IBM Software Group

IBM WebSphere® Application Server V5.1.1

Enterprise Workload Management (EWLM)

e business on demand.

Goals

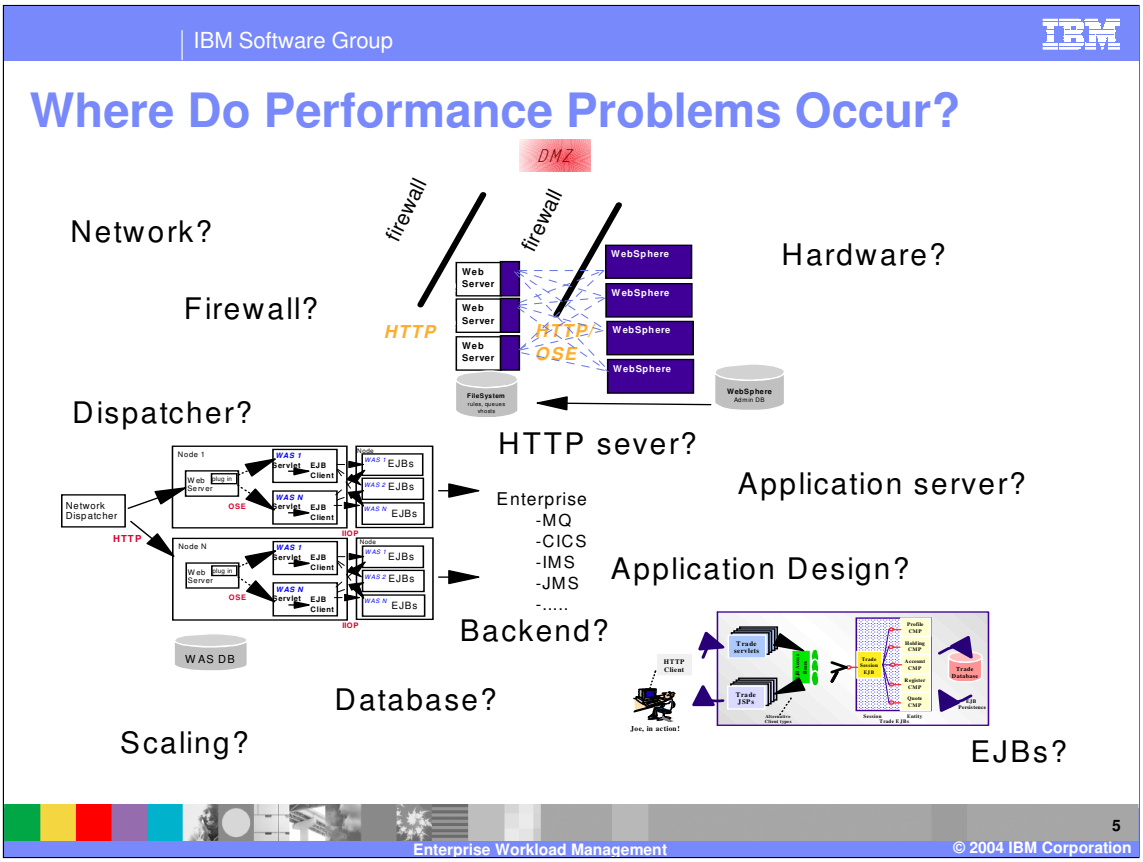
- Introduce Enterprise Workload Management (EWLM) concept
- Describe EWLM architecture and tooling

Agenda

- Overview
- Architecture
- Functionality

Section

Overview



The problem can be (and has been) almost anywhere within an application. The problem can be network and hardware related, backend system related, it can be actual product bugs, or quite often, application design issues.

With today's applications becoming increasingly complex it is getting harder and harder to track down these problems. Typical E-business applications now have one or several of the following attributes:

•Multiple logical and/or physical tiers

•Mixture of operating system platforms

•Mixture of hardware architectures

•Fuzzy application boundaries

Today's software is capable of producing massive quantities of numbers; CPU usage statistics, memory usage statistics, queuing statistics, transaction rate statistics, buffer pool statistics etc. The problem is that the large amount of statistics can actually be overwhelming and may hamper, rather than help, a person's ability to understand and analyze performance data.

Where Do Performance Problems Occur?

- Fundamental problems with looking at statistics only
- Data has no context
  - Example: If CPU utilization for a server is 100%, is this good or bad?
- Provide resource centric views – **not** how they relate to business goals
- Performance tuning based on statistics alone can sometimes lead to further performance degradation
- Focused on tuning resources on a server by server basis – results in over provisioning

Statistical data has no context associated with it. For example, if the CPU utilization of an individual server is 100%, is that good or bad?

•If the portion of the workload flowing through that server is performing as expected, then 100% utilization is good thing.

•If the workload flowing through that server is not performing as expected than 100% utilization may be a not good sign. But, one can not say so confidently without going through stacks of data.

The statistical data provides resource utilization indicators. There is no easy way to tie resource utilization to the business goal of the work.

If one assumes that work in not performing well because CPU is 100% busy, it is not correct. CPU could be 100% busy due to an I/O bottleneck or a physical memory shortage. Trying to solve the performance problem by adding more processing power and not resolving the I/O bottleneck or physical memory shortage problem, could further degrade the performance of the work.

There is no easy way to tie together performance statistics produced on separate machines for a specific business work request. The server based statistical data leads to performance tuning on each machine. The result of doing this would be the over provisioning of computing resources across all systems.

What Is EWLM?

- System monitoring solution that utilizes request metrics and ARM (separate product from WebSphere)
  - Tracks transaction times across application's infrastructure
  - Identifies SLA targets that are not being achieved
- Provides correlation between collected performance data and business goals
  - Relationships are held within **EWLM Domain Policies**
- Business goals are defined in the same terms as understood by the CIO
  - Closes the communication gap within the I/T infrastructure
- Goal definitions are server and application environment agnostic
  - Reuse goals across different environments

EWLM provides a solution to this problem by presenting information in a simple, logical way which allows analysts to rapidly and efficiently identify the likely origin of performance problems.

•EWLM is a separate product from WebSphere. The important thing to note here is that EWLM utilizes the Request Metrics functionality that WebSphere provides to track transactions as they move through the WebSphere environment.

EWLM makes it possible by providing a solid context in which statistical data can be matched to business goals. These relationships are held within EWLM domain policies.

The notion of context begins with the explicit definition of the performance expectations in the same terms that an analyst might explain to his or her CIO as to how well a portion of the I/T infrastructure has been performing.

•For example, a typical sale of company product over internet should complete on average within 2.5 seconds. The CIO might want to verify that such an expectation is being satisfied. The idea of an expectation, the 2.5 second goal, provides needed context to simplify every thing.

Different organizations servicing a specific business application can discuss the performance requirement in this common context.

EWLM domain policy supports various goal types that have been proven to be effective in most environments supported by z/OS workload manager.

The definition of a goal implies no physical relationship to the servers which deliver resources to the applications which process the work. Once a goal is established, the physical hardware, the server's operating systems and the application may be replaced, extended, upgraded or reconfigured while the external goal understood by the CIO remains in force.

Domain Policy

- Can be thought of as a contract which dictates how EWLM should treat “work” within a management domain
  - Describes how “work” should be classified, prioritized, managed, reported, etc.
- Officially, 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

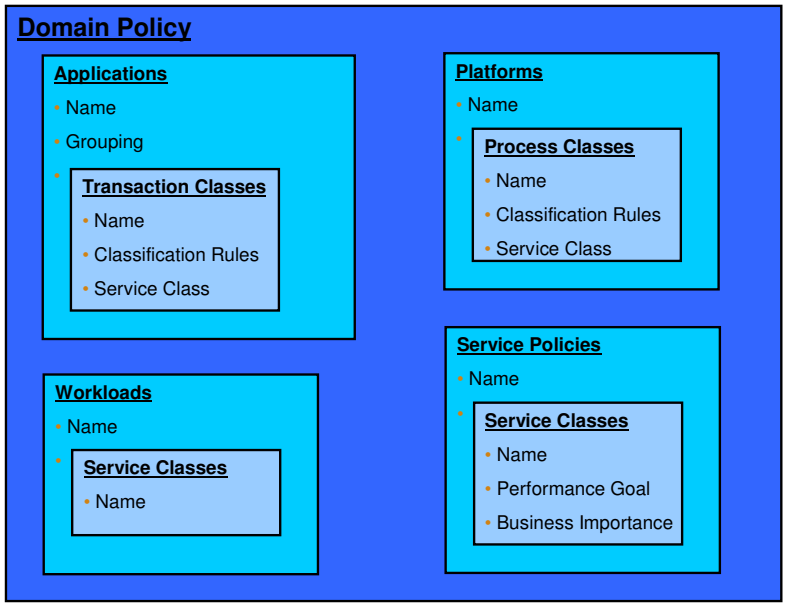
A domain policy enables the relationship of an application’s performance statistics (length of ShoppingCart EJB method execution, length of ShoppingCart servlet method execution, transaction per seconds, etc.) with its associated business goals (ex: all shopping cart orders must complete within 15 seconds. It also enables the definition of how important work is to the business.

•For example: In a brokerage firm there are two kinds of work – employee and customer work requests that execute on the same infrastructure. Using business importance you would be able to specify that all customer work requests be completed ahead of employee work requests when business goals are not being met.

•An important thing to note about business importance is that when all goals are being met, business importance is irrelevant. It does not indicate how to prioritize work in normal circumstances.

A management domain consists of a Control Center and Domain Manager and any combination of Managed Servers. A Management Domain consists of the Managed Servers defined in a Domain Policy.

Domain Policy Components



The following is a brief description of each type of component within a domain policy. For more information on each component see the Appendix.

- To understand the different components better we are going to use the example of a brokerage application. The application's infrastructure consists of an HTTP server (running on Linux), WebSphere (running on AIX), DB2 (running on separate AIX), and non-ARM-instrumented transaction processing backend (running on Windows). The application has four basic transactions: query stock prices, sell stock, buy stock, and change account information.

Service classes allow you to assign specific goals and importance to transactions within your application.

Classification rules allow you to define filters that classify like transactions into groups.

- Service classes and classification rules are the basic building blocks for the other components.

Transaction classes group related transactions with a service class.

- An example of this would be to group all incoming requests that buy or sell stock into one transaction class. The transaction class could then could then be paired with a service class that defines the appropriate business goals (transactions time under 5 seconds) and importance for the transactions (high importance).

Process classes group related system processes with a service class.

- An example of this would be to create a process class that contained the transaction processing backend. The transaction class could then be paired with a service class that defines the appropriate business goals (work should run at the fastest possible velocity) and importance for the transactions (high importance).

Applications combine on or more transaction classes into a common group. This component is used to monitor transactions running on an ARM-instrumented application.

- An example of this would be separate application definitions created for each of the ARM-instrumented processes in our brokerage applications. This would result in entries for the IBM HTTP server, WebSphere application server, and DB2.

Platforms combine on or more process classes into a common group. This component is defined for each platform that transactions will be monitored on. Also processes that are not ARM-instrumented will need to be included in the platform definition.

- An example of this would be separate platform definitions for each of the platforms in our brokerage applications infrastructure. This would result in entries for a Linux, two AIX, and a Windows system.

Service policies describe business performance objectives and importance of work within a management domain. Multiple service policies can be defined – each one can prioritize the importance of each service class differently and specify different goals.

- An example of this is if we defined two separate service policies for our brokerage application – one that outlines how work should be handled during weekdays and another that outlines how work should be handled on the weekend. On the weekend for example we may want to raise the priorities for stock quote lookups and account changes and on weekdays we may want the stock buy and sell requests to have a higher priority.

Workloads are an optional definition that combines service classes into common groupings for reporting purposes.

- An example of this would be to use it to group all stock activities (buy, sell and query) in our example application.

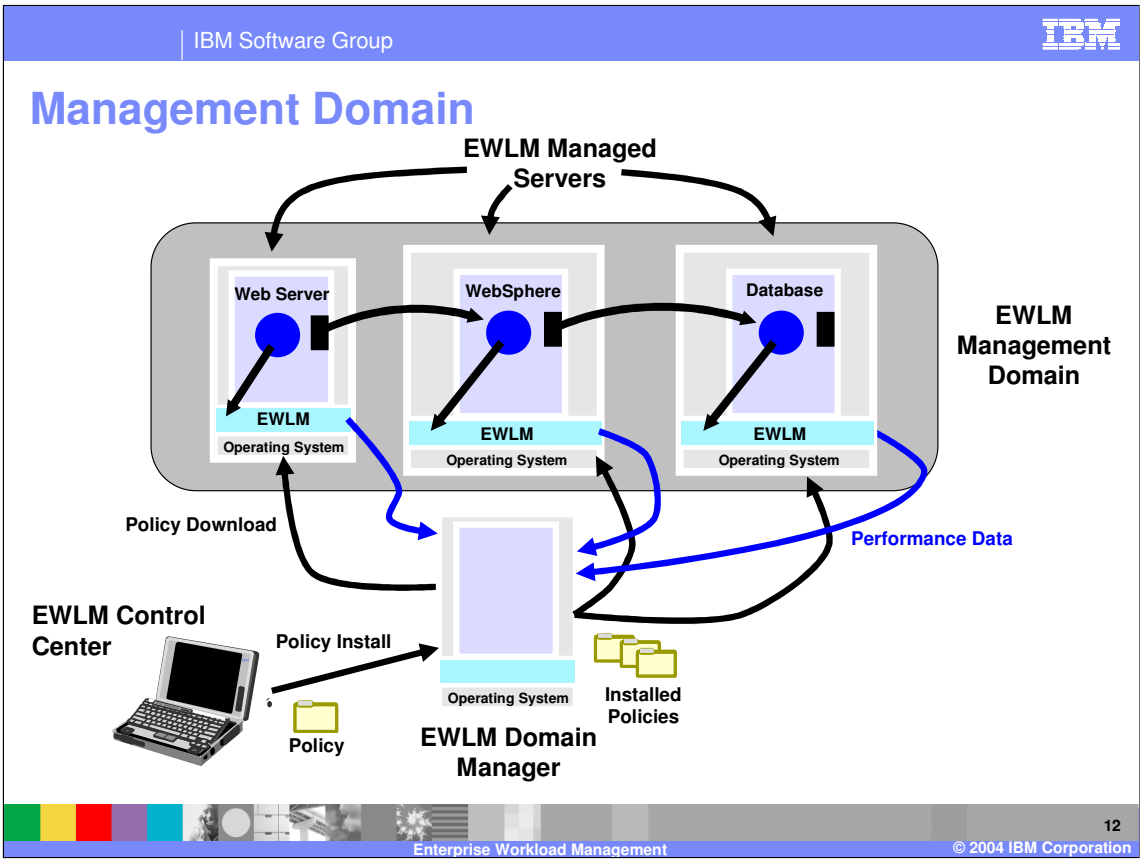
Section

Architecture

Configuring WebSphere To Utilize EWLM

- Install EWLM agent(s)
- Troubleshooting > PMI Request Metrics
  - Enable request metrics
  - Enable ARM
  - Define filters (optional) and trace level
- Servers > Application Servers > <%SERVER_NAME%> > Process Definition > Java Virtual Machine > Custom Properties
  - Disable logging
    - com.ibm.websphere.pmi.reqmetrics.loggingEnabled=false
  - Specify ARM implementation
    - com.ibm.websphere.pmi.reqmetrics.ARMIMPL=arm4
  - Specify ARM factory
    - ArmTransactionFactory=org.opengroup.arm40.sdk.ArmTransactionFactoryImpl
- Restart application server(s)

The steps above are the comprehensive steps needed to enable WebSphere 5.1.x to work EWLM.



The diagram above represents a typical Enterprise Workload Management (EWLM) management domain.

A management domain consists of the following components:

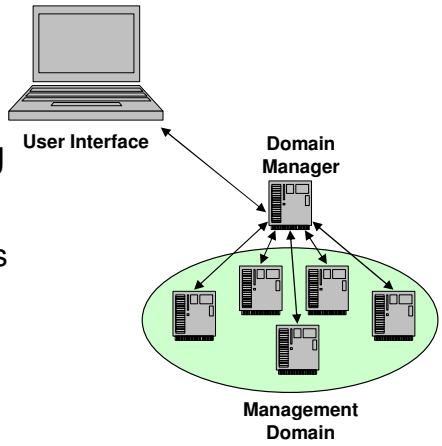
•Control Center – Web front end for the Domain Manager. The Control Center allows the user to define domain policies and their associated components. It also allows the user to view collected performance statistics and analysis data from the management domain.

•Domain Manager – Java based application that is responsible for the global coordination of business policies to managed servers in the management domain. Also retrieves request metric data via EWLM agents running within managed applications processes.

•Managed Servers – Systems participating in a management domain. Managed servers are the systems on which the monitored transaction is being executed on.

Control Center

- Web-enabled UI
  - Role based, authentication, encryption
  - Business policy creation & maintenance
- High level business-driven reporting
  - Display topology from business view
  - Drill-down into application & server views
- Response time and resource data
  - Server resource consumption & delays
  - Transaction and process-centric work



The Control Center is the web front end for the Domain Manager. Through it you are able to define domain policies, edit existing domain policies, monitor already defined management domains, view and analyze statistics collected from management domains.

The domain manager collects a wealth of statistics that can be viewed using the Control Center. Examples of these are:

•Server topology – Topology map of the different systems involved in the execution of a transaction.

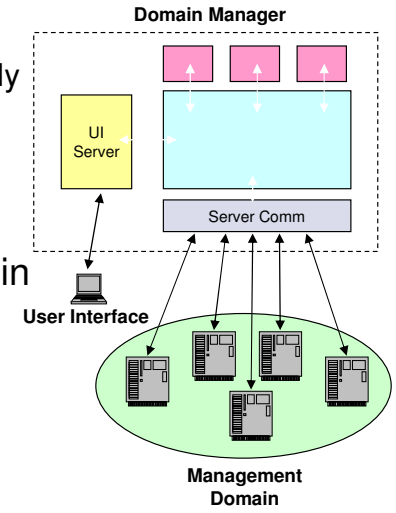
•Application topology – Topology map of the different applications involved in the execution of a transaction.

•Service class statistics – Statistics based on the defined service classes.

•Transaction rate and response time – Graphs number of transactions completed per second and the average time it took to complete transactions.

Domain Manager

- Java application
  - Accessed using Control Center
  - Secure messaging, firewall and DMZ friendly
- Global coordination of business policy actions
- Global view of the management domain
  - Data assimilation, learning of business topology
- Public APIs
  - Systems management products
  - Workload Balancing APIs

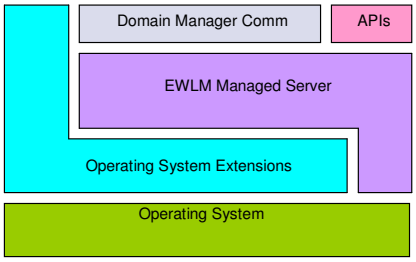


The domain manager is responsible for coordinating the activities of the management domain. It coordinates installed policies and retrieves/aggregates performance statistics across a management domain.

The domain manager exposes several public APIs that allow other system management software to query the domain manager for information. For example Tivoli Monitoring for Transaction Performance product could use information returned from the Domain Manager to build topology graphs, trigger alerts, display transaction response time graphs, etc. Additionally metrics collected by EWLM are very useful to workload balancing applications that need to look at all applications/systems involved when making work routing decisions.

Managed Servers

- Common Java implementation
- Domain Manager coordination
- Data aggregation
- Public APIs for local control and data export
- Platform-specific operating system extensions
  - ARM implementation
  - System and process resource use collections



Managed servers are the physical systems that transactions within a management domain execute on. The managed server aggregates some of the collected data before it is sent to the Domain Manager.

The Common Java implementation allows the managed server to communicate with the Domain Manager. It is used to collect operating system and transactional data from the managed system.

The ARM implementation and functionality to retrieve platform specific system and resource information is still dependent on platform-specific code.

Platform Support

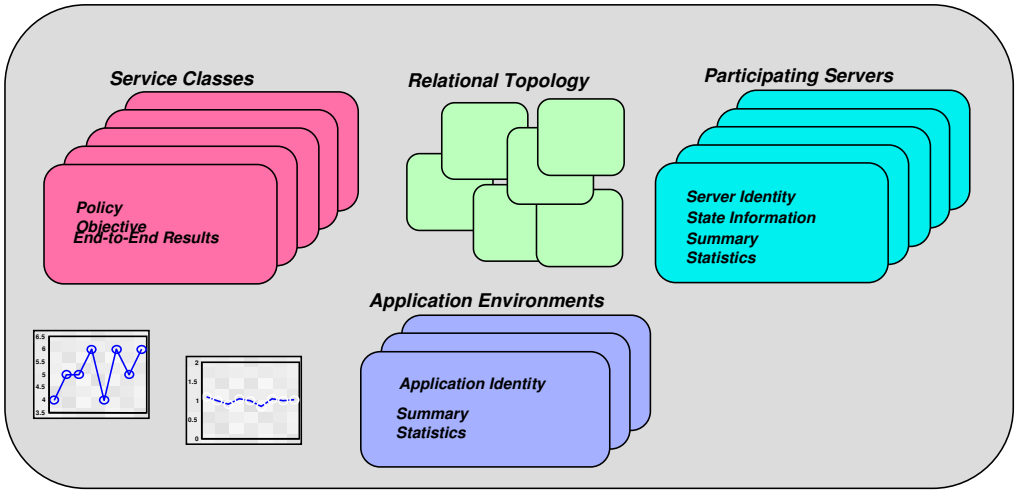
- Domain Managers
  - Red Hat / SuSe Linux (Platform set supported by WebSphere 5.1)
  - AIX 5.2F
  - OS/400 5.3
  - Windows 2000 / 2003
- Managed Servers
  - AIX 5.2F
  - OS/400 5.3
  - Windows 2000 / 2003
  - Solaris 8 / 9
- Load Balancing
  - AIX 5.2F
  - Windows 2000 / 2003
  - Solaris 8 / 9

Section

Functionality

Reporting And Analysis

- Domain manager provides the following statistics
  - Viewed through Control Center



The domain manager provides the following statistics.

•Server topology – Topology map of the different systems involved in the execution of a transaction.

•Individual server details – What type of server, current state information, statistics, etc.

•Application topology – Topology map of the different applications involved in the execution of a transaction.

•Individual application details – Which application, statistics, etc.

•Service class statistics – Statistics based on the defined service classes. Gives statistics on goal and whether or not the goal is currently be met.

•Transaction rate and response time – Graphs number of transactions completed per second and the average time it took to complete transactions.

•Exceptions – Statistics on which service classes are failing.

The statistics above can be used to answer the following questions about the application's current performance.

•How are we doing?

•Compared to business objectives

•What applications support each class of service?

•Graphical map of workflow topology

•Relationships between middleware instances

•Comparison of various instances

•What happens at each transaction "hop"?

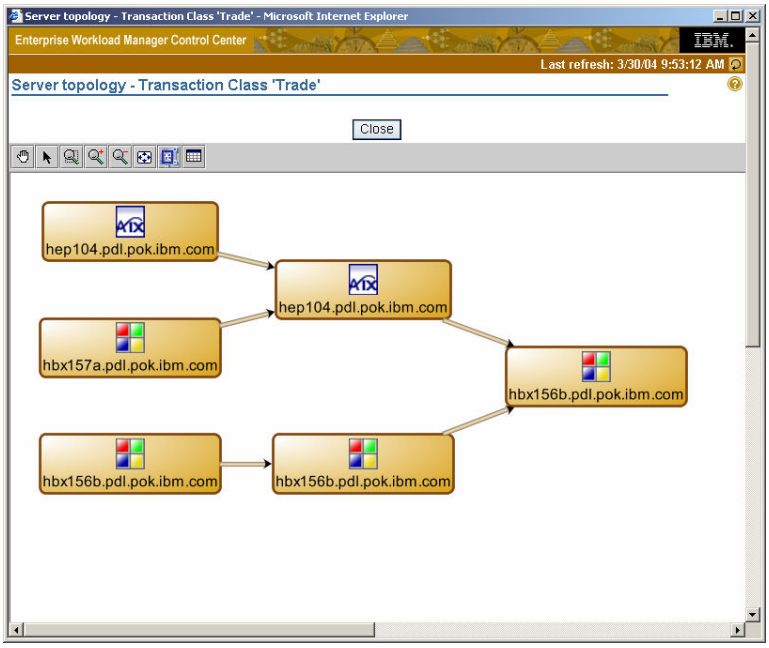
•Response times, distributions, etc.

•Resource consumption and delays

•What servers support each middleware instance?

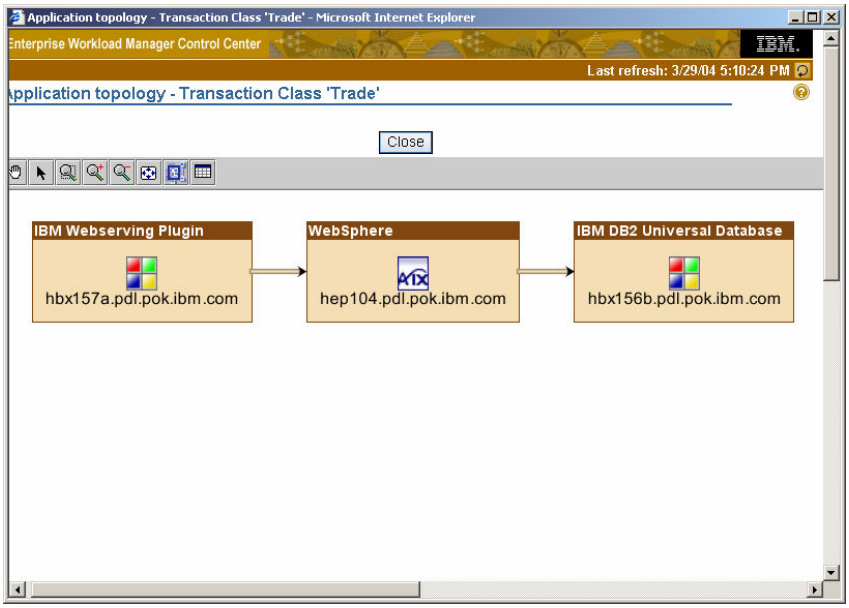
•General server-level statistics

Server Topology



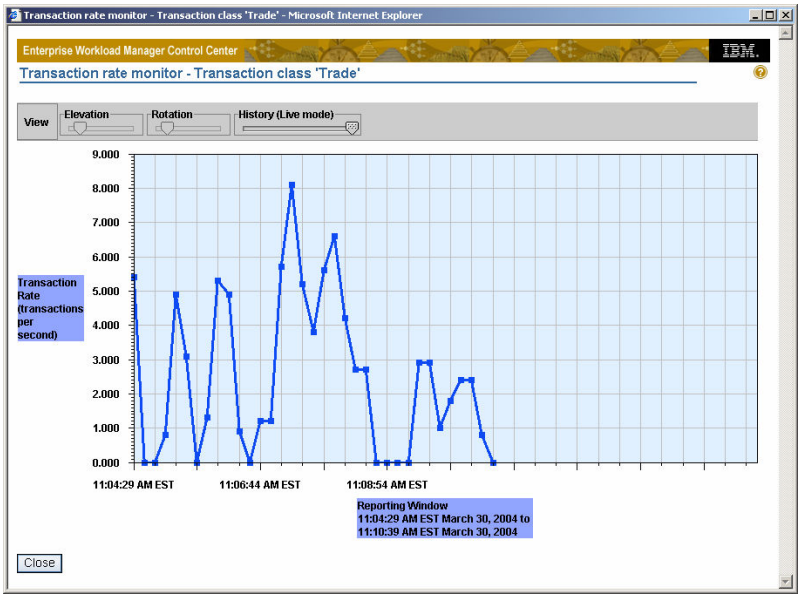
The screenshot above shows a server topology diagram. Each rectangle contains the name and type of server that was called.

Application Topology



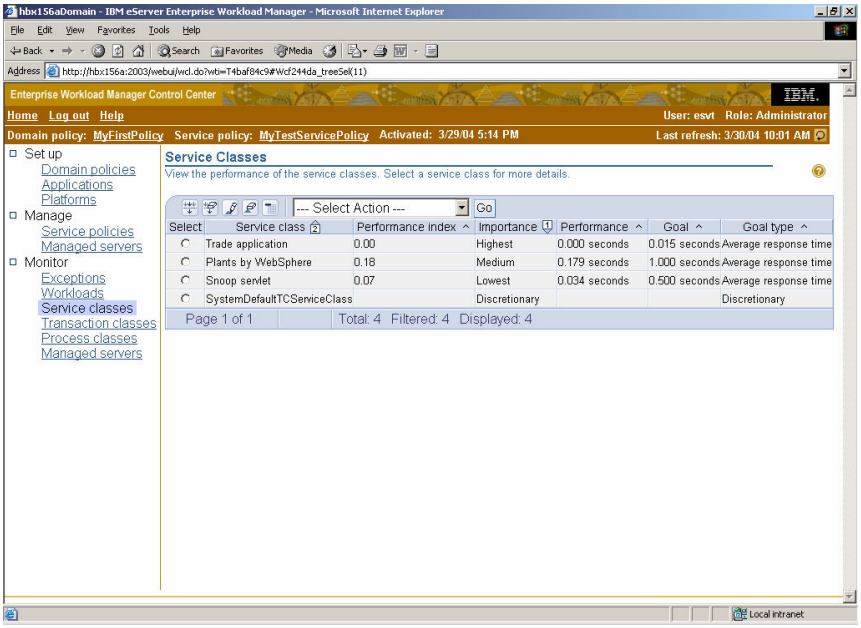
The screenshot above shows an application topology diagram. Each rectangle contains information on the application's name, server name, and server type.

Transaction Rate Monitor



The screenshot above shows a transaction rate diagram. The diagram shows the number of transactions called per second.

Service Class Report



The screenshot above shows a service class report. The report shows the service class settings (service class name, business importance, performance goal, etc.) and the actual performance values that are being achieved. The report also shows the service class's performance index. The performance index indicates how well the application is doing towards its goal.

- A performance index of 1.0 means that the application is meeting its goal.
- An index < 1.0 means that the application is exceeding it goal.
- An index > 1.0 means that the application is not exceeding its goal.

Additional Functionality

- Global data collection enables other opportunities
- Externalization of collected statistics to system management software (Ex: TMTP)
  - Use to display topology graphs, alerts, etc.
- Additionally used as input to
  - Provide detailed data for enhanced capacity planning tools
  - Influence workload routing & balancing mechanisms
  - Local server resource management algorithms
  - Dynamic management of logically partitioned hardware

Due to the APIs that the domain manager externalizes, data can be externalized from EWLM to other applications. System monitoring applications can use EWLM information to create server and application topology graphs, produce transaction response time graphs, create alerts, etc.

Because the information is collected globally across an entire application, EWLM information is very useful to workload balancing applications. The information allows the workload balancing application to route work around poorly performing or overloaded servers. It can also use the information to route work away from servers that don't have the application installed or on which the application is failing.

Section

Summary

Summary

- EWLM relates collected performance statistics to performance expectations and business goals
- Domain policies describe performance expectations in terms understood by everyone
- EWLM Domain Manager responsible for managing implemented policies and collecting performance data
- EWLM Control Center use to create policies and analyze collected data
- Public APIs allow EWLM to externalize collected data to other tooling

Section

Appendix

Section

Domain Policy Components

Service Class

- Service classes are a central concept to EWLM
- A group of work that has similar performance goals or business requirements
- Supported goals include: Percentile Response Time, Average Response Time, Velocity, and Discretionary
- You also specify how important it is to your business that the goal be met
- Importance plays a factor 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

Service Class: Goal Types

- Percentile Response Time Goal
  - Specifies what percentile of work requests should complete in a specified amount of time
- Average Response Time Goal
  - Specifies an average amount of time in which work requests should complete

Service Class: Goal Types (cont.)

- Velocity Goal
  - Specifies how fast work should run (relative to other work) when ready, without being delayed for EWLM-managed resources
  - Velocity may be specified as Fastest, Fast, Moderate, Slow, Slowest
    - In general one can expect that fastest would experience the least delay and slowest, the most delay
  - Appropriate for work for which a response time goal is not appropriate
    - For example: A non-ARM instrumented transactional server or a long running process

Service Class: Goal Types (cont.)

- Discretionary Goal
  - This type of goal receives what's left of the EWLM-managed resources after all the other goals have been satisfied
  - Does not have an associated Importance
  - Work assigned this service class goal is run as system resources are available and therefore is implicitly at an "Importance" level even lower than that of "Lowest"
  - Appropriate for workloads whose completion times are unimportant

Classification Rule

- Provides a way to identify the transaction class or process class to which work belongs
- Consists of a filter type, filter value, and an operation to apply to the filter type-value pair to determine if there is a match
- Filter types may be generic (as those provided by EWLM) or specific to an application or platform
- Filter values are specific to the work
- Supported filter operations are “Equal” or “Not Equal”

Transaction Class

- A named set of classification rules for mapping a group of transactions to a service class
- Use a transaction class to group related transactions
  - Those that have the same goal and are to be reported together
- This collection of transactions into a transaction class is made possible through the use of Rules
- Complex rule sets can be built, consisting of multiple nested rules, in order to accurately classify transactions
- 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

Process Class

- Similar to a transaction class
- Defines a group of system processes intended for reporting and management according to a service class
- A process class monitors work requests that a system initiates whereas a transaction class monitors work requests that users initiate from an application
- Note: Non-ARM instrumented transactional and application servers have to be monitored as part of a process class

Application

- An application must be ARM-instrumented in order for it to be monitored by EWLM as an application
- Consists of one or more transaction classes
- EWLM will attempt to match an application's work to a transaction class starting with position 1 and moving downward
- A transaction is classified into the transaction class where the first match is encountered
- Can define an application group which EWLM can manage as one entity

Platform

- The platform names of all EWLM-monitored processes must be added to the domain policy
- Consists of a list of process classes

Service Policy

- Describes the business performance objectives and the business importance of work within a management domain
- Consists of a list of service classes
- Each service policy can prioritize the importance of each service class differently and specify different goals

Workload

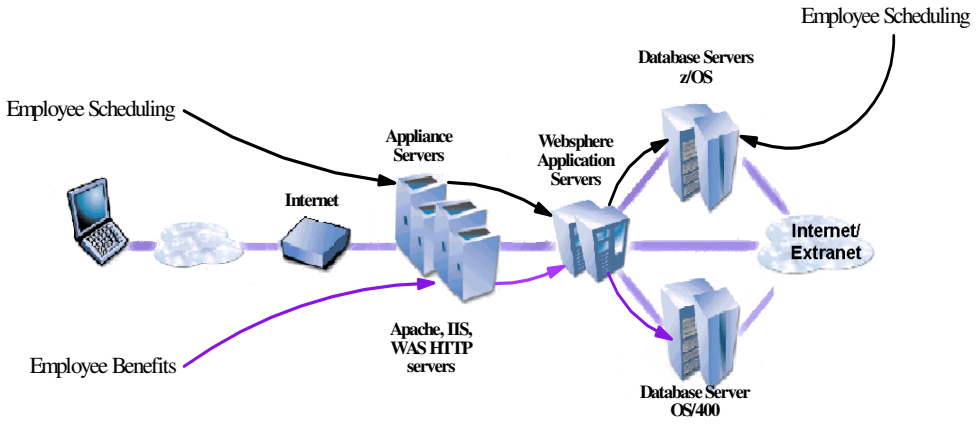
- An optional definition for a set of service classes that you want to group together for reporting purposes.

Section

Creating Domain Policies

Challenges

- Requires that the administrator has an understanding of every type of inbound work into the business environment



•The much needed simplification of performance statistics data requires an administrative action and that is informing EWLM of the performance goals for work flowing through various business environments.

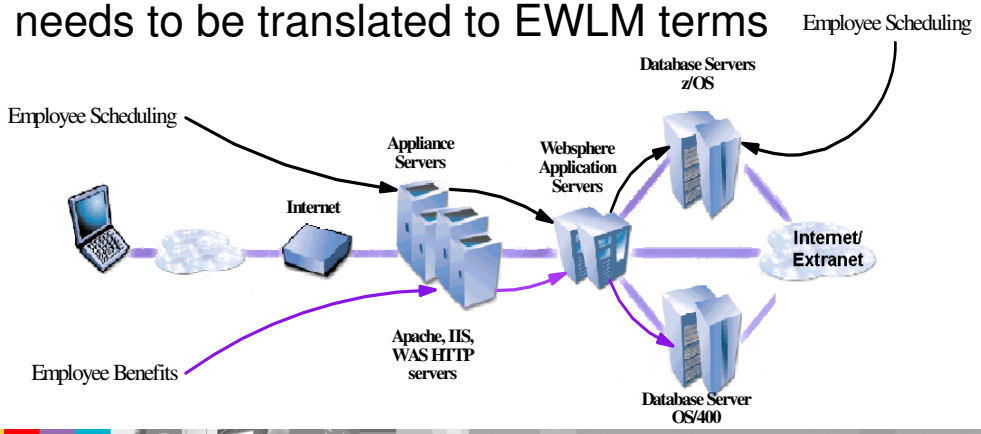
•The administrator must have an understanding of the complete set of business applications for a specific business environment.

•The work requests flowing through various applications must be grouped together for performance management and reporting.

- •Each group is known as a service class, having a name and an associated performance goal.
- •The name, such as Stock Trades, would be used for reporting purposes, while the goal, such as 1 second average response time, provides the definition of the difference between good performance and bad performance.

Challenges

- EWLM Management domain needs to be defined based on the business applications to be managed
- Service Level Agreement for each type of work needs to be translated to EWLM terms



•Customer's I/T infrastructure may include many diverse business environments. For each such business environment, EWLM management domain needs to be defined. For example, a customer might want to separate out a banking division from a brokerage division. An EWLM domain policy is only applicable to the domain it is serving.

•The work requests for a specific business environment is expected to be served by business applications running on the servers in the same EWLM domain. If work request crosses the EWLM domain, EWLM won't be able provide end-to-end statistics for the work.

•When work requests are served by business applications that spans across different business environments, it may be necessary to group different business environment in a single EWLM domain.

•Most installations do have a definition of what "good performance" is. It may be in SLA terms that is agreed upon by various groups in business and not necessarily in the same terms that EWLM expects.

•Each such internal goal needs to be translated to specific goal types supported by EWLM. EWLM supports the following goal types:

1.An average response time, such as 2 seconds.

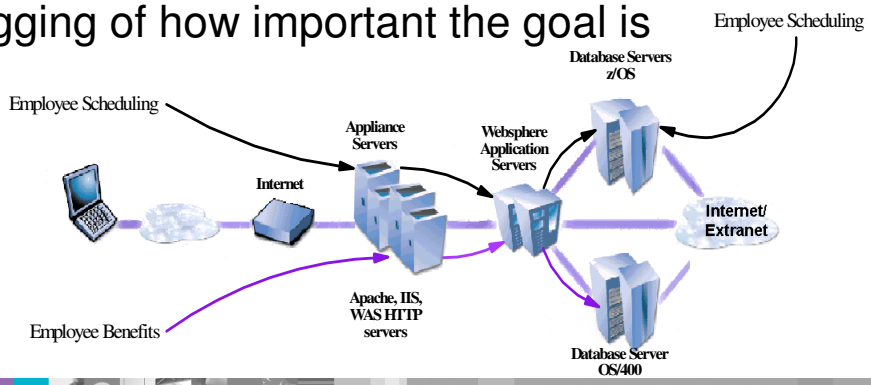
2.A percentile response time, such as 90% complete in 1.5 seconds.

3.A velocity goal that indicates how fast the work should move forward over time.

4.A discretionary goal, indicating that it doesn't really matter how fast the work is processed nor when it completes.

Challenges

- Not all goals are equal
  - How important is it to the business that the stated goal be achieved?
- Business Importance – A simple but powerful tagging of how important the goal is



•In a complex business environment that processes diverse workloads across a large server farm, the mere expression of classes of service and performance is not enough. Not all goals are equal.

•Besides performance goals, EWLM service class definitions include the notion of business importance levels which indicate:

  •How important is it to the business that the stated goal be achieved?

•It is likely that multiple classes of service can have an equal goal, however, one service class may be associated with internal users while another service class may support the application that interacts directly with customers and clients. So, when things go sour, which of these would one wish to worry about first?

•An important aspect of business importance is that when all goals are being met, business importance is totally irrelevant. It does not indicate how to prioritize the work in normal circumstances.

Creating Domain Policies

- Prepare a list of all the instrumented applications to be managed in the EWLM management domain
- Prepare a list of all the transactions to be managed for each business application
  - Assign business goal (based on SLA) to each type of transactions
  - Assign business importance to each type of transactions
- Prepare a list of OS platforms to be managed
- Prepare a list of various processes on each of the managed platform that will be processing the transactional or non-transactional work
  - Assign business goal
  - Assign business importance
- Group the transactions and processes based on the business goals and importance

•The complexity and importance of this administrative task of defining EWLM domain policy can not be overemphasized.

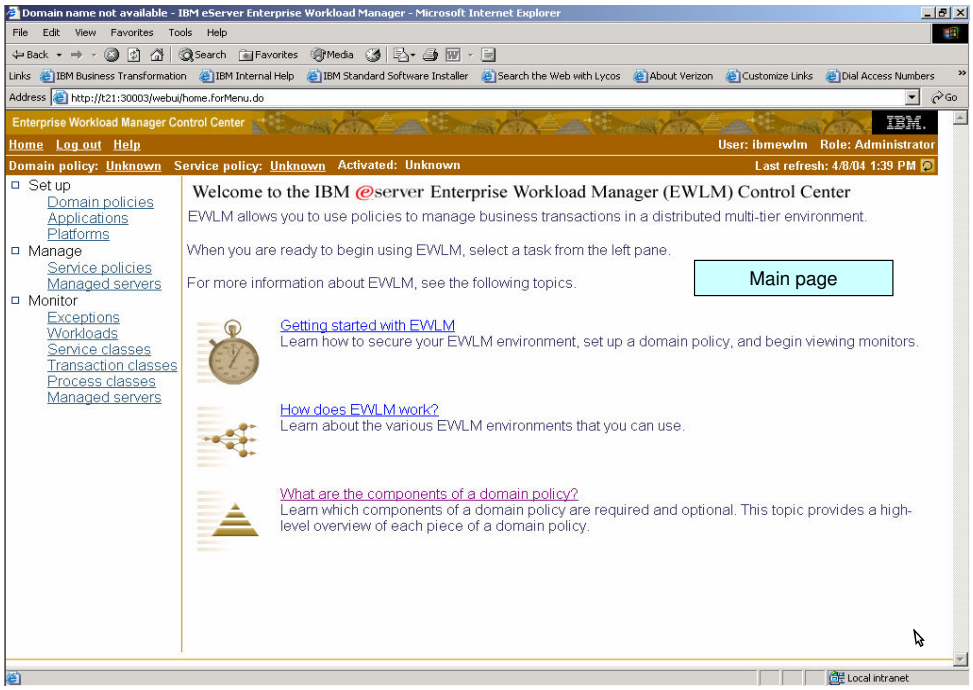
•This slide provides some recommendations to organize the information that can be used when you proceed with defining EWLM domain policy.

Section

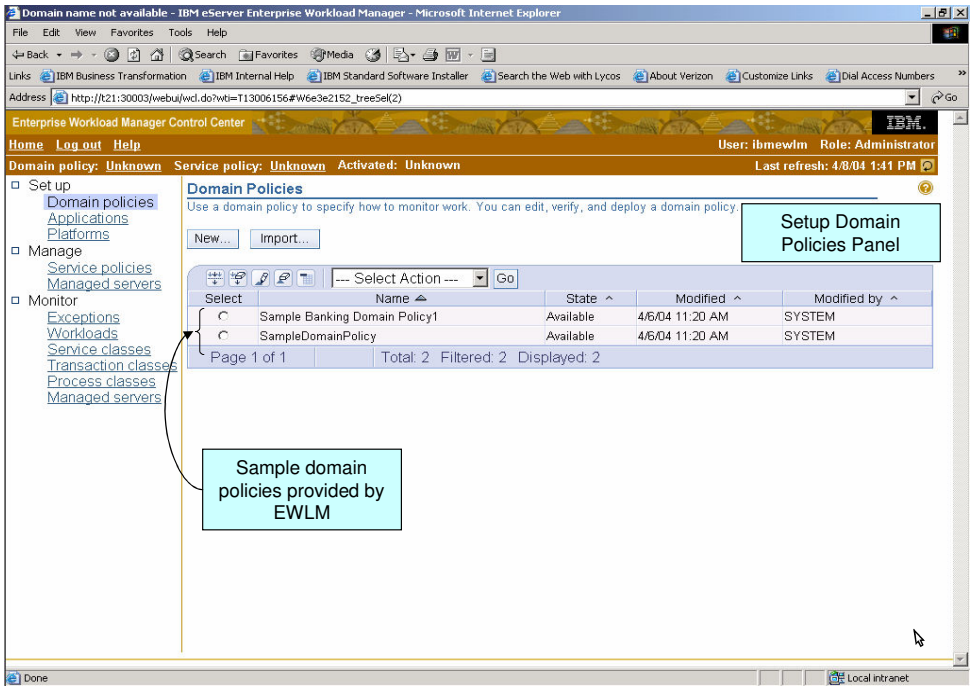
Control Center: Domain Policy Wizards

Domain Policy Wizard Overview

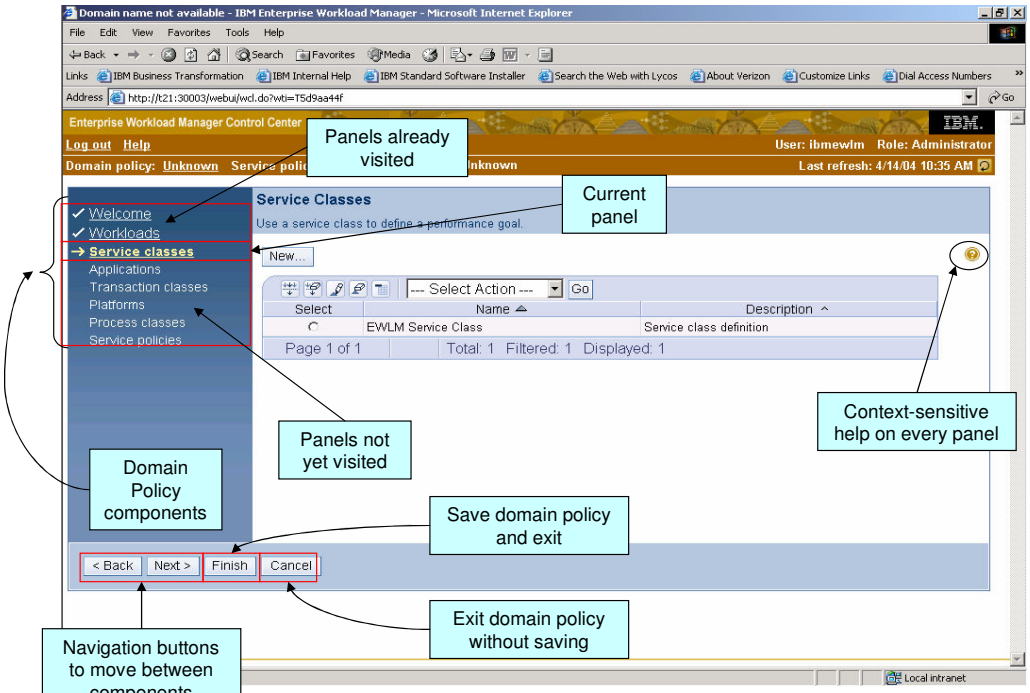
- Simple interface for creating domain policies
- Uses navigation buttons to move between individual domain policy components



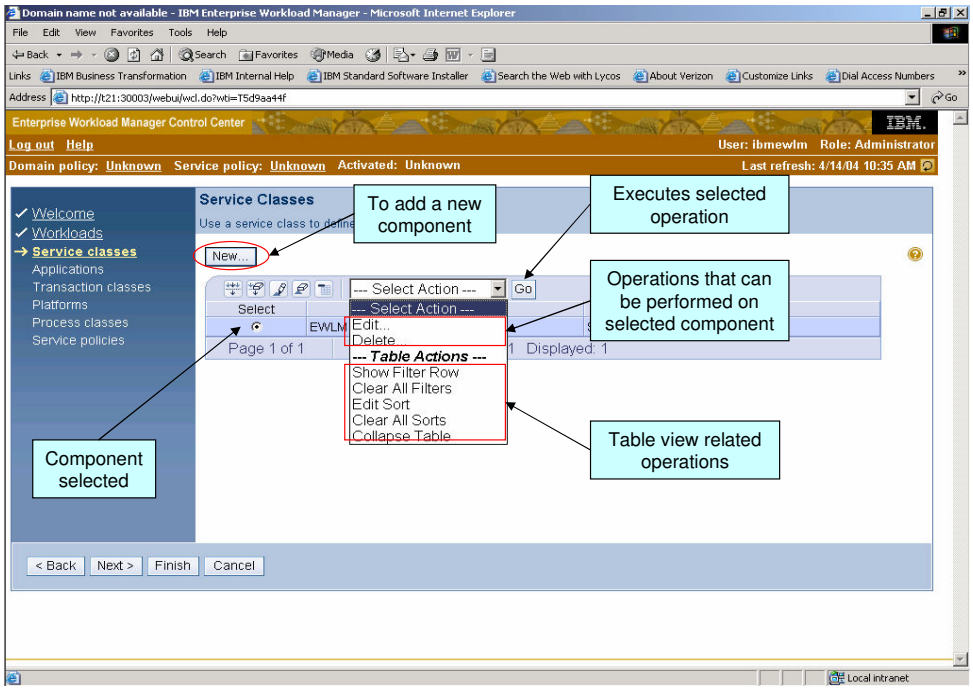
Main page after logging in. To get into the wizard we first click on Domain Policies from the left pane.



Then we click on New. This takes us into the wizard.



Note: This is not the first wizard panel but a sample panel within the wizard.



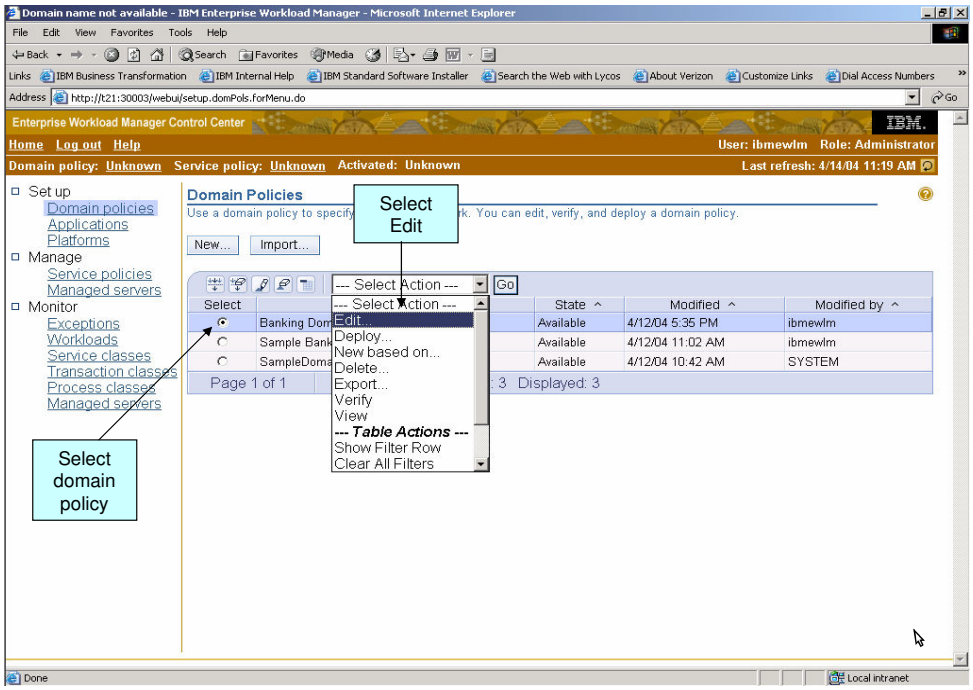
Sample panel continued.

Instead of New some panels have an Add button.

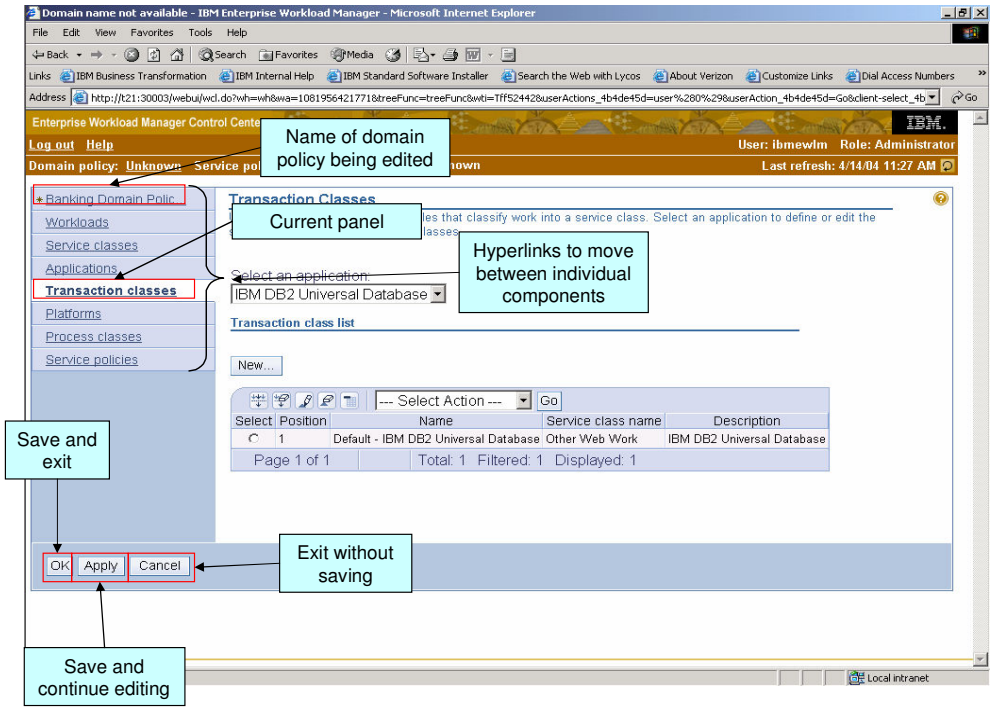
The operations that can be performed on a component varies from component to component.

Domain Policy Editor Overview

- Simple interface for editing domain policies
- Similar to the Domain Policy Wizard
- Uses hyperlinks instead of navigation buttons to move between individual domain policy components



From the main domain policies panel, select a domain policy, select Edit, then click Go.



Everything else between the wizard and the editor are the same.

Trademarks and Disclaimers

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

IBM	iSeries	OS/400	Informix	WebSphere
IBM(logo)	pSeries	AIX	Cloudscape	MQSeries
e(logo)business	xSeries	CICS	DB2 Universal Database	DB2
Tivoli	zSeries	OS/390	IMS	Lotus

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both. Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare 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 registered trademark of Linus Torvalds. Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program that does not infringe IBM's intellectual property rights may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

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

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.