	Instance		00.058
1	WebSphere [ewlm2Network]	Application		00.034
1	ewlm2	Server	Windows	00.034
1	ewlm2.TradeClusterServer1	Instance		00.034
2	IBM DB2 Universal Database [db2i...	Application		00.015
2	ewlm4.itso.ibm.com	Server	AIX	00.015
2	7	Instance		00.000
2	8	Instance		00.016
2	9	Instance		00.000
2	10	Instance		00.000
2	17	Instance		00.015
2	19	Instance		00.000
2	99	Instance		00.014

Java Applet Window

Figure 4-34 Tableview report

Server topology

Figure 4-35 is an example of a server topology report. If you click on 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’d expect. If not, then there’s most likely work that should be classified that was not known previously.


Server topology - Service Class 'SC_Trade3_account' - Microsoft Internet Explorer
Enterprise Workload Manager Control Center
IBM.
ewlm1
ewlm2
ewlm4.itso.ibm.com

Last refresh: 6/18/04 2:52:48 PM

Server topology - Service Class 'SC_Trade3_account'

View the servers that are processing work in the class. Click a server to view the applications that are processing work on the server. Then, click an application to view the application instances.

Figure 4-35 Server topology report

4.4.4 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 (JRE) release level is 1.4.1. All the following graphical reporting is based on the last 10 second interval:

- Application topology
- Server topology
- Performance index monitor
- Goal achievement monitor
- Transaction count 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 on using these monitors, refer to the help in the control center for each screen shown.

To start the monitor, select the monitor function from the pull-down menu and click **Go**. The monitor is based on one of the following components:

- Goal achievement
- Performance index
- Transaction count and rate
- Processor utilization

Once you see a graphical monitor, there are some features (shown in Figure 4-36) in the heading that are common through all the monitors, such as:

- View: Where you can:
  - Select the color
  - Show/hide annotations
  - Show monitor in 2D or 3D

- History: Where you can move back and forth on the data. Keeps an hour worth of history.
- Elevation/Rotation: These adjustments are useful in the 3D graphical view.

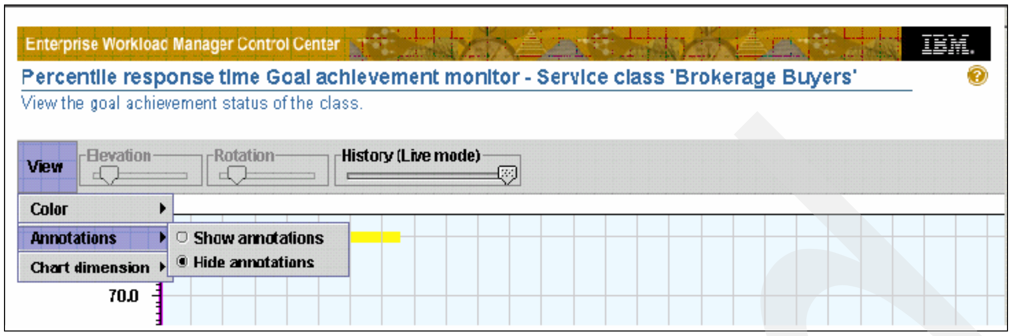

Enterprise Workload Manager Control Center
IBM.
Percentile response time Goal achievement monitor - Service class 'Brokerage Buyers'
View the goal achievement status of the class.
View
Elevation
Rotation
History (Live mode)
Color
Annotations
Chart dimension
Show annotations
Hide annotations
70.0

Figure 4-36 Report heading

Let's have a look at a sample of these graphical monitors. Figure 4-37 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.

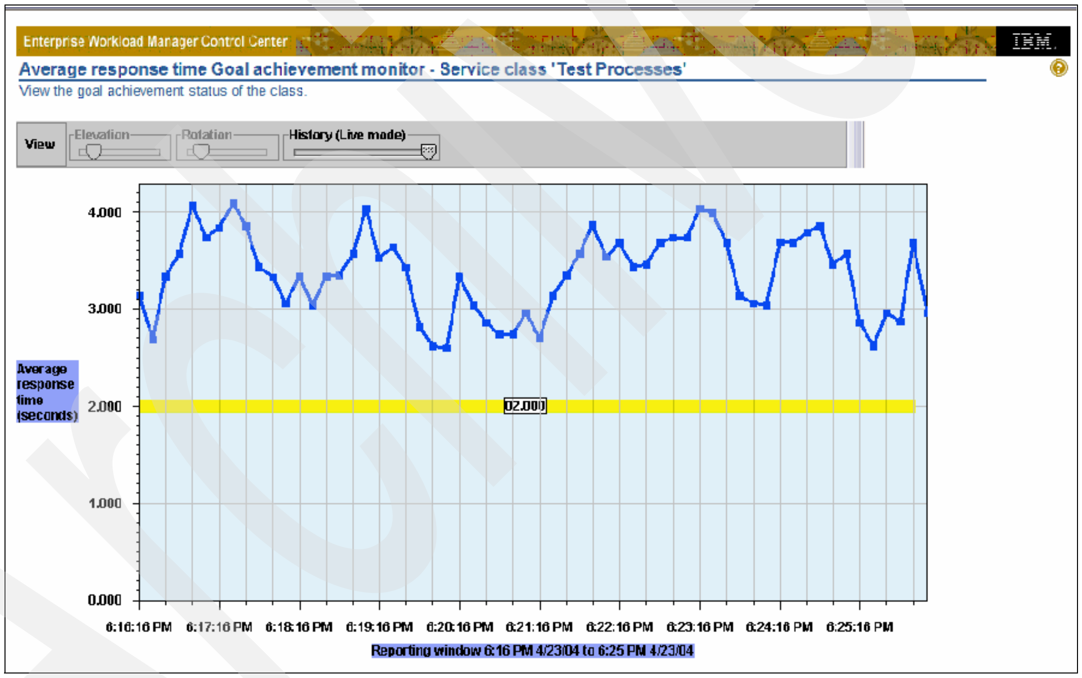

Enterprise Workload Manager Control Center
IBM.
Average response time Goal achievement monitor - Service class 'Test Processes'
View the goal achievement status of the class.
View
Elevation
Rotation
History (Live mode)
4.000
3.000
2.000
1.000
0.000
Average response time (seconds)
02.000
6:16:16 PM
6:17:16 PM
6:18:16 PM
6:19:16 PM
6:20:16 PM
6:21:16 PM
6:22:16 PM
6:23:16 PM
6:24:16 PM
6:25:16 PM
Reporting window 6:16 PM 4/23/04 to 6:25 PM 4/23/04

Figure 4-37 Average response time report

Figure 4-38 is a sample of the velocity goal report. Once again the goal is represented by the flat line in the middle, and the jagged line is the current trend of the service class. You can scroll the History bar if you want to see a different interval within the hour of collected data.

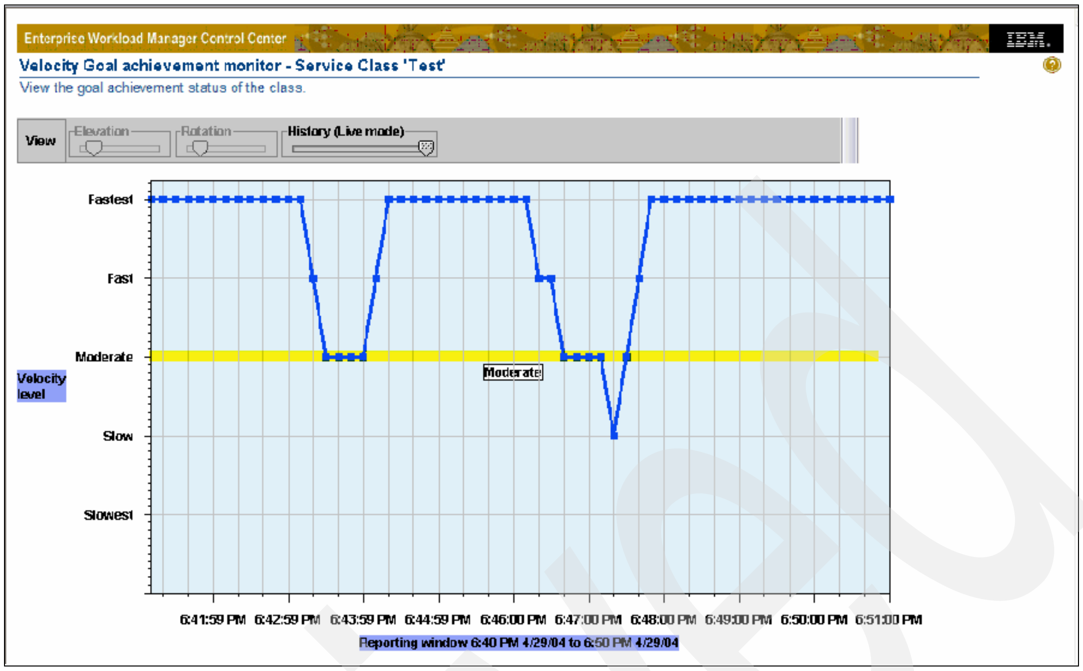

Enterprise Workload Manager Control Center
Velocity Goal achievement monitor - Service Class 'Test'
View the goal achievement status of the class.
View
Elevation
Rotation
History (Live mode)
Fastest
Fast
Moderate
Slow
Slowest
Velocity level
Moderate
6:41:59 PM
6:42:59 PM
6:43:59 PM
6:44:59 PM
6:46:00 PM
6:47:00 PM
6:48:00 PM
6:49:00 PM
6:50:00 PM
6:51:00 PM
Reporting window 6:40 PM 4/29/04 to 6:50 PM 4/29/04

Figure 4-38 Velocity goal report

Figure 4-39 shows a sample of a Transaction count report, where you can see the absolute number of successfully processed transactions in the interval of time.

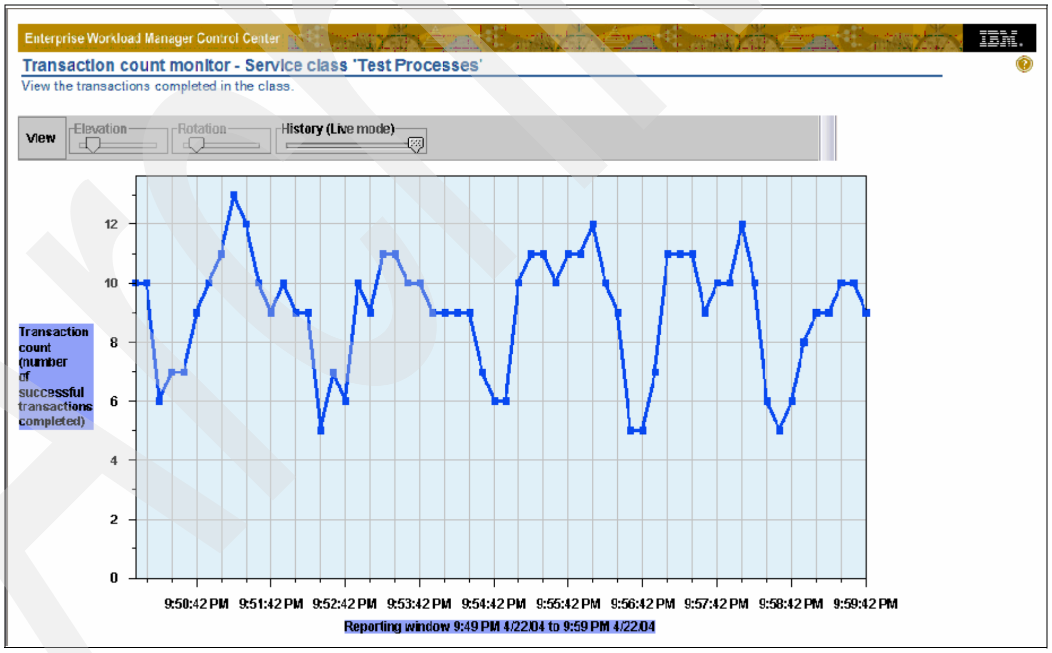
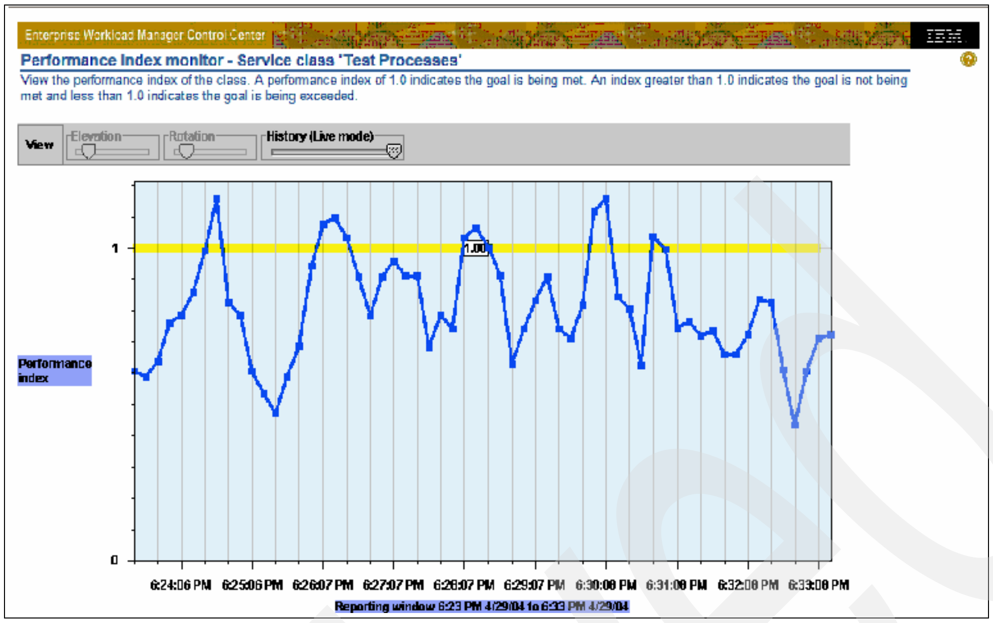

Enterprise Workload Manager Control Center
Transaction count monitor - Service class 'Test Processes'
View the transactions completed in the class.
View
Elevation
Rotation
History (Live mode)
12
10
8
6
4
2
0
Transaction count (number of successful transactions completed)
9:50:42 PM
9:51:42 PM
9:52:42 PM
9:53:42 PM
9:54:42 PM
9:55:42 PM
9:56:42 PM
9:57:42 PM
9:58:42 PM
9:59:42 PM
Reporting window 9:49 PM 4/22/04 to 9:59 PM 4/22/04

Figure 4-39 Transaction count report

The last report is the Performance Index monitor shown in Figure 4-40, 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.


Enterprise Workload Manager Control Center

Performance Index monitor - Service class 'Test Processes'

View the performance index of the class. A performance index of 1.0 indicates the goal is being met. An index greater than 1.0 indicates the goal is not being met and less than 1.0 indicates the goal is being exceeded.

Figure 4-40 PI monitor

In "Reporting" on page 150 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.

4.4.5 Summary table

Table 4-5 provides a summary of the available reporting. The areas of focus that reporting concentrates on are listed across the top, indicating the information available for each area. The specific information reported across the various focus areas is listed in the left-hand column.

Table 4-5 Available reporting

	Exceptions (By ServCl)	Workloads (By ServCl)	Service Classes	Transaction Classes	Process Classes
Service class detail	O	O	O		
Transaction class detail				O	
Process class detail					O
Application topology	O	O	O	O	
Server topology	O	O	O	O	O
Performance index monitor	O	O	O		
Goal achievement monitor	O	O	O	O	O
Transaction count monitor	O	O	O	O	O
Transaction rate monitor	O	O	O	O	
Performance index	O	O	O		
Importance	O	O	O		

	Exceptions (By ServCl)	Workloads (By ServCl)	Service Classes	Transaction Classes	Process Classes
Current performance	**O**	**O**	**O**		**O**
Performance goal	**O**	**O**	**O**		**O**

5

The ITSO EWLM scenario

In this chapter we describe the steps required to build an EWLM domain using the ITSO environment and its workload. We include some monitoring and reporting functions which give performance information indications. At this point we assume that you are familiar with the EWLM strategy, installation, application instrumentation, and operation described in the previous chapters.

This chapter is divided into the following topics:

- Building an EWLM domain in the ITSO environment
  - Assessment
  - Planning
  - Design
  - Classifying the work
  - Verification
- Example usage of EWLM monitoring and reporting functions
  - First-level report
  - Topology report
  - Details report

5.1 Overview of the process

Before building your domain policy you need to carefully assess your environment and plan your EWLM setup. As with the workload management functions of individual operating systems, the planning and preparation phase is the most important step to successfully use workload management capability. You need to collect information from various EWLM domain resources and decide what needs to be monitored.

The purpose of this chapter is to present an example implementation of EWLM and to give an explanation of some of the main considerations for designing and building it. The following list describes our five phase approach to building an EWLM domain. The goal of these steps is to produce a representation of EWLM domain policy with a measurable business goal that is easy to understand and that will serve as a basis for future expansion. These steps may not be a linear process and you may need to move several times between steps three and five.

1. Assessment
   a. Defining the EWLM domain
   b. Identifying all servers and middleware
   c. Identifying all applications
2. Planning
   a. Naming the domain policy
   b. Defining the user information
   c. Defining the directory and port configuration
3. Design
   a. Identifying the transactions that request business functions
   b. Identifying the processes that request business functions
   c. Grouping service classes into a workload
   d. Setting service class goals
4. Classifying the work
   a. Defining classification filters for each transaction and process class
   b. Creating the domain policy at the EWLM Control Center
5. Verification
   a. Verifying transaction request classification filters
   b. Verifying service class definition
   c. Defining and implementing service policy

In addition, we include some information on monitoring and reporting functionality once you have set up your EWLM domain. This includes reports and monitors which can help you understand and diagnose performance problems you may have or suspect in your environment.

We provide many tables which you can use to collect information when you organize your EWLM domain. To collect this information you need to contact various people from your company. This includes application designers, application developers, operations, system administrators, and persons who are responsible for particular component monitoring. These tables may be interim steps or they may be the final definition of your domain policy.

Note: There are two functions that we do not cover in this chapter:

- Integration with firewall and security
- Load balancing

Refer to Chapter 6, “Using a firewall and securing EWLM” on page 155 and Chapter 7, “Using load balancing” on page 187 for more information on these topics.

5.1.1 Business scenario

We implemented the following two Web applications in the ITSO environment:

- ***Trade3*** provides online trading service for stock broker.
- ***Plants by WebSphere*** provides online shopping service for those who are interested in gardening.

Why do we want to use EWLM? We want to view the performance of each workload individually and in detail, as well as the AIX batch application. We want to view the performance of shopping related transactions as a group. We want to define business goals for each service class after load testing, since we have no Service Level Agreements (SLAs).

The purpose of the monitoring and reporting section is to guide you in creating a way to understand performance considerations and possible performance problems in your environment.

5.1.2 Naming convention

All EWLM components like transaction class, process class, service class, workload, and service policy need names. These names must be in accordance with the EWLM naming requirements, which are the following:

- Up to 64 valid characters
- The valid characters include:
  - Upper case alphabetic characters (`A-Z`)
  - Lower case alphabetic characters (`a-z`)
  - Numeric characters (`0-9`)
  - Blank ( )
  - Period (`.`)
  - Underscore (`_`)
  - Hyphen (`-`)
- Start with alphabetic character
- End with alphabetic character, numeric character, period, underscore, or hyphen

EWLM components often have similar names. We recommend that you set up your own naming convention to avoid confusion. Once you decide on a naming convention, use it in your EWLM domain consistently. As soon as your environment includes more than just a handful of transactions it can get complicated. In Table 5-1 we show the naming convention we used in this chapter. The *ApplicationName* is optional for the workload and is replaced by appropriate application name like `trade3` or `plants`, and *_identifier* is optional and is used to distinguish the class from another within the same component. An example of a transaction class for our test environment is `TC_Trade3_shopping`.

Table 5-1 Naming convention for our EWLM components

EWLM component	Name begins with
Transaction class	TC_*ApplicationName[_identifier]*
Process class	PC_*ApplicationName[_identifier]*
Service class	SC_*ApplicationName[_identifier]*
Workload	WL*[_ApplicationName][_identifier]*

5.2 Assessment

Before we can design and plan our EWLM domain we need to assess our environment. This is the first step to develop a domain policy. At this stage, we assess the servers and applications of the ITSO environment. The design of your domain policy is dependent on the number of platforms, applications, workloads, and, in some situations, the Service Level Agreements you have, which makes this phase a very important one.

The assessment phase involves the following steps:

1. Defining the EWLM domain
2. Identifying all servers and middleware
3. Identifying all applications

5.2.1 Defining the EWLM domain

The first step of the assessment phase is to define an EWLM management domain and its boundary. Table 5-2 and Table 5-3 provide a simple overview of our EWLM management domain, which is the foundation for our domain policy planning.

The EWLM domain definition contains the following attributes:

- ► EWLM domain name and description
- ► Servers, which will be managed servers, and a simple description of each

Table 5-2 Definition of EWLM domain

EWLM domain name	Description
itsoewlm	Sample EWLM domain for Redbook.

Table 5-3 Boundary of EWLM domain

Server	Description
ewlmdm1.itso.ibm.com	EWLM domain manager
ewlm1.itso.ibm.com	Web server
ewlm2.itso.ibm.com	Application server, Deployment manager
ewlm3.itso.ibm.com	Application server
ewlm4.itso.ibm.com	Database server

You can see the EWLM domain name in the title bar of the EWLM Control Center in the browser window; this is shown in Figure 5-1.

itsoewlm - IBM Enterprise Workload Manager - Microsoft Internet Explorer

Figure 5-1 EWLM domain name in title bar of browser window

We recommend that you create a picture of your server topology for a better understanding of your environment, like what we did for our environment as shown in Figure 5-2. Note that we do not illustrate the logical relationship between the servers because EWLM creates it automatically as described at 4.4, “Monitor” on page 104.

We do not include the firewall, router, proxy server, DNS server, and laptops shown in Figure 5-2 in our EWLM domain because we do not want to monitor this hardware. We are interested in the workflow and a breakdown of transactions between the Web server, Application server and Database server.

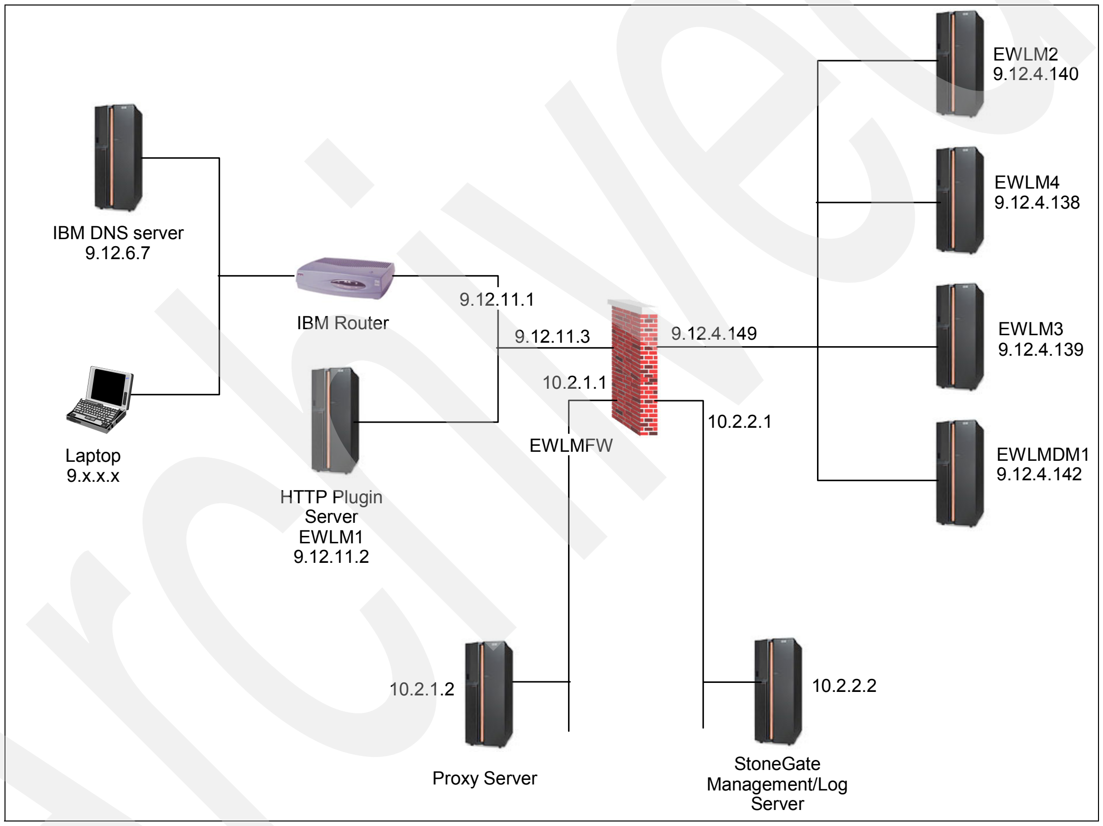


Figure 5-2 Topology of our lab environment

5.2.2 Identifying all servers and middleware

The next step of the assessment phase is to obtain more detailed information about the servers, such as IP address, operating system, and middleware that is running on each server. You have to collect operating system and middleware information carefully since current EWLM implementation depends on it. Refer to 1.3, “EWLM supported platforms and applications” on page 16 for supported operating system and middleware information.

The servers and middleware list includes the following attributes:

- Host name
- IP address

- Operating system name and version
- Middleware name and version

Our definition of servers and middleware is shown in Table 5-4. It consists of a Web Server tier running on `ewlm1`, an Application Server cluster tier running on servers `ewlm2` and `ewlm3`, and a Database Server tier running on `ewlm4`. These are our four managed servers which are on AIX and Windows operating systems.

Table 5-4 Definition of servers and middlewares

Host name	IP address	Operating system	Middleware
ewlm1.itso.ibm.com	9.12.4.141	Windows Server 2003	IBM HTTP Server 2.0.47
ewlm2.itso.ibm.com	9.12.4.140	Windows Server 2003	WebSphere Application Server 5.1.1, Deployment Manager 5.1.1
ewlm3.itso.ibm.com	9.12.4.139	AIX 5L Version 5.2 ML03	WebSphere Application Server 5.1.1
ewlm4.itso.ibm.com	9.12.4.138	AIX 5L Version 5.2 ML03	DB2 Universal Database 8.2

For an overview of your environment we recommend merging Table 5-3 and Table 5-4.

5.2.3 Identifying all applications

We have identified all servers and middleware in our environment. We now make an inventory of applications that are used by the business function. The applications include Web applications and client programs such as Internet Explorer. You have to identify all applications you would like to monitor using EWLM.

The application list contains the following attributes:

- Application name
- The purpose of the application
- The users of the application
- Related middleware or platform deployed

Our definition of applications is shown in Table 5-5. It includes our Web applications `Trade3` and `Plants by WebSphere` as well as an Internet Explorer and a batch application.

Table 5-5 Definition of applications

Application name	Users	Related middleware or platform
Trade3	Stock broker	IBM HTTP Server 2.0.47 WebSphere Application Server 5.1.1, DB2 Universal Database 8.2
Plants by WebSphere	Consumer	IBM HTTP Server 2.0.47, WebSphere Application Server 5.1.1, Cloudscape
Internet Explorer	WebSphere Application Server and EWLM administrators	Windows Server 2003
Calc batch application	In-house reporting system	AIX 5L Version 5.2 ML03

We are now ready to start planning our domain policy using the information we obtained in the previous sections.

5.3 Planning

Once you have assessed your environment, you can start planning your EWLM environment in detail. The following steps are part of the planning phase:

1. Naming the domain policy
2. Defining the user information
3. Defining the directory and port configuration

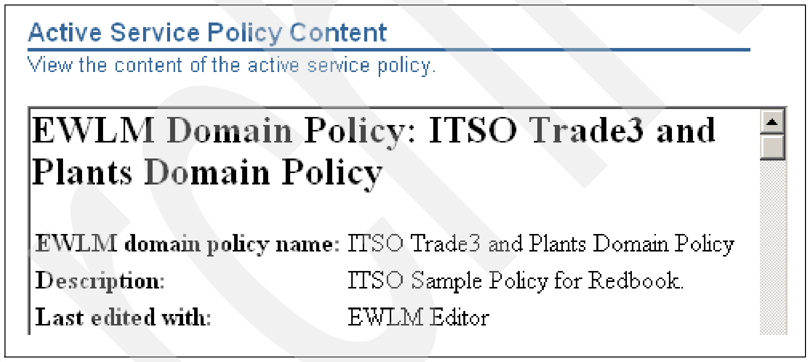
5.3.1 Naming the domain policy

We assign a name to the domain policy as shown in Table 5-6. This should also be the component you start with since all other EWLM components are part of this domain policy.

Table 5-6 Our domain policy description

EWLM domain policy name	Domain policy description
ITSO Trade3 and Plants Domain Policy	ITSO Sample Policy for Redbook.

It is called `ITSO Trade3 and Plants Domain Policy` for our test environment as shown in the Active Service Policy Content screen in Figure 5-3.


Active Service Policy Content
View the content of the active service policy.

EWLM Domain Policy: ITSO Trade3 and Plants Domain Policy

EWLM domain policy name:	ITSO Trade3 and Plants Domain Policy
Description:	ITSO Sample Policy for Redbook.
Last edited with:	EWLM Editor

Figure 5-3 Domain policy name and description at the EWLM Control Center

5.3.2 Defining the user information

After you set up your EWLM domain, you need to create several users. Table 5-7 describes the user information we used during installation, configuration, running, and administering EWLM components.

Refer to 2.3, "Virtualization Engine suite" on page 22, 2.4, "EWLM configuration in our ITSO environment" on page 32, 2.5, "Installing the domain manager on other operating systems" on page 50 and 4.1, "EWLM Control Center overview" on page 78 for more information.

Table 5-7   User information

Purpose	Host name	User name
Installing and configuring EWLM domain manager and EWLM managed servers	ewlmdm1.itso.ibm.com	root
	ewlm1.itso.ibm.com	Administrator
	ewlm2.itso.ibm.com	Administrator
	ewlm3.itso.ibm.com	root
	ewlm4.itso.ibm.com	root
Running WebSphere Application Server process on the domain manager	ewlmdm1.itso.ibm.com	root
Running domain manager process on the domain manager	ewlmdm1.itso.ibm.com	root
Administering WebSphere Application Server on the domain manager	ewlmdm1.itso.ibm.com	ibmewlm
Running managed server process on the managed server	ewlm1.itso.ibm.com	userid00
	ewlm2.itso.ibm.com	userid01
	ewlm3.itso.ibm.com	root
	ewlm4.itso.ibm.com	root
Administering EWLM domain using the EWLM Control Center	ewlmdm1.itso.ibm.com	ewlmadm
Operating EWLM domain using the EWLM Control Center	ewlmdm1.itso.ibm.com	ewlmops
Monitoring EWLM domain using the EWLM Control Center	ewlmdm1.itso.ibm.com	ewlmmon

5.3.3 Defining the directory and port configuration

We now list the EWLM components’ installation directory, working directory, and port configuration.

Table 5-8 shows our directory configuration. For this setup we used the default installation directory. An EWLM working directory is required when you issue the EWLM configuration and startup commands like **`createDM.sh`**, **`createMS.sh`**, **`changeCC.sh`**, and **`startMS.sh`**. The working directory should not exist before creating the domain manager or the managed server and must be provided as an absolute path. Refer to 2.3, “Virtualization Engine suite” on page 22, 2.4, “EWLM configuration in our ITSO environment” on page 32 and 2.5, “Installing the domain manager on other operating systems” on page 50 for more information.

Table 5-8   Directory configuration

Description	Host name	Directory name
EWLM domain manager installation directory	ewlmdm1.itso.ibm.com	/opt/IBM/VE/EWLM
EWLM domain manager working directory	ewlmdm1.itso.ibm.com	/opt/EWLMDM

Description	Host name	Directory name
EWLM managed server installation directory	ewlm1.itso.ibm.com	C:\Program Files\IBM\VE\EWLMMS
	ewlm2.itso.ibm.com	C:\Program Files\IBM\VE\EWLMMS
	ewlm3.itso.ibm.com	/opt/IBM/VE/EWLMMS
	ewlm4.itso.ibm.com	/opt/IBM/VE/EWLMMS
EWLM managed server working directory	ewlm1.itso.ibm.com	C:\ewlmMS
	ewlm2.itso.ibm.com	C:\ewlmMS
	ewlm3.itso.ibm.com	/opt/ewlmMS
	ewlm4.itso.ibm.com	/opt/ewlmMS

Our port configuration is shown in Table 5-9. The port for communication between the domain manager and the EWLM Control Center, for example, is specified as 9092. We chose these ports when running the **createDM** command in 2.4.1, “Configuring domain manager on Linux” on page 32. You have to ensure that these port numbers are not in use on your system or your connections will fail.

Table 5-9 Port configuration

Description	Port number
The port that the domain manager and managed server will use to communicate	3333
The port that the domain manager and the EWLM Control Center will use to communicate	9092
The first port in a range of fifteen ports that the WebSphere Application Server instance will use	20000

5.4 Design

Based on the information gathered during the previous phase, we begin to identify all transactions and processes to be monitored by EWLM.

The following steps constitute the design phase:

1. Identifying the transactions that request business functions
2. Identifying the processes that request business functions
3. Grouping service classes into a workload
4. Setting service class goals

Note: You should have functional EWLM domain manager, managed server, and ARM instrumented applications before starting this phase. Refer to Chapter 2, “Installing and configuring EWLM” on page 19 and Chapter 3, “Enabling middleware for EWLM” on page 57 for more information.

5.4.1 Identifying the transactions that request business functions

The first step of the design phase is to identify and classify the transactions.

It is difficult to identify all transactions in a distributed environment even if the EWLM management domain consists only of a few servers like in our test environment. The easiest way is to interview the application developer and ask the following questions:

- *How do clients use this application?*
- *What type of applications access this middleware?*

However, if you do not get answers to those questions you are in the same situation as we are with our test environment. We therefore provide an example study for such a case, describing how to identify transactions. It is, of course, dependent on your application environment.

To create transaction classes, you have to identify an edge application first since all work requests are classified to service classes by EWLM by its edge application. It is the entry point of your applications as described at 1.2, "EWLM concepts" on page 5. When we set up our servers, operating systems, middleware, applications, and firewall in our environment, we already know that the edge server of both Trade3 and Plants by WebSphere is the IBM HTTP Server. In the next section we describe how to find an edge server if such reference information is not available.

The overall process of identifying the transactions that request business functions involves the following steps:

1. Identifying an edge server of the work request
2. Identifying transactions
3. Defining transaction classes
4. Mapping transaction classes to service class

Identifying an edge server of the work request

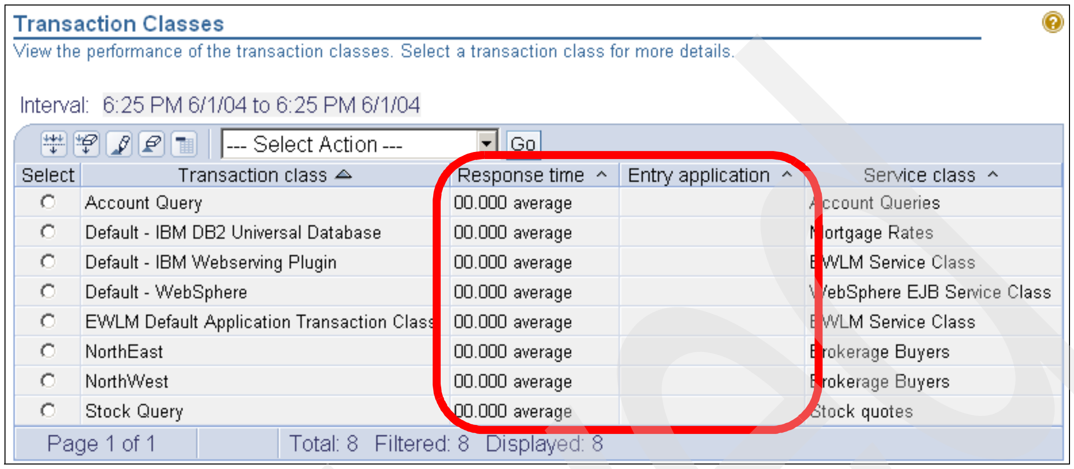
EWLM provides an application topology view which is a high-level view of the EWLM domain from a business transaction perspective. We use this view and an EWLM sample domain policy, `Sample Banking Domain Policy`, to identify the edge application of the work request. While using this policy, all work requests we generate are classified to a default transaction class in the domain policy because there is no transaction class suitable for our work request. There are, by default, three transaction classes for each application respectively:

- Default - IBM DB2 Universal Database
- Default - IBM Webserving Plugin
- Default - WebSphere

These transaction classes have the same classification filter, `(\*) Equal (\*)`, that matches to a work request which does not match any other transaction class filters. For more information on how filters work, refer to 4.2.1, "Classifying your workload" on page 88.

Tip: You can use EWLM default domain policy in this step, although we used `Sample Banking Domain Policy`. If the EWLM domain does not have an active service policy, the EWLM domain uses the default service policy, which consists of a discretionary goal for all transactions, but you cannot use the default policy once you have deployed some policy to your domain; it's gone.

Once the work request is classified by EWLM, the Response time and Entry application columns of the Transaction Classes monitor display some information as shown in Figure 5-4. We can also identify the edge application using the transaction class' application topology view, which is automatically generated by EWLM once we have a flow of transactions.


Transaction Classes

View the performance of the transaction classes. Select a transaction class for more details.

Interval: 6:25 PM 6/1/04 to 6:25 PM 6/1/04

Select	Transaction class	Response time	Entry application	Service class
○	Account Query	00.000 average		Account Queries
○	Default - IBM DB2 Universal Database	00.000 average		Mortgage Rates
○	Default - IBM Webserving Plugin	00.000 average		EWLM Service Class
○	Default - WebSphere	00.000 average		WebSphere EJB Service Class
○	EWLM Default Application Transaction Class	00.000 average		EWLM Service Class
○	NorthEast	00.000 average		Brokerage Buyers
○	NorthWest	00.000 average		Brokerage Buyers
○	Stock Query	00.000 average		Stock quotes

Page 1 of 1 Total: 8 Filtered: 8 Displayed: 8

Figure 5-4 Transaction classes monitor at the EWLM Control Center

To identify an edge application in our environment we need to follow these steps:

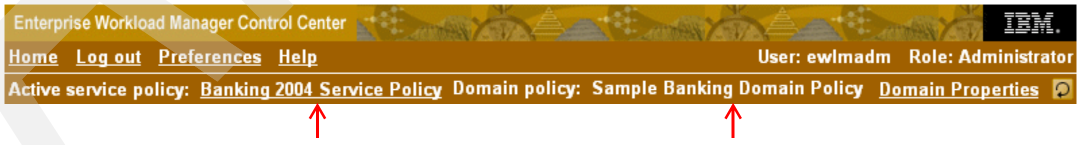
1. Deploy Sample Banking Domain Policy.
2. Run the workload to Trade3 and Plants by WebSphere.
3. Identify an edge server.

Deploy Sample Banking Domain Policy

To deploy Sample Banking Domain Policy:

1. Log in to the EWLM Control Center as a user who has the administrator role. In our case it is user ewlmadm.
2. Click **Domain policies** on the navigation panel and select **Sample Banking Domain Policy**.
3. Select **Deploy** from the action list and click **Go**.
4. Select **Banking 2004 Service Policy** to activate and click **Deploy**.

   You can verify deployed domain policy and activated service policy at the bottom of the EWLM Control Center banner area as shown in Figure 5-5.


Enterprise Workload Manager Control Center

User: ewlmadm Role: Administrator

Active service policy: Banking 2004 Service Policy Domain policy: Sample Banking Domain Policy Domain Properties

Figure 5-5 Current domain policy and active service policy

Run the workload to Trade3 and Plants by WebSphere

To make EWLM detect a work request and classify it to transaction class, we have to run some workload to the Trade3 and Plants by WebSphere applications. We use WebSphere Studio Workload Simulator for z/OS and OS/390 to generate the workload.

Our workloads have the following properties:

- Total number of concurrent users is 300.

- A user session for Trade3 includes:
  - Log into and log out from Trade3
  - Go to home page of Trade3
  - Update account information
  - Register new account to Trade3
  - Buy, sell, and quote stocks
  - View portfolio
- A user session for Plants by WebSphere includes:
  - Log into Plants by WebSphere
  - Go to each section: Flowers, Fruits & Vegetables, Trees and Accessories
  - Buy some stuff from each section
  - View shopping cart
  - Check out and log out from Plants by WebSphere

We do not provide details on how to configure and use this product. For more information, refer to the IBM WebSphere Studio Workload Simulator for z/OS and OS/390 home page at:

```
http://www.ibm.com/software/awdtools/studioworkloadsimulator/
```

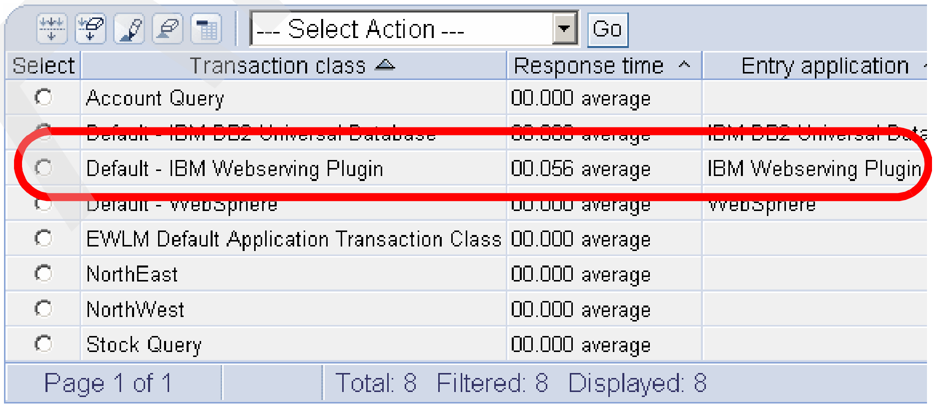
Note: You have to run the workload longer than the interval preference specified in the EWLM Control Center in order to detect your edge application successfully. Refer to 4.1, "EWLM Control Center overview" on page 78 for more information.

Identify an edge server

Now we can identify an edge application using the application topology view:

1. Log in to the EWLM Control Center as the user who has monitor role. In our environment this is user ewlmmon. However, users with administrator and operator role, like ewlmadm and ewlmops, can also view the monitor sections.
2. Click **Transaction classes** on the navigation panel.

   We can see that only the Default - IBM Webserving Plugin transaction class has a response time. This represents the general classification filter for a work request whose entry application is IBM HTTP server, as shown in Figure 5-6. If you were able to get a response time for either the WebSphere or DB2 transaction class this would then represent your entry application.



Select	Transaction class	Response time	Entry application
○	Account Query	00.000 average	
○	Default - IBM DB2 Universal Database	00.000 average	IBM DB2 Universal Data
○	Default - IBM Webserving Plugin	00.056 average	IBM Webserving Plugin
○	Default - WebSphere	00.000 average	WebSphere
○	EWLM Default Application Transaction Class	00.000 average	
○	NorthEast	00.000 average	
○	NorthWest	00.000 average	
○	Stock Query	00.000 average	

Page 1 of 1 Total: 8 Filtered: 8 Displayed: 8

Figure 5-6 Transaction class which receives work request

We have identified our edge application and want to view the application topology of this transaction class. We issue the following:

3. Select **Default - IBM Webserving Plugin**.
4. Select **Application Topology** from the action list and click **Go**.

A new browser window appears and you may see an application topology like the one we show in Figure 5-7. This is a topology view of both Trade3 and Plants by WebSphere at this point. The edge application is the IBM HTTP Server that marks the left side bound of the topology. This application topology is determined by EWLM itself. All we did was run some transactions for Trade3 and Plants by WebSphere.

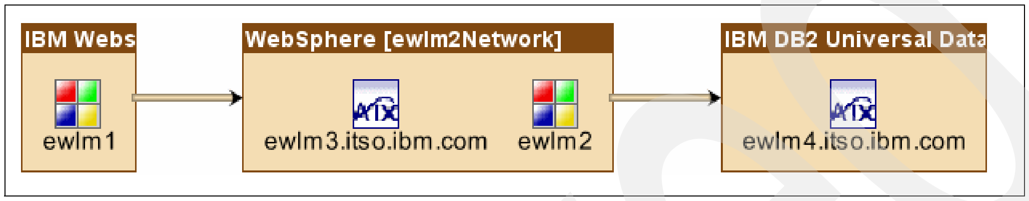


Figure 5-7 Application topology of Default - IBM Webserving Plugin transaction class

If your edge server is WebSphere Application Server you see an application topology as shown in Figure 5-8.

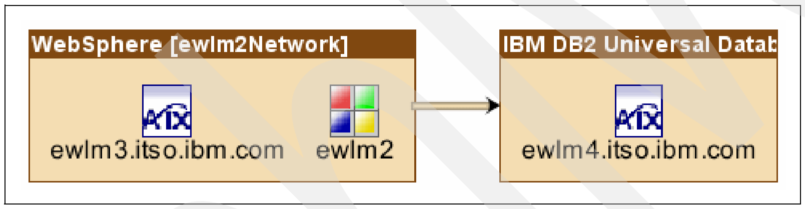


Figure 5-8 Application topology sample of Default - WebSphere transaction class

If your edge application is DB2, the Default - IBM DB2 Universal Database transaction class receives work requests. The application topology looks as shown in Figure 5-9.

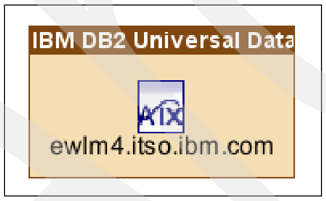


Figure 5-9 Application topology sample of Default - DB2 transaction class

Identifying transactions

In the previous section we provided you with a method to identify your edge application. In our environment we found that the edge application is the IBM HTTP Server. We now attempt to identify transactions using the IBM HTTP Server access log file in order to classify transactions.

Note: The steps described in the following discussion may not be suitable for your Web application. Each customer environment has unique implementations as well as requirements. Take those needs into account when you identify transactions.

Web applications provide business functions to customers by processing their HTTP requests from a Web browser by invoking servlets or JSPs and returning results to the Web browser. For example, when the client requests, the application server forwards it to an appropriate servlet using a requested URI which might be:

```
/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. Thus we can use them to identify transactions. They are logged in the IBM HTTP Server access log file.

We transfer the IBM HTTP Server access log file from the IBM HTTP Server (Windows) to our Linux machine using FTP, because Linux has good text processing utilities such as **grep**, **awk**, **sed** or **perl** and we are familiar with them. Click **Start** → **Programs** → **Accessories** → **Command Prompt** and follow the sequence shown in Example 5-1.

Example 5-1 Transfer IBM HTTP Server access log file from Windows to Linux

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\userid00>cd C:\Program Files\IBM HTTP Server 2.0\logs
C:\Program Files\IBM HTTP Server 2.0\logs>ftp ewlmdm1.itso.ibm.com
Connected to ewlmdm1.itso.ibm.com.
220 ewlmdm1.itso.ibm.com FTP server (Version wu-2.6.1-20) ready.
User (ewlmdm1.itso.ibm.com:(none)): ibmewlm
331 Password required for ibmewlm.
Password: ******
230 User ibmewlm logged in.
ftp> put access.log
200 PORT command successful.
150 Opening ASCII mode data connection for access.log.
226 Transfer complete.
ftp: 19659 bytes sent in 0.00Seconds 19659000.00Kbytes/sec.
ftp> bye
```

We use **grep** to print lines which contain a “?” character in the IBM HTTP Server access log file in order to remove requests for static content such as HTML pages or images. We found that both Trade3 and Plants by WebSphere seem to use the servlet name and `action` parameter to determine which business function is to be invoked as shown in Example 5-2.

Example 5-2 IBM HTTP Server access log file

```
[root]# grep '?' access.log
9.12.4.54 - - [14/May/2004:14:09:49 -0400] "GET /trade/app?action=quotes&symbols
=s:754 HTTP/1.1" 200 5177
9.12.4.54 - - [14/May/2004:14:09:49 -0400] "GET /trade/app?action=buy&symbol=s:1
00&quantity=21 HTTP/1.1" 200 3057
9.12.4.54 - - [14/May/2004:14:09:49 -0400] "GET /trade/app?action=quotes&symbols
=s:835 HTTP/1.1" 200 3057
9.12.4.54 - - [14/May/2004:14:09:49 -0400] "GET /trade/app?action=portfolio HTTP
/1.1" 200 3057
9.12.4.54 - - [14/May/2004:14:09:49 -0400] "GET /trade/app?action=quotes&symbols
=s:724 HTTP/1.1" 200 5182
9.12.4.54 - - [14/May/2004:14:09:49 -0400] "GET /trade/app?action=home HTTP/1.1"
200 3057
9.12.4.54 - - [14/May/2004:14:09:50 -0400] "GET /trade/app?action=home HTTP/1.1"
200 13411
```

```
9.12.4.142 - - [14/May/2004:14:09:50 -0400] "GET /PlantsByWebSphere/servlet/Imag
eServlet?action=getimage&inventoryID=A0001 HTTP/1.1" 200 75023
9.12.4.142 - - [14/May/2004:14:09:50 -0400] "GET /PlantsByWebSphere/servlet/Shop
pingServlet?action=shopping&category=3 HTTP/1.1" 200 8528
9.12.4.142 - - [14/May/2004:14:09:50 -0400] "GET /PlantsByWebSphere/servlet/Imag
eServlet?action=getimage&inventoryID=A0002 HTTP/1.1" 200 70368
9.12.4.54 - - [14/May/2004:14:09:50 -0400] "GET /trade/app?action=home HTTP/1.1"
 200 13415
9.12.4.142 - - [14/May/2004:14:09:50 -0400] "GET /PlantsByWebSphere/servlet/Imag
eServlet?action=getimage&inventoryID=A0003 HTTP/1.1" 200 74635
9.12.4.54 - - [14/May/2004:14:09:50 -0400] "GET /trade/app?action=account HTTP/1
.1" 200 3057
9.12.4.142 - - [14/May/2004:14:09:50 -0400] "GET /PlantsByWebSphere/servlet/Imag
eServlet?action=getimage&inventoryID=A0004 HTTP/1.1" 200 77269
9.12.4.142 - - [14/May/2004:14:09:50 -0400] "GET /PlantsByWebSphere/servlet/Imag
eServlet?action=getimage&inventoryID=A0005 HTTP/1.1" 200 76553
```

We created a simple shell script named **analyzelog** as shown in Example 5-3. It extracts and prints out the context root, servlet name, and action parameter in tab-delimited format from the IBM HTTP Server access log file.

Example 5-3 The analyzelog script

```
#!/bin/sh

LOGFILE=$1

awk '/\?/{print $7}' ${LOGFILE} |
perl -pe 's/\/([^\/]*)\/([^\?]*)\?.*action=([^&]*)&?.*/$1\t$2\t$3/' | sort | uniq

exit 0
```

Example 5-4 shows how the **analyzelog** script works. We found that we can separate work requests by context root and `action` parameter, instead of servlet name.

Example 5-4 Extracting context root, servlet name and action parameter

```
[root]# ./analyzelog access.log
PlantsByWebSphere       servlet/AccountServlet  account
PlantsByWebSphere       servlet/AccountServlet  login
PlantsByWebSphere       servlet/AccountServlet  register
PlantsByWebSphere       servlet/ImageServlet    getimage
PlantsByWebSphere       servlet/ShoppingServlet gotocart
PlantsByWebSphere       servlet/ShoppingServlet initcheckout
PlantsByWebSphere       servlet/ShoppingServlet orderinfodone
PlantsByWebSphere       servlet/ShoppingServlet productdetail
PlantsByWebSphere       servlet/ShoppingServlet register
PlantsByWebSphere       servlet/ShoppingServlet shopping
trade   app     account
trade   app     buy
trade   app     home
trade   app     logout
trade   app     portfolio
trade   app     quotes
trade   app     register
trade   app     sell
trade   app     update_profile
```

After that, we look into which `action` parameter corresponds to which customer transaction in both Trade3 and Plants by WebSphere by executing them manually using Internet Explorer, and collect the result as shown in Table 5-10. As most `action` parameters represent business functions explicitly, you can predict the usage of them, but we recommend that you verify all customer's usage manually or using interviews with the application designer or application developer. This usage information is important when you map the transaction classes to the service class described in "Mapping transaction classes to service class" on page 136 or group the service classes into the workload described 5.4.3, "Grouping service classes into a workload" on page 138.

Table 5-10 List of identified transactions for Trade3 and Plants by WebSphere

Application name	Action parameter	Customer's usage
Trade3	account	View account information
	buy	Buy stocks
	home	Go to home page
	logout	Log out from Trade3
	portfolio	View portfolio
	quotes	Quote stocks
	register	Register to Trade3
	sell	Sell stocks
	update_profile	Update account profile
Plants by WebSphere	addtocart	Add product to shopping cart
	getimage	Get products' image
	gotocart	View shopping cart
	initcheckout	Start checkout
	login	Log in to Plants by WebSphere
	productdetail	Show product detail information
	register	Register to Plants by WebSphere
	shopping	View procucts by category

Tip: You can use action mappings defined in `struts-config.xml` to identify transactions if your Web application uses the Apache Struts Web Application Framework. For more information about this framework, refer to the Apache Struts Web Application Framework homepage at:

`http://jakarta.apache.org/struts/`

Note: The workload we used in the ITSO configuration was mainly a three tier workload with the “edge” hop as HTTP and we used the default definitions for workload that could be generated on the WebSphere Application Server or DB2 application, such as the WebSphere Admin activity. In your configuration, you might have workloads that have either WebSphere or DB2 as an edge hop. In such a situation, the filters for the classification of your transaction will be different and will depend upon the application.

For the WebSphere Application Server you can use:

- EJB Name
- EWLM:URI for servlets
- Port
- QueryString

For DB2 you can use:

- DB2 Platform
- Database Name
- Program Name
- Client Protocol
- Application Hostname
- System Auth Id
- Execution Id
- Application Id

Defining transaction classes

A transaction class defines a group of similar transactions; they are combined into a service class for reporting. The work request can be identified by a pattern or by a specific identifier called filter defined within the transaction class.

Although we could have defined a transaction class at a more generic level, we defined transaction classes for each identified transaction. Table 5-11 shows our definition of transaction classes. Refer to “Naming convention” on page 121 for details about naming rules we used in this table.

Table 5-11 Definition of transaction classes

Application name	Action parameter	Transaction class name	Transaction class description
Trade3	account	TC_Trade3_account	View account information
	buy	TC_Trade3_buy	Buy stocks
	home	TC_Trade3_home	Go to home page
	logout	TC_Trade3_logout	Log out from Trade3
	portfolio	TC_Trade3_portfolio	View portfolio
	quotes	TC_Trade3_quotes	Quote stocks
	register	TC_Trade3_register	Register to Trade3
	sell	TC_Trade3_sell	Sell stocks
	update_profile	TC_Trade3_update_profile	Update account profile
	N/A	TC_Trade3_general	General Transaction class for Trade3.
Plants by WebSphere	addtocart	TC_Plants_addtocart	Add products to cart
	getimage	TC_Plants_getimage	Get products' image
	gotocart	TC_Plants_gotocart	View shopping cart
	initcheckout	TC_Plants_initcheckout	Start checkout
	login	TC_Plants_login	Log in to Plants
	productdetail	TC_Plants_productdetail	Show product detail information
	register	TC_Plants_register	Register to Plants
	shopping	TC_Plants_shopping	View products by category
	N/A	TC_Plants_general	General transaction class for Plants

Note that `TC_Trade3_general` and `TC_Plants_general` do not correspond to any `action` parameter of the query string. These transaction classes are used for work requests which correspond to requests for static content such as HTML pages or images. If, for example, `home` transaction does not need to be monitored by EWLM, remove the definition of `TC_Trade3_home` and it will be classified to `TC_Trade3_general`. We discuss this general transaction class in more detail in "Defining filters for transaction classes" on page 141.

Mapping transaction classes to service class

A service class is a group of transaction classes that have similar performance goals or business requirements. We define eleven service classes and map our transaction classes to these service classes. This is shown in Table 5-12.

As an example, we assume that `TC_Trade3_buy` and `TC_Trade3_sell` should have the same performance goal since these transaction classes represents similar customer usage: buy stocks and sell stocks. We verify the validity of these groupings in 5.6.2, "Verifying service class definition" on page 148.

Table 5-12 Transaction classes mappings to service class

Transaction class name	Service class name	Service class description
TC_Trade3_buy	SC_Trade3_shopping	Shopping service class for Trade3.
TC_Trade3_sell		
TC_Trade3_account	SC_Trade3_account	Account service class for Trade3.
TC_Trade3_register	SC_Trade3_profile	Profile service class for Trade3.
TC_Trade3_update_profile		
TC_Trade3_home	SC_Trade3_default	Default service class for Trade3.
TC_Trade3_logout		
TC_Trade3_portfolio	SC_Trade3_portfolio	Portfolio service class for Trade3.
TC_Trade3_quotes	SC_Trade3_quotes	Quotes service class for Trade3.
TC_Plants_login	SC_Plants_login	Login service class for Plants.
TC_Plants_register		
TC_Plants_getimage	SC_Plants_default	Default service class for Plants.
TC_Plants_productdetail		
TC_Plants_addtocart	SC_Plants_shopping	Shopping service class for Plants.
TC_Plants_gotocart		
TC_Plants_initcheckout		
TC_Plants_shopping		
TC_Trade3_general	SC_Trade3_general	General service class for Trade3.
TC_Plants_general	SC_Plants_general	General service class for Plants.

5.4.2 Identifying the processes that request business functions

The other step of the design phase is to identify and to classify processes. As described in "Process class details" on page 108, a process class is similar to a transaction class with the difference that it defines a group of system processes intended for reporting and management according to a service class. All non-ARM instrumented applications or middleware must be monitored as process classes.

This step is quite simple since processes are static applications and were already identified in 5.2.3, "Identifying all applications" on page 124, while transactions are dynamic applications.

This step involves the following:

1. Identifying processes
2. Defining process classes
3. Mapping process classes to service class

Identifying processes

Identifying processes is a relatively easy step compared to identifying transactions. In many cases, you can create a list of processes like the one we show in Table 5-13 by copying standalone program lists like the one in Table 5-5 described in 5.2.3, “Identifying all applications” on page 124.

If you have a high CPU utilization problem and would like to monitor processes which cause high CPU usage, you can find the processes using operating system specific process monitors such as **`ps`**, **`vmstat`**, **`svmon`** command, or Windows Task Manager, and add to this table.

Table 5-13 List of processes

Application name	Description
Internet Explorer	Web browser process for WebSphere Application Server and EWLM administrators
Calc batch application	Batch application run on AIX for in-house reporting system

Defining process classes

We create a process class for each identified process as shown at Table 5-14. For our test environment we have two process classes, one for Internet Explorer on Windows and one for a batch application on AIX.

Table 5-14 Definitions of process classes

Application name	Process class name	Description
Internet Explorer	PC_InternetExplorer	Internet Explorer process class for Windows Server 2003.
Calc batch application	PC_Calc_Batch	Calc batch process class for AIX.

Mapping process classes to service class

We then map each process class to its own service class as shown in Table 5-15. Note that we create two new service classes for the process classes, since a service class can be assigned to either a transaction class or a process class. It cannot be assigned to both.

Table 5-15 Process classes mappings to service class

Process class name	Service class name	Description
PC_InternetExplorer	SC_InternetExplorer	Internet Explorer service class for Windows Server 2003.
PC_Calc_Batch	SC_Calc_Batch	Calc batch service class for AIX.

5.4.3 Grouping service classes into a workload

Workload is a group of service classes for the purpose of reporting. Grouping service classes into workload is optional, but it is useful when you use EWLM reporting and monitoring functions.

At this step, we define a set of service classes as our workload to be monitored as shown in Table 5-16. This takes into consideration the purpose of our business scenario.

Table 5-16 Definition of workloads

Service class name	Workload name	Workload description
SC_Trade3_shopping	WL_shopping	Shopping workload for Trade3 and Plants.
SC_Plants_shopping		
SC_Trade3_default	WL_default	Default workload for Trade3 and Plants.
SC_Plants_default		
SC_Trade3_account	WL_Trade3_others	General workload for Trade3.
SC_Trade3_profile		
SC_Trade3_portfolio		
SC_Trade3_quotes		
SC_Plants_login	WL_Plants_others	General workload for Plants.

5.4.4 Setting service class goals

The final steps of the design phase are to set the performance goals and business importance of each service class. The performance goal and business importance describe how important each service class is to our business. The service policy is a set of these definitions as described at 4.3.1, “Service policies” on page 102.

These values play a factor only when a service class is not achieving its goal. 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 a performance goal as follows:

- *Percentile Response* - Specifies a time in which a percentile of work requests should complete.
- *Average Response* - Specifies an average amount of time in which work requests should complete.
- *Velocity Goal* - Specifies how fast work should run relative to other work when ready, without being delayed for EWLM managed resources.
- *Discretionary Goal* - Receives what’s left of the EWLM managed resources after all the other goals have been met. This goal type does not have any importance nor goal definition.

For *importance* you can choose between the following values. Be aware that the importance value is a business question reflecting how important it is to your business that the goal is achieved:

- Highest
- High
- Medium
- Low
- Lowest

Important: The best starting point for defining the transaction response time goal is by using the values specified in a Service Level Agreement. After all, that is what EWLM is trying to do – help your business meet the end-to-end business goal. If an SLA has not been defined, the next best thing would be to use the response time results from a workload simulator tool, such as WebSphere Studio Workload Simulator, Load runner, SilkPerformer, Tivoli® monitoring, and so forth. Finally, if those tools are not available, a temporary solution would be to create testing service classes for each transaction class and map them one by one to obtain the response time for each transaction class. Once response times are known, the transaction classes can be redefined and grouped to the appropriate production service class.

In Table 5-17 we show our first service policy definition. You can see that we specify the same importance and goal for all transaction and related service classes. This is because we do not have any Service Level Agreements and, therefore, no defined goal which needs to be achieved. We insert these definitions as the initial goal, run the workload and view the achieved response time. We then redefine the goal which we want these classes to achieve.

T

Note: Current EWLM implementation does not control operating system or hardware resources according to these importance or performance goals except through integration with Load Balancers. These goals are only used for monitoring and reporting purposes. Operating system or hardware management capabilities will be enhanced as the product evolves, with a focus on automated provisioning.

Table 5-17 Our first definition of service policy

Service class name	Importance	Goal type	Goal definition
SC_Trade3_shopping	Medium	Average Response Time	5 sec.
SC_Trade3_account	Medium	Average Response Time	5 sec.
SC_Trade3_profile	Medium	Average Response Time	5 sec.
SC_Trade3_default	Medium	Average Response Time	5 sec.
SC_Trade3_portfolio	Medium	Average Response Time	5 sec.
SC_Trade3_quotes	Medium	Average Response Time	5 sec.
SC_Plants_login	Medium	Average Response Time	5 sec.
SC_Plants_default	Medium	Average Response Time	5 sec.
SC_Plants_shopping	Medium	Average Response Time	5 sec.
SC_Trade3_general	N/A	Discretionary	N/A
SC_Plants_general	N/A	Discretionary	N/A
SC_InternetExplorer	Medium	Velocity	Fast
SC_Calc_Batch	Medium	Velocity	Slowest

5.5 Classifying the work

In this phase, we create classification filters first and gather them into our domain policy using the EWLM Control Center.

This phase includes these steps:

1. Defining classification filters for each transaction and process class
2. Creating the domain policy at the EWLM Control Center

5.5.1 Defining classification filters for each transaction and process class

As mentioned before, transaction and process classes are a named set of classification filters for mapping work requests to service classes based on transaction or process characteristics as defined by the middleware or operating system. If a work request's properties match the filter, the work request is classified to the service class. We create classification filters for all transaction classes and process classes. For more information on how filter works, refer to 4.2.1, "Classifying your workload" on page 88.

Defining filters for transaction classes

The definition of filters for transaction classes contains the following attributes:

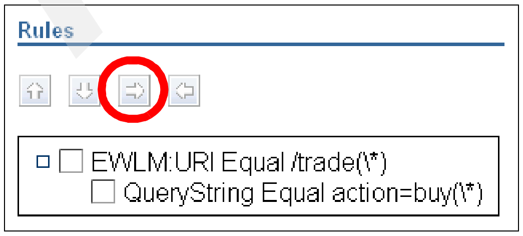
- *Position* - Specifies an order for EWLM to use the transaction classes.

  Work requests are compared to the positional filters specified in the transaction class, starting at position one. We have two general transaction classes, *TC_Trade3_general* and *TC_Plants_general,* which classify the work request for each Web application that has no action parameter. These transaction classes have higher numbered positions than any other transaction classes.

  Note: Use high numbered positions for filters that are more general and low numbered position for filters that are more specific. The highest numbered position specifies the default transaction class in position that uses the default service class. Transaction classes are applied on a first match basis for transactions. The recommendation that transaction classes for the most frequent transactions should be in the lower position is not always possible because of the first match approach. Because of first match, the most specific transaction classes need to be first.

- *Transaction class name* - Defined in "Defining transaction classes" on page 135.
- *Application* - For all of our transaction classes, our application is the IBM Webserving Plugin since all of our workloads come through the edge application, which is the IBM HTTP Server.
- *Filter type* - is specific to the application or platform that processes the work request.

  A plus (+) character placed in front of a filter type means this filter is nested inside the filter above. You can nest a filter rule inside of another filter rule by clicking the right arrow icon when you create rules at the EWLM Control Center. This is shown in Figure 5-10. If the workload matches the first rule, it compares to the nested rule. We use as first rule `EWLM:URI` to distinguish the Web application. As second rule we use a `QueryString` to identify the `action` parameter in the query string of the work request.



EWLM:URI Equal /trade(*)
QueryString Equal action=buy(*)

Figure 5-10 Nested rules

- *Filter operation* - `Equal` or `Not Equal`.
- *Filter value* - Specifies a value to use for the rule.

  You can specify mask `(\?)` or wildcard `(\*)` in the filter value. The mask can be used to represent any single character. The wildcard can be used to represent a series of any characters. Filter value cannot contain a leading question mark in the query string. Note that our general transaction classes, `TC_Trade3_general` and `TC_Plants_general`, have only `EWLM:URI` filter type in order to classify work requests which do not have `action` parameter in the query string.

  Note: Wildcard `(\*)` must be used at the end of a filter value because it represents multiple characters.

Table 5-18 shows our definition of filters for our transaction classes. These are done similarly for all transaction classes as described in the previous example.

Table 5-18 Definition of filters for transaction classes

Position	Transaction class name	Application	Filter type	Filter operation	Filter value
1	TC_Trade3_buy	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	action=buy(*)
2	TC_Trade3_sell	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	action=sell(*)
3	TC_Trade3_account	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	action=account(*)
4	TC_Trade3_register	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	action=register(*)
5	TC_Trade3_update_profile	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	action=update_profile(*)
6	TC_Trade3_home	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	action=home(*)
7	TC_Trade3_logout	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	action=logout(*)
8	TC_Trade3_portfolio	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	action=portfolio(*)
9	TC_Trade3_quotes	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	action=quotes(*)
10	TC_Plants_login	IBM Webserving Plugin	EWLM:URI	Equal	/PlantsByWebSphere(*)
			+ QueryString	Equal	action=login(*)
11	TC_Plants_register	IBM Webserving Plugin	EWLM:URI	Equal	/PlantsByWebSphere(*)
			+ QueryString	Equal	action=register(*)

Position	Transaction class name	Application	Filter type	Filter operation	Filter value
12	TC_Plants_getimage	IBM Webserving Plugin	EWLM:URI	Equal	/PlantsByWebSphere(*)
			+ QueryString	Equal	action=getimage(*)
13	TC_Plants_productdetail	IBM Webserving Plugin	EWLM:URI	Equal	/PlantsByWebSphere(*)
			+ QueryString	Equal	action=productdetail(*)
14	TC_Plants_gotocart	IBM Webserving Plugin	EWLM:URI	Equal	/PlantsByWebSphere(*)
			+ QueryString	Equal	action=gotocart(*)
15	TC_Plants_initcheckout	IBM Webserving Plugin	EWLM:URI	Equal	/PlantsByWebSphere(*)
			+ QueryString	Equal	action=initcheckout(*)
16	TC_Plants_shopping	IBM Webserving Plugin	EWLM:URI	Equal	/PlantsByWebSphere(*)
			+ QueryString	Equal	action=shopping(*)
17	TC_Trade3_general	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
18	TC_Plants_general	IBM Webserving Plugin	EWLM:URI	Equal	/PlantsByWebSphere(*)

Defining classification filters for process classes

The definition of filters for process classes has similar attributes to those of filters for transaction classes. These attributes are:

- Position
- Process class name
- Platform
- Filter type
- Filter operation
- Filter value

Although in our setup we specify each process by its executable path, you can specify the process by user name, group name, and so on. Refer to 4.2.1, "Classifying your workload" on page 88 for more information. The definition of filters for our process classes is shown in Table 5-19.

Table 5-19 Definition of filters for process classes

Position	Process class name	Platform	Filter type	Filter operation	Filter value
1	PC_InternetExplorer	Windows Server 2003	ExecutablePath	Equals	C:\Program Files\Internet Explorer\IEXPLORE.EXE
2	PC_Calc_Bath	AIX 5.1.6 ML03	ExecutablePath	Equals	/usr/sbin/calc.batch

5.5.2 Creating the domain policy at the EWLM Control Center

You are now ready to start the implementation of the domain policy. We assume that you already understand how to create domain policy; therefore, this section does not describe how to do it screen-by-screen. Refer to "Building a domain policy" on page 92 for a detailed description of the steps to create a domain policy.

5.6 Verification

In this phase, we verify our new domain policy and defined service policy.

Before you deploy a new domain policy into your production environment, you have to verify that there is no mistake or error in your domain policy. Possible mistakes are often found at:

- Work request classification
- Service class definition

When all the following steps are completed successfully, you can use the EWLM monitoring and reporting functions.

5.6.1 Verifying transaction request classification filters

There may be some mistakes in your classification filter no matter how carefully you implemented it. EWLM does not provide a tool to show what transaction request is classified to what transaction class, thus you have to check manually using the EWLM Control Center. Our verification steps are follows:

1. Run the workload again.
2. Review classification result using Transaction Classes monitor.
3. Redefine the classification filter.

Run the workload again

We run the workload to Trade3 and Plants by WebSphere again using WebSphere Studio Workload Simulator for z/OS and OS/390. We use the same workload for this step that we described in "Identifying an edge server of the work request" on page 128.

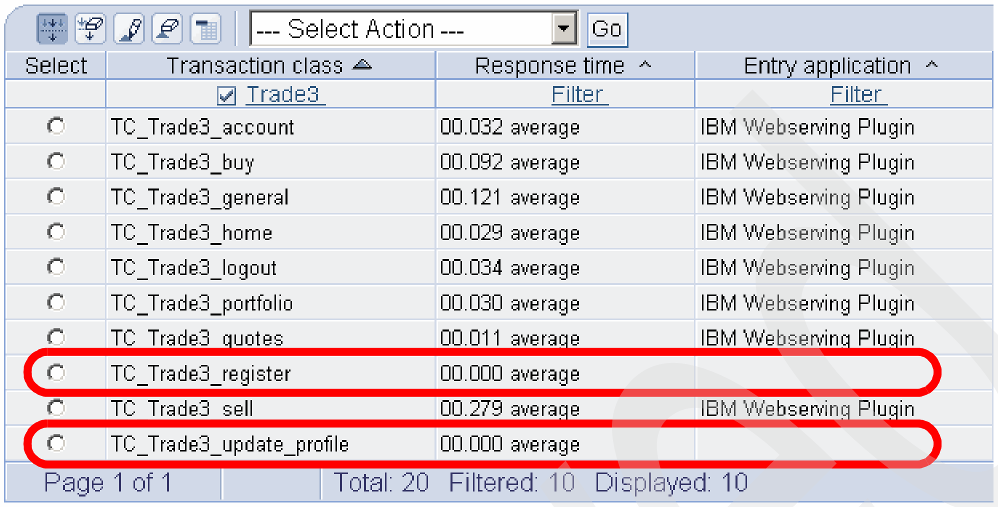
Note: You have to run the workload longer than the interval specified in the EWLM Control Center preferences in order to make EWLM detect all work requests and classify them to transaction class successfully. Refer to 4.1, "EWLM Control Center overview" on page 78 for more information.

Review classification result using Transaction Classes monitor

Next, we review the Transaction Classes monitor to verify our classification filter. To see the monitor:

1. Log in to the EWLM Control Center as the user who has monitor role. In our case we log on as user ewlmmon.
2. Click **Transaction classes** on the navigation panel.

   Figure 5-11 shows the classification result. We found that the Response time value for two transaction classes, `TC_Trade3_register` and `TC_Trade3_update_profile`, shows `00.000 average`, which means these transaction classes do not receive any requests. 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.



--- Select Action --- Go

Select	Transaction class	Response time	Entry application
	☑ Trade3	Filter	Filter
○	TC_Trade3_account	00.032 average	IBM Webserving Plugin
○	TC_Trade3_buy	00.092 average	IBM Webserving Plugin
○	TC_Trade3_general	00.121 average	IBM Webserving Plugin
○	TC_Trade3_home	00.029 average	IBM Webserving Plugin
○	TC_Trade3_logout	00.034 average	IBM Webserving Plugin
○	TC_Trade3_portfolio	00.030 average	IBM Webserving Plugin
○	TC_Trade3_quotes	00.011 average	IBM Webserving Plugin
○	TC_Trade3_register	00.000 average	
○	TC_Trade3_sell	00.279 average	IBM Webserving Plugin
○	TC_Trade3_update_profile	00.000 average	

Page 1 of 1 Total: 20 Filtered: 10 Displayed: 10

Figure 5-11 Result of classification filter at Transaction Classes monitor

Redefine the classification filter

We can verify whether Trade3 receives the work request related to the actions or not using the IBM HTTP Server access log file. We transfer the IBM HTTP Server access log file to our Linux machine as we did at “Identifying transactions” on page 131.

Then, we use **grep** again to print lines which contain “?” and “`register`” or “`update_profile`” to verify if the IBM HTTP Server received the work request related to these actions. Example 5-5 and Example 5-6 show us that the work requests were coming to the application but were not classified to the appropriate transaction class. There may be some mistakes in our classification filter, therefore EWLM classifies these work requests to the general transaction class, `TC_Trade3_general`.

Example 5-5 Access log of register action for Trade3

```
[root]# grep '?' access.log | grep register | head -3
9.12.4.54 - - [17/May/2004:17:01:16 -0400] "GET /trade/app?Full+Name=first%3A176
+last%3A3209&snail+mail=1925+mystreet&email=ru%253A1111960%4038.com&user+id=ru%3
A1111960&passwd=222&confirm+passwd=yyy&money=1000000&Credit+Card+Number=123-fake
-ccnum-456&action=register HTTP/1.1" 200 13413
9.12.4.54 - - [17/May/2004:17:01:39 -0400] "GET /trade/app?Full+Name=first%3A573
+last%3A2840&snail+mail=337+mystreet&email=ru%253A2135272%4057.com&user+id=ru%3A
2135272&passwd=222&confirm+passwd=yyy&money=1000000&Credit+Card+Number=123-fake-
ccnum-456&action=register HTTP/1.1" 200 13413
9.12.4.54 - - [17/May/2004:17:01:46 -0400] "GET /trade/app?Full+Name=first%3A46+
last%3A2146&snail+mail=2122+mystreet&email=ru%253A7142375%4010.com&user+id=ru%3A
7142375&passwd=222&confirm+passwd=yyy&money=1000000&Credit+Card+Number=123-fake-
ccnum-456&action=register HTTP/1.1" 200 13413
```

Example 5-6 Access log of update_profile action for Trade3

```
[root]# grep '?' access.log | grep update_profile | head -3
9.12.4.54 - - [17/May/2004:17:01:17 -0400] "GET /trade/app?userID=uid%3A20&fulln
ame=rnd5114277&password=111&address=rndAddress&cpassword=xxx&creditcard=rndCC&em
ail=rndEmail&action=update_profile HTTP/1.1" 200 13601
9.12.4.54 - - [17/May/2004:17:01:26 -0400] "GET /trade/app?userID=uid%3A15&fulln
ame=rnd7123842&password=111&address=rndAddress&cpassword=xxx&creditcard=rndCC&em
```

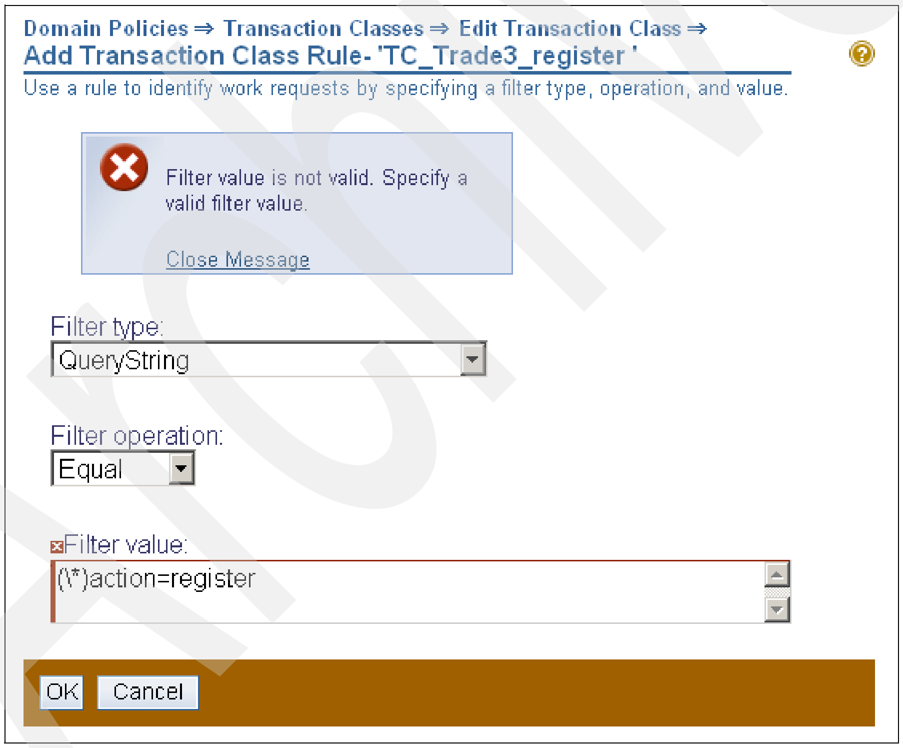
```
ail=rndEmail&action=update_profile HTTP/1.1" 200 13607
9.12.4.54 - - [17/May/2004:17:01:29 -0400] "GET /trade/app?userID=ru%3A486074&fu
llname=rnd4126159&password=111&address=rndAddress&cpassword=xxx&creditcard=rndCC
&email=rndEmail&action=update_profile HTTP/1.1" 200 10615
[root@ewlmdm1 access]#
```

We found that the query string of these work requests is a bit different than we thought when we created the classification filter at "Defining filters for transaction classes" on page 141. We assumed that all query strings for both Trade3 and Plants by WebSphere *begin* with `action=actionname` and defined filters value as `action=actionname(\*)` in Table 5-18 on page 142. However, query strings for `register` and `update_profile` actions *end* with `action=actionname`. Thus, `TC_Trade3_register` and `TC_Trade3_update_profile` could not classify any work requests.

We then fixed the classification filter. As described at "Defining filters for transaction classes" on page 141, a wildcard `(\*)` must be used at the end of a filter value, so you cannot specify such filters as:

```
(\*)action=register
(\*)action=update_profile
```

If you specify a filter that way in the Domain Manager Wizard, you may see an error message like that shown in Figure 5-12.


Domain Policies ⇒ Transaction Classes ⇒ Edit Transaction Class ⇒

Add Transaction Class Rule- 'TC_Trade3_register '

Use a rule to identify work requests by specifying a filter type, operation, and value.

Filter value is not valid. Specify a valid filter value.

Close Message

Filter type: QueryString

Filter operation: Equal

Filter value: (*)action=register

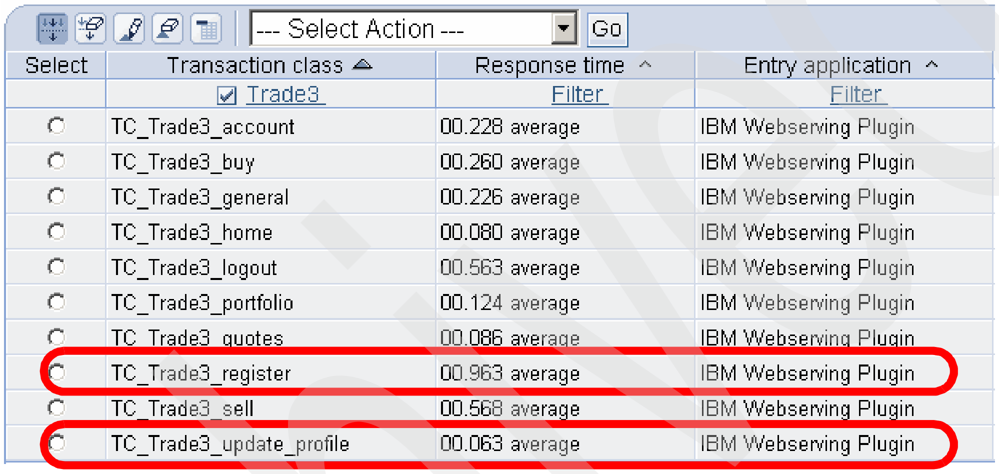
Figure 5-12 Invalid wild card usage

We corrected these filter definitions as shown in Table 5-20 and modified the transaction class implementation using the EWLM Control Center. The changed filter values are highlighted.

Table 5-20 Reviced definition of filters for transaction classes

Position	Transaction class name	Application	Filter type	Filter operation	Filter value
4	TC_Trade3_register	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	Full+Name=(*)
5	TC_Trade3_update_profile	IBM Webserving Plugin	EWLM:URI	Equal	/trade(*)
			+ QueryString	Equal	userID=(*)

Finally, we get the correct classification and therefore an entry in the Entry application column and a Response time as shown in Figure 5-13.



Select	Transaction class	Response time	Entry application
	☑ Trade3	Filter	Filter
○	TC_Trade3_account	00.228 average	IBM Webserving Plugin
○	TC_Trade3_buy	00.260 average	IBM Webserving Plugin
○	TC_Trade3_general	00.226 average	IBM Webserving Plugin
○	TC_Trade3_home	00.080 average	IBM Webserving Plugin
○	TC_Trade3_logout	00.563 average	IBM Webserving Plugin
○	TC_Trade3_portfolio	00.124 average	IBM Webserving Plugin
○	TC_Trade3_quotes	00.086 average	IBM Webserving Plugin
○	TC_Trade3_register	00.963 average	IBM Webserving Plugin
○	TC_Trade3_sell	00.568 average	IBM Webserving Plugin
○	TC_Trade3_update_profile	00.063 average	IBM Webserving Plugin

Figure 5-13 Result of revised classification filter at Transaction Classes monitor

Application topology revisited

At the point of "Identify an edge server" on page 130, Trade3 and Plants by WebSphere had the same application topology as shown in Figure 5-7 on page 131. But actually, they have different application topologies. As indicated in Table 5-5 on page 124, Plants by WebSphere is running with IBM HTTP Server and WebSphere Application Server, while Trade3 is running with IBM HTTP Server, WebSphere Application Server, and DB2 Universal Database. Now we can see this difference in the application topology view:

1. Log in to the EWLM Control Center as the user who has monitor role.

   Refer to 4.1, "EWLM Control Center overview" on page 78 for more information about EWLM Control Center roles.

2. Click **Transaction classes** on the navigation pane.
3. Select **TC_Trade3_login**.
4. Select **Application Topology** from the action list and click **Go**.

   You will see an application topology like the one pictured in Figure 5-14.

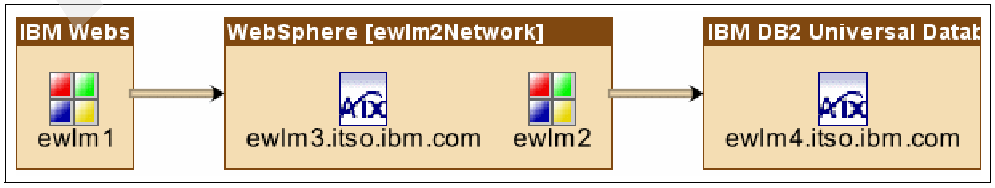


Figure 5-14 Application topology of TC_Trade3_login

5. Close the application topology view window and select **TC_Plants_login**.
6. Select **Application Topology** from the action list and click **Go** again.

   You will see an application topology like the one pictured in Figure 5-15. This demonstrates that Plants by WebSphere is:

   – Running with IBM HTTP Server and WebSphere Application Server.

   – Only deployed at ewlm2.itso.ibm.com while Trade3 is deployed at both ewlm2.itso.ibm.com and ewlm3.itso.ibm.com.

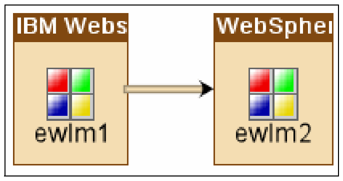
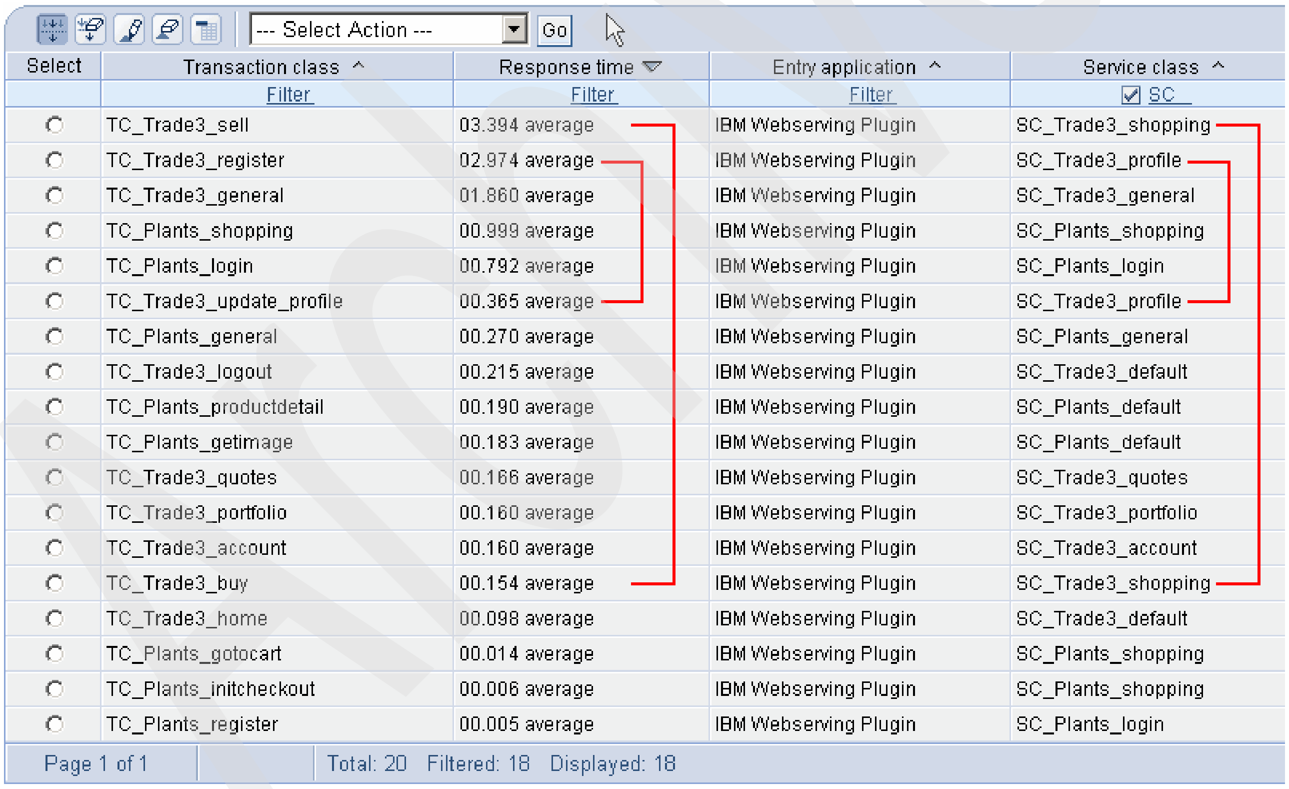


Figure 5-15 Application topology of TC_Plants_login

5.6.2 Verifying service class definition

We found that some transaction classes mapped to the same service class showed very different values in Response time, as shown in Figure 5-16.



Select	Transaction class ^	Response time ▽	Entry application ^	Service class ^
	Filter	Filter	Filter	✓ SC
◯	TC_Trade3_sell	03.394 average	IBM Webserving Plugin	SC_Trade3_shopping
◯	TC_Trade3_register	02.974 average	IBM Webserving Plugin	SC_Trade3_profile
◯	TC_Trade3_general	01.860 average	IBM Webserving Plugin	SC_Trade3_general
◯	TC_Plants_shopping	00.999 average	IBM Webserving Plugin	SC_Plants_shopping
◯	TC_Plants_login	00.792 average	IBM Webserving Plugin	SC_Plants_login
◯	TC_Trade3_update_profile	00.365 average	IBM Webserving Plugin	SC_Trade3_profile
◯	TC_Plants_general	00.270 average	IBM Webserving Plugin	SC_Plants_general
◯	TC_Trade3_logout	00.215 average	IBM Webserving Plugin	SC_Trade3_default
◯	TC_Plants_productdetail	00.190 average	IBM Webserving Plugin	SC_Plants_default
◯	TC_Plants_getimage	00.183 average	IBM Webserving Plugin	SC_Plants_default
◯	TC_Trade3_quotes	00.166 average	IBM Webserving Plugin	SC_Trade3_quotes
◯	TC_Trade3_portfolio	00.160 average	IBM Webserving Plugin	SC_Trade3_portfolio
◯	TC_Trade3_account	00.160 average	IBM Webserving Plugin	SC_Trade3_account
◯	TC_Trade3_buy	00.154 average	IBM Webserving Plugin	SC_Trade3_shopping
◯	TC_Trade3_home	00.098 average	IBM Webserving Plugin	SC_Trade3_default
◯	TC_Plants_gotocart	00.014 average	IBM Webserving Plugin	SC_Plants_shopping
◯	TC_Plants_initcheckout	00.006 average	IBM Webserving Plugin	SC_Plants_shopping
◯	TC_Plants_register	00.005 average	IBM Webserving Plugin	SC_Plants_login

Page 1 of 1 Total: 20 Filtered: 18 Displayed: 18

Figure 5-16 Wrong transaction class mapping

Since a service class is a group of transaction classes that have similar performance goals or business requirements, these mappings should be corrected. We changed these mappings and modified the service class implementation. The revised transaction classes mappings are shown at Table 5-21. Transaction classes that were changed are highlighted.

Table 5-21 Transaction classes mappings to service class revised

Transaction class name	Service class name
TC_Trade3_sell	SC_Trade3_shopping
TC_Trade3_register	
TC_Trade3_account	SC_Trade3_account
TC_Trade3_update_profile	SC_Trade3_profile
TC_Trade3_buy	
TC_Trade3_home	SC_Trade3_default
TC_Trade3_logout	
TC_Trade3_portfolio	SC_Trade3_portfolio
TC_Trade3_quotes	SC_Trade3_quotes
TC_Plants_login	SC_Plants_login
TC_Plants_register	
TC_Plants_getimage	SC_Plants_default
TC_Plants_productdetail	
TC_Plants_gotocart	SC_Plants_shopping
TC_Plants_initcheckout	
TC_Plants_shopping	
TC_Trade3_general	SC_Trade3_general
TC_Plants_general	SC_Plants_general

5.6.3 Defining and implementing service policy

According to our purpose for using EWLM, described in 5.1.1, “Business scenario” on page 121, we defined the service policy shown in Table 5-22 and implemented it using the EWLM Control Center.

Note that we do not have any basis or grounds for defining this service policy. Defining importance and goals deeply depends on the individual customer’s environment, as well as their unique requirements. This topic is covered in more in detail in 4.2.3, “Building a domain policy” on page 92.

Table 5-22 Definition of service policy

Service class name	Importance	Goal type	Goal definition
SC_Trade3_shopping	Medium	Percentile Response Time	90% in 10 sec.
SC_Trade3_account	Medium	Average Response Time	700 msec.
SC_Trade3_profile	Medium	Average Response Time	1 sec.
SC_Trade3_default	Medium	Average Response Time	1.2 sec.
SC_Trade3_portfolio	Medium	Average Response Time	500 msec.
SC_Trade3_quotes	Medium	Average Response Time	800 msec.

Service class name	Importance	Goal type	Goal definition
SC_Plants_login	Medium	Percentile Response Time	90% in 200 msec.
SC_Plants_default	Medium	Average Response Time	100 msec.
SC_Plants_shopping	Medium	Percentile Response Time	80% in 200 msec.
SC_Trade3_general	N/A	Discretionary	N/A
SC_Plants_general	N/A	Discretionary	N/A
SC_InternetExplorer	Medium	Velocity	Fast
SC_Calc_Batch	Medium	Velocity	Slowest

5.7 Reporting

In this section we want to discuss monitoring performance data using the EWLM Control Center. We concentrate on reports related to our example policy which we previously discussed. We want to focus on how to use the reports and monitors to find a performance problem or bottleneck within our application topology. This is not the only value of the monitors; for a general discussion on reporting functionality refer to 4.4, “Monitor” on page 104.

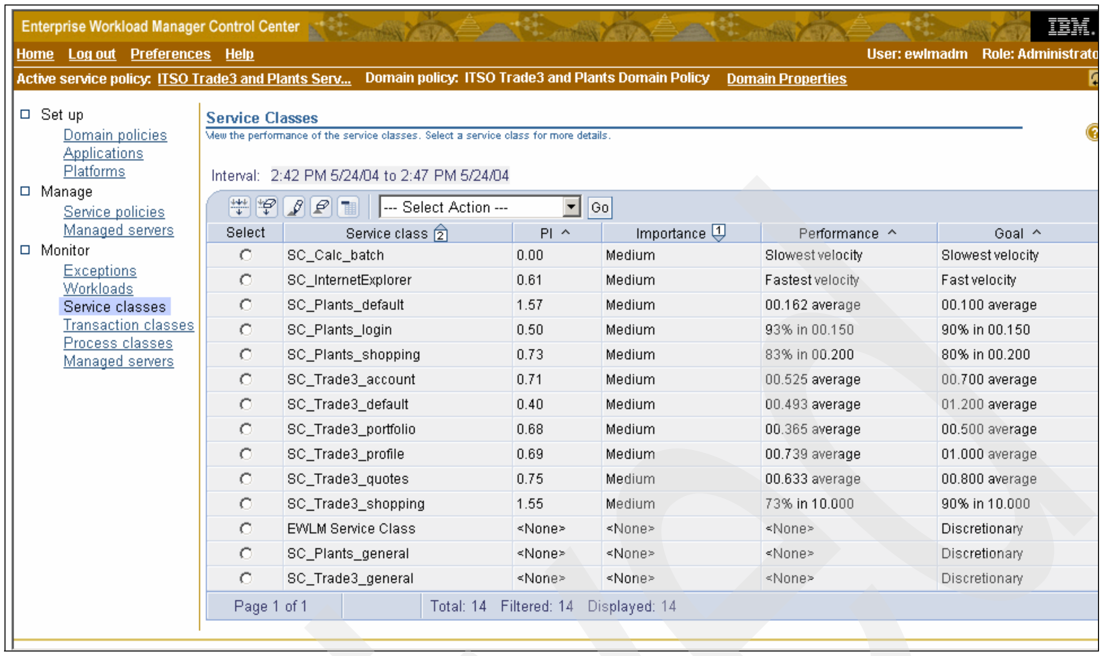
We include in our discussion on performance data the following reports:

- First-level analysis
- Topology view
- Details view

EWLM collects and aggregates performance statistics so that you can compare a high-level goal to the actual performance at all times. The EWLM Control Center provides a user-friendly way of viewing this information. EWLM captures relevant performance statistics on each OS instance within a management domain. It then reports an aggregated view of the real-time state of the domain to the EWLM domain manager. In the EWLM Control Center you can view detailed statistics of the actual results that were achieved, as opposed to the desired results. You can capture performance statistics in a tabular form for offline analysis or for integration into other performance reporting tools.

The available monitoring and reporting functions are useful to determine whether or not there is a performance problem for a service class. They also provide a view of the logical tiers and the application environment and help to determine if applications or servers are experiencing any problems. The statistics include detailed resource usage and delay information, as well as correlation to specific operating system processes that support each middleware application.

Make sure you have the Preference setting set to a reasonable value, for example one minute, for monitoring your service class goal achievement. A good starting point then is the overview of the assigned performance goals and the actual achievement as shown in Figure 5-17. For this you need to log in to the EWLM Control Center as a user with the monitor role like our user ewlmmon. Select **Service classes** from the monitor task list to view the performance based on the goals defined in the service policy. There are sorting and filtering options to customize the view. The values to look at are the performance index (PI), the importance, and the goal you have set, as well as the actual performance which is achieved.


Enterprise Workload Manager Control Center
IBM.
Home Log out Preferences Help
User: ewlmadm Role: Administrato
Active service policy: ITSO Trade3 and Plants Serv... Domain policy: ITSO Trade3 and Plants Domain Policy Domain Properties
Set up
Domain policies
Applications
Platforms
Manage
Service policies
Managed servers
Monitor
Exceptions
Workloads
Service classes
Transaction classes
Process classes
Managed servers
Service Classes
View the performance of the service classes. Select a service class for more details.
--- Select Action ---
Go

Interval: 2:42 PM 5/24/04 to 2:47 PM 5/24/04

Select	Service class	PI	Importance	Performance	Goal
○	SC_Calc_batch	0.00	Medium	Slowest velocity	Slowest velocity
○	SC_InternetExplorer	0.61	Medium	Fastest velocity	Fast velocity
○	SC_Plants_default	1.57	Medium	00.162 average	00.100 average
○	SC_Plants_login	0.50	Medium	93% in 00.150	90% in 00.150
○	SC_Plants_shopping	0.73	Medium	83% in 00.200	80% in 00.200
○	SC_Trade3_account	0.71	Medium	00.525 average	00.700 average
○	SC_Trade3_default	0.40	Medium	00.493 average	01.200 average
○	SC_Trade3_portfolio	0.68	Medium	00.365 average	00.500 average
○	SC_Trade3_profile	0.69	Medium	00.739 average	01.000 average
○	SC_Trade3_quotes	0.75	Medium	00.633 average	00.800 average
○	SC_Trade3_shopping	1.55	Medium	73% in 10.000	90% in 10.000
○	EWLM Service Class	<None>	<None>	<None>	Discretionary
○	SC_Plants_general	<None>	<None>	<None>	Discretionary
○	SC_Trade3_general	<None>	<None>	<None>	Discretionary

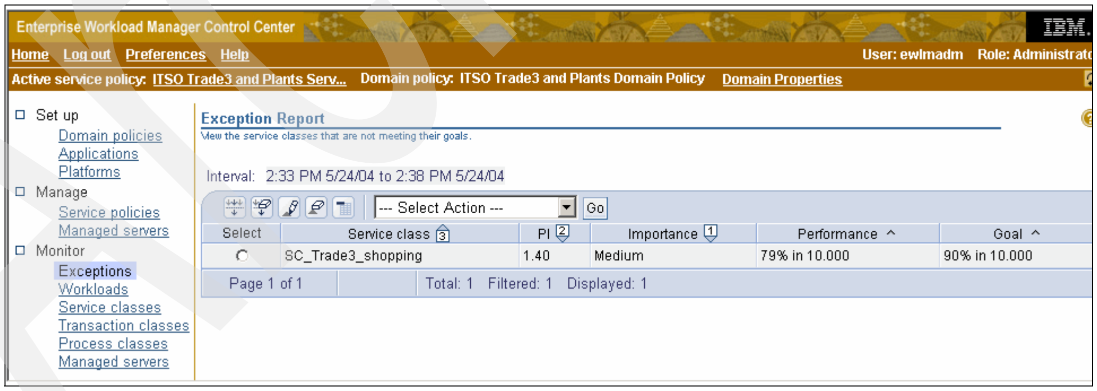
Page 1 of 1 Total: 14 Filtered: 14 Displayed: 14

Figure 5-17 Service class report to identify performance problems

If you click on the PI column header, the service classes are sorted by PI. This will help you focus on the rows having a PI greater the 1.00

First-level analysis

To verify if the service classes are meeting their goals you can look at the performance index in this view. Alternatively, you can go to the Exceptions monitor. For this you need to select **Exceptions** from the monitor option to get a view as shown in Figure 5-18 on page 151. If there are no entries in this section all service classes are meeting the business response time goals you have set.


Enterprise Workload Manager Control Center
IBM.
Home Log out Preferences Help
User: ewlmadm Role: Administrato
Active service policy: ITSO Trade3 and Plants Serv... Domain policy: ITSO Trade3 and Plants Domain Policy Domain Properties
Set up
Domain policies
Applications
Platforms
Manage
Service policies
Managed servers
Monitor
Exceptions
Workloads
Service classes
Transaction classes
Process classes
Managed servers
Exception Report
View the service classes that are not meeting their goals.
--- Select Action ---
Go

Interval: 2:33 PM 5/24/04 to 2:38 PM 5/24/04

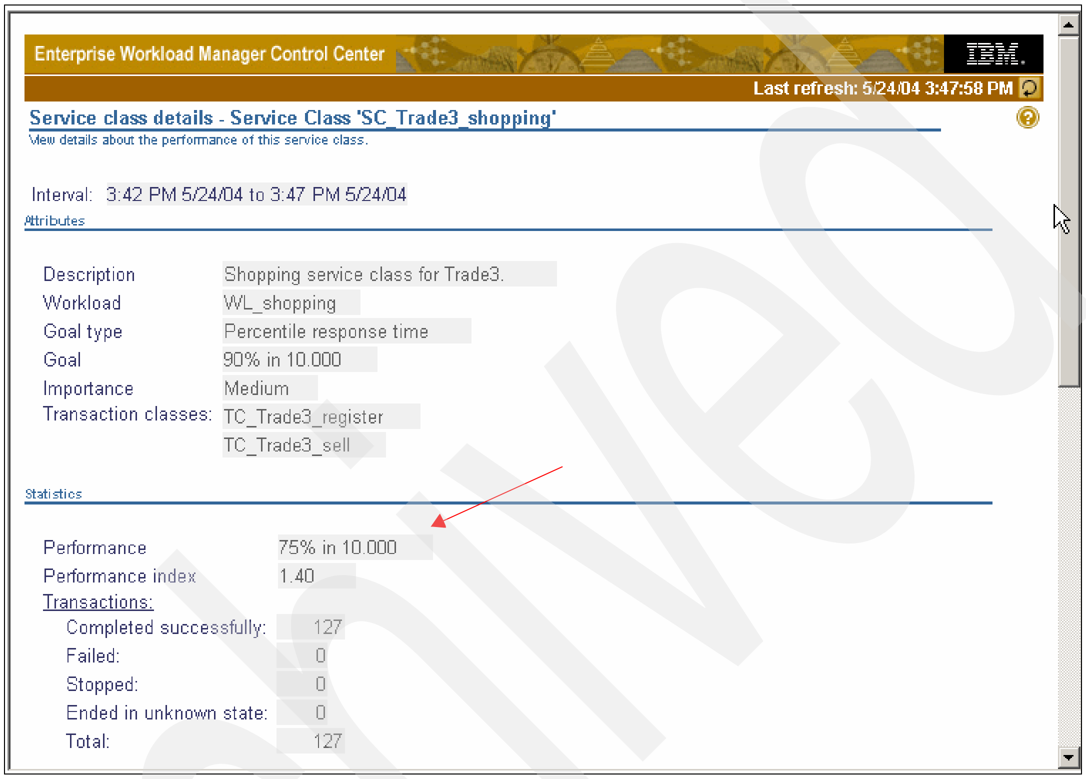
Select	Service class	PI	Importance	Performance	Goal
○	SC_Trade3_shopping	1.40	Medium	79% in 10.000	90% in 10.000

Page 1 of 1 Total: 1 Filtered: 1 Displayed: 1

Figure 5-18 Exceptions report to identify performance problems

We have service class `SC_Trade3_shopping` listed in this exception section because it has a performance index greater than one, which means the business goal has been missed. Either we have set an unrealistic goal or we need to do some further investigation to find out what caused the bad performance numbers.

For this we select the service class `SC_Trade3_shopping` and choose **Service class details** from the action bar. We press the **Go** button and are able to view further details on failed, successfully completed, and total number of transactions. This is shown in Figure 5-19. The Statistics section shows that all transactions are completed successfully, but that only 75% of the total number is achieving the target response time of 10.00 seconds, while the desired goal was that 90% of the total transactions be completed within 10.00 seconds. By scrolling down, we could see a graphical display of the historical response time. Since it does not show any indication of a problem, we did not include it here.



Enterprise Workload Manager Control Center
IBM.
Last refresh: 5/24/04 3:47:58 PM
Service class details - Service Class 'SC_Trade3_shopping'
View details about the performance of this service class.

Interval: 3:42 PM 5/24/04 to 3:47 PM 5/24/04

Attributes

Description	Shopping service class for Trade3.
Workload	WL_shopping
Goal type	Percentile response time
Goal	90% in 10.000
Importance	Medium
Transaction classes:	TC_Trade3_register
	TC_Trade3_sell

Statistics

Performance	75% in 10.000
Performance index	1.40
Transactions:	
Completed successfully:	127
Failed:	0
Stopped:	0
Ended in unknown state:	0
Total:	127

Figure 5-19 Viewing service class details to identify the performance problem

All transactions seem to be processed successfully, which means we need to look at some other performance problem indicators.

Topology view

We select service class `SC_Trade3_shopping` again from the option **Exceptions**, select **Application topology** from the action bar, and click **Go**. We expect to see the workflow going from the HTTP plug-in running on server `ewlm1` to the Web Server tier running on servers `ewlm2` and `ewlm3` and then go to the Database Server `ewlm4`. This is shown in Figure 5-20.

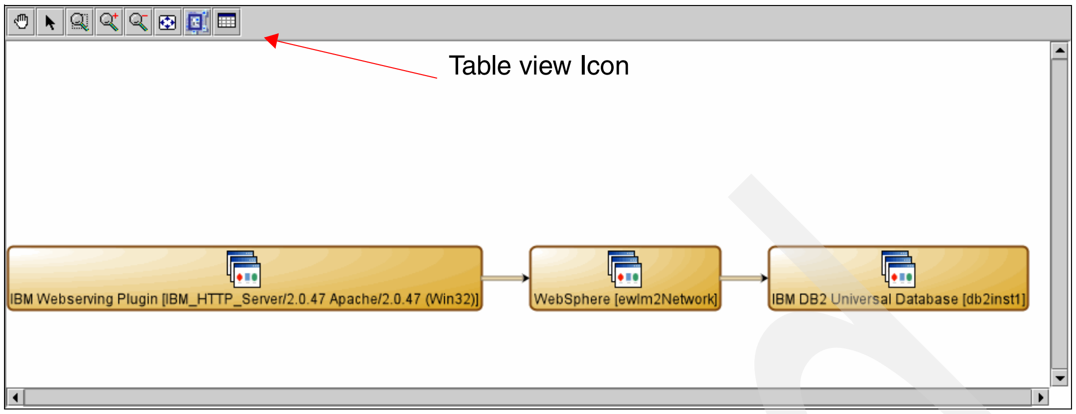
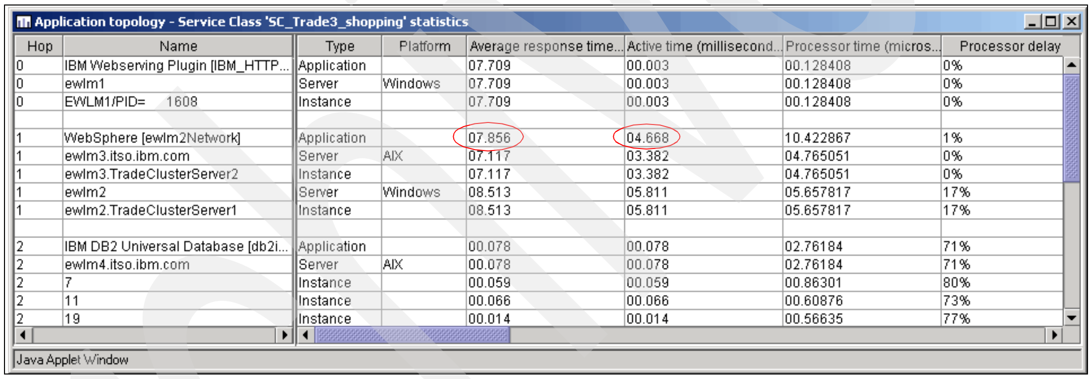


Table view Icon
IBM Webserving Plugin [IBM_HTTP_Server/2.0.47 Apache/2.0.47 (Win32)]
WebSphere [ewlm2Network]
IBM DB2 Universal Database [db2inst1]

Figure 5-20 Application topology for service class SC_Trade3_shopping

The next step after verifying that your application topology is correct is to look at the table view to find the application or the server experiencing problems. This view is shown in Figure 5-21. For each node, that is for each application, server, or instance, the tableview shows total response time, total active time, the system on which the instance is running, the number of completed, failed, and abended transactions, and some additional performance data.



Application topology - Service Class 'SC_Trade3_shopping' statistics

Hop	Name	Type	Platform	Average response time...	Active time (millisecond...	Processor time (micros...	Processor delay
0	IBM Webserving Plugin [IBM_HTTP...	Application		07.709	00.003	00.128408	0%
0	ewlm1	Server	Windows	07.709	00.003	00.128408	0%
0	EWLM1/PID= 1608	Instance		07.709	00.003	00.128408	0%
1	WebSphere [ewlm2Network]	Application		07.856	04.668	10.422867	1%
1	ewlm3.itso.ibm.com	Server	AIX	07.117	03.382	04.765051	0%
1	ewlm3.TradeClusterServer2	Instance		07.117	03.382	04.765051	0%
1	ewlm2	Server	Windows	08.513	05.811	05.657817	17%
1	ewlm2.TradeClusterServer1	Instance		08.513	05.811	05.657817	17%
2	IBM DB2 Universal Database [db2i...	Application		00.078	00.078	02.76184	71%
2	ewlm4.itso.ibm.com	Server	AIX	00.078	00.078	02.76184	71%
2	7	Instance		00.059	00.059	00.86301	80%
2	11	Instance		00.066	00.066	00.60876	73%
2	19	Instance		00.014	00.014	00.56635	77%

Java Applet Window

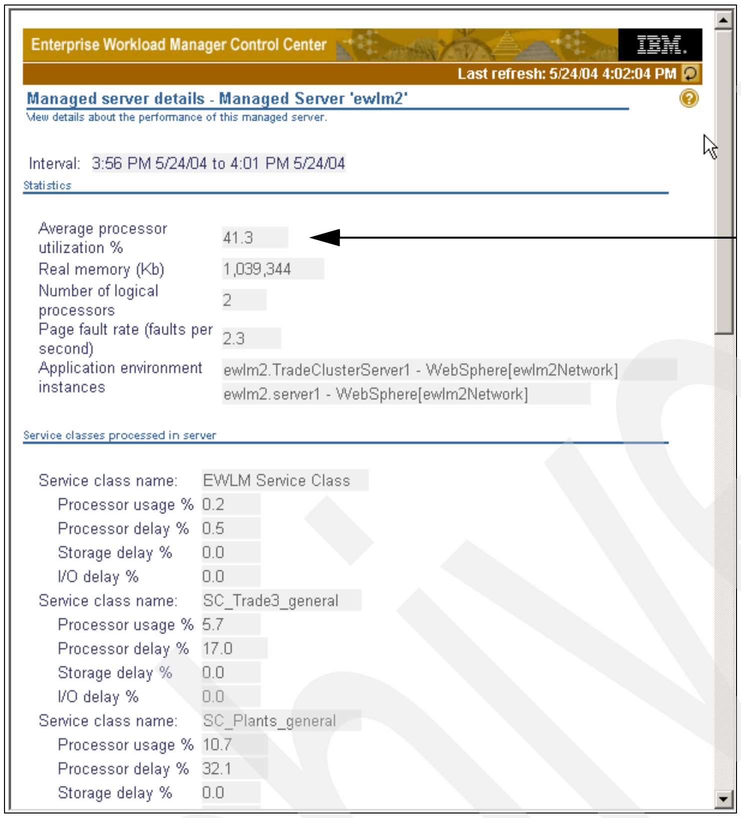
Figure 5-21 Table view for service class SC_Trade3_shopping

There is a lot of data in this tableview that can indicate the performance of the transactions and diagnose potential problems. First we check the longest active time, which in this example is on the WebSphere cluster (ewlm2Network). We know that WebSphere Application Server runs on servers `ewlm2` and `ewlm3`. If the longest active time was showing high values for our database server we would need to log on there and check if some processes or applications take up a large amount of server resources.

An indication of a performance problem is a higher active time for one node relative to the other nodes within the hop. Another indication of a problem is if the successful transaction count in the table view shows a much lower value for one node than for the others. The failed transaction count is worth looking at to see if one node has a much higher count relative to the others or relative to what you expect. Some of these fields, such as transactions counts, quiesce time, page %, delay time, and so forth, are omitted from the figure because of formatting but you can see them by scrolling right on the panel.

Managed server details view

We have identified that our WebSphere cluster servers have a relatively high longest active time value. We can log on to those two servers and check if they have a performance problem because other applications have a high CPU utilization. Alternatively, we can look at a managed server details view. We select **Managed servers** from the Monitor option, select `ewlm2`, for example, and select **Managed Server Details** from the action bar. After pressing **Go** we can examine the Managed Servers details view as shown in Figure 5-22.


Enterprise Workload Manager Control Center
IBM.
Last refresh: 5/24/04 4:02:04 PM

Managed server details - Managed Server 'ewlm2'

View details about the performance of this managed server.

Interval: 3:56 PM 5/24/04 to 4:01 PM 5/24/04

Statistics

Average processor utilization %	41.3
Real memory (Kb)	1,039,344
Number of logical processors	2
Page fault rate (faults per second)	2.3
Application environment instances	ewlm2.TradeClusterServer1 - WebSphere[ewlm2Network]
	ewlm2.server1 - WebSphere[ewlm2Network]

Service classes processed in server

Service class name:	EWLM Service Class
Processor usage %	0.2
Processor delay %	0.5
Storage delay %	0.0
I/O delay %	0.0
Service class name:	SC_Trade3_general
Processor usage %	5.7
Processor delay %	17.0
Storage delay %	0.0
I/O delay %	0.0
Service class name:	SC_Plants_general
Processor usage %	10.7
Processor delay %	32.1
Storage delay %	0.0

Figure 5-22 Details report of WebSphere server ewlm2

This detailed view is available for each server. It provides you with information on the amount of processors and real memory, the application environment instances, and the average processor utilization. If this number has a high value, it is a good indication that this may cause the transactions to fail to complete within the desired response time. In our example the processor utilization is on average at 41%, which is reasonable. We still log on to the system to monitor the system more closely and to verify that there are no other applications using up a large amount of resources.

Another number to look at is the processor delay, which you can see when you scroll down the view shown in Figure 5-22. A high processor delay could also indicate a reason for your performance problem.

There are many more details reports and monitors available for performance administrators to look at. We introduced you to these in 4.4, “Monitor” on page 104, but for finding performance problems the steps indicated here are a good place to start. This should be sufficient to narrow down the source of the problem. As described in this section you need to start with an overall service class view. You can then look further into the operating system instances and application processes supporting the service class. This allows you to understand where potential problems exist and to identify the additional support personnel that you may need to contact to resolve the problem.

6

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

6.1 Firewalls

In today's e-business environment most companies are extremely 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 configuration did not include all 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.

6.1.1 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 about 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 too. You do not need an external firewall, because StoneGate firewall operates as a virtual firewall inside iSeries or zSeries machine.

6.1.2 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. 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 “EWLM configuration in our ITSO environment” on page 32, 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 starting port: `-wasPort`

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 Admin Console. Since the `wasPort` parameter is a starting port for up to fifteen ports that WebSphere Application Server uses, you need to execute the **`displayCC`** command at the domain manager as shown in Example 6-1, after the **`createDM`** command is successful. This provides you with the four ports assigned to the Admin Console and Control Center traffic.

Example 6-1 displayCC to get assigned ports

```
$ ./displayCC.sh -ports /opt/EWLMDM -adminUser ibmewlm -adminPW 111111

Processing displayCC -ports request.  Please be patient as this may
take a while...

...processing 33% complete
...processing 66% complete

 ...Ports assigned to EWLM Control Center:
      - HTTP    20003
      - HTTPS   20004
...Use -changeCC -controlCenterPorts to change these ports if desired.

...Ports assigned to WebSphere Admin Console
      - HTTP    20001
      - HTTPS   20000
...Use -changeCC -adminConsolePorts to change these ports if desired.

...Port assigned to WebSphere Admin: 20009
...Use -changeCC -adminPort to change this port if desired.

PROCESSING COMPLETE
```

If you need to check the domain manager port, the **`displayDM`** command will show this in the DomainManagerPort/mp as shown in Example 6-2.

Example 6-2 displayDM command to get assigned ports

```
# cd /opt/IBM/VE/EWLM/bin
# ./displayDM.sh /opt/EWLMDM
Processing displayDM request.  Please be patient as this may take a while...
WLMConfig - configurable property settings:
   ViaProxyPort/vp(null)
   TracePlugin/tlog(Off)
   InterBrokerPort/dp(null)
   InterBrokerAddress/da(null)
   JmxPort/jp(9092)
   FirewallBrokerList/fb(null)
   ReportingTrace/rt(250)
   ViaProxyHost/va(null)
   DomainName/dn(itsoewlm)
   JniTrace/jt(250)
   SSLKeystore/sslks(null)
   ComponentTrace/ct(250)
   DomainManagerPort/mp(3333)
   MessageLog/ml(250)
   CommunicationTrace/nt(250)
   TraceDistHubBroker/tcomm(0)
   SocksPort/sp(null)
   FirewallBrokerPort/fp(null)
   FailureLimit/fl(50)
   SSLKeystorePassword/sslpw(password suppressed)
   DomainManagerAddress/ma(9.12.4.142)
   AuthorityLevel/auth(None)
   ProcessMode/mode(DomainManager)
   LBPublicPort/lbp(Off)
   LBSecurePort/lbs(Off)
   EventTrace/et(250)
   TraceLevel/tl(Min)
   TestComponent/t(null)
   DumpRetentionQuantity/dpn(25)
   DumpRetentionAge/dpa(30)
   SocksHost/sa(null)
WLMConfig - non-configurable property settings:
   ManagedServerFailureTime(null)
   ManagedServerIdentity(b05b63dad0404b3a5f460829852701c6)
   ManagedServerId(-1)
   StatisticsInterval(10)
   DomainManagerIdentity(null)
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 6-1 on page 160 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.

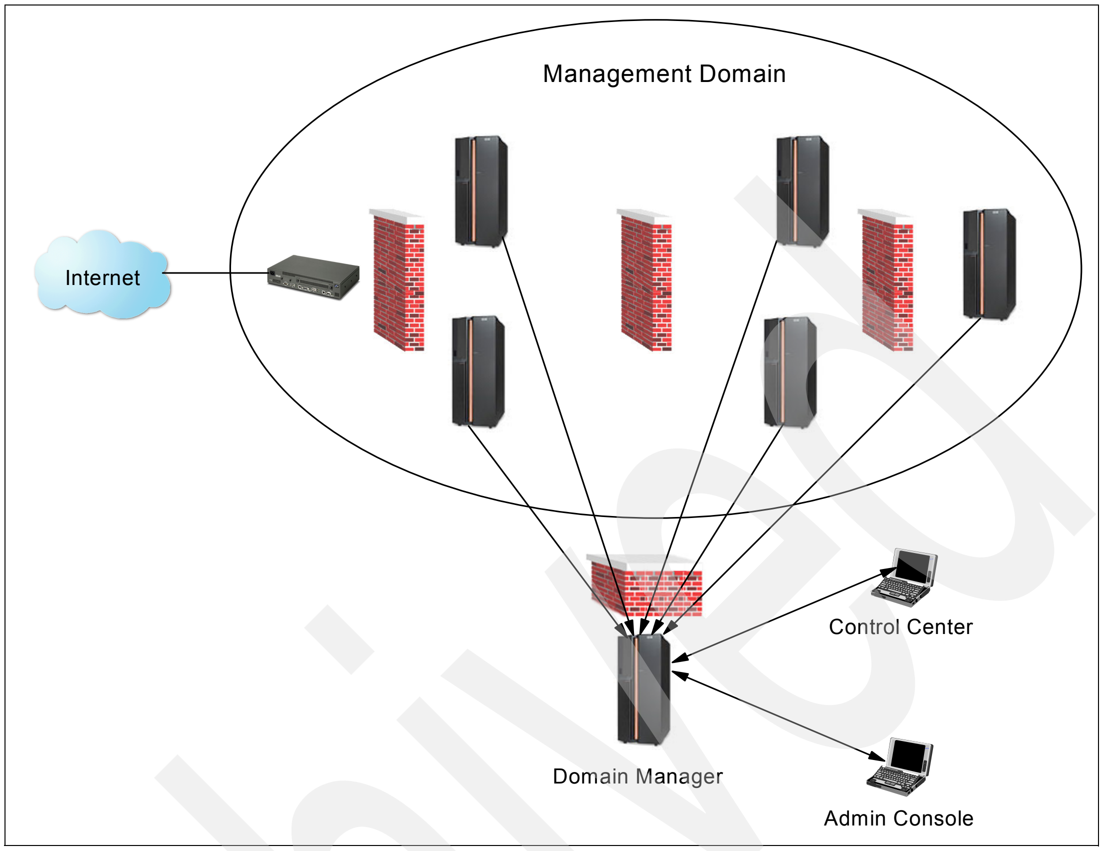


Figure 6-1 EWLM network traffic

Stateful inspection and EWLM

In Figure 6-1, 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 6-1 and Example 6-2 respectively:

- Managed server(s) to domain manager at address/port, ewlmdm1.itso.ibm.com/3333
- Control Center browser to UI server at address/port(s), ewlmdm1.itso.ibm.com/20003,20004
- Admin Console browser to WebSphere Application Server at address/port(s), ewlmdm1.itso.ibm.com/20001,20002

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 6-2. The managed server must be configured with two additional parameters, `-va` and `-vp`, using the **`changeMS`** command.

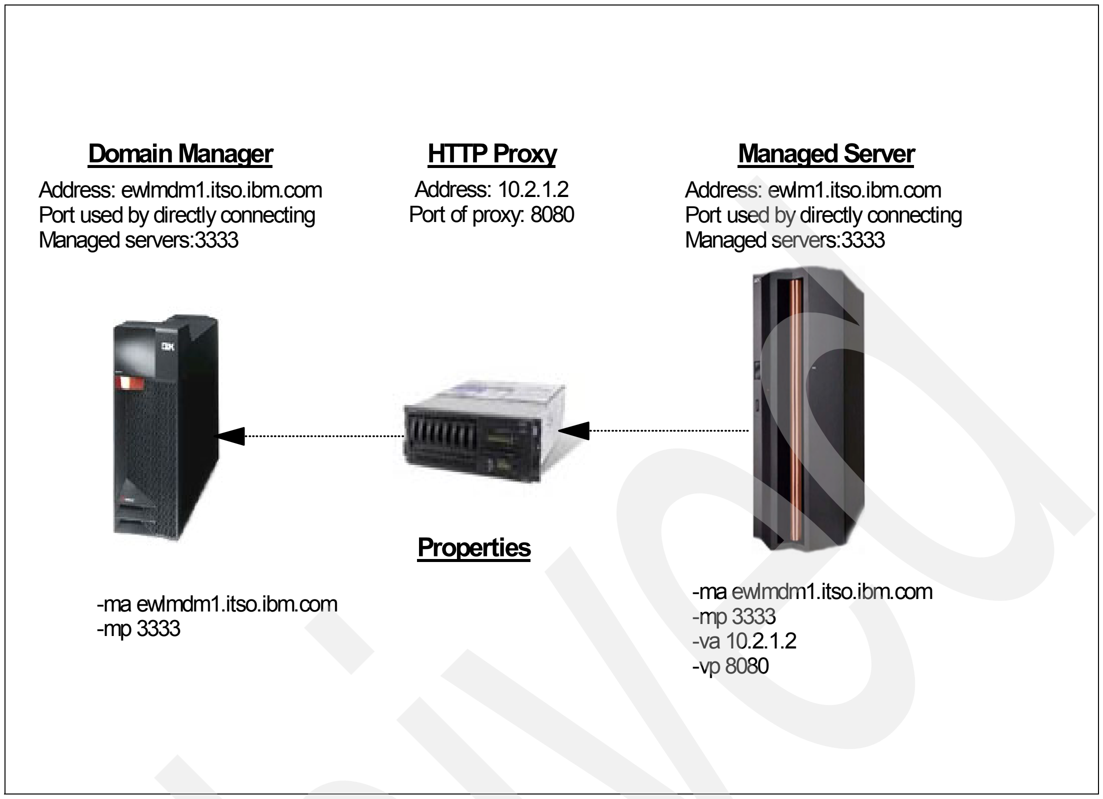


Figure 6-2 HTTP Proxy example

Figure 6-2 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 6-3 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 6-3 changeMS command for Windows, Solaris, AIX

```
changeMS <workingDir>
                       [-auth [None : ServerSSL : ClientServerSSL]]
                       [-sslks [path] -sslpw [password]]
                       [-ma address] [-mp port]
                       [-va address] [-vp port]
                       [-nt [value]] [-ct [value]]
                       [-et [value]] [-ml [value]]
                       [-tl [value]] [-tlog [class]]
                       [-dpa [value]] [-dpn [value]]
                       [-fl [value]]

For i5/OS, you need to use the STRWLM command.
```

In order to add the Proxy address and port, Example 6-4 shows the custom **changeMS** command that would be used based on the addresses and ports in Figure 6-2. 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 6-4 changeMS for Proxy support, Windows example

```
changeMS c:\ewlmMS -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/VE/EWLMMS directory. You can then plan to use the firewall broker on the managed server itself or distribute it on the appropriate platform.

Table 6-1 is a summary of the different configurations with a SOCKS server and how they affect the EWLM code, followed by two detailed examples.

Table 6-1 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	- Need firewall broker. - Domain manager must be configured to identify firewall broker using the changeDM command with -fp and -fb parameters. - Firewall broker must be configured to use SOCKS server using changeFB command with the -sa and -sp parameters.
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	- 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 firewall broker using changeDM command with the -fp and -fb parameters. - The domain manager -fb list must use SOCKS tag for this firewall broker.

In the following discussion, two sample SOCKS server configurations are shown that are mainly describing different architecture solutions for the security zones.

Figure 6-3 shows the managed servers accessing the domain manager through a SOCKS server through a firewall broker. In this configuration, the SOCKS server is protecting the zone where the domain manager is located.

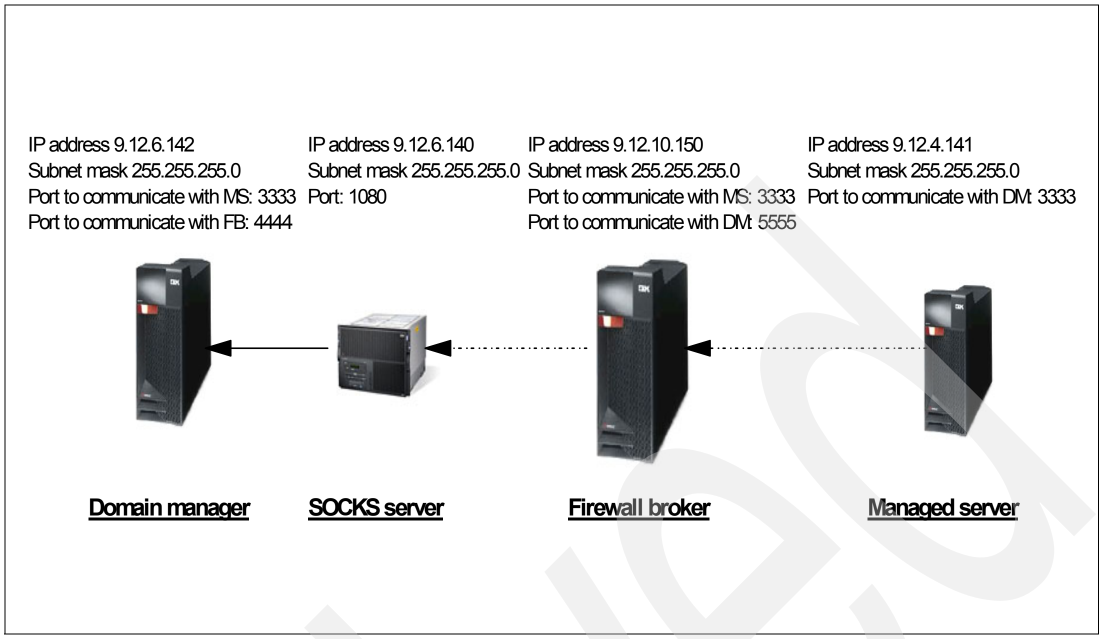

IP address 9.12.6.142
Subnet mask 255.255.255.0
Port to communicate with MS: 3333
Port to communicate with FB: 4444
IP address 9.12.6.140
Subnet mask 255.255.255.0
Port: 1080
IP address 9.12.10.150
Subnet mask 255.255.255.0
Port to communicate with MS: 3333
Port to communicate with DM: 5555
IP address 9.12.4.141
Subnet mask 255.255.255.0
Port to communicate with DM: 3333
Domain manager
SOCKS server
Firewall broker
Managed server

Figure 6-3 SOCKS Server protecting the domain manager zone

To define this configuration the following changes are required (sample commands are in Example 6-5):

- On the managed servers specify the following parameters on the **`createMS`** command:
  - `-ma` and `-mp` must point to the IP address or hostname and port of the firewall broker.
- On the firewall broker, run the **`createFB`** script with the following parameters:
  - `-ma`, the IP address or hostname 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 6-5 Sample commands to define the SOCKS server environment

```
on the Domain manager:
./changeDM.sh /opt/ewlmDM -ma 9.12.6.142 -mp 3333 -fp 4444 -fb 9.12.10.150:5555:SOCKS

on the Firewall broker:
./createFB.sh /opt/ewlmFB -ma 9.12.10.150 -mp 3333 -da 9.12.6.142 -dp 4444 -fp 5555
             -auth None
./changeFB.sh /opt/ewlmFB -sa 9.12.6.140 -sp 1080

on the Managed server:
./createMS.sh /opt/ewlmMS -ma 9.12.10.150 -mp 3333 -auth None
```

In Figure 6-4 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.

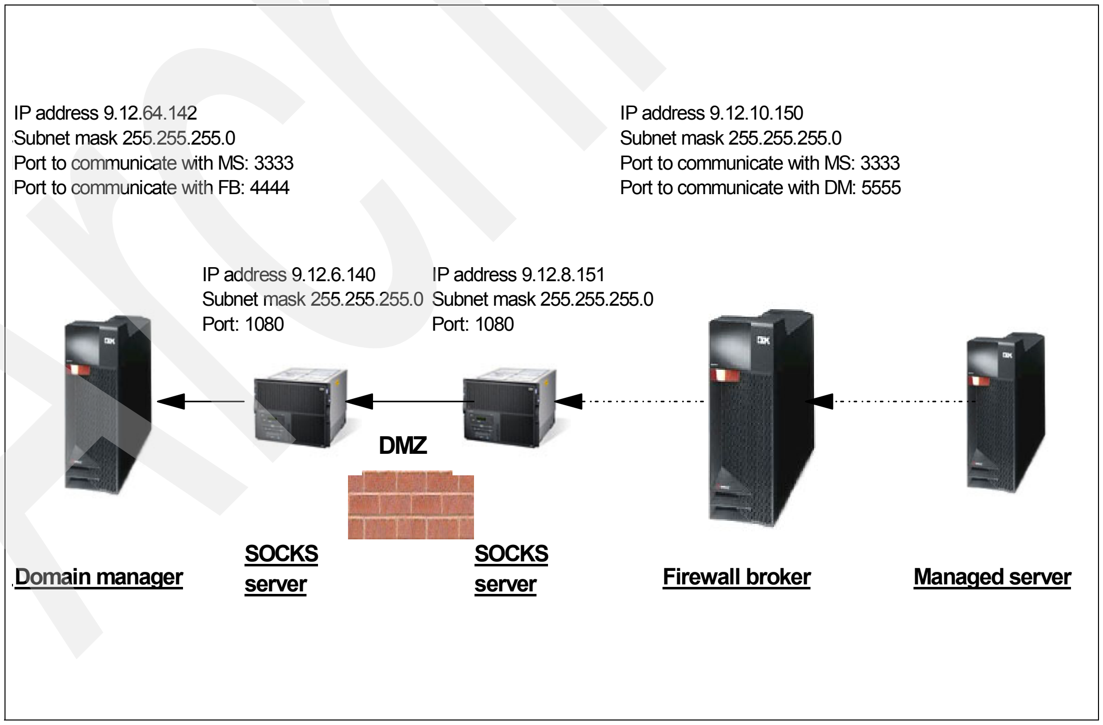

IP address 9.12.64.142
Subnet mask 255.255.255.0
Port to communicate with MS: 3333
Port to communicate with FB: 4444
IP address 9.12.10.150
Subnet mask 255.255.255.0
Port to communicate with MS: 3333
Port to communicate with DM: 5555
IP address 9.12.6.140
Subnet mask 255.255.255.0
Port: 1080
IP address 9.12.8.151
Subnet mask 255.255.255.0
Port: 1080
DMZ
Domain manager
SOCKS server
SOCKS server
Firewall broker
Managed server

Figure 6-4 SOCKS server with DMZ

This configuration is similar to the one described in Figure 6-3 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 in shown in Example 6-6.

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

```
on the Domain manager:
./changeDM.sh /opt/ewlmDM -ma 9.12.6.142 -mp 3333 -fp 4444 -fb 9.12.10.150:5555:SOCKS
            -sa 9.12.4.140 -sp 1080

on the Firewall broker:
./createFB.sh /opt/ewlmFB -ma 9.12.10.150 -mp 3333 -da 9.12.6.142 -dp 4444 -fp 5555
            -auth None
./changeFB.sh /opt/ewlmFB -sa 9.12.8.151 -sp 1080

on the Managed server:
./createMS.sh /opt/ewlmMS -ma 9.12.10.150 -mp 3333 -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 AIX, Solaris, and Windows. The only difference is that AIX and Linux require a .sh extension while Windows requires a .bat extension. To create a firewall broker on OS/400, you must use the STRWLM CL command.

For the command syntax, refer to the InfoCenter at:

`http://public.boulder.ibm.com/eserver/`

6.1.3 Our EWLM firewall configurations

This section shows the two firewall configurations we used in our lab 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 6-5, the only managed server to domain manager communication across the firewall is from the HTTP Plug-in managed server to the domain manager.

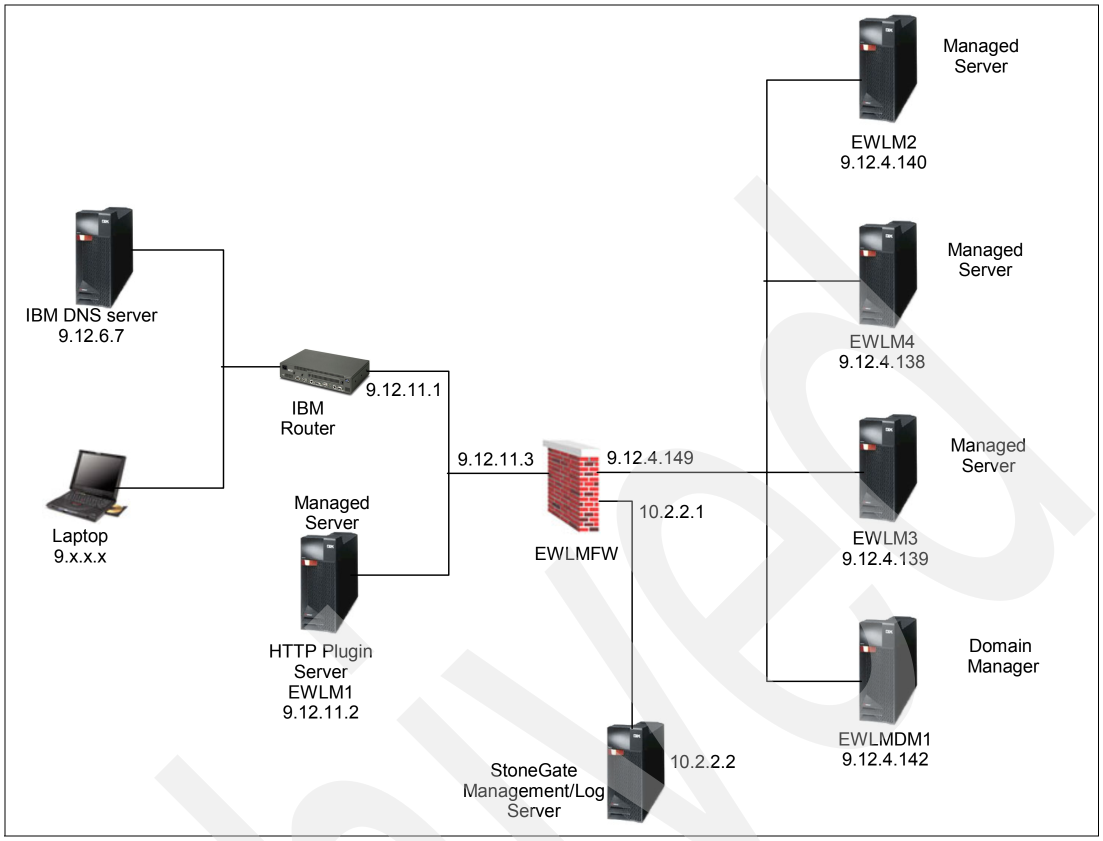
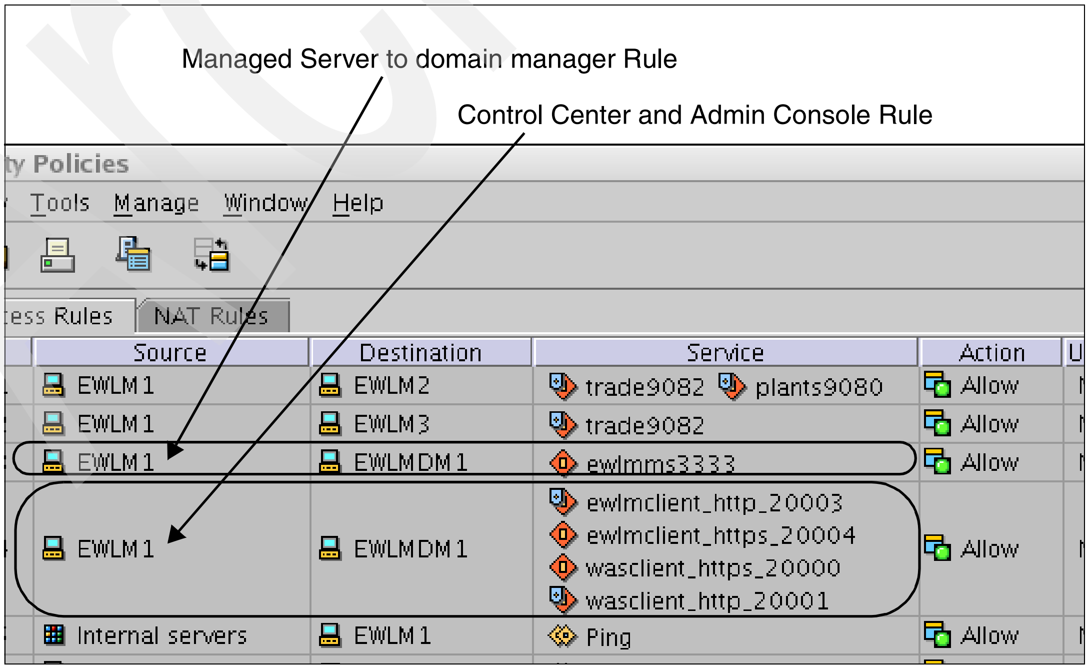


Figure 6-5 Stateful inspection firewall lab configuration

Since there is no Proxy, no additional EWLM configuration needs to take place, but as highlighted in "EWLM firewall support" on page 157, the stateful inspection firewall rules may need to be updated to include this traffic.


Managed Server to domain manager Rule
Control Center and Admin Console Rule
ty Policies
Tools Manage Window Help
ess Rules
NAT Rules

Source	Destination	Service	Action	U
EWLM1	EWLM2	trade9082 plants9080	Allow	
EWLM1	EWLM3	trade9082	Allow	
EWLM1	EWLMDM1	ewlmms3333	Allow	
EWLM1	EWLMDM1	ewlmclient_http_20003 ewlmclient_https_20004 wasclient_https_20000 wasclient_http_20001	Allow	
Internal servers	EWLM1	Ping	Allow	

Figure 6-6 StoneGate stateful inspection firewall and EWLM

Figure 6-6 identifies the two rules that 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 on EWLMDM1 using port 3333. In addition, we were using EWLM1 as a browser in the lab to get to the domain manager's Control Center and the WebSphere Application Server Admin Console. Therefore, we needed to add this traffic to EWLMDM1 using ports 20000, 200001, 200003, and 200004.

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 6-7.

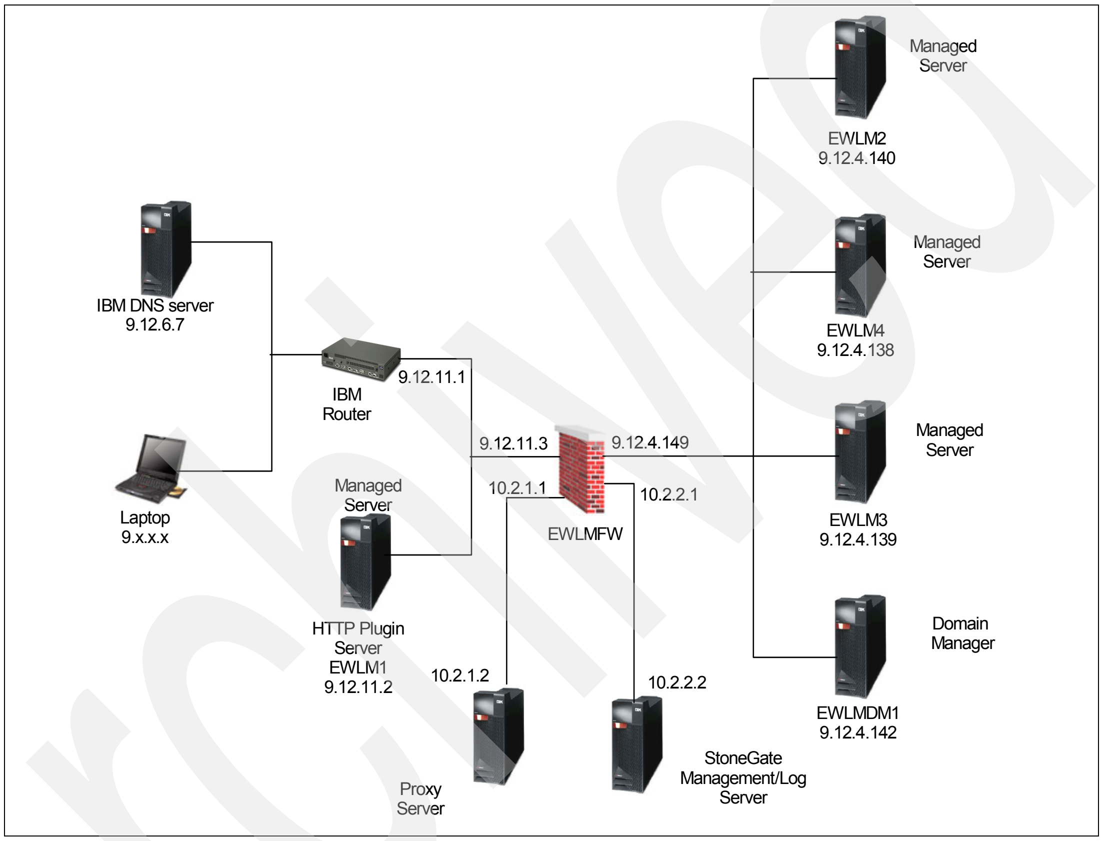
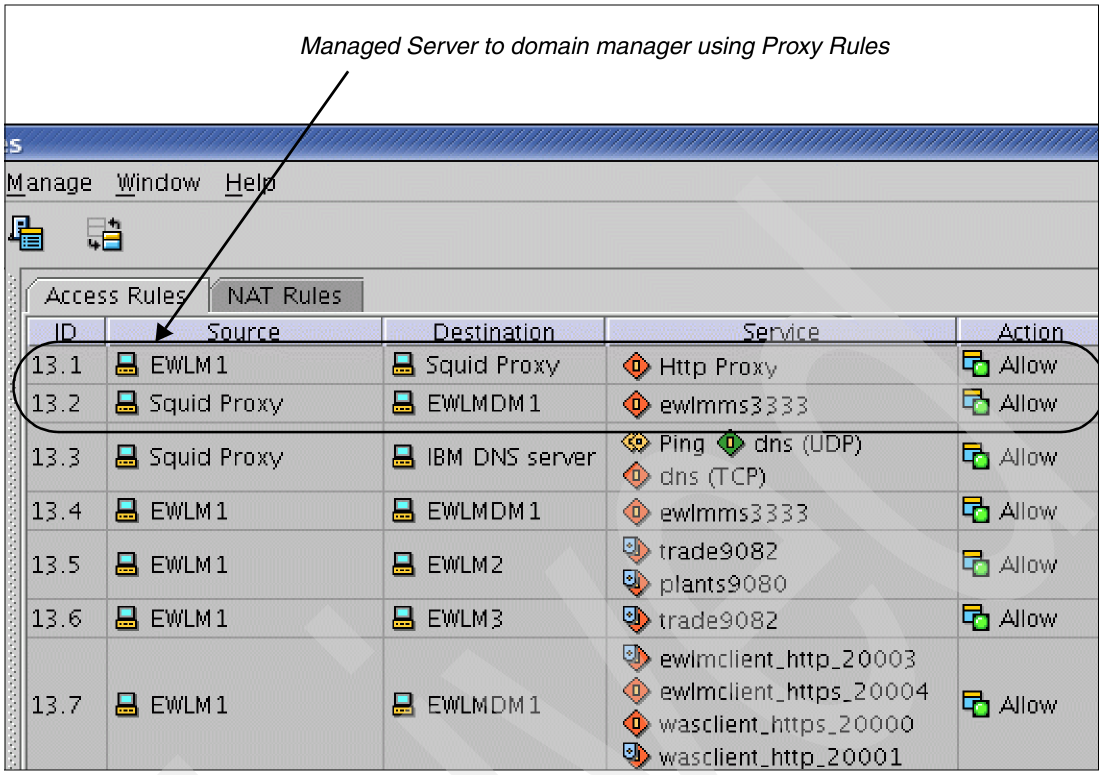


Figure 6-7 Proxy firewall ITSO configuration

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, as shown in Figure 6-8.



ID	Source	Destination	Service	Action
13.1	EWLM1	Squid Proxy	Http Proxy	Allow
13.2	Squid Proxy	EWLMDM1	ewlmms3333	Allow
13.3	Squid Proxy	IBM DNS server	Ping dns (UDP) dns (TCP)	Allow
13.4	EWLM1	EWLMDM1	ewlmms3333	Allow
13.5	EWLM1	EWLM2	trade9082 plants9080	Allow
13.6	EWLM1	EWLM3	trade9082	Allow
13.7	EWLM1	EWLMDM1	ewlmclient_http_20003 ewlmclient_https_20004 wasclient_https_20000 wasclient_http_20001	Allow

Figure 6-8 StoneGate with Proxy configuration and EWLM

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 6-7 Managed server configuration for Proxy server

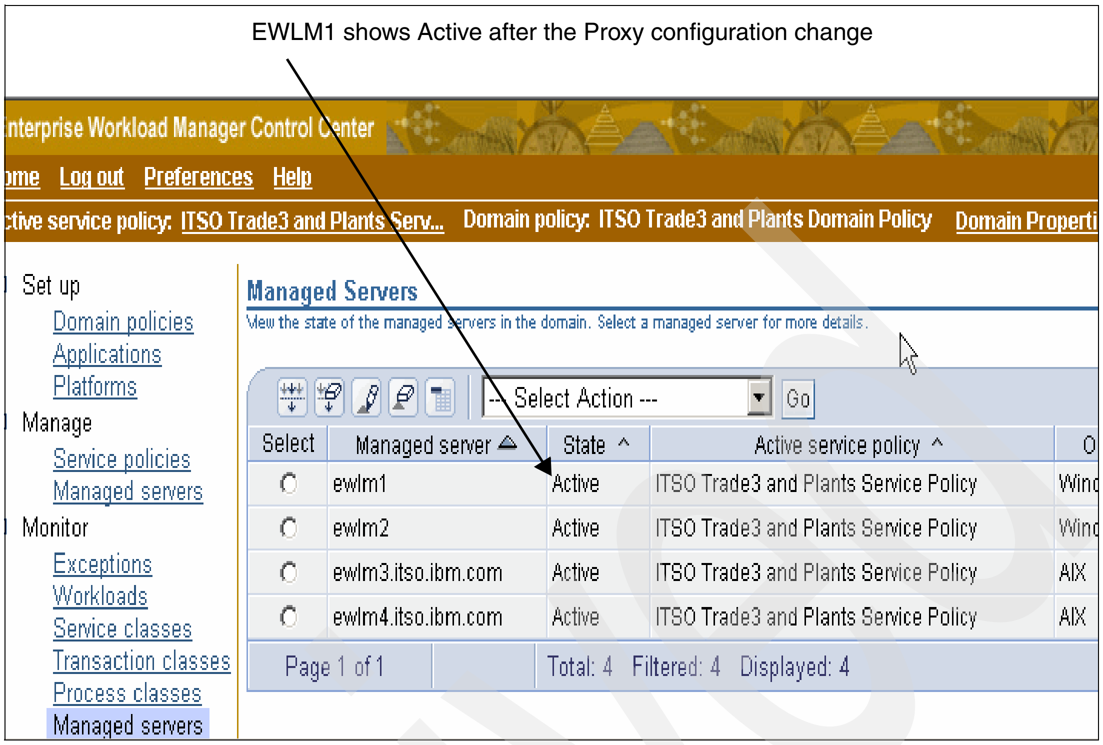
```
C:\Program Files\IBM\VE\EWLMMS\bin>changeMS c:\ewlmMS -va 10.2.1.2 -vp 8080

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

PROCESSING COMPLETE
```

Now you can execute a **displayMS** command that shows the proxy address and proxy port that the managed server is communicating with as shown in Example 6-8.

Example 6-8 displayMS at the managed server

```
C:\Program Files\IBM\VE\EWLMMS\bin>displayMS c:\ewlmMS

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

WLMConfig - configurable property settings:
   ViaProxyPort/vp(8080)
   TracePlugin/tlog(Off)
   InterBrokerPort/dp(null)
   InterBrokerAddress/da(null)
   JmxPort/jp(Off)
   FirewallBrokerList/fb(null)
   ReportingTrace/rt(250)
   ViaProxyHost/va(10.2.1.2)
   DomainName/dn(itsoewlm)
   JniTrace/jt(250)
   SSLKeystore/sslks(null)
   ComponentTrace/ct(250)
   DomainManagerPort/mp(3333)
   MessageLog/ml(250)
   CommunicationTrace/nt(250)
   TraceDistHubBroker/tcomm(0)
   SocksPort/sp(null)
   FirewallBrokerPort/fp(null)
   FailureLimit/fl(50)
   SSLKeystorePassword/sslpw(password suppressed)
   DomainManagerAddress/ma(ewlmdm1.itso.ibm.com)
   AuthorityLevel/auth(None)
   ProcessMode/mode(ManagedServer)
   LBPublicPort/lbp(Off)
   LBSecurePort/lbs(Off)
   EventTrace/et(250)
   TraceLevel/tl(Min)
   TestComponent/t(null)
   DumpRetentionQuantity/dpn(25)
   DumpRetentionAge/dpa(30)
   SocksHost/sa(null)
WLMConfig - non-configurable property settings:
   ManagedServerFailureTime(null)
   ManagedServerIdentity(439b9b2f6e14278da3f06539fe0bc139)
   ManagedServerId(1)
   StatisticsInterval(10)
   DomainManagerIdentity(b05b63dad0404b3a5f460829852701c6)
PROCESSING COMPLETE
```

In order to check whether the managed server and domain manager are communicating as expected, sign into the Control Center with a userid 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.



nterprise Workload Manager Control Center

ome Log out Preferences Help

ctive service policy: ITSO Trade3 and Plants Serv... Domain policy: ITSO Trade3 and Plants Domain Policy Domain Properti

Set up
- Domain policies
- Applications
- Platforms

Manage
- Service policies
- Managed servers

Monitor
- Exceptions
- Workloads
- Service classes
- Transaction classes
- Process classes
- Managed servers

Managed Servers

View the state of the managed servers in the domain. Select a managed server for more details.

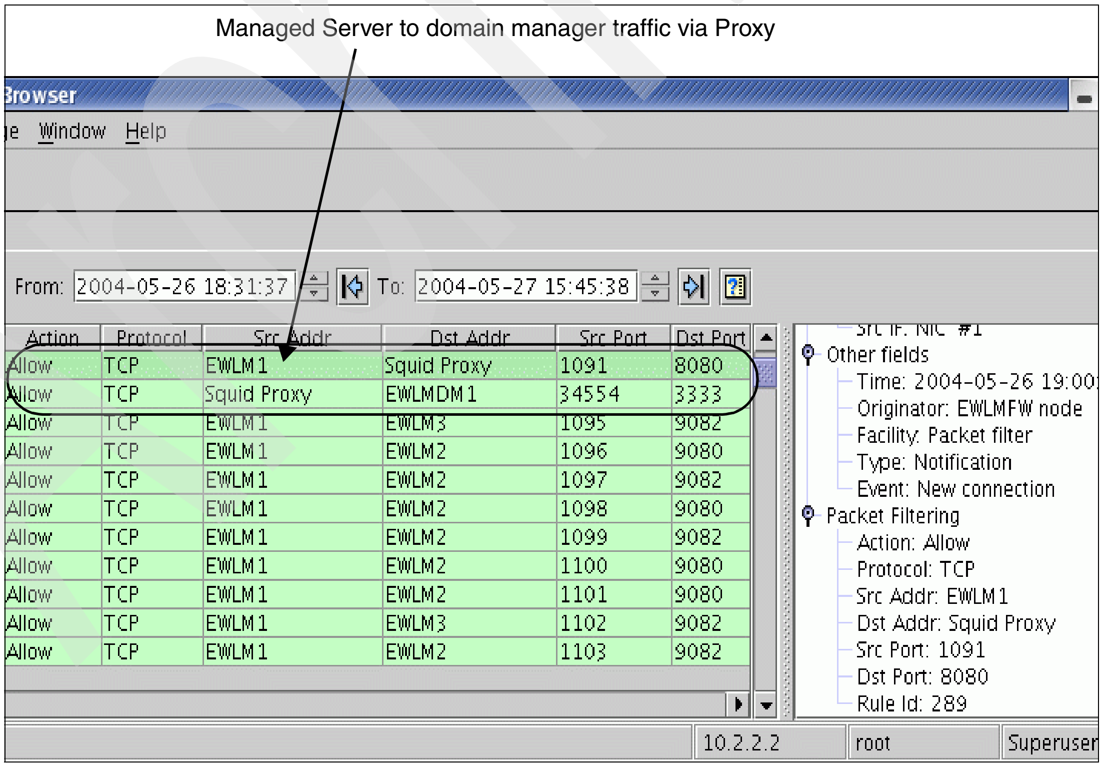
-- Select Action -- Go

Select	Managed server	State	Active service policy	O
○	ewlm1	Active	ITSO Trade3 and Plants Service Policy	Wind
○	ewlm2	Active	ITSO Trade3 and Plants Service Policy	Wind
○	ewlm3.itso.ibm.com	Active	ITSO Trade3 and Plants Service Policy	AIX
○	ewlm4.itso.ibm.com	Active	ITSO Trade3 and Plants Service Policy	AIX

Page 1 of 1   Total: 4 Filtered: 4 Displayed: 4

Figure 6-9 Managed servers Monitor View

The firewall log in Figure 6-10 also shows the traffic from the EWLM1 managed server going to the Proxy and then the EWLMDM1 domain manager.



From: 2004-05-26 18:31:37 To: 2004-05-27 15:45:38

Action	Protocol	Src Addr	Dst Addr	Src Port	Dst Port
Allow	TCP	EWLM1	Squid Proxy	1091	8080
Allow	TCP	Squid Proxy	EWLMDM1	34554	3333
Allow	TCP	EWLM1	EWLM3	1095	9082
Allow	TCP	EWLM1	EWLM2	1096	9080
Allow	TCP	EWLM1	EWLM2	1097	9082
Allow	TCP	EWLM1	EWLM2	1098	9080
Allow	TCP	EWLM1	EWLM2	1099	9082
Allow	TCP	EWLM1	EWLM2	1100	9080
Allow	TCP	EWLM1	EWLM2	1101	9080
Allow	TCP	EWLM1	EWLM3	1102	9082
Allow	TCP	EWLM1	EWLM2	1103	9082

Src IF: NIC #1
- Other fields
  - Time: 2004-05-26 19:00
  - Originator: EWLMFW node
  - Facility: Packet filter
  - Type: Notification
  - Event: New connection
- Packet Filtering
  - Action: Allow
  - Protocol: TCP
  - Src Addr: EWLM1
  - Dst Addr: Squid Proxy
  - Src Port: 1091
  - Dst Port: 8080
  - Rule Id: 289

10.2.2.2 root Superuser

Figure 6-10 StoneGate Network Traffic Log

In our configuration we did not test the SOCKS server for firewall protection.

6.2 Security

In our sample scenario, we had the Trade3 workload running through three tiers of servers: HTTP, 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 domain manager to each managed server and vice versa, and from the domain manager to the user interface. This information about transactions, as shown in Figure 6-11 on page 172, 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 mostly 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 options depending on your installation: None, Server SSL, and Client/Server 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

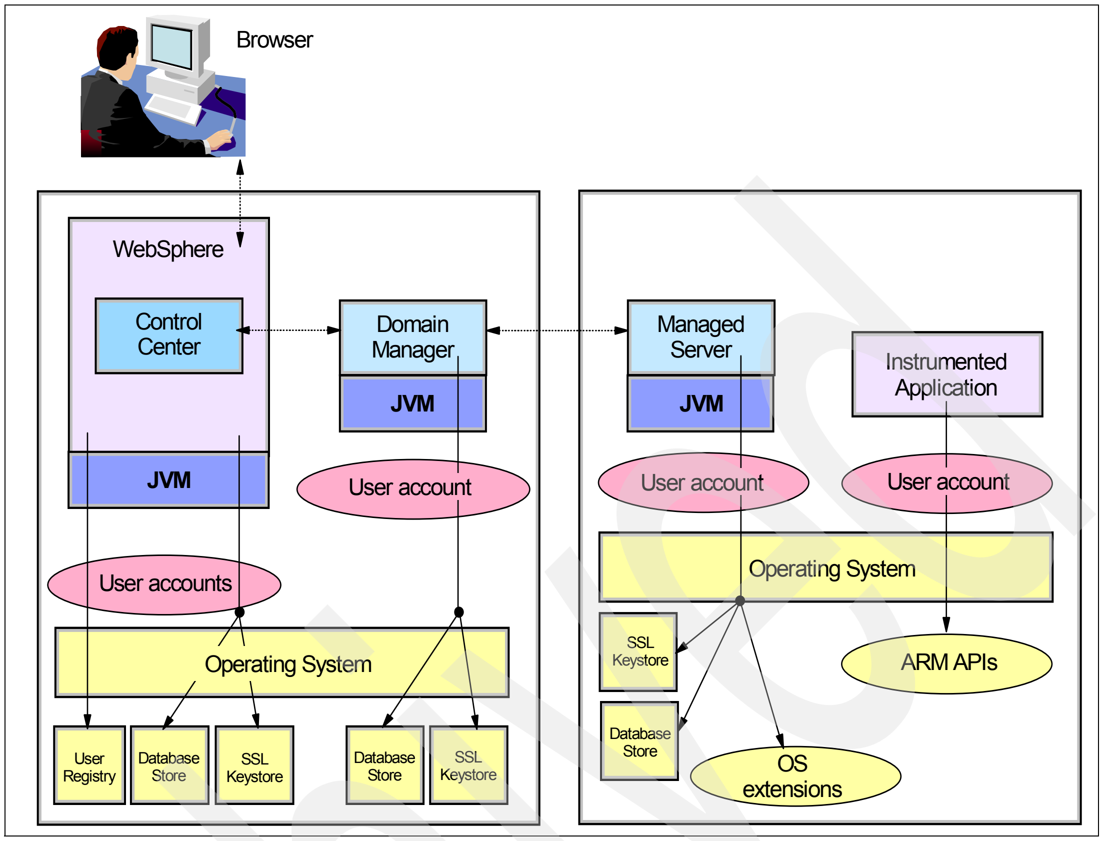


Figure 6-11 Security context and boundaries

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 (discussed in "EWLM and workload balancing overview" on page 188)

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.

6.2.1 Browser to Control Center

Figure 6-12 on page 173 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 userid 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 been already created using the **`createDM`** command, which creates the Control Center along with the domain manager. For i5/OS, the **`createCC`** shell command creates the Control Center. In both cases, the command establishes the userid passed in the adminUser parameter as an administrator for the WebSphere Application Server instance established for the site.

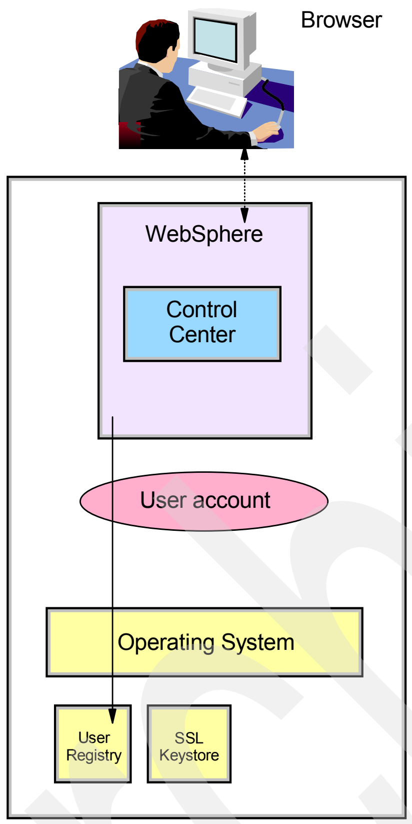
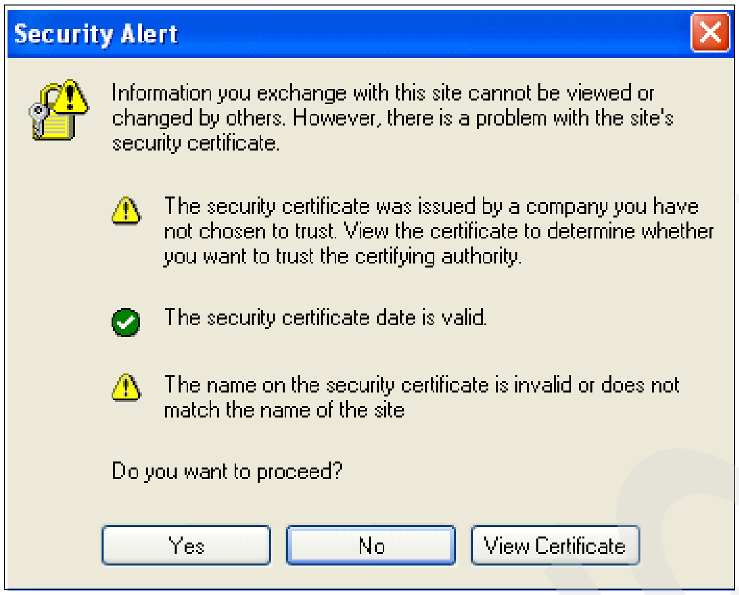


Figure 6-12   Communication between the browser and the Control Center

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 20003), you notice that EWLM immediately routes your connection on the SSL port 20004 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 to ask you to accept the default certificate, as shown in Figure 6-13.


Security Alert

Information you exchange with this site cannot be viewed or changed by others. However, there is a problem with the site's security certificate.

The security certificate was issued by a company you have not chosen to trust. View the certificate to determine whether you want to trust the certifying authority.

The security certificate date is valid.

The name on the security certificate is invalid or does not match the name of the site

Do you want to proceed?

Figure 6-13 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 6-9, 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 6-9 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 6-10 ChangeCC command to enable security

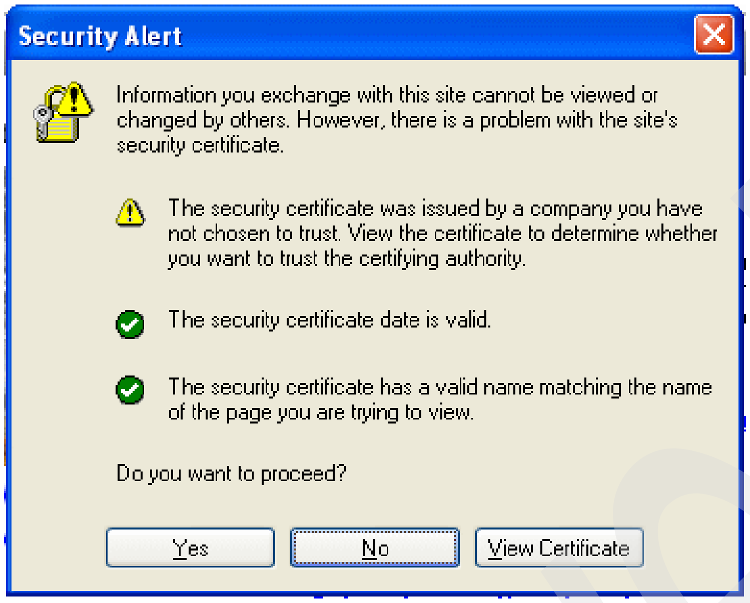
```
[ibmewlm@ewlmdm1 bin]$ ./changeCC.sh -sslSecurity /opt/EWLMDM/ -adminUser ibmewlm -adminPW
111111 -level adminDefined -keystore /home/ibmewlm/ks -keystorePW 111111

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

...processing 33% complete:
...processing 66% complete
 SSLSECURITY END
```

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 (such as Verisign). 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 6-14, 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.


Security Alert

Information you exchange with this site cannot be viewed or changed by others. However, there is a problem with the site's security certificate.

The security certificate was issued by a company you have not chosen to trust. View the certificate to determine whether you want to trust the certifying authority.

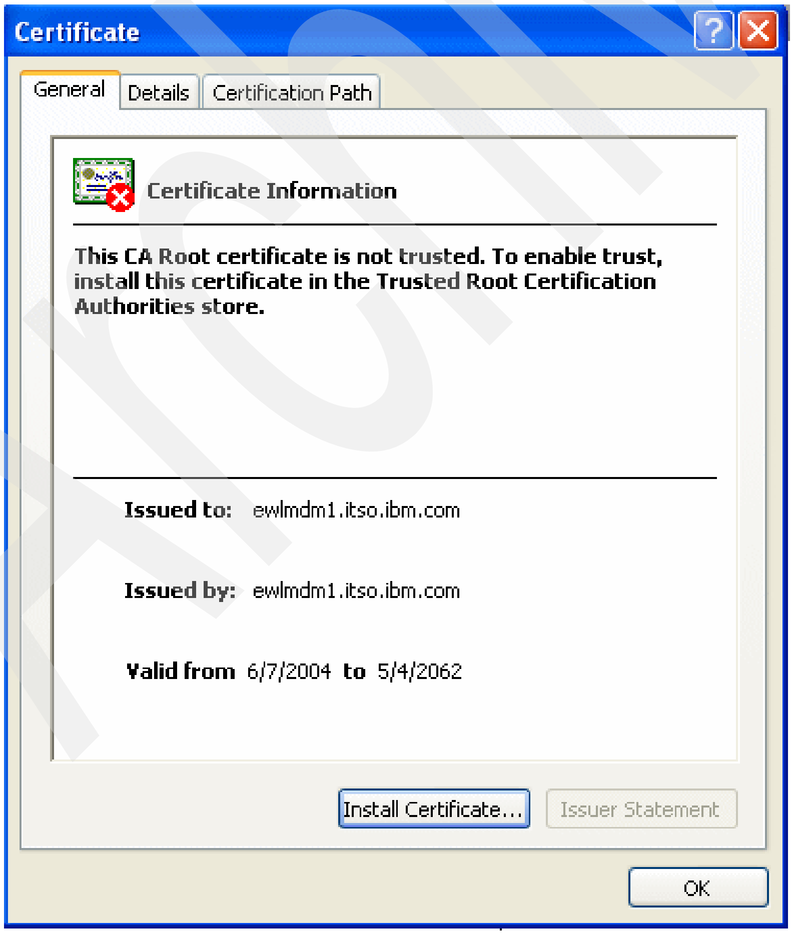
The security certificate date is valid.

The security certificate has a valid name matching the name of the page you are trying to view.

Do you want to proceed?

Figure 6-14 Securing warning after certificate installation

If you click **View Certificate** you will receive the Certificate shown in Figure 6-15, which gives you the opportunity to launch the wizard to install the Certificate on your workstation.



Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

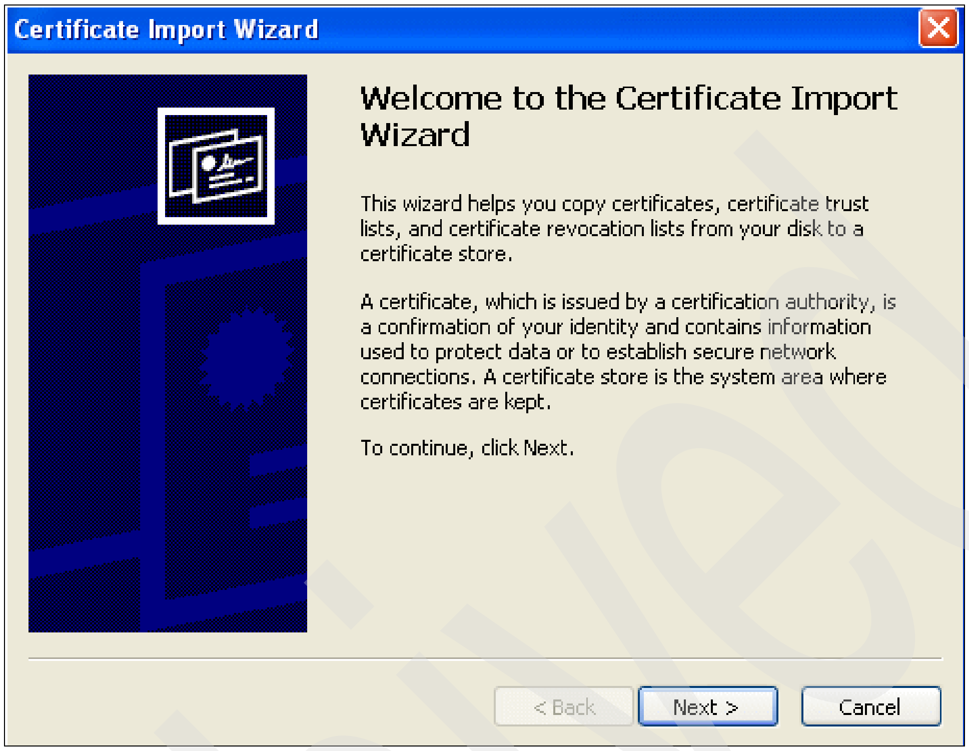
Issued to: ewlmdm1.itso.ibm.com

Issued by: ewlmdm1.itso.ibm.com

Valid from 6/7/2004 to 5/4/2062

Figure 6-15 Certificate

The **Install Certificate** option starts the wizard as shown in Figure 6-16.



Certificate Import Wizard

Welcome to the Certificate Import Wizard

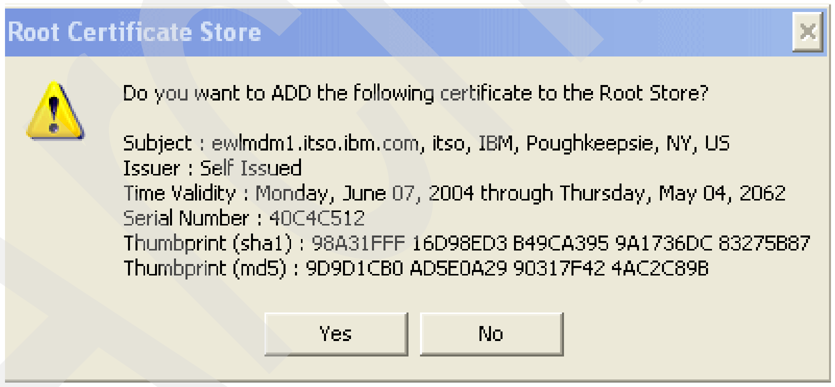
This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

Figure 6-16 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 6-17.



Root Certificate Store

Do you want to ADD the following certificate to the Root Store?

Subject : ewlmdm1.itso.ibm.com, itso, IBM, Poughkeepsie, NY, US
Issuer : Self Issued
Time Validity : Monday, June 07, 2004 through Thursday, May 04, 2062
Serial Number : 40C4C512
Thumbprint (sha1) : 98A31FFF 16D98ED3 B49CA395 9A1736DC 83275B87
Thumbprint (md5) : 9D9D1CB0 AD5E0A29 90317F42 4AC2C89B

Figure 6-17 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.

6.2.2 Control Center to domain manager

As indicated in Figure 6-18 on page 178, the Control Center and domain manager communicate through a port. The port is defined via the `–jp` parameter on the **`createDM`** and

createCC commands, or via the CONPORT parameter on the STRWLM command on an OS/400 installation.

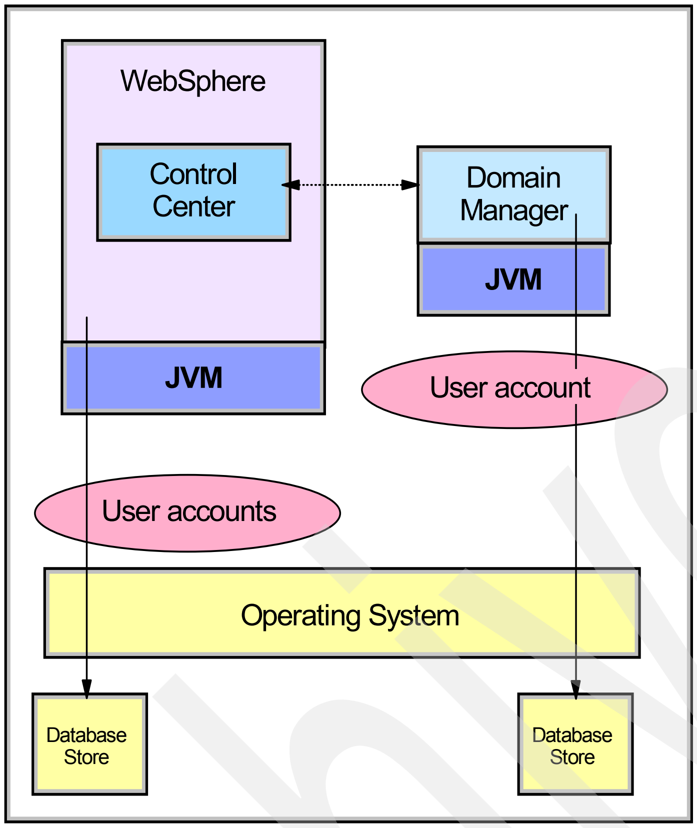


Figure 6-18 Communication between Control Center and Domain Manager

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` and CONPORT port should be blocked. On systems that provide means to restrict internal access to ports, the `–jp` and CONPORT 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.

6.2.3 Domain manager to managed servers

Figure 6-19 on page 179 shows the domain manager and managed servers communicating through addresses and ports. SSL encryption provides confidentiality of the data that flows between them.

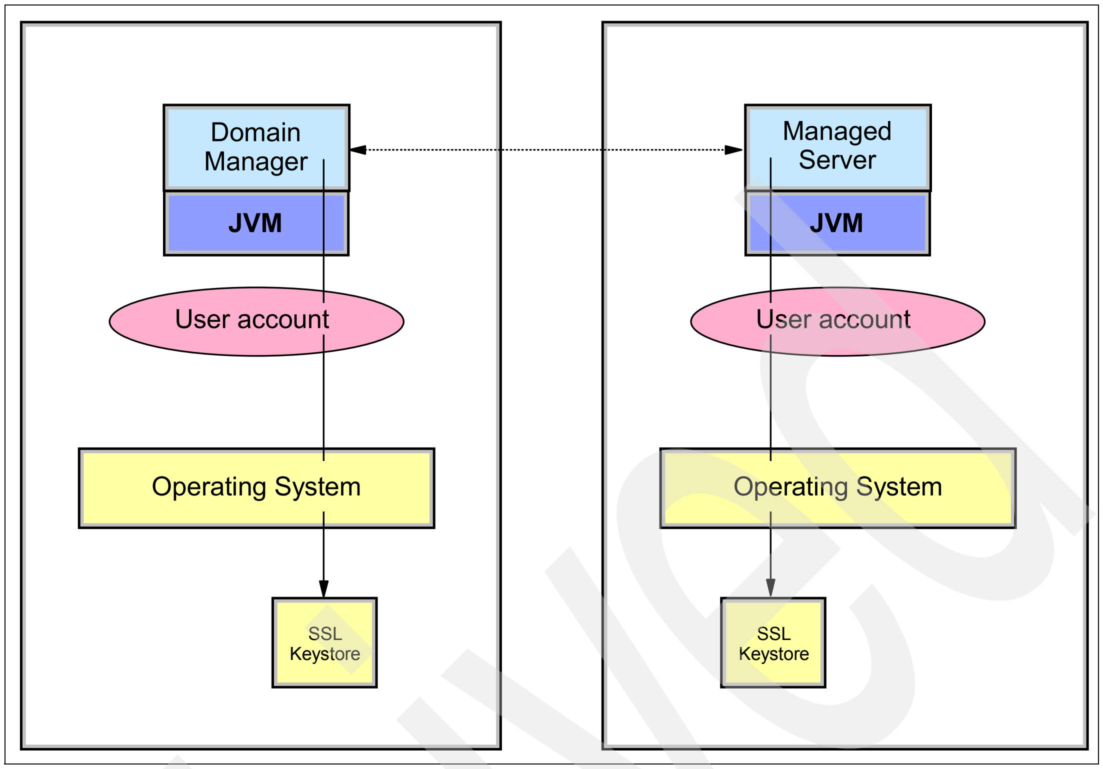


Figure 6-19 Communication between Domain Manager and Managed Server

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 mainly two 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 mostly where authentication is not a strong issue. 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:

- 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 6-11.

Example 6-11 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
```

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

Example 6-12 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>
```

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

Example 6-13 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
```

- At this point, we are ready to send (in binary) the certificate ks.cert to all the managed nodes in our configuration.
- On each managed node, we then import the certificate ks.cert into the managed node keystore. Example 6-14 shows the import step on the ewlm4 server.

Example 6-14 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:D0:84: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 6-15):

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 6-15 Enabling domain manager security

```
[root@ewlmdm1 bin]# ./changeDM.sh /opt/EWLMDM/  -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]# ./displayDM.sh /opt/EWLMDM/

Processing displayDM request.  Please be patient as this may take a while...
WLMConfig - configurable property settings:
   ViaProxyPort/vp(null)
   TracePlugin/tlog(Off)
   InterBrokerPort/dp(null)
   InterBrokerAddress/da(null)
   JmxPort/jp(9092)
   FirewallBrokerList/fb(null)
   ReportingTrace/rt(250)
   ViaProxyHost/va(null)
   DomainName/dn(itsoewlm)
   JniTrace/jt(250)
   SSLKeystore/sslks(/home/ibmewlm/ks_dm)
   ComponentTrace/ct(250)
   DomainManagerPort/mp(3333)
   MessageLog/ml(250)
   CommunicationTrace/nt(250)
   TraceDistHubBroker/tcomm(0)
   SocksPort/sp(null)
   FirewallBrokerPort/fp(null)
   FailureLimit/fl(50)
   SSLKeystorePassword/sslpw(password suppressed)
   DomainManagerAddress/ma(9.12.4.142)
   AuthorityLevel/auth(ServerSSL)
   ProcessMode/mode(DomainManager)
   LBPublicPort/lbp(Off)
   LBSecurePort/lbs(Off)
   EventTrace/et(250)
   TraceLevel/tl(Min)
   TestComponent/t(null)
   DumpRetentionQuantity/dpn(25)
   DumpRetentionAge/dpa(30)
   SocksHost/sa(null)
WLMConfig - non-configurable property settings:
   ManagedServerFailureTime(null)
   ManagedServerIdentity(b05b63dad0404b3a5f460829852701c6)
   ManagedServerId(-1)
   StatisticsInterval(10)
   DomainManagerIdentity(null)
PROCESSING COMPLETE
[root@ewlmdm1 bin]# ./startDM.sh /opt/EWLMDM/
```

At this point all the connected managed servers might need to be stopped—if they are not already stopped—in order to make the changes. The next step is to alter their security configuration.

On each managed node (see Example 6-16):

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 UI under the Monitoring → Managed Servers task.

Example 6-16 EWLM4 managed node

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

root@ewlm4:/opt/IBM/VE/EWLMMS/bin>./displayMS.sh /opt/ewlmMS
WLMConfig - configurable property settings:
   ViaProxyPort/vp(null)
   TracePlugin/tlog(Off)
   InterBrokerPort/dp(null)
   InterBrokerAddress/da(null)
   JmxPort/jp(Off)
   FirewallBrokerList/fb(null)
   ReportingTrace/rt(250)
   ViaProxyHost/va(null)
   DomainName/dn(itsoewlm)
   JniTrace/jt(250)
   SSLKeystore/sslks(/usr/java14/bin/ks_ms)
   ComponentTrace/ct(250)
   DomainManagerPort/mp(3333)
   MessageLog/ml(250)
   CommunicationTrace/nt(250)
   TraceDistHubBroker/tcomm(0)
   SocksPort/sp(null)
   FirewallBrokerPort/fp(null)
   FailureLimit/fl(50)
   SSLKeystorePassword/sslpw(password suppressed)
   DomainManagerAddress/ma(9.12.4.142)
   AuthorityLevel/auth(ServerSSL)
   ProcessMode/mode(ManagedServer)
   LBPublicPort/lbp(Off)
   LBSecurePort/lbs(Off)
   EventTrace/et(250)
   TraceLevel/tl(Min)
.......................
root@ewlm4:/opt/IBM/VE/EWLMMS/bin>./startMS.sh /opt/ewlmMS
```

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:

- We create a public/private keypair for the managed server and place it into 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 6-17 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
```

- We then export the public key into a certificate:

Example 6-18 Exporting the key

```
root@ewlm4:/opt/IBM/VE/EWLMMS/bin> keytool -export -alias ms1-keystore ks_ms -rfc -file
ks_ms1.cert
Enter keystore password: 111111
Certificate stored in file <ks_ms1.cert>
```

- We send the certificate (ks_ms1.cert) to the domain manager as a binary file.
- We import the certificate (ks_ms1.cert) into the domain manager’s keystore. In our configuration the keystore is called ks_dm.

Example 6-19 Import the certificate on the domain manager

```
[root@ewlmdm1 bin]# ./changeDM.sh /opt/EWLMDM/ keytool -import -alias ms1 -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
```

- 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 6-20):

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 6-20 Enabling Client/Server SSL on domain manager

```
[root@ewlmdm1 bin]# ./changeDM.sh /opt/EWLMDM/  -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]# ./displayDM.sh /opt/EWLMDM/

Processing displayDM request.  Please be patient as this may take a while...
WLMConfig - configurable property settings:
   ViaProxyPort/vp(null)
   TracePlugin/tlog(Off)
   InterBrokerPort/dp(null)
   InterBrokerAddress/da(null)
   JmxPort/jp(9092)
   FirewallBrokerList/fb(null)
   ReportingTrace/rt(250)
   ViaProxyHost/va(null)
   DomainName/dn(itsoewlm)
   JniTrace/jt(250)
   SSLKeystore/sslks(/home/ibmewlm/ks_dm)
   ComponentTrace/ct(250)
   DomainManagerPort/mp(3333)
   MessageLog/ml(250)
   CommunicationTrace/nt(250)
   TraceDistHubBroker/tcomm(0)
   SocksPort/sp(null)
   FirewallBrokerPort/fp(null)
   FailureLimit/fl(50)
   SSLKeystorePassword/sslpw(password suppressed)
   DomainManagerAddress/ma(9.12.4.142)
   AuthorityLevel/auth(ClientServerSSL)
   ProcessMode/mode(DomainManager)
   LBPublicPort/lbp(Off)
   LBSecurePort/lbs(Off)
   EventTrace/et(250)
   TraceLevel/tl(Min)
   TestComponent/t(null)
   DumpRetentionQuantity/dpn(25)
   DumpRetentionAge/dpa(30)
   SocksHost/sa(null)
WLMConfig - non-configurable property settings:
   ManagedServerFailureTime(null)
   ManagedServerIdentity(b05b63dad0404b3a5f460829852701c6)
   ManagedServerId(-1)
   StatisticsInterval(10)
   DomainManagerIdentity(null)
PROCESSING COMPLETE

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

On each managed server (see Example 6-21):

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 the Monitoring → Managed Servers task.

Example 6-21 Enabling Client/Server SSL on managed server

```
root@ewlm4:/opt/IBM/VE/EWLMMS/bin>./changeMS.sh /opt/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>./displayMS.sh /opt/ewlmMS
WLMConfig - configurable property settings:
   ViaProxyPort/vp(null)
   TracePlugin/tlog(Off)
   InterBrokerPort/dp(null)
   InterBrokerAddress/da(null)
   JmxPort/jp(Off)
   FirewallBrokerList/fb(null)
   ReportingTrace/rt(250)
   ViaProxyHost/va(null)
   DomainName/dn(itsoewlm)
   JniTrace/jt(250)
   SSLKeystore/sslks(/usr/java14/bin/ks_ms)
   ComponentTrace/ct(250)
   DomainManagerPort/mp(3333)
   MessageLog/ml(250)
   CommunicationTrace/nt(250)
   TraceDistHubBroker/tcomm(0)
   SocksPort/sp(null)
   FirewallBrokerPort/fp(null)
   FailureLimit/fl(50)
   SSLKeystorePassword/sslpw(password suppressed)
   DomainManagerAddress/ma(9.12.4.142)
   AuthorityLevel/auth(ClientServerSSL)
   ProcessMode/mode(ManagedServer)
   LBPublicPort/lbp(Off)
   LBSecurePort/lbs(Off)
   EventTrace/et(250)
   TraceLevel/tl(Min)
.......................
root@ewlm4:/opt/IBM/VE/EWLMMS/bin>./startMS.sh /opt/ewlmMS
```

6.2.4 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 QWLM 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).

6.2.5 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 Local System Account or in an account that belongs to the Administrators group or the EWLMArm4Users group.

7

Using load balancing

This chapter provides a description of how Enterprise Workload Manager (EWLM) influences load balancing through the use of dynamic weight generation capabilities.

It also describes the configuration steps needed to accomplish the load balancing and data flow that takes place.

7.1 EWLM and workload balancing overview

The EWLM domain manager has a global view of all the servers and middleware technologies which make up the management domain that supports the business applications, as described in 1.2.1, “Domain manager” on page 6. The domain manager obtains performance information on 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 Configuring an EWLM load balancing setup

Figure 7-1 shows the components of an EWLM management domain which consists of managed servers, a domain manager, and a load balancer.

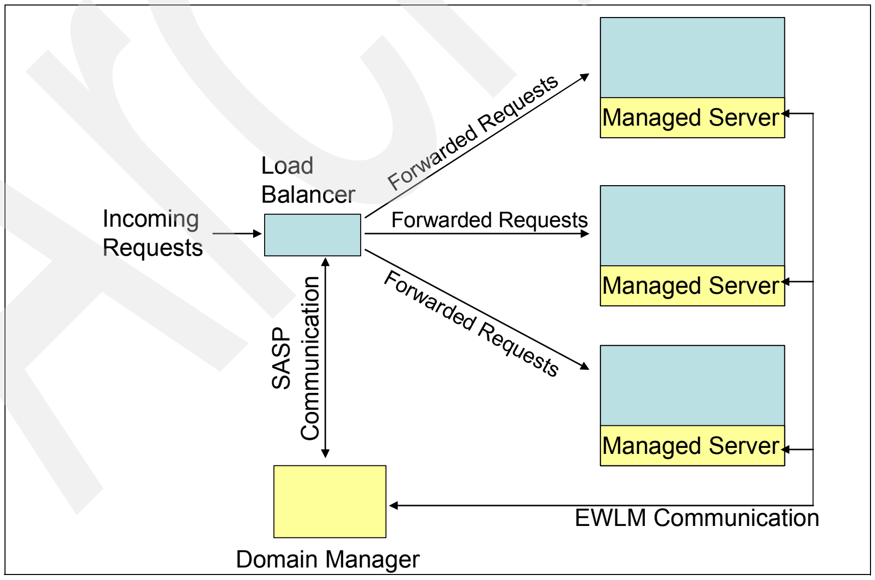


Figure 7-1 EWLM workload balancing components

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.

7.2.1 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 “Configuring managed servers on AIX and Windows” 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. For Solaris, refer to “Managed server on Solaris” on page 54.

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 7-1 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 which 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 7-1 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
```

7.2.2 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 hostname 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(s) 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.

7.2.3 Load balancer

Today, without EWLM, a load balancer routes work to multiple servers (or applications). The load balancer would define a server group which 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.

7.3 Interfaces

This section describes the types of interfaces between the different elements.

7.3.1 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

7.3.2 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

7.3.3 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

7.4 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.

7.4.1 System-level identification

The flow illustrated in Figure 7-2 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.

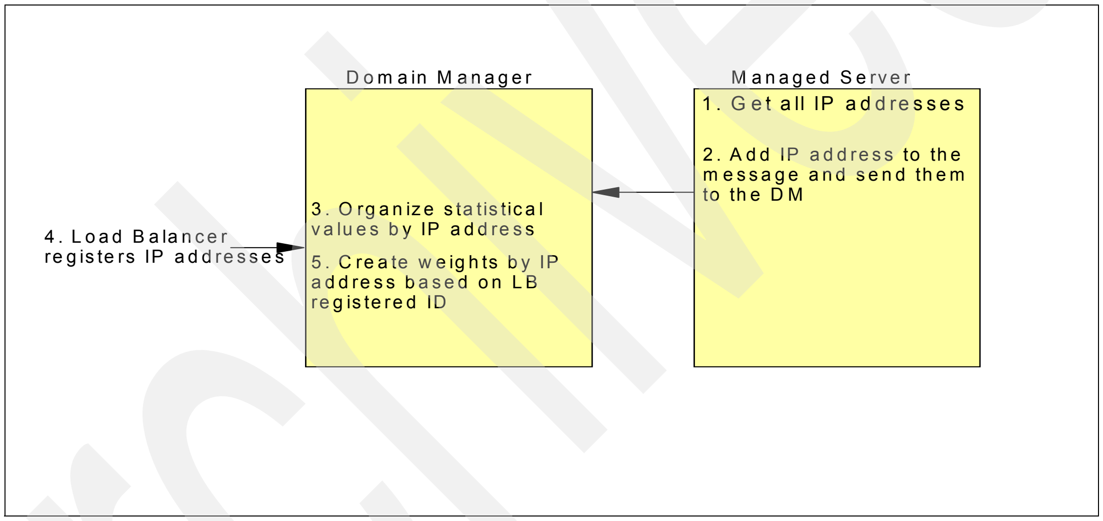

Domain Manager
Managed Server
1. Get all IP addresses
2. Add IP address to the message and send them to the DM
3. Organize statistical values by IP address
5. Create weights by IP address based on LB registered ID
4. Load Balancer registers IP addresses

Figure 7-2 System-level identification

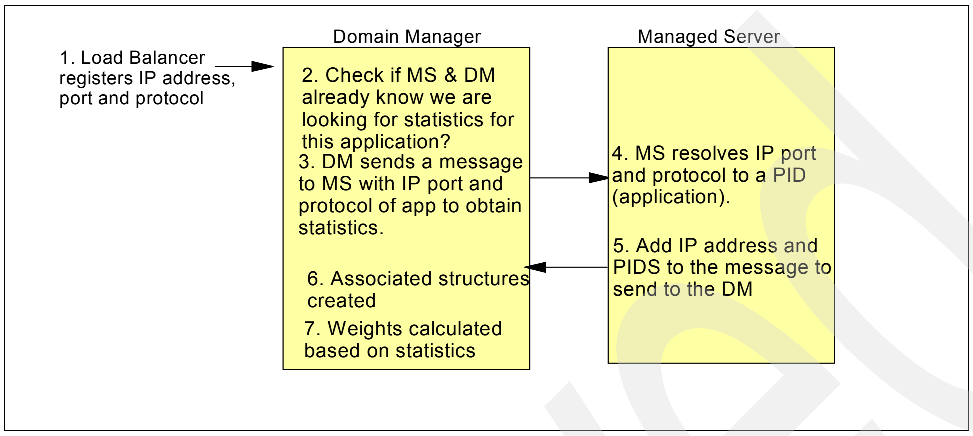
1. The managed server determines all the IP addresses associated with the platform. This occurs every 30 seconds.
2. The IP addresses are sent from the managed server to the domain manager.
3. The domain manager organizes statistical values by the IP addresses passed by the managed server in the message.
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.

7.4.2 Application-level identification

The flow illustrated in Figure 7-3 shows the mechanism for application-level identification. The application-identification mechanism organizes similar data by the corresponding application criteria registered by the load balancer.



Domain Manager
Managed Server
1. Load Balancer registers IP address, port and protocol
2. Check if MS & DM already know we are looking for statistics for this application?
3. DM sends a message to MS with IP port and protocol of app to obtain statistics.
4. MS resolves IP port and protocol to a PID (application).
5. Add IP address and PIDS to the message to send to the DM
6. Associated structures created
7. Weights calculated based on statistics

Figure 7-3 Application-level identification

1. The load balancer registers 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 doesn't 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 192), 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.

7.5 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.

7.5.1 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 doesn't 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 $\rightarrow$ will receive no credits.

  This may occur when the domain manager has missed the last several statistical update messages or if the managed server hasn’t yet connected to the domain manager.

- Instance is quiesced: set weight = 0, and quiesced flag = 1 $\rightarrow$ 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 currently is determined by:

  ```
  CPU Utilization=Total CPU Time used/(CPU Count * Time Interval)
  ```

- Importance level of work running on systems.

  This is specified in the service policy for each service class as described in “Setting service class goals” on page 139.

- 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 (see “Service class details” on page 108).

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 least important work.

Once steps 1 though 3 are completed, the three normalized reward distributions will be statistically combined to form a general reward distribution, and weights will be calculated, prioritized in the following order:

- The highest weights should go to the most under-utilized systems.
- Of these systems, preferences should be given to those that are meeting the goals of the majority of their work.
- Of these systems, preferences should be given to those that are meeting goals of the most important work.
- Of these systems, preferences should be given to those that are running the least important work.

7.5.2 Application-level balancing

Application-level balancing is similar to system-level balancing in that requests or transactions are routed to applications which 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 shouldn't 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 who have had successful transactions. To do this, a Success Factor (SF) is created which measures the success rate:

   ```
   SF = 1 - (Failed Transaction Count / Total Transaction Count)
   ```

   Note: If the SF is too low, the instance is placed into a 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 which 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 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 195.

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 made, 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, prioritized in the following order:

- The highest weights should be given to under-utilized systems.
- Of these systems, preference should be given to those which have been very successful in completing transactions.
- Of these systems, preference should be given to applications which meet their goals for the highest percentage of their work.
- Of these systems, preferences should be given to applications which meet their goals for their most important work.
- Of these systems, preference should be given to those which are blocked the least.
- Of these systems, preference should be given to those which are running the least important work.

7.6 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:

a. Non-encrypted port (`-lbp <port>`)

b. SSL Port (authentication with encryption, `-lbs <port>`)

c. Both ports (`-lbp <port> -lbs <port>`)

- If option b or c is selected, the domain manager's keystore (`-sslks <keystore path>`) as well as the keystore password (`-sslpw <keystore password>`) are needed. These options can be specified as configuration parameters on the **`changeDM`** command.

7.7 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.

7.7.1 Load balancer

The load balancer should provide the following:

- The ability to display the server groups which 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.

7.7.2 Managed server

Commands

To list the applications registered to ARM on an AIX:

```
/usr/sbin/lsarm -a
```

To see if the netwlm daemon is started on AIX:

```
ps -ef netwlm
```

To determine if the common code and the netwlm code was installed (refer to "Configuring managed servers on AIX and Windows" on page 46):

```
AIX - smitty.
```

To display current properties of the managed server:

```
displayMS commands
```

To determine how many transactions a particular system or application has processed:

This 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 which 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

7.7.3 Domain manager

EWLM Control Center

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 "First-level reports" on page 104.

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.

8

Troubleshooting and diagnostics

This chapter describes some of the challenges we encountered while we were installing Enterprise Workload Manager, configuring the environment, and verifying that our configuration was correct. It also covers the commands or steps used to resolve the errors. This chapter also describes what facilities are available for you to verify that your environment is set up correctly.

8.1 Problems with installing/uninstalling the EWLM code

INST Problem

Problem reinstalling the IBM Virtualization Engine.

Error received

No specific error message is received. The previous message received was:

```
The IBM Virtualization Engine uninstaller has successfully completed.
```

Expanded description

After uninstalling the IBM Virtualization Engine and trying to reinstall it we ran into a problem where the new install could not install using the same default VE directory (directory already exists).

Diagnosis and solution

We navigated to the default directory for IBM VE and it still existed. This is because the uninstall Virtualization Engine wizard does not remove the VE directory or driver, even though it says it does. We manually deleted the directory /opt/IBM/VE and ran the IBM VE install again using the wizard. This time we were successful.

8.2 Problems with the domain manager

The following paragraphs describe common problems you can observe operating on the domain manager.

8.2.1 Configuration

Problem

Processing **createDM** request failed.

Error received

```
Processing createDM request. Please be patient as this may take a while...
...Configuring EWLM Control Center
BUILD FAILED
file:/opt/IBM/VE/WebSphere/AppServer/bin/wsinstance/instance.xml:206: Cannot write to
/ewlmDM/WAS/logs/updatePorts_ewlmdm1_/ewlmDM2001814211.log
Total time: 6 seconds
...processing 20% complete
     [exec] Result: 255
ERROR: Unable to start WebSphere instance. Possible port collision detected.
Cleaning up after createDM failure. Please be patient as this may take a while...
Copy WebSphere logs into createDM log file before deleting directory
```

Expanded description

When we installed the EWLM code on our machines using the VE Installation Wizard, we specified the IBM Virtualization Engine install directory as /ewlmdm. Although the installation ran successfully, when we tried to use the directory (by issuing the **createDM** command, which in turn was creating the WebSphere Application Server on the Linux machine), we received the error described.

Diagnostics and solution

After looking through the **`createDM`** log, and some trial and error, we discovered that the VE install directory cannot be defined to be in the root directory(/). We created a new directory, /opt/ewlmdm, and used it instead of /ewlmdm.

8.2.2 Operations

Problem

EWLM activation error opening internal database.

Error received

```
EWLM activation error: opening internal database
EWLM activation error: db2j.p.j
EWLM activation error: Failed to start database ‘PolicyDB’ see the next exception for
details.
PROCESING COMPLETE
```

Expanded description

When the domain manager is started it uses an internal database. Lock files are created for the various databases in use to assure integrity of the data. When the domain manager does not stop successfully there are locks open on the database. Therefore, we are not able to successfully restart the domain manager because it cannot open the database since the database has locks against it.

Diagnosis and solution

In our directory, /opt/EWLMDM/eWLMData, there are several subdirectories for each of the databases we are using. If we go into one of the directories (PolicyDB, for example), we see a file called dbex.lck. This is the lock on the PolicyDB database. In order to safely restart the domain manager, we need to successfully close these locks. We recommend rebooting the machine which the domain manager is trying to start because this will release the locks safely.

Problem

EWLM activation error starting the EWLM domain manager.

Error received

```
Starting EWLM Domain Manager ...
EWLM activation error: opening internal databases
EWLM activation error: db2j.p.j
EWLM activation error: Failed to start database ‘ReportingDB’, see the next exception for
details.
PROCESSING COMPLETE
```

Expanded description

After encountering a problem with the domain manager, we killed the ./startDM.sh process via the **`kill`** command. However, this did not terminate all of the domain manager processes that were running. When we tried to restart the domain manager, one (or more) of these processes were using the database and thus we were not able to start it.

Killing the shell script doesn't stop the actual Java process, especially if you are running the domain manager on Linux where each Java thread looks like a Java process in "`ps -ef`". In this case you need to issue a "`ps -eH`" and kill all the Java processes under the startDM.sh script to completely shut down the domain manager.

If you are running on AIX, you should kill the Java process, and since there is only one for the domain manager, then the startDM.sh script will exit gracefully and the database will be released too.

Diagnosis and solution

It is easier to start domain manager and managed server commands in the foreground so if you need to stop the domain manager or managed servers, you can issue a CONTROL-C from the window where the server was started. If you need to run these processes in the background, you need to be familiar with the operating system in order to be able to stop the correct process.

Problem

The domain manager directory was not deleted during the uninstall.

Error received

```
EWLM Domain Manager ...
EWLM activation error: opening internal databases
EWLM activation error: db2j.p.j
EWLM activation error: Failed to start database ‘ReportingDB’, see the next exception for
details.

PROCESSING COMPLETE
```

Expanded description

We tried to delete the domain manager and recreate it but were unable to clean up the entire environment with the **deleteDM.sh** command. We then tried to create a new DM using the same directory but could not since the directory was still there from our previous install.

Diagnosis and solution

To bypass the problem you can either delete the old directory and reissue the **createDM** command or you can use a different working directory on the **createDM** command if you need to preserve data in the old working directory.

Problem

Cannot display the domain manager properties when the domain manager is running.

Error received

```
EWLM customization error: loading customized properties
ERROR: Domain Manager must be stopped in order to display properties
```

Expanded description

At this time, you cannot display the properties of the domain manager when the domain manager is running.

Diagnosis and solution

At this time, you cannot display or change any of the EWLM components dynamically. In order to display the properties of the domain manager you need to first stop the domain manager. Since the domain manager was started in the foreground, there should be a window open where it was started. Go to that window and stop the domain manager with the CONTROL-C function. Then issue the **./displayDM.sh** command again to display the properties.

8.3 Problems with WebSphere Application Server

The following paragraphs describe a common problem that you can encounter on the WebSphere Application Server.

8.3.1 Starting

Problem

Starting the WebSphere Application Server failed initialization.

Error received

```
startWAS command started at:  Wed May 19 12:00:20 EDT 2004 Platform =  Linux
STEP 1: Access EWLM working directory
EWLM working directory: /opt/EWLMDM/
STEP 2: Start WebSphere
ADMU0116I: Tool information is being logged in file
           /opt/EWLMDM/WAS/logs/server1/startServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3011E: Server launched but failed initialization. Server log files should
           contain failure information.
Unable to start server
PROCESSING COMPLETE
```

Expanded description

We were unable start the WebSphere Application Server.

Diagnosis and solution

There are typically two reasons why we received the error:

1. Another program is using the port which WebSphere requires to start.
2. The userid which is issuing the **`startWAS`** command is not root.

8.4 Problems with the managed servers

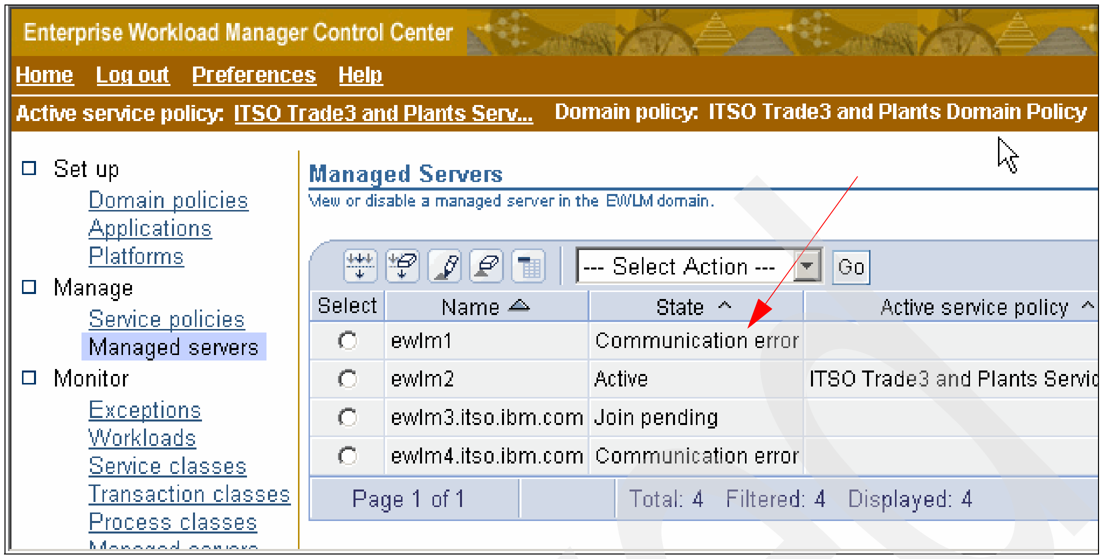
The following paragraphs describe a common problem that you can encounter when operating on the managed servers.

Problem

Communication error to the managed server.

Error received

When using the EWLM Control Center, we selected the managed server option under Manage to view our managed servers. Instead of seeing “Active” for each of our servers, we received a “Communication error.”



Active service policy: ITSO Trade3 and Plants Serv... Domain policy: ITSO Trade3 and Plants Domain Policy

Managed Servers

View or disable a managed server in the EWLM domain.

Select	Name	State	Active service policy
○	ewlm1	Communication error	
○	ewlm2	Active	ITSO Trade3 and Plants Servic
○	ewlm3.itso.ibm.com	Join pending	
○	ewlm4.itso.ibm.com	Communication error	

Page 1 of 1 Total: 4 Filtered: 4 Displayed: 4

Figure 8-1 Communication error to the managed servers

Expanded description

The State column provides an explanation of what operating state the managed server is in. Refer to “Managed servers” on page 103 for a description of the different states.

Diagnostics

There are several reasons that a communication error might appear in this column:

1. The operating systems are up (ewlm1, elwm2) but the managed server has not been started.
2. The managed server is hung.
3. On our AIX system, we enabled the EWLM services (see “Configuring managed servers on AIX and Windows” on page 46), but we rebooted the machine and EWLM services were not enabled permanently.
4. The domain manager lost connectivitiy to the managed server.

Solutions

1. Make sure the `startMS` command has been issued on the various operating system(s) if it has not been started already.
2. If the managed server is hung, stop the managed server with the CONTROL-C command and restart it with the `startMS` command.
3. Use `smitty` to enable the EWLM services and set it so that it will automatically restart upon the next boot.
4. For connectivity problems to the managed server, check the IP configurations with the `ipconfig` command on both sides of the communication link. Try to `ping` the other server to make sure there is connectivity. If the `ping` is unsuccessful check any routers and networking that comes between the two systems with the `route` command. Make sure the managed server has been started.
5. Wait a few seconds and refresh the screen. More than likely the managed server was just starting, which would show “Join pending” on the EWLM Control Center screen. This should turn to “Active” shortly.

8.5 Problems enabling ARM on middleware applications

The following paragraphs describe a common problem that you can observe while enabling instrumentations on middleware applications, such as WebSphere Application Server, Web server, and UDB DB2.

Problem

HTTP Plug-in is not ARM instrumented correctly.

Error received

```
- ERROR: ws_arm: _armInitialize: 5: -1021
- ERROR: ws_arm: _armInitialize: 15: -1015
```

Expanded description

We were unable start the WebSphere Application Server.

Diagnosis and solution

The user that runs the http server (nobody) is not defined in the /etc/EWLM/auth file. Check the following in the HTTP log:

```
- PLUGIN: Plugins loaded.
- PLUGIN: --------------------System Information-----------------------
- PLUGIN: Bld version: 5.1.0
- PLUGIN: Bld date: May 11 2004, 13:39:44
- PLUGIN: Webserver: IBM_HTTP_Server/2.0.47 Apache/2.0.47 (Unix) DAV/2
       or
- PLUGIN: Webserver: IBM_HTTP_SERVER/1.3.28.1  Apache/1.3.28 (Unix)
- PLUGIN: Hostname = hes017
- PLUGIN: NOFILES = hard: 65536, soft: 65536
- PLUGIN: MAX COREFILE SZ = hard: INFINITE, soft: INFINITE
- PLUGIN: DATA = hard: INFINITE, soft: INFINITE
- PLUGIN: --------------------------------------------------------------
```

8.6 Problems using the EWLM Control Center

The following paragraphs describe common problems that you might observe on the EWLM Control Center:

8.6.1 General use

Problem

We were unable to log in to the EWLM Control Center with a valid userid.

Error received

```
Unable to process login. Specify a valid user name and password and try again.
```

Expanded description

We could not log in to the EWLM Control Center when we tried logging in using a valid userid and password. After several attempts at loggin on, sometimes we were successful.

Diagnosis

Possible causes of this problem are the following:

- EWLM Control Center is supported on IE V6 SP1. If you are on a previous version, it may work, but you may have intermittent problems.
- You need to enable cookies.
- You need to enable Javascript.
- Do not set security to the maximum on IE because this renders your login unsuccessful.
- You need a JVM plug-in to display graphs.
- The EWLM Control Center uses SSL/TLS for communications.

Solution

- We recommend you scan your computer and install all critical Internet Explorer updates. For example, from Internet Explorer, choose **Tools** → **Windows Update**. This will take you to the Microsoft Window Update internet site. Click **Scan for updates**. Apply the latest critical updates, including KB831167, which is needed to allow a valid userid to log in without first waiting one full minute after presentation of the login page.
- You need to set the security in your browser to use TLS. To do this follow these steps:

  a. In IE, click **Tools** → **Internet Options** → **Advanced**.

  b. Scroll down to Security and select **Use TLS 1.0**.

  c. Click **OK**.

8.6.2 Manage options

Problem

Communication error

Error received

See "Problems with the managed servers" on page 205.

8.6.3 Monitor options

Problem

Application topology window does not show expected results.

Error received

The Application topology for our transaction class did not show our entire topology. The Application Topology window should have looked similar to Figure 8-2 since we had an HTTP Server as hop 0, two WebSphere Application Servers as hop 1, and the DB2 Universal Database as hop 2 in our configuration. However, when we selected Application Topology from the pull-down list from the **Monitor** → **Transaction Class** selection, we got a view that looked either like Figure 8-3 (missing the DB2 hop) or Figure 8-4 (missing the HTTP hop).

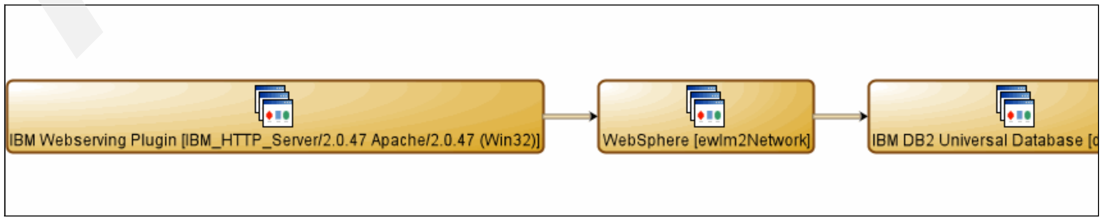

IBM Webserving Plugin [IBM_HTTP_Server/2.0.47 Apache/2.0.47 (Win32)]
WebSphere [ewlm2Network]
IBM DB2 Universal Database [d

Figure 8-2 Correct Application Topology view

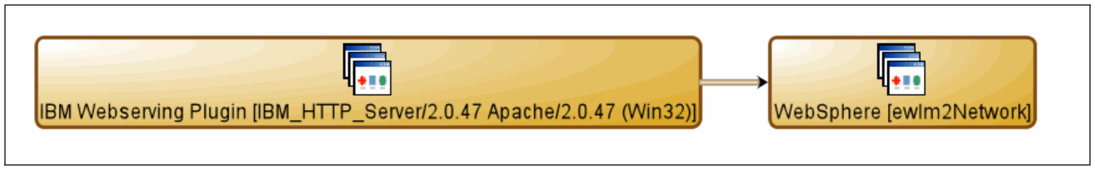
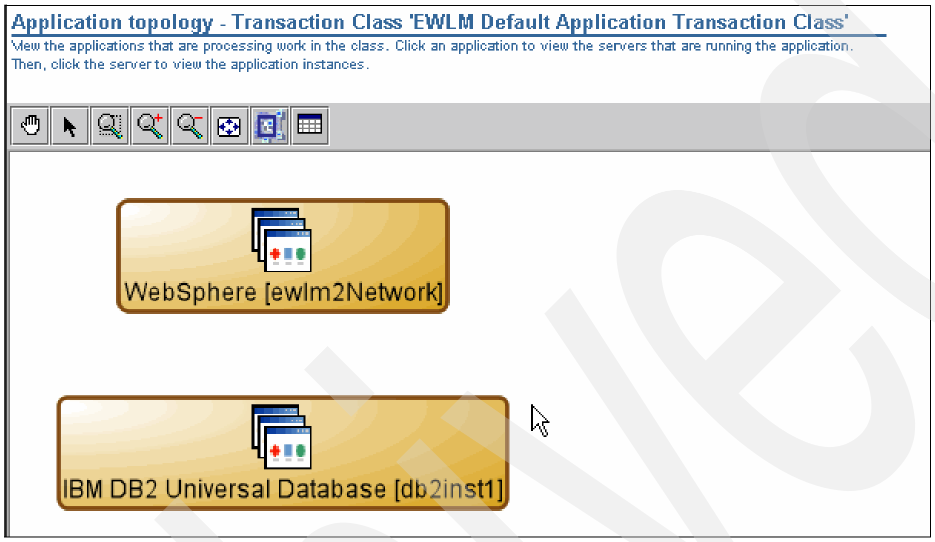

IBM Webserving Plugin [IBM_HTTP_Server/2.0.47 Apache/2.0.47 (Win32)]
WebSphere [ewlm2Network]

Figure 8-3 Broken Application Topology view 1


Application topology - Transaction Class 'EWLM Default Application Transaction Class'

View the applications that are processing work in the class. Click an application to view the servers that are running the application. Then, click the server to view the application instances.

Figure 8-4 Broken Application Topology view 2

Expanded description

In our configuration, we have an HTTP Server (hop 0), two WebSphere Application Servers (hop 1), and a DB2 database (hop 2). In the Enterprise Workload Manager Control Center, under the **Monitor** option, we clicked **Transaction classes**, selected our transaction class, clicked the pull-down, and chose **Application Topology**. We expected to see a chart in the Application Topology view that looked like Figure 8-2, but one of the WebSphere Application Servers and the DB2 were not there.

Since neither the WebSphere Application Server nor the DB2 were set up in a special transaction class, they showed up as hop 0s in the EWLM Default Application Transaction Class, as shown in Figure 8-4.

Diagnosis and solution

There are a number of reasons that this scenario may occur:

1. Managed server-related

   The managed server was not started on the machine where the application was running.

2. DB2-related

   The DB2 Universal JDBC driver supports JDBC type 2 and type 4. Since we were testing the trade3 installation script, it was using the legacy DB2 JDBC driver in type 2 mode by default (db2java.zip). The legacy driver does not propagate the ARM correlator to the next hop, so we installed the new JCC driver in type 4 mode.

   To verify that instrumentation has been set up correctly on AIX, issue the **`lsarm -a`** command. You should see:

   ```
   APPL: IBM DB2 Universal Database
   ```

If ARM is not set up correctly for DB2, see “Enabling DB2 Universal Database for ARM” on page 60.

3. HTTP-related

   Verify that the HTTP is instrumented correctly. Turn tracing on for the HTTP Server so that you can verify that the HTTP server has ARM enabled.

   a. Edit the plug-in:
      C:\Program Files\WebSphere\AppServer\configuration\cells\plugin-cfg.xml
   b. Change the log level from whatever it is to Trace <Log Loglevel=**"Trace"** Name= "C:\Program Files\WebSphere\Deployment Manager\logs\http_plugin.log"/>
   c. Restart the HTTP Server.

   In the HTTP log (specified by the Name= variable in step b) there should be new entries stating that the ARM library was successfully initialized.

   If you see the ARM entries, the HTTP Server has been configured correctly and you should turn TRACE off again since it will produce enormous amounts of data. If you do not see the ARM messages in the log, review the steps in “Enabling IBM HTTP Server for ARM” on page 65.

4. WebSphere Application Server-related

   The WebSphere Application Server may not be instrumented correctly as described in “Enabling WebSphere Application Server for ARM” on page 63.

   To verify that instrumentation has been set up correctly on AIX, issue the **`lsarm -a`** command. You should see:

   ```
   APPL: WebSphere
   ```

5. Policy-related

   Check your service class, transaction class, and policy definitions, as described in 5.5.2, “Creating the domain policy at the EWLM Control Center” on page 144. It is possible that there is a typo or an error in defining the transaction class to the correct service class.

6. Workload-related

   Verify that the workload is currently running. If there is no workload or it is at the beginning or end of the workload, EWLM might still be in the process of building the topology and is not capable of displaying the full picture yet. If this is the case, verify that the workload is running and wait at least the transaction average response time plus 10 seconds to let EWLM build the application topology graphic and check the preference setup.

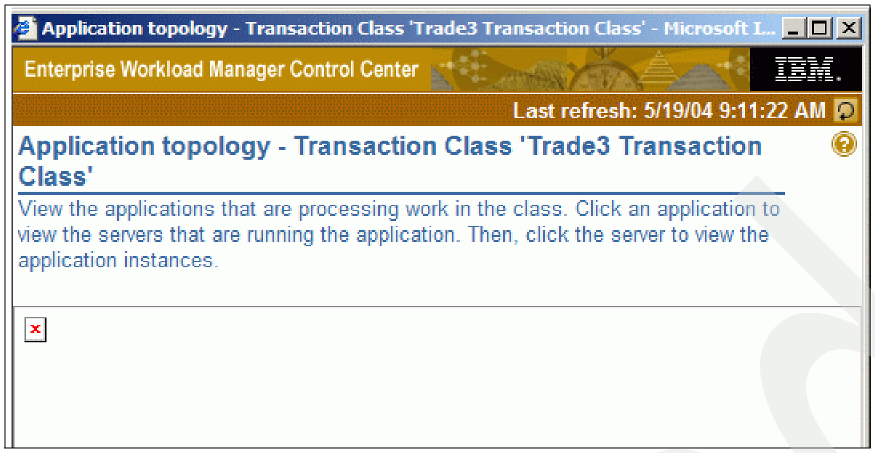
Related information is in 1.3, “EWLM supported platforms and applications” on page 16.

Problem

You cannot see the graphical displays in the EWLM Control Center.

Error received

The error received is shown in Figure 8-5.


Application topology - Transaction Class 'Trade3 Transaction Class' - Microsoft I...

Enterprise Workload Manager Control Center

Last refresh: 5/19/04 9:11:22 AM

Application topology - Transaction Class 'Trade3 Transaction Class'

View the applications that are processing work in the class. Click an application to view the servers that are running the application. Then, click the server to view the application instances.

Figure 8-5 Application Topology: Lack of Java graphical data

Expanded description

When trying to look at the Application topology of a transaction class, the view that was presented did not show the Java logo, then present the graphical display. Instead, the only view that was received is shown in Figure 8-5.

Diagnosis and solution

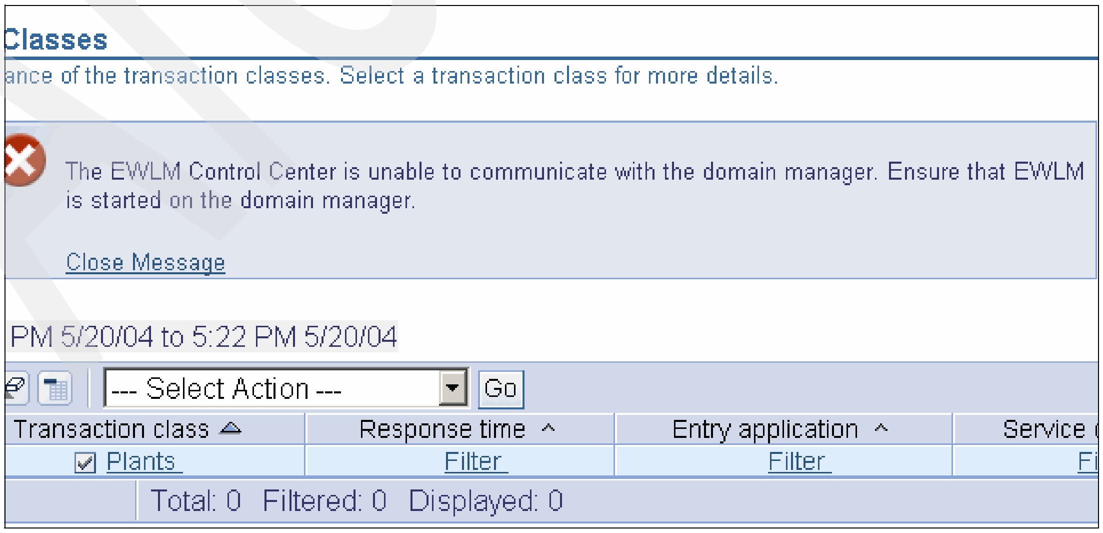
JVM 1.4.1 needs to be installed on the system where the browser is running that is trying to display the topology graph from the EWLM Control Center.

Problem

The domain manager did not provide topology data.

Expanded description

After selecting **Monitor** → **Transaction class**, and selecting **Application topology** from the pull-down list, an error message pops up stating there is a communication problem with the domain manager.


Classes

ance of the transaction classes. Select a transaction class for more details.

The EWLM Control Center is unable to communicate with the domain manager. Ensure that EWLM is started on the domain manager.

Close Message

PM 5/20/04 to 5:22 PM 5/20/04

Transaction class	Response time	Entry application	Service
☑ Plants	Filter	Filter	Fi

Total: 0 Filtered: 0 Displayed: 0

Figure 8-6 Communication problem with the domain manager

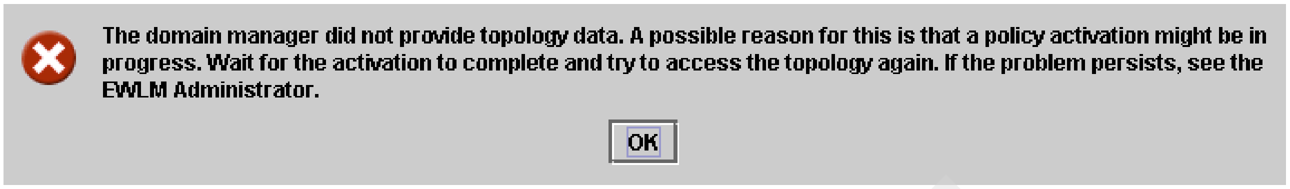

The domain manager did not provide topology data. A possible reason for this is that a policy activation might be in progress. Wait for the activation to complete and try to access the topology again. If the problem persists, see the EWLM Administrator.

Figure 8-7 Application topology data missing

Diagnosis and solution

One of the following might be the cause of the problem:

- A policy activation is in progress and we need to wait till it completes.
- The domain manager has not been started.

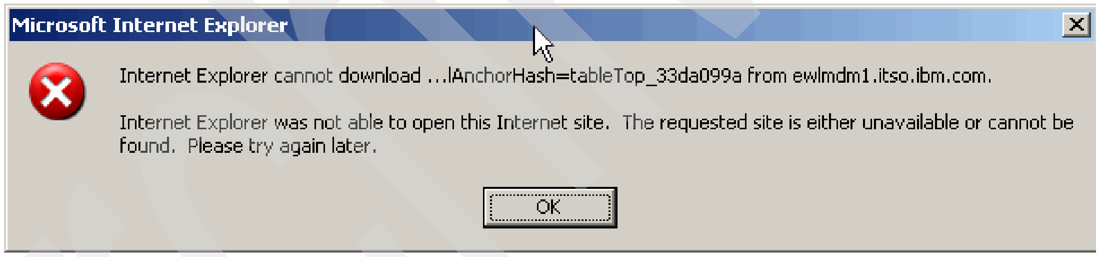
8.6.4 Domain policies

Problem

```
Unable to download. Internet Explorer was unable to open this internal site. The
requested site is either unavailable or cannot be found. Please try again later.
```

Expanded description

We were unable to export the domain policy over SSL from a remote browser to the domain manager. When we try to export the domain policy from the local browser we have no problem.



Internet Explorer cannot download ...lAnchorHash=tableTop_33da099a from ewlmdm1.itso.ibm.com.

Internet Explorer was not able to open this Internet site. The requested site is either unavailable or cannot be found. Please try again later.

Figure 8-8 Detailed description of export problem

Diagnosis and solution

We determined that Internet Explorer (IE) file downloads over SSL do not work with the cache control headers, so we needed to take the following steps for IE:

1. Start the Registry Editor on Windows.

2. For per-user setting, locate the following registry key:

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings

For per-user setting, locate the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings

3. One the Edit menu, click **Add Value**, and then add the following registry values:

```
"BypassSSLNoCacheCheck"=Dword:00000001
```

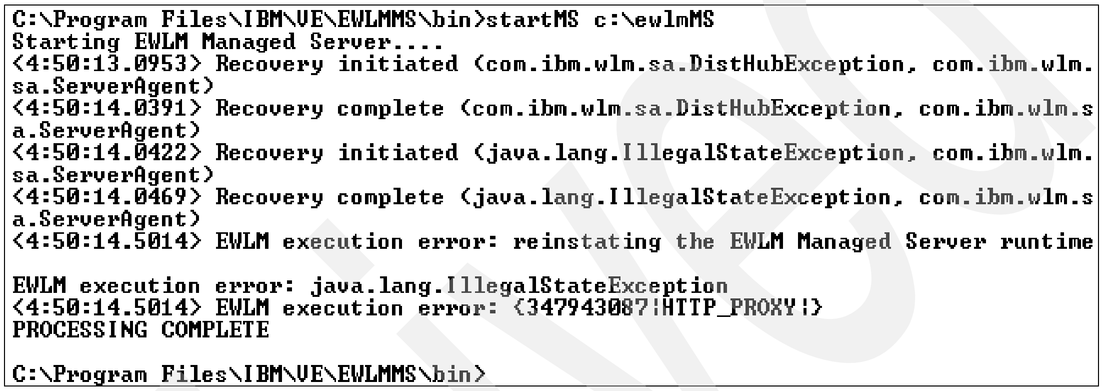
4. Quit the Registry Editor.

For additional information see the Microsfot Web site at:

```
http://support.microsoft.com/?kbid=323308#appliesto
```

8.7 Defining security and firewall

From the managed server, the **changeMS** command is issued to add the IP address and port of the HTTP proxy server. This is required if the managed server must access the domain manager via an HTTP proxy server. When the **changeMS** command is issued there needs to be a route from the managed server to the proxy server, otherwise you will see an error similar to Figure 8-9.



```
C:\Program Files\IBM\VE\EWLMMS\bin>startMS c:\ewlmMS
Starting EWLM Managed Server....
<4:50:13.0953> Recovery initiated (com.ibm.wlm.sa.DistHubException, com.ibm.wlm.
sa.ServerAgent)
<4:50:14.0391> Recovery complete (com.ibm.wlm.sa.DistHubException, com.ibm.wlm.s
a.ServerAgent)
<4:50:14.0422> Recovery initiated (java.lang.IllegalStateException, com.ibm.wlm.
sa.ServerAgent)
<4:50:14.0469> Recovery complete (java.lang.IllegalStateException, com.ibm.wlm.s
a.ServerAgent)
<4:50:14.5014> EWLM execution error: reinstating the EWLM Managed Server runtime

EWLM execution error: java.lang.IllegalStateException
<4:50:14.5014> EWLM execution error: {347943087!HTTP_PROXY!}
PROCESSING COMPLETE

C:\Program Files\IBM\VE\EWLMMS\bin>
```

Figure 8-9 Communication error to the firewall

8.8 Diagnostic information

This section describes some sources of information to help you diagnose problems and debug your systems.

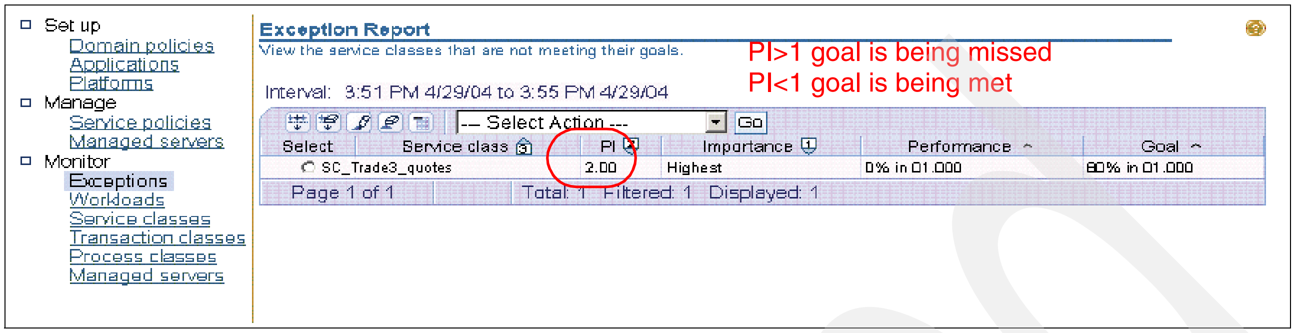
8.8.1 EWLM Control Center

The EWLM Control Center can be used to monitor the management domain when a service policy has been activated, as described in "Reporting" on page 150. From the Monitor option, you can look at exceptions, the performance goals versus the actual performance results for service classes, transaction classes, and process classes, as well as the state of the managed servers (such as active or if it has a communication error). From there, you can look at more detailed information regarding the various monitoring results.

The EWLM Control Center provides a consolidated view of how well things are performing across your enterprise. If a service class is not meeting the performance goals (work has a PI greater than 1) it will show up in the Exceptions Report as shown in Figure 8-10. It is from here that you can select an item from the pull-down to view:

- Service class details
- Application topology
- Server topology
- Performance index monitors
- Goal achievement monitors
- Transaction count monitors
- Transaction rate monitors

Providing that there are no error in your classification, you can use the EWLM Control Center to drill down further to find what is causing the problem. Please refer to “Reporting” on page 150 which describes the drill down steps to identify the cause of the problem.


- Set up
  - Domain policies
  - Applications
  - Platforms
- Manage
  - Service policies
  - Managed servers
- Monitor
  - Exceptions
  - Workloads
  - Service classes
  - Transaction classes
  - Process classes
  - Managed servers

Exception Report

View the service classes that are not meeting their goals.

PI>1 goal is being missed
PI<1 goal is being met

Interval: 3:51 PM 4/29/04 to 3:55 PM 4/29/04

-- Select Action --   Go

Select	Service class	PI	Importance	Performance	Goal
○	SC_Trade3_quotes	2.00	Highest	0% in 01.000	80% in 01.000

Page 1 of 1   Total: 1 Filtered: 1 Displayed: 1

Figure 8-10 Exception report

8.8.2 Commands

`displayDM`	This command displays the current properties of the domain manager.
`displayMS`	This command displays the current configurable and non-configurable properties settings of the managed server. This includes serviceability options such as trace sizes, failure limits, trace levels, and dump retention age.
`displayFB`	This command is used to display the current properties of the Firewall Broker.
`displaySSEWLM`	This command is used to display the current properties of the Single System EWLM.
`displayCC`	This command is used to display the ports, users, groups, and security currently used by WebSphere for the EWLM Control Center.

Note: 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 change command, then restart the service.

8.8.3 Logs

Common command logs

For each configuration command, there is a corresponding log that gets created in the same directory as the command itself. The configuration commands are:

- create*XX*
- change*XX*
- delete*XX*
- display*XX*
- start*XX*
- stop*XX*

The *XX* should be replaced for 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

Trace logs

For diagnostics purposes, EWLM provides several trace logs which can be used for debugging. These logs are kept in storage and exist within the EWLM dumps themselves. They can be written out to a trace log using the EWLM file trace plug-in and formatted as an XML document.

The trace logs can be found in the EWLM working directory in the Diagnostics folder.

WebSphere logs

Additional information can be found in the WebSphere Application Server logs when tracing has been turned on. These logs can be found in:

- DM_WORKING_DIR/WAS/logs/server1
  - systemOut.log
  - systemErr.log

Error logs

If an error occurs while EWLM is starting, causing EWLM to terminate, the error will go in:

- EWLM_root\Diagnostics

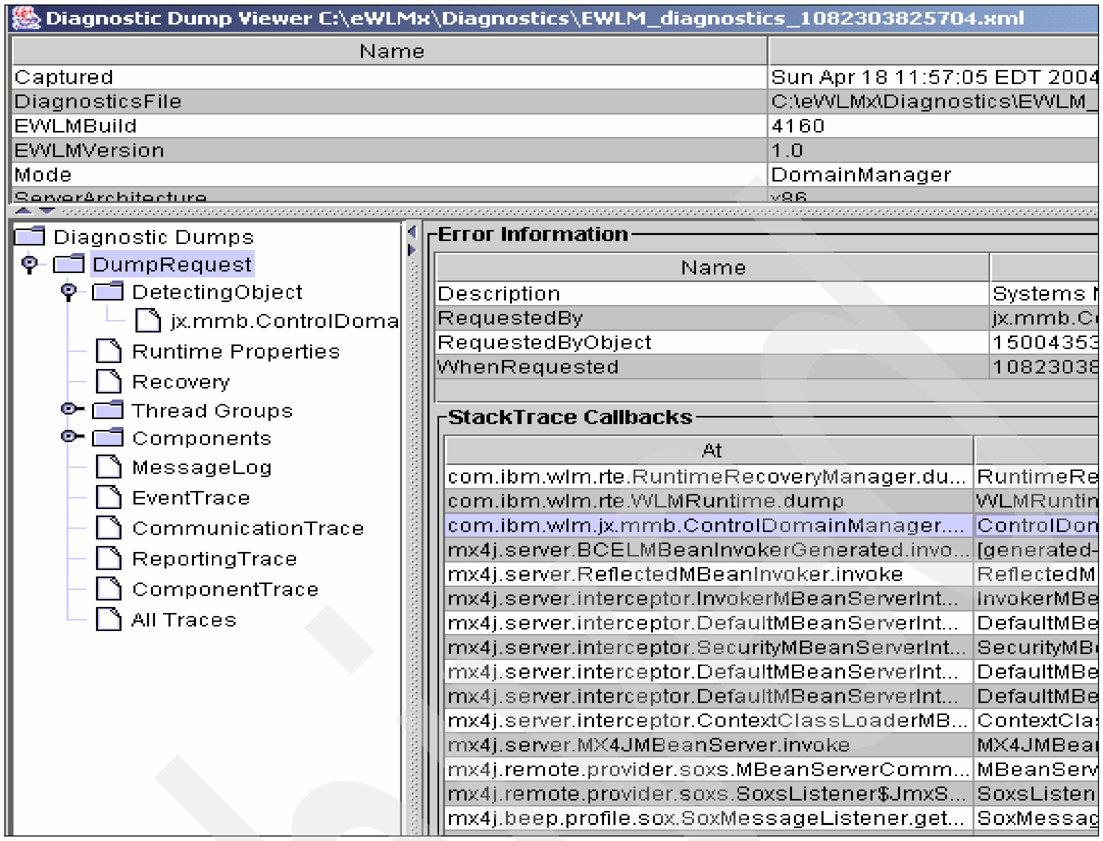
8.8.4 Dumps

Dumps are automatically generated for EWLM error conditions and are saved as XML documents. The dumps are saved in the EWLM working directory in the Diagnostics folder. The dumps can then be sent to IBM level 2 for debugging.

The EWLM dumps contain the following information:

- Basic service information such as the EWLM version and build number
- Error information and callback stack (where the error occurred)
- Runtime properties, such as authority level, trace sizes, firewall broker information
- Recovery information
- Thread identification (what threads are running and their names)
- EWLM component information
- Some trace entries dump the trace logs

Figure 8-11 shows an example of the dump information showing the error information and stack trace.


Diagnostic Dump Viewer C:\eWLMx\Diagnostics\EWLM_diagnostics_1082303825704.xml

Name	
Captured	Sun Apr 18 11:57:05 EDT 2004
DiagnosticsFile	C:\eWLMx\Diagnostics\EWLM_
EWLMBuild	4160
EWLMVersion	1.0
Mode	DomainManager
ServerArchitecture	x86

Diagnostic Dumps
- DumpRequest
  - DetectingObject
    - jx.mmb.ControlDoma
  - Runtime Properties
  - Recovery
  - Thread Groups
  - Components
  - MessageLog
  - EventTrace
  - CommunicationTrace
  - ReportingTrace
  - ComponentTrace
  - All Traces

Error Information

Name	
Description	Systems I
RequestedBy	jx.mmb.C
RequestedByObject	15004353
WhenRequested	10823038

StackTrace Callbacks

At	
com.ibm.wlm.rte.RuntimeRecoveryManager.du...	RuntimeRe
com.ibm.wlm.rte.WLMRuntime.dump	WLMRuntin
com.ibm.wlm.jx.mmb.ControlDomainManager....	ControlDon
mx4j.server.BCELMBeanInvokerGenerated.invo...	[generated-
mx4j.server.ReflectedMBeanInvoker.invoke	ReflectedM
mx4j.server.interceptor.InvokerMBeanServerInt...	InvokerMBe
mx4j.server.interceptor.DefaultMBeanServerInt...	DefaultMBe
mx4j.server.interceptor.SecurityMBeanServerInt...	SecurityMB
mx4j.server.interceptor.DefaultMBeanServerInt...	DefaultMBe
mx4j.server.interceptor.DefaultMBeanServerInt...	DefaultMBe
mx4j.server.interceptor.ContextClassLoaderMB...	ContextCla
mx4j.server.MX4JMBeanServer.invoke	MX4JMBeat
mx4j.remote.provider.soxs.MBeanServerComm...	MBeanServ
mx4j.remote.provider.soxs.SoxsListener$JmxS...	SoxsListen
mx4j.beep.profile.sox.SoxMessageListener.get...	SoxMessag

Figure 8-11 EWLM dump showing error and stacktrace

8.8.5 Traces

EWLM provides trace information which is resident in storage. Trace information exists within the EWLM dump and can be written out to a trace log using the EWLM file trace plug-in. Trace files are formatted as XML documents.

There are seven types of traces:

1. Message log

   The Message log provides a resident log of information status messages generated by components within an EWLM process. The log maintains a history of events and activities deemed important by the originating components.

2. Communications trace

   Communications trace provides a runtime trace object that records summary information about the transmission and reception of communication messages between the EWLM domain manager and an EWLM managed server instance.

3. Event trace

   Event trace records state transitions of Event objects within the EWLM execution process.

4. Component trace

   Component trace records state transitions of component objects within the EWLM execution process.

5. Report trace

   Report trace records generated performance reporting data within the EWLM execution process.

6. JNI trace

   JNI trace records interactions between components of the EWLM managed server environment with the underlying operating system platform.

The level of tracing can be set from the web.xml file in the following directory: ewlmRoot\WAS\config\cells\userName\applications\wlmwebui.ear\deployments\wlmwebui\webui.war

Example 8-1

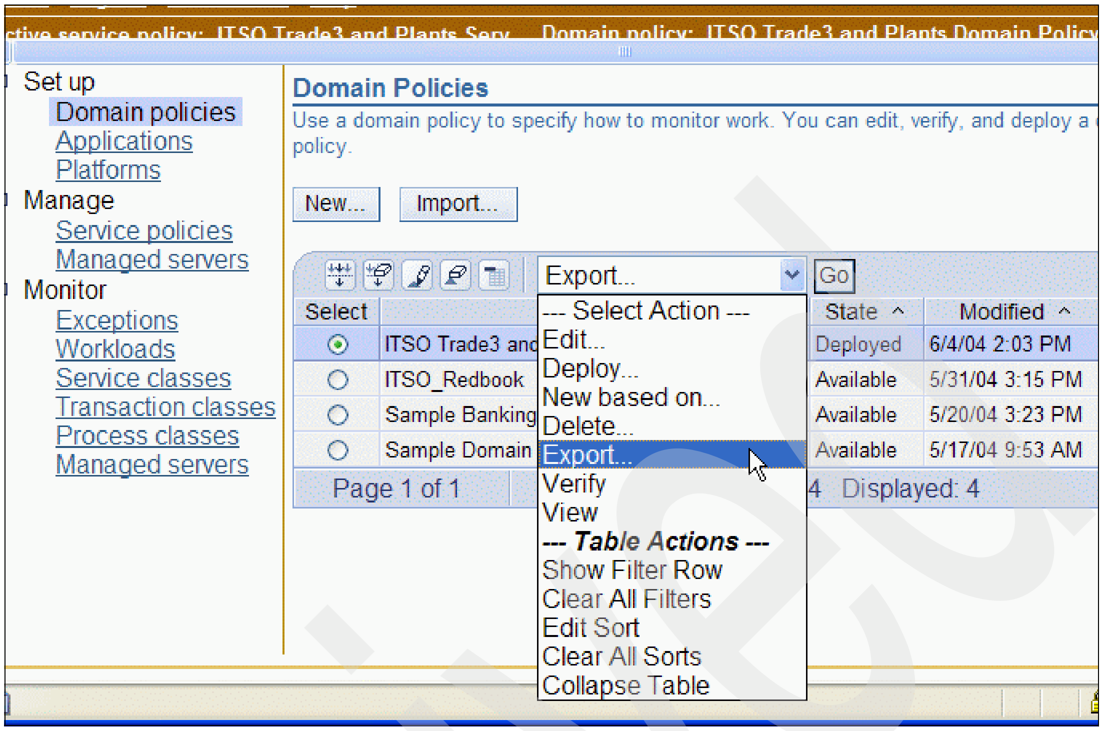
```
</init-param>
      <init-param id="InitParam_1084284254472">
         <param-name>wlm.trace.level</param-name>
         <param-value>3</param-value>
      </init-param>
      <load-on-startup>2</load-on-startup>
```

7. LoadBalancingTrace

   The LoadBalancingTrace provides information concerning the Load Balancing function in EWLM.

8.9 Database backup and recovery

Before updating your domain policy with new or changed definitions, you will want to save the old policy so that you can fall back to it in case there is a problem with the new policy or the new policy does not perform as expected. To do this, you want to **Export** your domain policy by logging into the EWLM Control Center and selecting **Domain policies** under the **Set up** option. A list of the domain policies appears in the right-hand pane. Select the domain policy that you would like to update, select the **Export** function from the pull-down list, and click **Go**, as shown in Figure 8-12 on page 218.



Domain Policies

Use a domain policy to specify how to monitor work. You can edit, verify, and deploy a policy.

Select		State	Modified
●	ITSO Trade3 and	Deployed	6/4/04 2:03 PM
○	ITSO_Redbook	Available	5/31/04 3:15 PM
○	Sample Banking	Available	5/20/04 3:23 PM
○	Sample Domain	Available	5/17/04 9:53 AM

Figure 8-12 Export domain policy

9

Initial 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

This chapter mainly focuses on the domain manager performance measurement.

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.

9.1 Domain manager resource description

The domain manager is essentially comprised of two Java processes. The domain manager process is responsible for aggregating data statistics from the managed servers; distributing policies to the managed servers; and providing reporting data to the Control Center and other potential requestors. The WebSphere Application Server process runs the EWLM Control Center application, which provides the user interface into the domain manager. On independent 10 second intervals, each managed server sends data statistics 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 keeps a live version of the aggregated statistics in memory and one hour's worth of one-minute summary data hardened to disk. Both the live and hardened data is utilized to provide reporting data to the Control Center or other requestors.

Data at the managed server contains information on the specific server only, while at the domain manager, it is aggregated to provide you with an end-to-end view of the transaction.

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.

9.1.1 Sizing factors

The key resources you need to size for the domain manager are CPU, memory, and disk storage for the internal database. Following 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 depend on any given domain configuration and active policy. In our testing we selected a subset of these that span 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. For instance, if you have 80 transaction classes defined, but only 10 typically utilized, then resource utilization will be much less.
- Application/Server topology in term of number of Application Environments, number of hops, how transactions 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.

Note that the transaction rate is not a factor in the domain manager sizing.

Initial measurements were conducted to evaluate the performance and scalability of the EWLM domain manager. This chapter contains the detailed performance results of our 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.

9.2 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.

9.2.1 Environment

All supported platforms were chosen and configured as domain managers as illustrated in Table . Domain manager server configuration

Platform	Processor model	Processor speed	Physical memory	Network adapter
Windows Server 2003 Enterprise Edition	x255	Xeon 4-way @2GHz w/ht	3.8GB	Gbit
AIX 5.2 ML03 (32bit)	p630 7028-6C4	Power 64bit 1.4Ghz 4-way	7.0GB	Gbit
i5/OS	4way of an i825 6way	4way of a 6way rated at 6600 CPW	5.7GB	Gbit

The number of managed servers was respectively: 63, 126, 252, 504. The same set of measurements were conducted on all three different platforms.

Two different service policies were used during the measurements and they are summarized in Table 9-1. All the service classes are defined with Response Time goals.

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 9-1 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 OS instances (servers) with each server being responsible for a single function (for example Web serving, application serving, database) as illustrated in Figure 9-1.

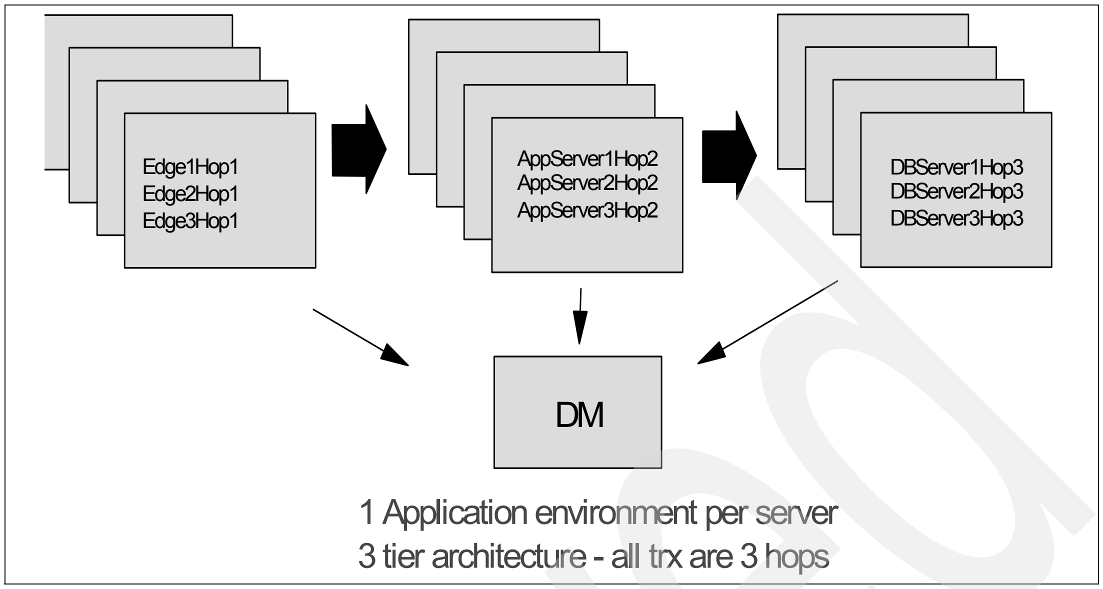


Figure 9-1 Application/server topology

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. We use here 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 9-2 shows some additional characteristics of the combined Policy/Application/Server topologies utilized in our testing.

Table 9-2 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 eighty 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 are not recommending an installation try to predict all of these characteristics. 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 and domain manager processing and memory resources.

9.2.2 Domain manager on Windows

Following are some charts that show the domain manager resources usage when running on Windows.

Figure 9-2 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.

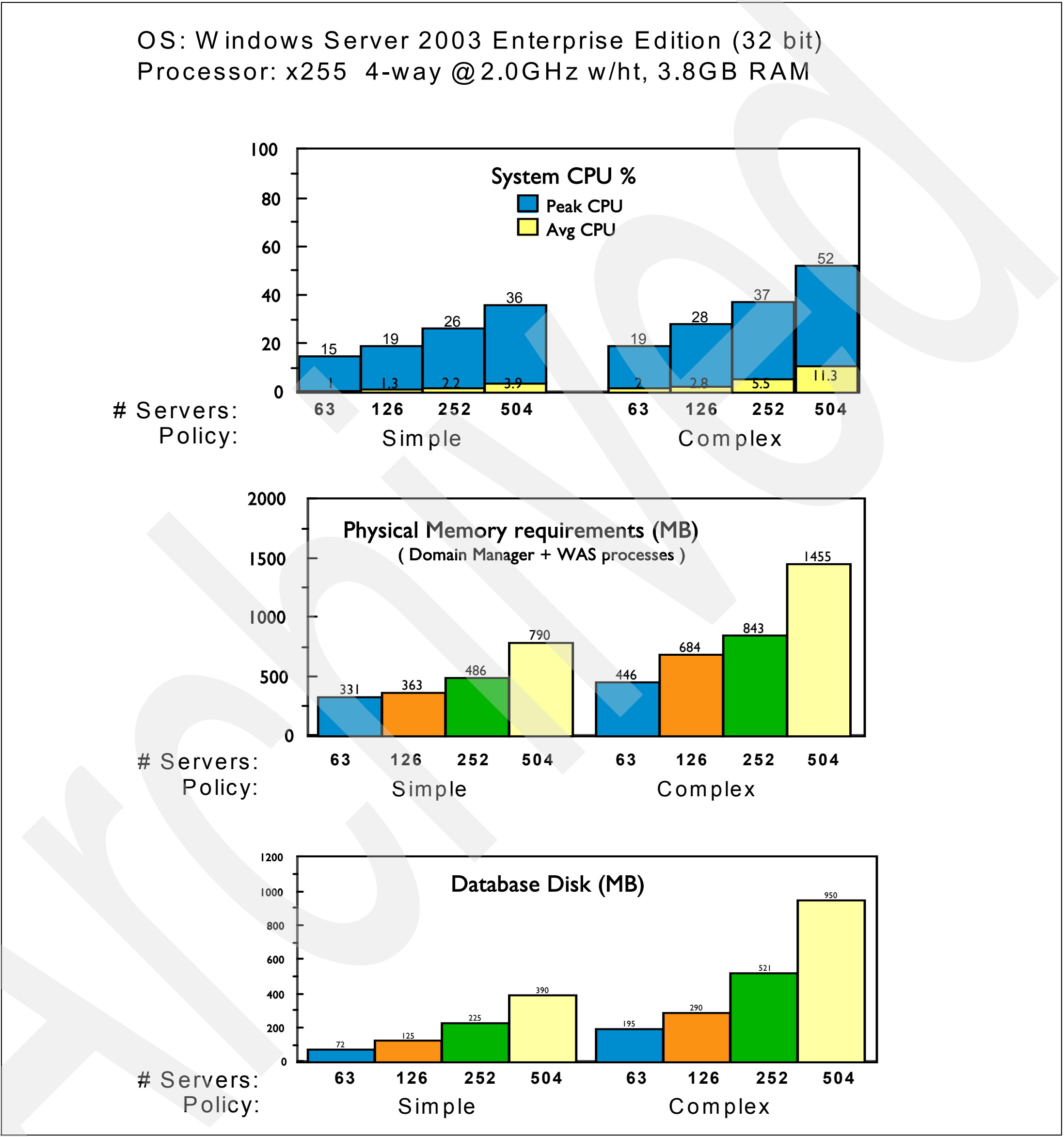


Figure 9-2 Resources usage on Windows domain manager

9.2.3 Windows test methodology

The following were our methodology considerations:

- The Windows performance monitor was used to capture most of the metrics in Table 9-3 and Table 9-4. 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 (1800MB), 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% free space). We left the -minHeap at the EWLM default of 128 (128MB).
- 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 which was comparable in size to the current active policy.
- A 10 minute measurement interval was utilized 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 9-3 and Table 9-4.

*Table 9-3 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 9-4 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% 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% 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 50-70MB greater then 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% 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.

9.2.4 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 512MB 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 2way 2Ghz with 2 GB RAM or comparable hardware. 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.
- 200 to 500 managed servers with a policy up to 80 transaction/process classes and 20 service classes should safely fit on an xSeries 4way 2Ghz w/2GB 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 out-of-memory condition for java heap. Beyond product install disk requirements, an additional 2GB 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 - 512MB, WebSphere Application Server - 256MB. 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 320MB.

9.2.5 Domain manager on AIX

Following are some charts that show the domain manager resources usage when running on AIX.

Figure 9-3 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.

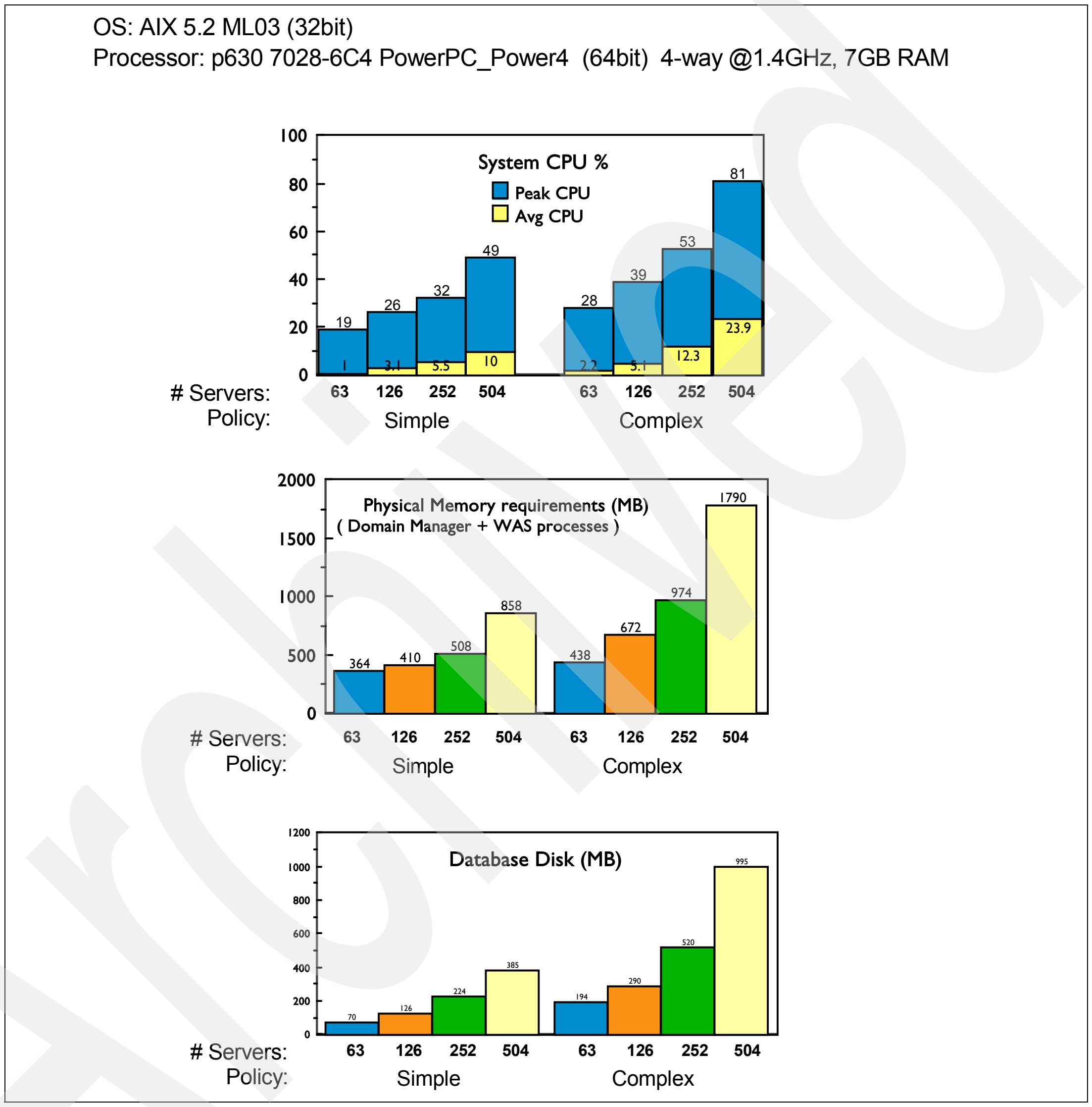

OS: AIX 5.2 ML03 (32bit)
Processor: p630 7028-6C4 PowerPC_Power4 (64bit) 4-way @1.4GHz, 7GB RAM

Figure 9-3 Resources usage on AIX domain manager

9.2.6 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 9-5 and Table 9-6. The domain manager Java process was started with "`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 (1800MB),

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% free space). We left the -minHeap at the EWLM default of 128 (128MB).

- In order to set the -maxHeap to 1800 on AIX (32bit), 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 2GB maximum java heap, though as we said, we set our maximum to 1800MB.

- 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.
- A 10 minute measurement interval was utilized to capture most of the nmon 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.

Detailed results are displayed in Table 9-5 and Table 9-6:

Table 9-5 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)	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 9-6 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)	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% 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 50-70MB greater then 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 1GB 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% 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.

9.2.7 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™ 1.4GHz (or comparable) with 512MB 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 1.4GHz (or comparable) with 2.5GB 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 2GB 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 - 512MB, WebSphere Application Server - 256MB. 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.

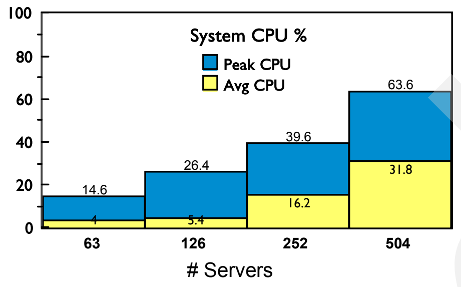
The 504 managed server/simple policy and the 252/504 managed server/complex policy test scenarios all required increasing the java maximum heap for the domain manager process. Additionally, the 504 managed server/complex policy test scenario required increasing the maximum heap for the WebSphere Application 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 320MB.

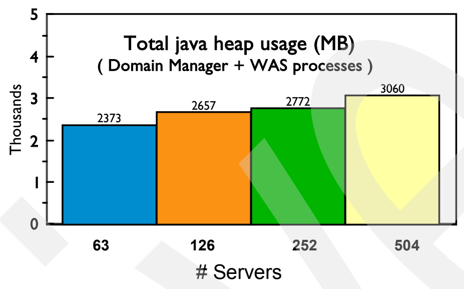
9.2.8 Domain manager on i5/OS

Following are some charts that show the domain manager resources usage when running on i5/OS.

Figure 9-4 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.

OS: i5/OS
Processor: 4-way partition of an iSeries Model 825 6-way (6600 CPW), 5.7GB RAM
Policy: Complex





Note: See 'i5/OS Test Methodology' & 'General Sizing recommendations on i5/OS'.
These Java heap statistics do NOT represent recommended java heap settings.

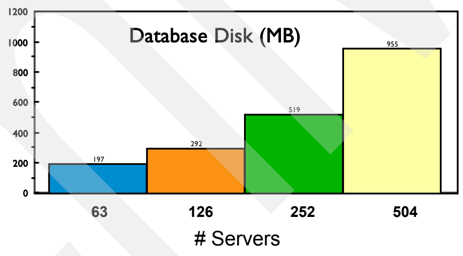


Figure 9-4   Resources usage on i5/OS domain manager

9.2.9 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 1200MB. In practice, domain managers managing fewer than 300 servers would most likely run with the initial heap size at 256MB 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 9-7.

Table 9-7 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% 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% 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% 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.

9.2.10 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 1.5GB 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

- 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 2GB 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 2.5GB 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 1GB 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.

9.3 Additional considerations

- Don't place 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 EWLM working directory on a higher performance disk. This is to improve access times of EWLM's internal database.

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 on ordering these publications, see "How to get IBM Redbooks" on page 237. Note that some of the documents referenced here may be available in softcopy only.

- *DB2 UDB V8 and WebSphere V5 Performance Tuning and Operation 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

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 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/`

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

IBM Enterprise Workload Manager

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

IBM Enterprise Workload Manager

How to configure operating systems and enable middleware for EWLM

Sample scenario for building a real-world domain policy

Integrate with security, load balancers, and firewalls

This IBM Redbook provides an introduction to the Enterprise Workload Manager (EWLM). In addition to describing the overall product concept and functionality, it presents a detailed discussion of the elements that are part of the 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 and reporting capabilities.

A sample scenario implemented in the ITSO environment is used to guide you through the process of classifying workload, and defining and deploying your own EWLM policy. This scenario is then used to demonstrate techniques for securing your implementation.

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.

SG24-6350-00 ISBN 0738490490