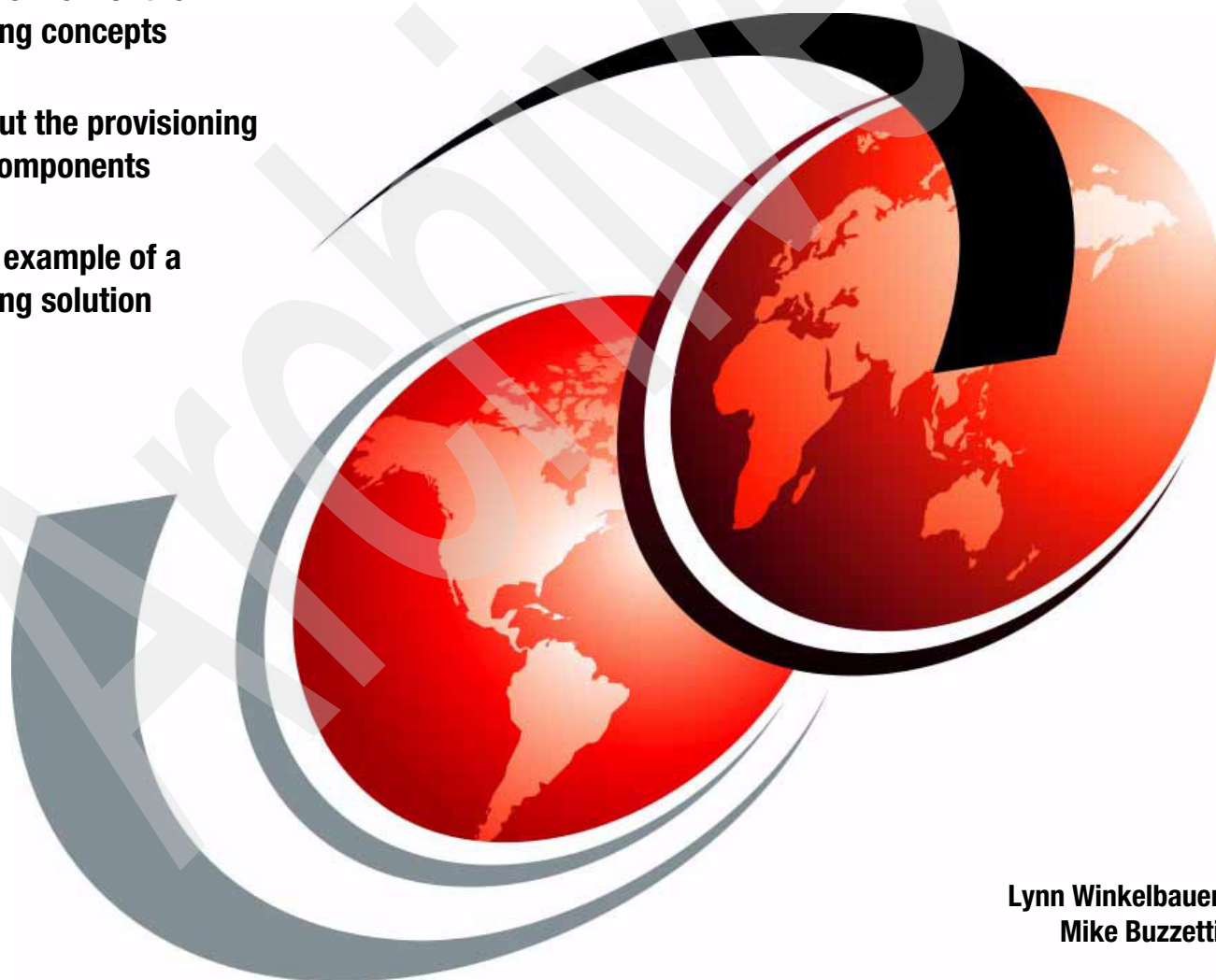


# Automated Provisioning using Tivoli Intelligent Orchestrator and Enterprise Workload Manager

Gain an overview of the  
provisioning concepts

Learn about the provisioning  
solution components

Follow an example of a  
provisioning solution



Lynn Winkelbauer  
Mike Buzzetti





International Technical Support Organization

**Automated Provisioning using Tivoli Intelligent  
Orchestrator and Enterprise Workload Manager**

July 2007

Archived

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

## **Second Edition (July 2007)**

This edition applies to Version 2 Release 1 of IBM Virtualization Engine Enterprise Workload Manager, product number 5733-EWM.

© Copyright International Business Machines Corporation 2005, 2007. All rights reserved.

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

# Contents

<b>Notices</b> .....	v
Trademarks .....	vi
<b>Preface</b> .....	vii
The team that wrote this Redpaper .....	vii
Become a published author .....	viii
Comments welcome .....	viii
<b>Chapter 1. Product overview</b> .....	1
1.1 Enterprise Workload Manager .....	2
1.1.1 Enterprise Workload Manager Control Center .....	2
1.1.2 Domain manager .....	2
1.1.3 Managed server .....	3
1.2 Tivoli Intelligent Orchestrator .....	3
1.3 Tivoli Provisioning Manager .....	4
<b>Chapter 2. Components of the Enterprise Workload Manager provisioning solution</b> ..	5
2.1 Component overview .....	6
2.2 Provisioning manager .....	7
2.2.1 The resource optimizer (Tivoli Intelligent Orchestrator) .....	7
2.2.2 Deployment engine (Tivoli Provisioning Manager) .....	7
2.2.3 Database (data center model) .....	7
2.2.4 Configuration summary .....	9
2.3 Objective analyzer .....	10
2.4 Performance manager (Enterprise Workload Manager) .....	12
<b>Chapter 3. Environment for testing the provisioning solution</b> .....	15
3.1 Test environment .....	16
3.1.1 Application environment .....	16
3.1.2 Management server .....	16
3.2 Enterprise Workload Manager .....	17
3.2.1 Enterprise Workload Manager domain manager .....	17
3.2.2 Enterprise Workload Manager managed server .....	18
3.2.3 Software .....	19
3.3 Tivoli Intelligent Orchestrator .....	19
3.3.1 Our network infrastructure setup .....	19
3.3.2 Our customer relationship model .....	20
3.3.3 Configuring SSH .....	21
3.3.4 Summary of our environment .....	21
3.4 Using objective analyzer to automate provisioning .....	25
3.4.1 Starting and stopping Tivoli Intelligent Orchestrator .....	25
3.4.2 Installing the objective analyzer driver .....	26
3.4.3 Avoiding JMX conflicts .....	27
3.4.4 Configuring the JMX credentials .....	28
3.4.5 Configuring JMX connect at Tivoli Intelligent Orchestrator .....	28
3.4.6 Associating the objective analyzer with an application .....	29
3.4.7 Configuring SSH .....	30
3.4.8 Starting the Enterprise Workload Manager domain manager .....	32
3.4.9 Starting the Enterprise Workload Manager Control Center .....	32

3.4.10 Starting Tivoli Intelligent Orchestrator . . . . .	32
3.4.11 Sample EWLMOA workflows . . . . .	32
3.4.12 Using objective analyzer workflows . . . . .	33
3.4.13 Changes required in the objective analyzer workflows . . . . .	33
3.4.14 Sample policyengine console.log . . . . .	34
3.4.15 Objective analyzer logs . . . . .	35
3.4.16 Objective analyzer workflows . . . . .	36
3.4.17 The provisioning test . . . . .	37
3.5 Summary . . . . .	40
<b>Related publications</b> . . . . .	41
IBM Redbooks . . . . .	41
Other publications . . . . .	41
Online resources . . . . .	41
How to get IBM Redbooks . . . . .	42
Help from IBM . . . . .	42

# Notices

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

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

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

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

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

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

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

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

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


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

## **COPYRIGHT LICENSE:**

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

## Trademarks

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

Redbooks (logo) ®

AIX®

CICS®

DB2 Universal Database™

DB2®

IBM®

Parallel Sysplex®

POWER5™

Redbooks™

System z™

Tivoli®

Virtualization Engine™

WebSphere®

The following terms are trademarks of other companies:

Java, JMX, and all Java-based trademarks are 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.

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

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



# Preface

IBM® Tivoli® Provisioning Manager dynamically provisions (and deprovisions) server instances. It does so based on the end-to-end service-level achievement that is provided by IBM Enterprise Workload Manager and coordinated by IBM Tivoli Intelligent Orchestrator.

This IBM Redpaper explains how the integration of Enterprise Workload Manager and Tivoli Intelligent Orchestrator can proficiently and automatically provision systems and application middleware. In doing so, we demonstrate how business goals continue to be achieved, regardless of varying transaction rates, with efficient use of the system resources in an environment. The paper begins with an overview of the products, followed by the Enterprise Workload Manager and Tivoli Intelligent Orchestrator provisioning solution components. Then it describes our test environment and the installation and customization that we performed.

## The team that wrote this Redpaper

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

**Lynn Winkelbauer** is an IT Specialist for IBM in Poughkeepsie, New York. She has 21 years of experience in the IBM System z™ area, working in performance, Parallel Sysplex®, and On Demand Business. She specializes in the Linux® and Virtualization Engine™ areas. Lynn holds a Bachelor of Science degree in computer science and business administration. Her areas of expertise include Linux for System z, CICS® Transaction Server, WebSphere®, performance analysis and Virtualization Engine.

**Mike Buzzetti** updated this second edition. He is an IT Specialist at the Design Center in Poughkeepsie, New York. He specializes in IBM Virtualization Engine and security. He is a vocal advocate of Open Source and Linux. Mike holds a Bachelor of Science degree in computer science from Clarkson University.

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

John F. Brady  
IBM Software Group

Donna Dillenberger  
IBM Research

Mark Hulber  
IBM Research

## 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](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

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

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

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an e-mail to:

[redbook@us.ibm.com](mailto: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



## Product overview

This chapter is an overview of the products that were used in our solution:

- ▶ IBM Enterprise Workload Manager
- ▶ IBM Tivoli Intelligent Orchestrator
- ▶ IBM Tivoli Provisioning Manager

## 1.1 Enterprise Workload Manager

With IBM Enterprise Workload Manager, you can define business-oriented performance goals for work requests that are running on your servers. Enterprise Workload Manager continually monitors that work. It also reports performance statistics based on the work requests that are running on the applications and servers in its management domain. The performance goals specify how the work should run, while the performance statistics show how the work is actually running.

An Enterprise Workload Manager management domain consists of a domain manager and managed servers (Figure 1-1), which are displayed by Enterprise Workload Manager Control Center.

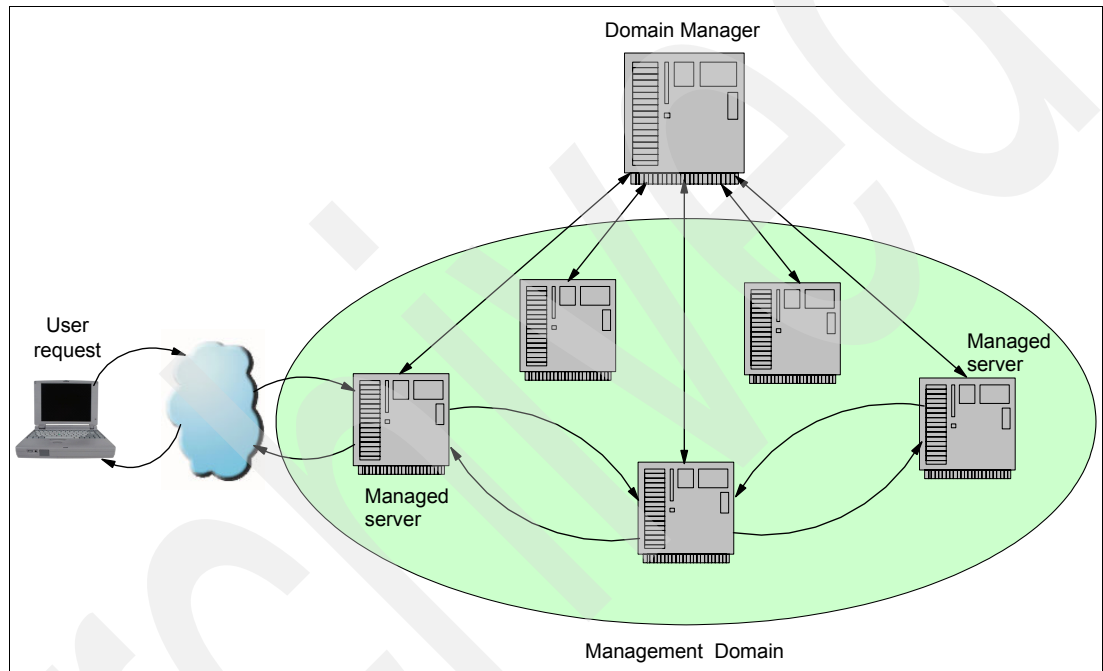


Figure 1-1 Enterprise Workload Manager environment

In the following sections, we take a closer look at the Enterprise Workload Manager components.

### 1.1.1 Enterprise Workload Manager Control Center

The Enterprise Workload Manager Control Center is a Web-based user interface that provides visibility into the performance goals and statistics. In the Enterprise Workload Manager Control Center, the service policies, including the goals, are defined and deployed to the managed servers. The Control Center has various views that show server, transaction class, and service class performance information.

### 1.1.2 Domain manager

The Enterprise Workload Manager domain manager is a software stack that consists of two operating system processes: one supports the Control Center and one coordinates the policy for the managed servers and aggregates performance statistics. The domain manager also provides management functions, including influencing network routers and dynamically adjusting LPAR CPU allocation on IBM POWER5™ servers.

The Enterprise Workload Manager domain manager has a complete and comprehensive view of the performance characteristics in the Enterprise Workload Manager management domain. Therefore, it can surface the performance data to Tivoli Intelligent Orchestrator through the objective analyzer, which receives and processes the data and optimizes resources based on that data.

### 1.1.3 Managed server

The managed server provides the Enterprise Workload Manager platform support for collecting performance data and sharing the data with the domain manager. The managed server is a common component that is installed in each operating system instance that must be managed.

After the managed server is started, it immediately makes contact with the domain manager to indicate that it is part of the management domain and starts sharing the performance data from the system with the domain manager.

For more information about Enterprise Workload Manager, see:

<http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/ewlminfo/eicaakickoff.htm>

## 1.2 Tivoli Intelligent Orchestrator

Tivoli Intelligent Orchestrator is an automated resource management solution. Using orchestrated provisioning, it can manage the IT environment in real time, according to defined business policies, to achieve desired business goals. For the purpose of this document, the defined business policies are obtained from Enterprise Workload Manager.

Tivoli Intelligent Orchestrator configures resources in a multiapplication environment to balance user traffic demands, excess capacity, and service-level targets. Using an adaptive control technology, the system predicts capacity fluctuations and facilitates dynamic infrastructure reallocation. Tivoli Intelligent Orchestrator can perform the following functions:

- ▶ Define and configure inventory resources and manage resource pools.
- ▶ Gather information about the performance of all application clusters and build a workload model that can predict future resource requirements.
- ▶ Manage resources over a set of application clusters to improve resource usage and delivery of service levels.
- ▶ Automate the deployment of the optimal computing resources to each application environment to ensure consistent response to change that is based on best practices.

Tivoli Intelligent Orchestrator automatically triggers the provisioning, configuration and deployment done by Tivoli Provisioning Manager, which is part of Tivoli Intelligent Orchestrator but is also a stand-alone product.

For more information about Tivoli Intelligent Orchestrator, see:

<http://www.ibm.com/software/tivoli/products/intell-orch/>

## 1.3 Tivoli Provisioning Manager

Tivoli Provisioning Manager automates the manual tasks of provisioning and configuring these entities:

- ▶ Servers and virtual servers
- ▶ Operating systems
- ▶ Middleware
- ▶ Applications
- ▶ Storage
- ▶ Network devices that are acting as routers, switches, firewalls, and load balancers

You can use Tivoli Provisioning Manager to create, customize, and quickly use best-practice automation packages. Pre-built automation packages provide control and configuration of major vendor products, while customized automation packages can implement and automate the best practices and procedures of your data center. In fact, when these automation packages are used, Tivoli Provisioning Manager can provision and deploy a server with the single push of a button.

For more information about Tivoli Provisioning Manager, see:

<http://www.ibm.com/software/tivoli/products/prov-mgr/>



## **Components of the Enterprise Workload Manager provisioning solution**

Now that you understand the functions of IBM Enterprise Workload Manager and IBM Tivoli Intelligent Orchestrator, you can look further into the details of the Enterprise Workload Manager provisioning solution. This chapter provides information about the components of this solution.

## 2.1 Component overview

Functionally, the provisioning solution has the following components:

- Provisioning manager

The provisioning manager provides additional resources when those available to performance managers are not enough. It consists of these areas:

- Resource optimizer (Tivoli Intelligent Orchestrator)

This decision function calculates *the best* allocation of resources (servers) to the present workload. If multiple inputs are used in this process, they might conflict. The resource optimizer resolves conflicts between various contributors to conclude which of the various reallocation choices is *best* through arbitration.

- Deployment engine (IBM Tivoli Provisioning Manager)

This implementation function manages the workflows that are associated with a decision that the resource optimizer has made.

- Database of available hardware and software (data center model)

This database performs a relationship registry function that contains available resources and the relationships between those resources (a configuration database).

- Objective analyzer

The objective analyzer has the interfaces that the provisioning manager uses. It periodically pulls resource information using the configuration information from the database. It processes performance data into a *probability of breach*, which is then periodically requested and used by the resource optimizer. The objective analyzer function is logically part of the performance manager. Therefore, we have an Enterprise Workload Manager objective analyzer.

In our environment, the Enterprise Workload Manager objective analyzer uses Tivoli Intelligent Orchestrator to complete Enterprise Workload Manager provisioning.

- Performance manager

A performance manager is used to define business goals and the importance of those goals relative to other goals. It also provides a complete view of the actual performance relative to those goals. Some performance managers can manage resource allocations, and some can surface this performance information to an objective analyzer, which in turn manages the resource allocation.

In our solution (Figure 2-1), the performance manager is Enterprise Workload Manager.

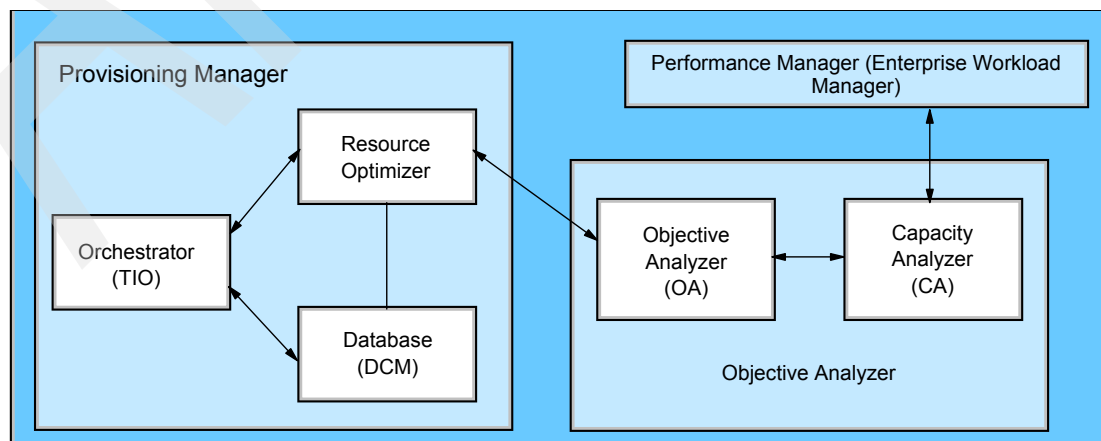


Figure 2-1 Enterprise Workload Manager and Tivoli Intelligent Orchestrator provisioning solution



In the following sections, we provide more details about the Enterprise Workload Manager and Tivoli Intelligent Orchestrator provisioning components, starting with the provisioning manager.

## 2.2 Provisioning manager

A system administrator can use provisioning software (such as Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager) to provision resources dynamically, improving the return of IT assets and server use. Tivoli Provisioning Manager provides workflows for many manual repetitive tasks performed by system, network, and storage administrators. Administrators who have implemented an objective analyzer and Tivoli Intelligent Orchestrator can automatically analyze the Enterprise Workload Manager performance data, provide this information to Tivoli Intelligent Orchestrator, and have Tivoli Intelligent Orchestrator coordinate the automation of provisioning servers, if necessary.

The provisioning manager consists of three areas:

- ▶ **Tivoli Intelligent Orchestrator:** Resource optimization
- ▶ **Tivoli Provisioning Manager:** Deployment engine
- ▶ **Data center model:** Tivoli Provisioning Manager database

### 2.2.1 The resource optimizer (Tivoli Intelligent Orchestrator)

The resource optimizer makes capacity allocation decisions that are based on data in the form of a *probability of breach surface*, or a *P(B) surface*. For each interval, the resource optimizer requests P(B) surfaces for all clusters that are associated with the application. Because the membership to each cluster might have changed since the last interval, the database must be probed for each cluster to determine which servers are members.

Administrators can set the scope (minimum and maximum bounds) of the resources that are available to a cluster. The resource optimizer is then free to provision and deprovision those resources to the cluster within the specified range. The resource optimizer assumes that all resources in a pool share the same computation capacity. Finally, the optimizer sends its provisioning decisions to Tivoli Provisioning Manager for execution.

### 2.2.2 Deployment engine (Tivoli Provisioning Manager)

To carry out a decision that is made by a resource optimizer, Tivoli Provisioning Manager looks up the application cluster for each application, takes a server from the resource pool that is associated with that cluster, and provisions the server to it. To accomplish this, the administrator implements a workflow called *Cluster.AddServer*. As part of the workflow, Tivoli Provisioning Manager can install and start the Enterprise Workload Manager managed server and any of the middleware products.

### 2.2.3 Database (data center model)

All of the servers and resources (the physical and logical) that make up an application in the data center are defined in a database called a *data center model*. The data center model keeps track of the data center hardware, applications, and configuration changes. When a Tivoli workflow successfully completes a requested change to the data center, the updated data center model reflects the current data center infrastructure.

The data center model structure (Figure 2-2) shows the objects and their relationships in the data center model for both the network infrastructure and the customer relationship model.

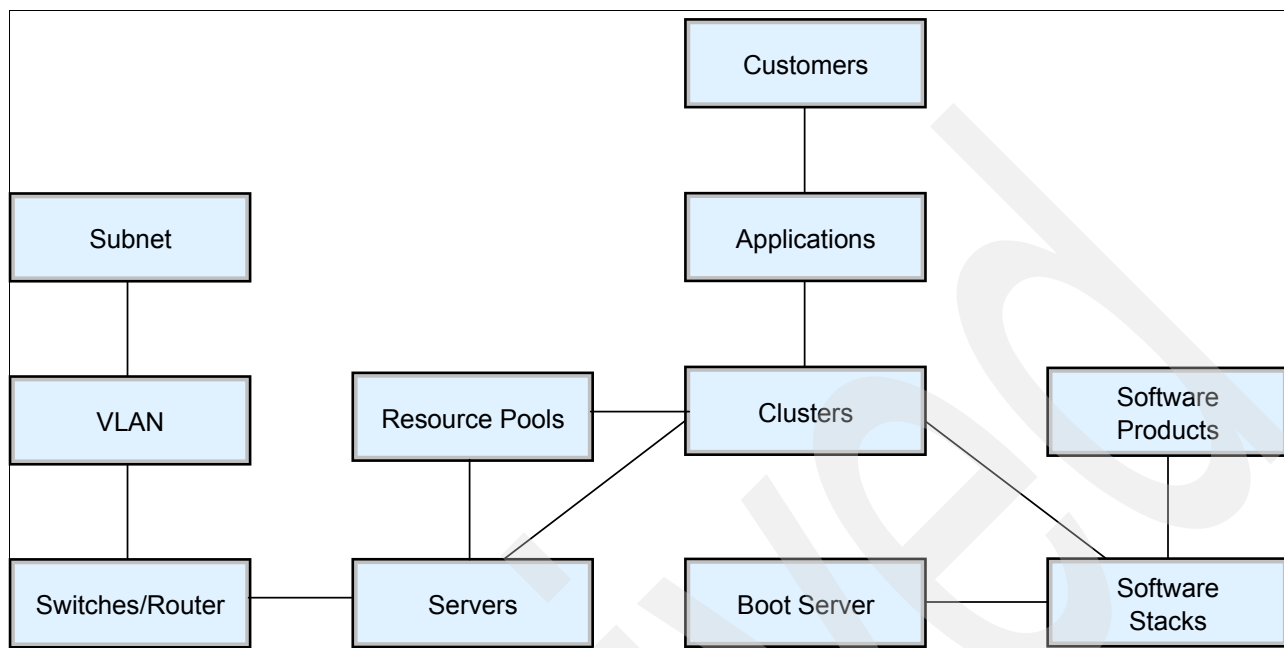


Figure 2-2 Structure of the data center model

## Network infrastructure

On the left side of the data center model structure in Figure 2-2, you see the network infrastructure. A data center model can contain one or more subnets and associated *virtual LANs* (VLANs). VLANs are a logical association of switch ports that are based on a set of rules or criteria, such as a MAC address, protocol, network address, or multicast address. This concept permits the logical grouping of devices in a network to form a single domain without requiring physical rearrangement. To create and configure a resource pool or an application cluster, you must have a VLAN.

A VLAN is associated with a *subnet* definition. Therefore, you must create a subnet definition first. We used the following logical order to create and configure our assets and resources:

1. Subnet
2. VLAN
3. Switch
4. Resource pools
5. Servers

Example 2-1 shows the definition of the switch fabric, subnet, and VLAN in the data center model.

### Example 2-1 Data center model definition of switch fabric, subnet, and VLAN

```

<switch-fabric name="DesignCenter_Fabric" />
<subnetwork name="DesignCenter181subnet" ipaddress="129.40.181.0" netmask="255.255.255.0"
in-maintenance="false">
<vlan vlan-number="3" fabric="DesignCenter_Fabric" in-maintenance="false" />
</subnetwork>

```

## Customer relationship model

On the right side of the data center model structure in Figure 2-2 on page 8, you see the customer relationship infrastructure with the customer definition at the top. The customer relationship model consists of the following components:

- ▶ **Customer:** A customer owns applications. Customers can be unique corporations or departments within a single corporation.
- ▶ **Application:** This is a group of one or more clusters. A service level priority (silver, gold, or platinum) is assigned at this level. The object analyzer is also associated at this level.
- ▶ **Application cluster:** This is a grouping or container for like resources or servers that support an application. Automated resource allocation and deallocation occurs at the cluster level.
- ▶ **Resource pool:** This is a grouping or container of available (deallocated) servers that support one or more application clusters. This is also called a *spare pool*.
- ▶ **Servers:** Servers refers to the physical definition of servers. They belong to or are assigned to pools and clusters.

We used the following logical order for our customer definition:

1. Customers
2. Application
3. Application clusters and resource pool
4. Servers
5. Software stacks
6. Software products

### 2.2.4 Configuration summary

In this section, we present a simple example of the provisioning manager using the configuration in Figure 2-3.

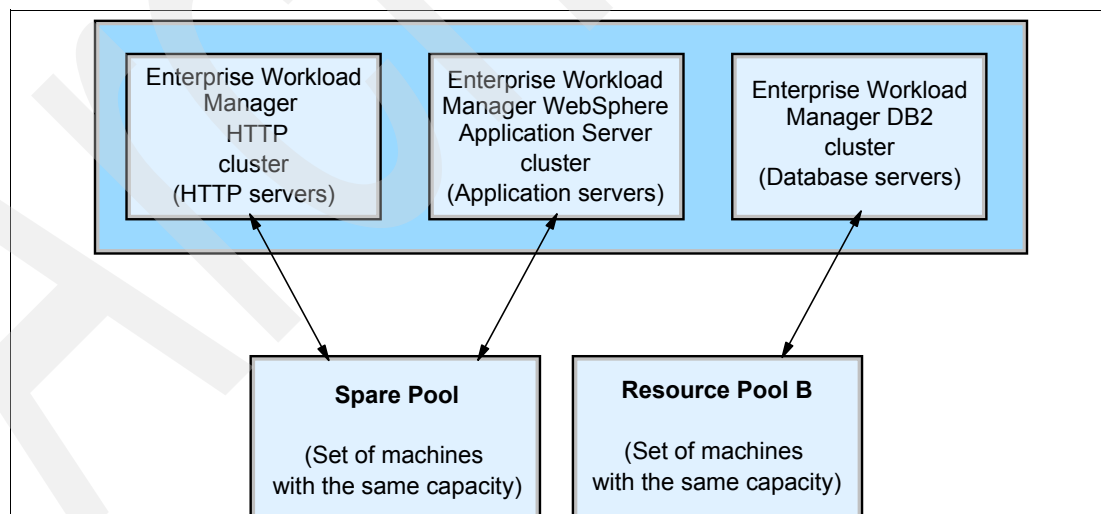


Figure 2-3 Single multi-tier application and resource pools

We defined three clusters in the data center model for a customer application:

- ▶ One cluster is defined for the HTTP servers.
- ▶ One cluster is defined for the WebSphere Application Server profiles.
- ▶ One cluster is defined for the database server.

All servers that are part of the application are also part of the same Enterprise Workload Manager management domain.

We also defined two resource pools. Resource pools consist of a list of available servers that Tivoli Intelligent Orchestrator can use to provision into the clusters as needed. In this example, a *spare pool* is available for provisioning into the HTTP cluster or the WebSphere Application Server cluster. *Resource Pool B* is available for provisioning into the DB2® cluster.

Suppose that the application is running at a higher rate than normal. The objective analyzer determines that Cluster B is contributing to the performance goal not being met based on Enterprise Workload Manager performance information. The objective analyzer determines that there is a CPU delay on the servers that belong to Cluster B. Then it suggests to Tivoli Intelligent Orchestrator that it should provision a new server from Resource Pool A into Cluster B to offset some of the work, distributing some of the CPU consumption throughout an expanded Cluster B.

The provisioning can include anything from building the operating system to installing middleware to setting up IP addresses to starting the necessary software. It can also include the spare server in the cluster, depending on the workflows that are included in the Cluster.AddServer logical device operation.

## 2.3 Objective analyzer

Tivoli Intelligent Orchestrator obtains Enterprise Workload Manager performance information using a technology that is called an *objective analyzer*. It enables Tivoli Intelligent Orchestrator to make better provisioning decisions based on performance information that it receives from Enterprise Workload Manager for the resources in the clusters. The objective analyzer determines how the applications and servers in this environment are affecting the business goals for a particular workload.

The objective analyzer is associated with an application. Periodically the performance manager (Enterprise Workload Manager) sends performance information to the objective analyzer. The objective analyzer tells Tivoli Intelligent Orchestrator how likely the cluster is to fail to meet its performance goals using various allocations of machines. By repeatedly calling all the currently assigned objective analyzers with a different number of machines, Tivoli Intelligent Orchestrator can determine the best course of action for this cycle.

Tivoli Intelligent Orchestrator determines cluster requirements based on data that it receives from Enterprise Workload Manager about the resources using the Enterprise Workload Manager objective analyzer. Although Tivoli Intelligent Orchestrator comes with a default objective analyzer called *capacity on demand*, it is based on a Web workload environment. The objective analyzer that we use is based on performance metrics that are obtained from Enterprise Workload Manager, which we discuss later.

**Note:** Although Tivoli Intelligent Orchestrator has a default objective analyzer, when we refer to objective analyzer in this paper, we are referring to the Enterprise Workload Manager objective analyzer unless otherwise noted.

Using the objective analyzer technology, users can model their workloads to determine the resources that a particular cluster needs. Tivoli Intelligent Orchestrator, in turn, determines the priority of the cluster to determine if resources are available to service the requirements of the cluster. Resources that are not in use can be provisioned into the cluster. If no resources are available, Tivoli Intelligent Orchestrator might determine that clusters with a lower priority must give up resources so that they can be provisioned into higher-priority clusters.

For more information about objective analyzer, see:

[http://publib.boulder.ibm.com/infocenter/wxdinfo/v6r0/index.jsp?topic=/com.ibm.websphere.xd.doc/info/odoe\\_task/codoetioa.html](http://publib.boulder.ibm.com/infocenter/wxdinfo/v6r0/index.jsp?topic=/com.ibm.websphere.xd.doc/info/odoe_task/codoetioa.html)

## Point of contact

The objective analyzer is the point of contact between the capacity analyzer and the resource optimizer. This component exposes all of the required external interfaces to Enterprise Workload Manager and Tivoli Intelligent Orchestrator.

## Capacity analyzer

The function of a capacity analyzer is to process performance data. The capacity analyzer turns the data into recommendations that represent the likelihood that, for a range of system allocations over time, there will be a breach of performance goals that are defined in the Enterprise Workload Manager service policy.

The capacity analyzer is the *hub* of the provisioning model, the destination of the performance reporting data, and creator of P(B) surfaces for the resource optimizer.

The capacity analyzer is responsible for working through the Enterprise Workload Manager reporting data and generating P(B) surfaces for each service class in a cluster. The P(B) surface represents the probability that the service-level objective for a particular service class on a cluster will meet its performance goal.

Instead of using Enterprise Workload Manager performance indexes to communicate the performance impact of capacity changes, a probability value is used. This value represents the probability that a goal will be missed (breached), hence the name probability of breach. To project the value, the goal (the service class period) and the number of allocated servers must be known. For the resource optimizer to be able to evaluate a range of possible values using data from a single interaction, a set of probabilities must be calculated. For the capacity analyzer perspective, the relevant variables are:

- ▶ The amount of resources that are available to the cluster (number of servers)
- ▶ The identity of the servers in the cluster by fully qualified host name
- ▶ The length of time to consider in the P(B) surface (time)
- ▶ The service class that contains the goal

The capacity analyzer is interested in the servers that comprise a cluster, but not in the specific details about the cluster.

## Generic interfaces

The following interfaces that are provided by the objective analyzer must be compatible with Tivoli Intelligent Orchestrator, Tivoli Provisioning Manager, the data center model, and any other third-party provisioning manager:

- ▶ Initialize an objective analyzer: The resource optimizer requests an instance of the CA to be initialized.
- ▶ Start a decision cycle: The resource optimizer notifies the objective analyzer of the start of a decision cycle for the cluster. The underlying data provided in the P(B) surfaces must not change during the course of a decision cycle. It is used to initiate a collection and computation cycle.
- ▶ Get P(B) surface: The resource optimizer requests a P(B) surface from the objective analyzer for a certain cluster.

- ▶ **End a decision cycle:** The resource optimizer notifies the objective analyzer of the end of a decision cycle for the cluster. The underlying data provided in the P(B) surfaces may now be discarded for this decision cycle.
- ▶ **Change managed status:** The resource optimizer notifies the objective analyzer of a managed status change. When a cluster has the *isManaged==false* Tivoli setting, then the resource optimizer excludes this cluster from the decision cycle, and the objective analyzer may stop any data collection and computation for this cluster. When the setting is *isManaged==true*, the objective analyzer is required to collect and compute data.
- ▶ **Shut down an objective analyzer:** The resource optimizer notifies the objective analyzer to shut down the instance because it is no longer required.

## 2.4 Performance manager (Enterprise Workload Manager)

In the Enterprise Workload Manager environment, a performance goal specifies how fast work should run in your environment and the relative importance of goals to others within the domain. Performance goals are defined for each service class in the Enterprise Workload Manager service policy. The following performance goals are available:

- ▶ **Percentile response time:** Specifies the percentage of work requests that should be completed within a specified amount of time
- ▶ **Average response time:** Specifies the average amount of time that work in the service class should be completed
- ▶ **Velocity:** Defines how fast work should run when ready, without being delayed for processor, storage, or I/O, for managed system resources)

Velocity goals are intended for work for which response time goals are not appropriate, such as service processes, daemons, and long-running batch work. You can also use velocity goals for applications that have not been Application Response Measurement (ARM) instrumented. These work requests are normally initiated by the system automatically rather than from a user.

- ▶ **Discretionary:** Specifies that work associated with this goal is to complete when resources are available

Enterprise Workload Manager provides a convenient measurement, called the performance index (PI), to indicate whether the work is meeting its goal.

From an external point of view, the PI is viewed as:

- ▶  $PI < 1$ : Exceeding the goal
- ▶  $PI = 1$ : Meeting the goal
- ▶  $PI > 1$ : Missing the goal

If the work is meeting its business goal (denoted by  $PI = 1$ ), the amount of resources for the work being processed is sufficient. If the work is exceeding its business goal ( $PI < 1$ ), you can assume that you have enough resources and perhaps have more resources than you need. If the work is not meeting its goal ( $PI > 1$ ), then an action needs to be taken to help achieve that goal. This action could mean that new resources must be provisioned to accommodate the additional processing of the work.

### Analyzing the Enterprise Workload Manager performance data

Before the integration of Enterprise Workload Manager and Tivoli Intelligent Orchestrator, system administrators could manually provision new resources based on the performance information provided by Enterprise Workload Manager.

The Enterprise Workload Manager Control Center shows performance statistics for defined service classes. If the service class shows  $PI > 1$ , (or increasing toward a  $PI > 1$ ), system administrators could analyze the performance data in the Enterprise Workload Manager Control Center to determine which server was the cause for not meeting the performance goal. For a detailed description of how system administrators can use the Enterprise Workload Manager Control Center views to determine the root cause of the performance problem, see *Enterprise Workload Manager - Interpreting Control Center Performance Reports*, REDP-3963. By understanding the root cause, you can perform the appropriate steps, which might include the provisioning of a new server to absorb the additional workload.

With the integration of Enterprise Workload Manager and Tivoli Intelligent Orchestrator, the Enterprise Workload Manager performance information in the form of a P(B) surface is passed to Tivoli Intelligent Orchestrator through the objective analyzer. Therefore, the provisioning manager can automatically perform determination and provisioning. The following list identifies some of the topology fields used:

- ▶ **Name:** Specifies the name of the server, application, or instance that is processing work requests for a particular hop
- ▶ **Average response time:** Specifies the amount of time, on average, that the work requests processed on the server, application, or instance incur from the moment they enter the hop  
The average response time includes all transactions processed in a particular hop.
- ▶ **Average active time:** Specifies how long, on average, the server or application takes to process a work request
- ▶ **Processor time:** Indicates the actual cumulative amount of time, in seconds, that all of the transactions for the specific time interval incur on the processor
- ▶ **Processor delay:** Specifies the percentage of non-idle time that the work requests incur when they request to use the processor but are unable to do so
- ▶ **Processor using percentage:** Specifies the percentage of time that the server, application, or instance can use the processor when it requests to do so
- ▶ **Stopped transactions:** Specifies the number of transactions that stopped before completing
- ▶ **Failed transactions:** Specifies the number of transactions that failed on each server or application for the specified hop
- ▶ **Successful transactions:** Specifies the number of transactions that complete successfully in the associated hop
- ▶ **I/O delays:** Specifies the percentage of non-idle time where I/O delays the work requests in the associated hop
- ▶ **Page delays:** Specifies the percentage of non-idle time that transactions incur while waiting for page faults to resolve
- ▶ **Idle:** Specifies the percentage of time that the processor is available but not processing any transactions
- ▶ **Hard page faults:** Specifies the number of page faults that the operating system needs to resolve by accessing hard disk I/O rather than virtual memory
- ▶ **Average queue time:** Specifies the amount of time, on average, that incurs from the moment that the middleware application receives the work request to the moment that it begins processing it

For more information about these Enterprise Workload Manager topology fields, see:

[http://publib.boulder.ibm.com/infocenter/eserver/v1r1/en\\_US/index.htm?info/icmain.htm](http://publib.boulder.ibm.com/infocenter/eserver/v1r1/en_US/index.htm?info/icmain.htm)

Archived





## Environment for testing the provisioning solution

This chapter provides specific details about the environment that we used for testing the IBM Enterprise Workload Manager and IBM Tivoli Intelligent Orchestrator provisioning solution.

## 3.1 Test environment

This section explains the details of the application environment and the management server used for this solution.

### 3.1.1 Application environment

We created a three-tier infrastructure from the four servers in our environment: a Web server tier, an application server tier, and a database server tier. The application server tier contains a two-server WebSphere Application Server cluster. These three tiers map to the three application cluster definitions in the data center model. We also included two spare servers (D14V01 and D14V02) in the infrastructure, and they are part of the Tivoli Intelligent Orchestrator resource pool that we used for provisioning. Figure 3-1 shows the servers, operating systems, applications, connectivity, and some of the Tivoli definitions.

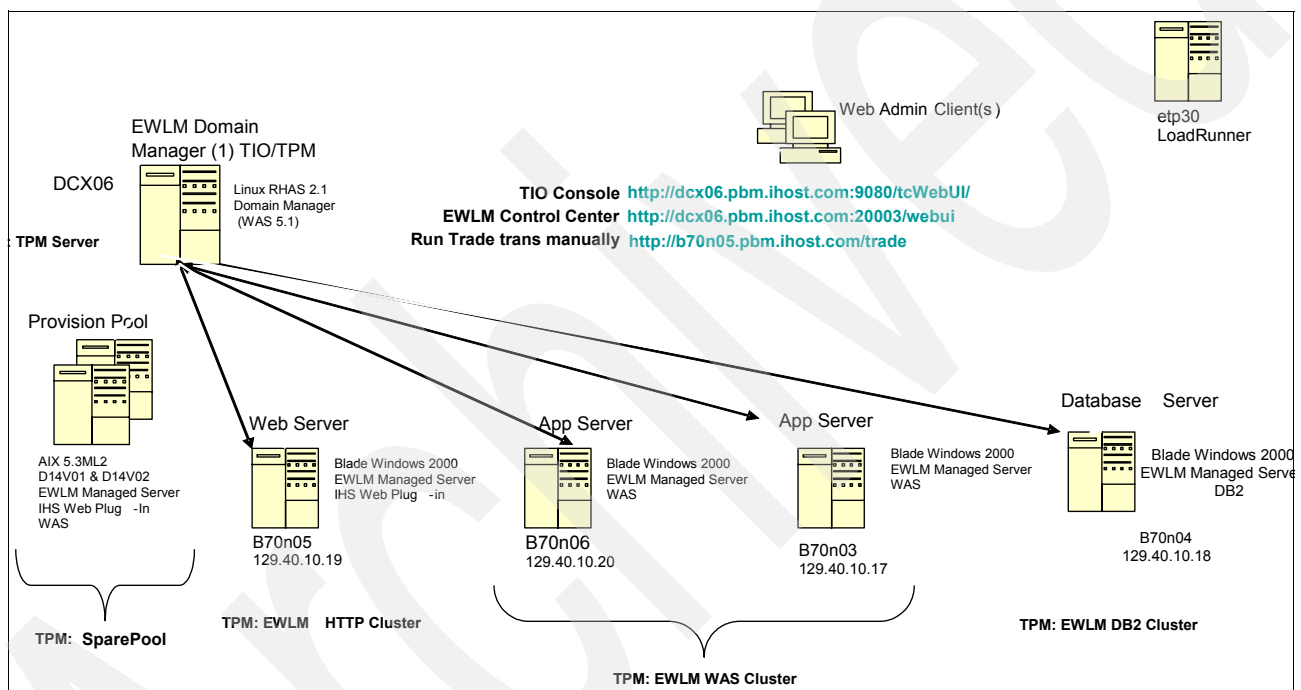


Figure 3-1 Enterprise Workload Manager and Tivoli Intelligent Orchestrator environment

### 3.1.2 Management server

The management server is the system that is designated as the hub of the Enterprise Workload Manager and Tivoli Intelligent Orchestrator provisioning solution. It coordinates the Enterprise Workload Manager policy for the managed servers and aggregates performance statistics from them. It uses the performance statistics to provision new servers into the Enterprise Workload Manager management domain to ensure that the Enterprise Workload Manager business goals are met. Our management server, DCX06, hosted the domain manager, Tivoli Intelligent Orchestrator, and the objective analyzer driver. Our server was a Red Hat Enterprise Linux AS 2.1 running Enterprise Workload Manager domain manager V1.1, Tivoli Intelligent Orchestrator V2.1.2, and Enterprise Workload Manager objective analyzer V1.0.2.

## 3.2 Enterprise Workload Manager

Enterprise Workload Manager is our performance manager as explained in 2.4, “Performance manager (Enterprise Workload Manager)” on page 12. We defined one Enterprise Workload Manager domain manager and six managed servers.

### 3.2.1 Enterprise Workload Manager domain manager

Using the Enterprise Workload Manager Control Center, we set up our *domain policy*, which specifies the performance goals for the work running in the Enterprise Workload Manager management domain. It has information about the entry application, platforms, transaction classes, service classes (Figure 3-2), and associated service policy performance goals. We deployed the domain policy to each managed server and activated the service policy.



Service Classes					
View the performance of the service classes. Select a service class for more details.					
Interval: 2:24 PM 9/20/05 to 2:34 PM 9/20/05					
 -- Select Action --- 					
Select	Service class <sup>2</sup>	PI <sup>^</sup>	Importance <sup>1</sup>	Performance <sup>^</sup>	Goal <sup>^</sup>
<input type="radio"/>	SC_Calc_batch	0.00	Medium	Slowest velocity	Slowest velocity
<input type="radio"/>	SC_InternetExplorer	0.00	Medium	Slowest velocity	Fast velocity
<input type="radio"/>	SC_Plants_default	0.00	Medium	00.000 average	00.100 average
<input type="radio"/>	SC_Plants_login	0.00	Medium	0% in 00.150	90% in 00.150
<input type="radio"/>	SC_Plants_shopping	0.00	Medium	0% in 00.200	80% in 00.200
<input type="radio"/>	SC_Trade3_account	0.62	Medium	00.094 average	00.150 average
<input type="radio"/>	SC_Trade3_default	0.53	Medium	00.096 average	00.180 average
<input type="radio"/>	SC_Trade3_portfolio	0.69	Medium	00.097 average	00.140 average
<input type="radio"/>	SC_Trade3_profile	0.59	Medium	00.059 average	00.100 average
<input type="radio"/>	SC_Trade3_quotes	0.64	Medium	00.090 average	00.140 average
<input type="radio"/>	SC_Trade3_shopping	0.50	Medium	89% in 00.110	63% in 00.110

Figure 3-2 Service Classes view

When the service policy was activated, the domain manager started receiving data from the Enterprise Workload Manager managed servers periodically and provided the data to Tivoli Intelligent Orchestrator through the objective analyzer. This information can also be viewed from the Enterprise Workload Manager Control Center (Figure 3-3.)



 -- Select Action --- 					
Select	Managed server <sup>^</sup>	Processor utilization % <sup>^</sup>	State <sup>^</sup>	Active service policy <sup>^</sup>	Operating sy:
<input type="radio"/>	b70n03	33.4	Active	DC Trade3 and Plants Service Policy	Windows
<input type="radio"/>	b70n04	26.4	Active	DC Trade3 and Plants Service Policy	Windows
<input type="radio"/>	b70n05	6.2	Active	DC Trade3 and Plants Service Policy	Windows
<input type="radio"/>	b70n06	60.8	Active	DC Trade3 and Plants Service Policy	Windows
<input type="radio"/>	d14v01.pbm.ihost.com		Communication error	DC Trade3 and Plants Service Policy	AIX
Page 1 of 1		Total: 5 Filtered: 5 Displayed: 5			

Figure 3-3 Managed servers

Notice that there is a communication error for d14v01 because the Enterprise Workload Manager managed server was not started.

To obtain more detailed information about each of the service classes, select the service class and then select **Application Topology**. From the application topology view, select the tableview window (Figure 3-4) to see detailed information about the service class, as explained in “Analyzing the Enterprise Workload Manager performance data” on page 12. We see granular information for each hop of the service class, including the application, server, and instance of the hop.

Hop	Name	Type	Platform	Average response time...	Average active time (se...	Processor time (secon...	Pr
0	IBM Webserver Plugin [IBM_HTTP...	Application		00.098	00.000	00.15625	0%
0	b70n05	Server	Windows	00.098	00.000	00.15625	0%
0	B70N05/PID=0000000520	Instance		00.139	00.018	00.15625	0%
0	B70N05/PID=0000001240	Instance		00.003	00.000	00.0	0%
1	WebSphere [TradeClusterServer1]	Application		00.001	00.001	08.820941	0%
1	b70n06	Server	Windows	00.001	00.001	08.820941	0%
1	b70n06.TradeClusterServer1	Instance		00.001	00.001	08.820941	0%
1	WebSphere [TradeClusterServer2]	Application		00.001	00.001	09.657295	0%
1	b70n03	Server	Windows	00.001	00.001	09.657295	0%
1	b70n03.TradeClusterServer2	Instance		00.001	00.001	09.657295	0%

Figure 3-4 Application Topology tableview window

The performance information gathered at the Enterprise Workload Manager domain manager is sent to the objective analyzer and processed. Through the objective analyzer P(B) surfaces, Tivoli Intelligent Orchestrator can make decisions about whether a new server needs to be provisioned and where to provision it to meet the specified goals. For example, instead of the table in Figure 3-4 showing that there is 0% processor delay, we might see a 40% processor delay for WebSphere (TradeClusterServer1), which would lead us to believe that we need more processing power in the WebSphere cluster.

### 3.2.2 Enterprise Workload Manager managed server

Initially, there were four managed servers in our management domain with the potential to go to six. The two additional servers would be provisioned into our Enterprise Workload Manager management domain if our Enterprise Workload Manager business goals were not being met. Figure 3-5 shows the initial application topology, which we captured from the Enterprise Workload Manager Control Center application topology view.

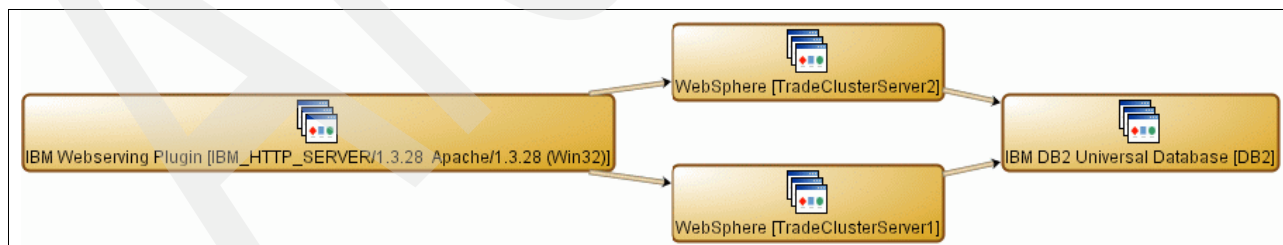


Figure 3-5 Application topology view

Note how transactions flow through our three-tier environment. The entry application in our Enterprise Workload Manager management domain (Hop 0) is the IBM HTTP server. The next hop (Hop 1) is at the WebSphere Application Server cluster, which consists of the two WebSphere Application Server profiles: TradeClusterServer1 and TradeClusterServer2. The last hop (Hop 2) is to the DB2 Universal Database™ server. All of the managed servers run on Microsoft® Windows® 2000 Server SP4.

Servers D14V01 and D14V02, which are part of the Tivoli Intelligent Orchestrator Resource Pool, were logical partitions (LPARs) on an IBM System p590 running AIX® Version 5.3, maintenance level 2. These two servers could be provisioned as part of either the HTTP cluster or the WebSphere Application Server cluster. They do not appear in the diagram in Figure 3-5 because they have not yet been provisioned.

### 3.2.3 Software

Our environment consisted of the following middleware products:

- ▶ IBM HTTP server 1.3.28 (on B70N05, D14V01, and D14V02)
- ▶ WebSphere Application Server V5.1.1.1 (on B70N06, B70N03, D14V01, and D14V02)
- ▶ DB2 UDB Version 8.2 (on B70N04)

## 3.3 Tivoli Intelligent Orchestrator

After installing Tivoli Intelligent Orchestrator on our management server, our next step was to define our resources in the data center model. First, we set up the network infrastructure, then the customer relationship model. Finally, we associated the objective analyzer to the application.

### 3.3.1 Our network infrastructure setup

Figure 3-6 shows the resource definitions and the relationships that were defined in the data center model. The switch fabric had three subnets and associated VLANs. The VLANs have a logical association with a switch port, which we based on the IP address. The servers are part of one of the switch ports based on the IP address.

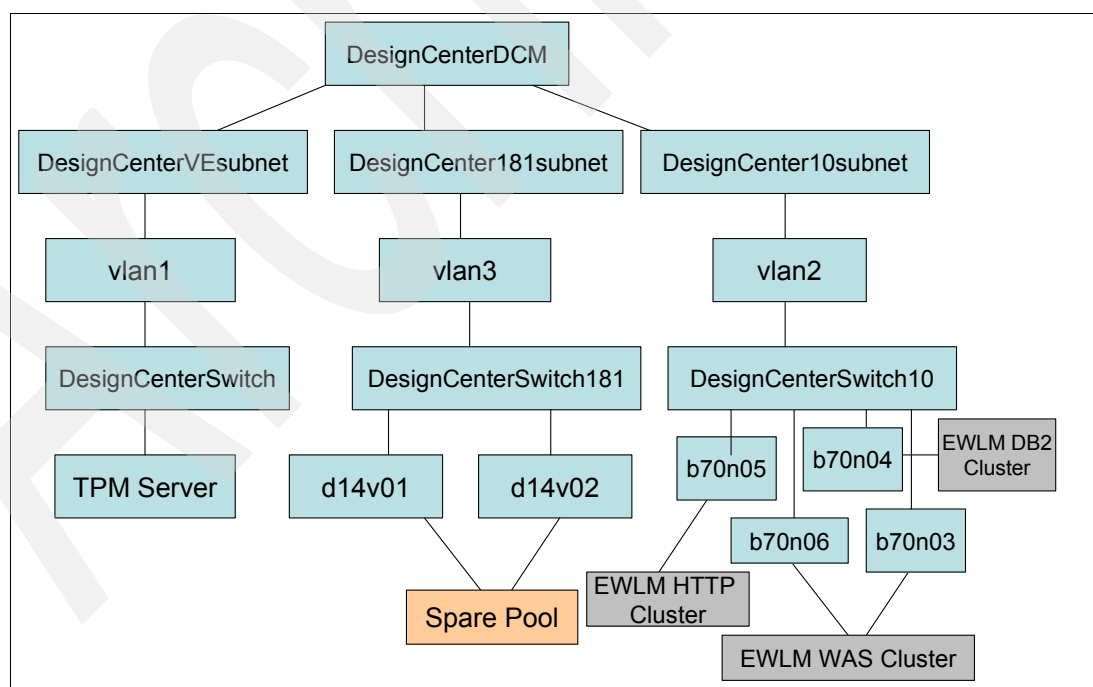


Figure 3-6 Our network infrastructure in the DCM

We defined two resource pools: *TPM Server* and *Spare Pool*. *TPM Server* was defined for the management server, DCX06. *Spare Pool* was a container for our available servers that

support both the HTTP cluster and the WebSphere Application Server cluster. Spare Pool consists of two AIX images, d14v01 and d14v02. The other servers (b70n03 through b70n06) were dedicated to one of the three application clusters: Enterprise Workload Manager HTTP cluster, Enterprise Workload Manager WebSphere Application Server cluster, or Enterprise Workload Manager DB2 cluster.

When P(B) surfaces are being created with the Enterprise Workload Manager reporting data, the servers listed in the data center model must map correctly to the servers that appear in the Enterprise Workload Manager data. This is done by storing the Enterprise Workload Manager Universal Unique Identifier of the managed server in the data center model for each server.

### 3.3.2 Our customer relationship model

Continuing with the definitions in the data center model, we had to associate the physical servers with the logical function (the trade3 stock application). Figure 3-7 illustrates the customer relationship model defined in the data center model. We started with the definition of a customer who owns the Enterprise Workload Manager Trade Application. The Enterprise Workload Manager Trade Application consists of three application clusters, one for each of the application tiers (HTTP, WebSphere Application Server, and DB2).

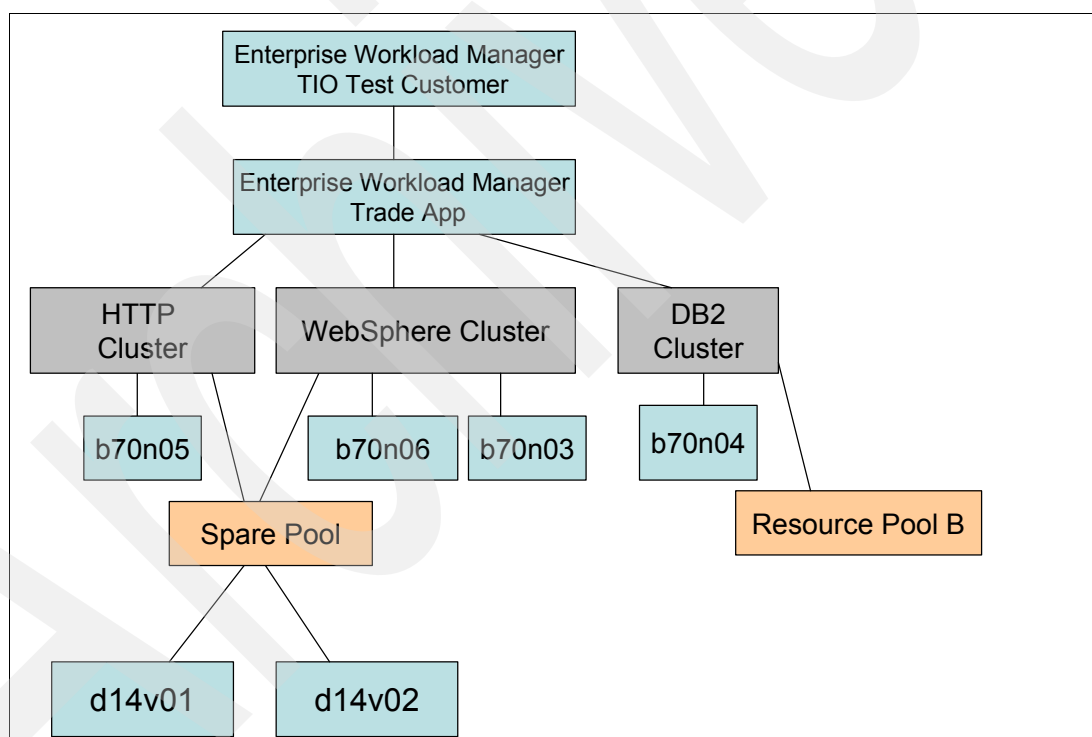


Figure 3-7 Our customer relationship model defined in the DCM

Each cluster had a dedicated server or servers defined to it. The resource pool, *Spare Pool*, is a grouping of two available (deallocated) servers that support the HTTP and the WebSphere Application Server clusters, in case it was determined that a new server had to be provisioned into that cluster. We also defined another resource pool, *Resource Pool B*. We did not define servers to it, because we did not customize DB2 to span multiple operating systems. We left that for another test.

Figure 3-8 on page 21 shows some of the definitions that we extracted from the data center model. It includes the customer (Enterprise Workload Manager TIO Test Customer) definition,



the associated application (Enterprise Workload Manager Trade Application), the application cluster (Enterprise Workload Manager HTTP Cluster), and a server (b70n05). The figure also shows how we associated the application cluster with the resource pool (SparePool) that the application cluster uses to draw resources. In addition, the *software stack* is associated with the server. The software stack is defined with the software products associated with it.

```
<customer name="EWLM TIO Test Customer">
  <application name="EWLM Trade App" locale="en_US" priority="10.0" in-
    maintenance="false">
    <cluster name="EWLM HTTP Cluster" locale="en_US" managed="true" virtual-tcp-port="-1"
      min-servers="1" max-servers="6" in-maintenance="false" pool="SparePool" vlan="2"
      fabric="DesignCenter_Fabric" tier="0" />
    <server name="b70n05" locale="en_US" is-device-model="SSH Service Access Point"
      in-maintenance="false" failed="false" software-stack="Stack2104" tcp-port="0">
      . . .

  <objective-analyzer type-name="EWLMOA">
    <parameter name="weighting" value="1" />
  </objective-analyzer>
</application>
</customer>

. . .

<software-stack name="Stack2104" stack-type="Inherited" is-image="false">
  <software-stack-product product-name="HTTPServer" position="10" expected-
    state="Installed" />
</software-stack>
<software name="HTTPServer" locale="en_US" service="false" type="SOFTWARE"
  version="1.3.28" package_path="C:\cygdrive\install\IHS-1.3.28" category="WebSphere">
  <property component="DEPLOYMENT_ENGINE" name="http.installLocation"
    value="/usr/IBMHttpServer" />
  <property component="DEPLOYMENT_ENGINE" name="http.service.name"
    value="apachectl" />
  <property component="DEPLOYMENT_ENGINE" name="ihs.service.name"
    value="IBMHttpServer" />
</software>
```

Figure 3-8 Customer relationship data center model definitions

### 3.3.3 Configuring SSH

Before you run any of the Tivoli workflows, SSH communication must be set up between the management server (that Tivoli Intelligent Orchestrator is running on) and the managed servers so that the Tivoli Intelligent Orchestrator administrator is authorized to perform the functions it needs for provisioning. For more information about the steps that are involved, see “Installing and configuring SSH” in the *IBM Tivoli Provisioning Manager Installation Guide for Linux Version 2.1.0*, GC32-1616.

### 3.3.4 Summary of our environment

This section summarizes our environment before we configured the Enterprise Workload Manager objective analyzer.

#### Application configuration

For the application, we updated the WebSphere plug-in on B70N05 to round-robin to B70N03, B70N06, d14v01, and d14v02. Because d14v01 and d14v02 were not yet provisioned, the HTTP Server routed only to B70N03 and B70N06 initially. There is a WebSphere plug-in configuration parameter called *RetryInterval* that specifies the length of

time that the plug-in waits after marking a server as unavailable, before sending it an additional request. We set ours to 60 seconds.

On d14v01 and d14v02, we preconfigured the HTTP server and the WebSphere Application Server to run the trade3 application. After it is provisioned, the HTTP server or WebSphere Application Server can automatically be part of the new application configuration and start running the workload once the RetryInterval has elapsed.

**Note:** We could have created Tivoli Provisioning Manager workflows to install and configure Enterprise Workload Manager, the HTTP server, and WebSphere Application Server on d14v01 and d14v02 to run the trade3 application. However, we chose to set this up beforehand because we were testing the Enterprise Workload Manager and Tivoli Intelligent Orchestrator provisioning function, not the Tivoli Provisioning Manager workflows. We intended to use Tivoli Provisioning Manager workflows to start and stop Enterprise Workload Manager, the HTTP server, the WebSphere Application Server, or all three, depending on the provisioning that took place.

## Enterprise Workload Manager environment

Figure 3-9 shows how we set up our Enterprise Workload Manager environment.

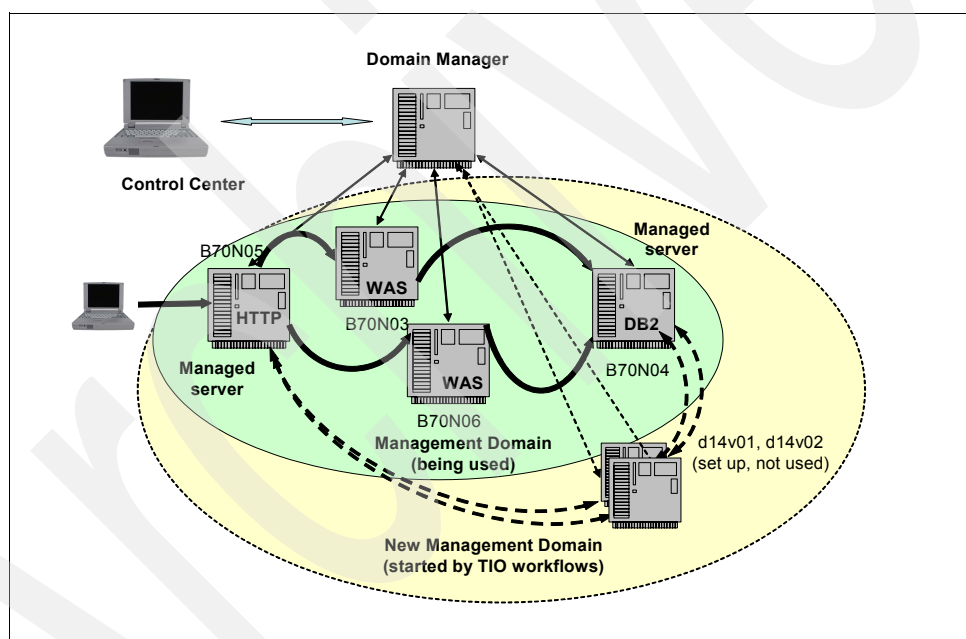


Figure 3-9 Our Enterprise Workload Manager environment

Our management domain consisted of four managed servers (an HTTP server, two WebSphere Application Server profiles, and a DB2 server). We defined our Enterprise Workload Manager service policy to help us see the managed server information and the trade3 application with different service classes. We installed the Enterprise Workload Manager managed server code on all six of the servers. Because d14v01 and d14v02 were not provisioned, the Enterprise Workload Manager managed server was not started, and therefore, was not part of the current Enterprise Workload Manager management domain.

In summary, we could use Enterprise Workload Manager Control Center to determine if we were meeting our service goals and to obtain detailed information about the problem area if we were not. We could then log in to the Tivoli Intelligent Orchestrator administrative console and use that data to provision servers into our environment manually, which includes starting



Enterprise Workload Manager managed server and either HTTP server or WebSphere Application Server. When the Enterprise Workload Manager managed server starts, it communicates with the domain manager and sends performance data. Because our application configuration was set up to recognize the new resources dynamically, we could automatically take advantage of them as soon as they were provisioned.

## Tivoli Intelligent Orchestrator

The Tivoli Intelligent Orchestrator database has the complete customer application definition and the network infrastructure with all the resources that potentially could be used by the application (Figure 3-10).

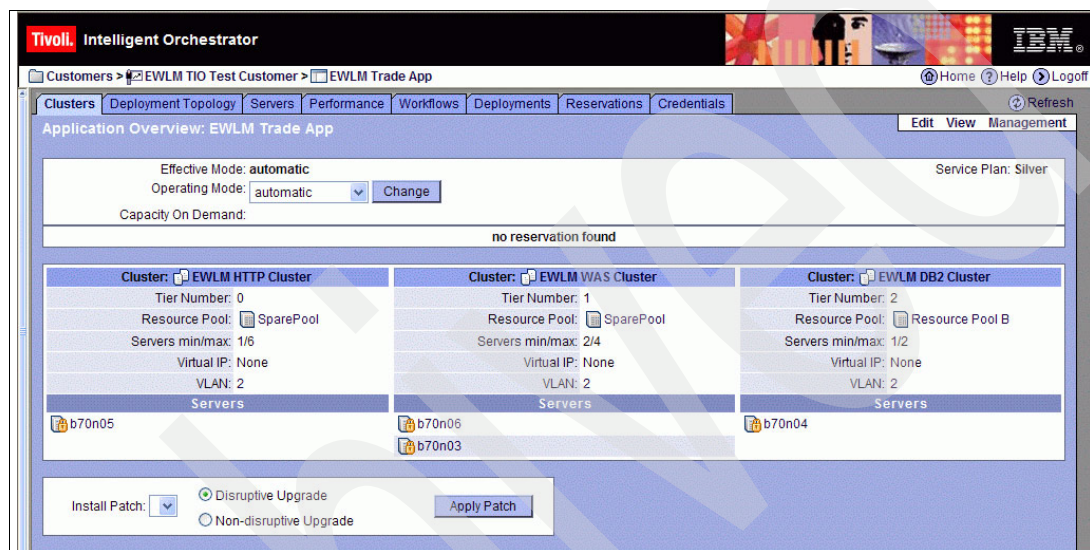


Figure 3-10 Tivoli Intelligent Orchestrator Application Overview

For each cluster, we defined the tier (hop), the resource pool, and the minimum and maximum number of servers that it can use. Tivoli Intelligent Orchestrator uses this information when provisioning to determine configuration limitations. We used the Tivoli Intelligent Orchestrator administrative console to provision servers manually into an application cluster (Figure 3-11) based on the determination we made from Enterprise Workload Manager performance data.

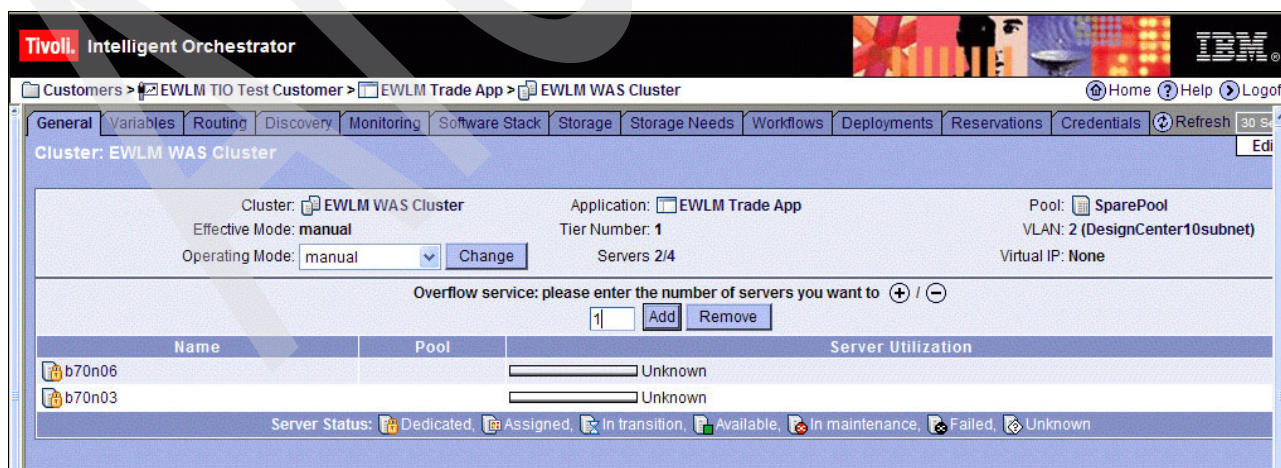


Figure 3-11 Manual provisioning into the Enterprise Workload Manager WebSphere Cluster

In this example, we provisioned one server into the WebSphere Application Server cluster. When we clicked the Add button, the logical device operation called *Cluster.AddServer* started. It provisioned the server into the WebSphere Application Server cluster and started the Enterprise Workload Manager managed server and the WebSphere Application Server automatically. For more information, see 3.4.16, “Objective analyzer workflows” on page 36.

The server that was provisioned came from SparePool. The server was “in transition” until the entire *Cluster.AddServer* workflow completed. Then, one of the SparePool servers appeared in the list of WebSphere cluster servers (Figure 3-12).

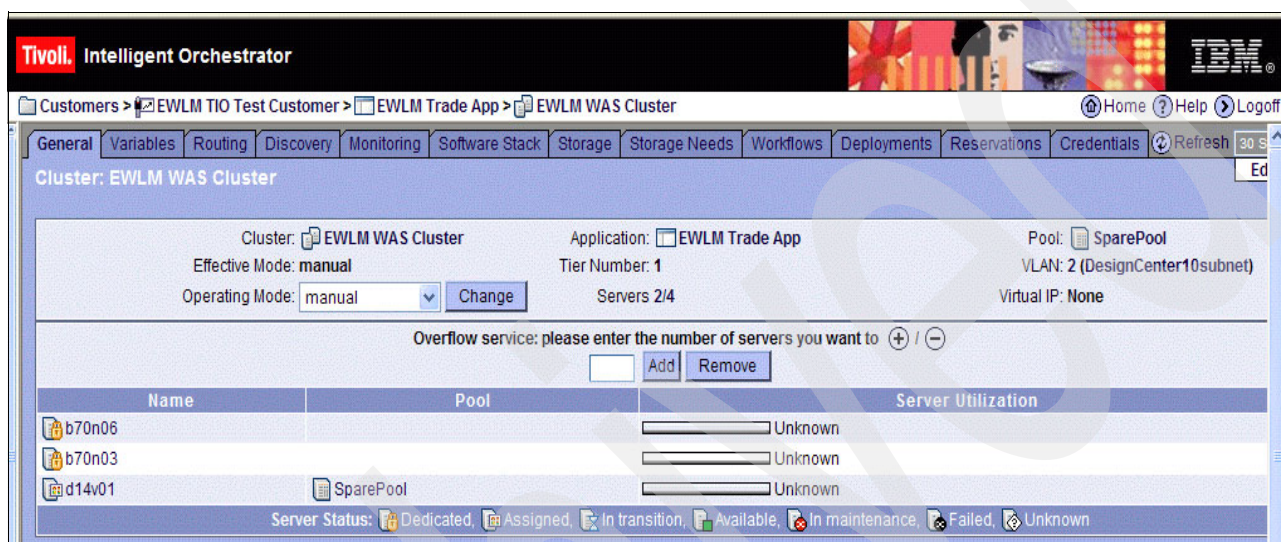


Figure 3-12 Manual provisioning completed

In summary, Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager were configured to perform *Cluster.AddServer* and *Cluster.RemoveServer* for either the HTTP cluster or the WebSphere Application Server cluster. Figure 3-13 shows the workflow configuration for the WebSphere Application Server cluster.

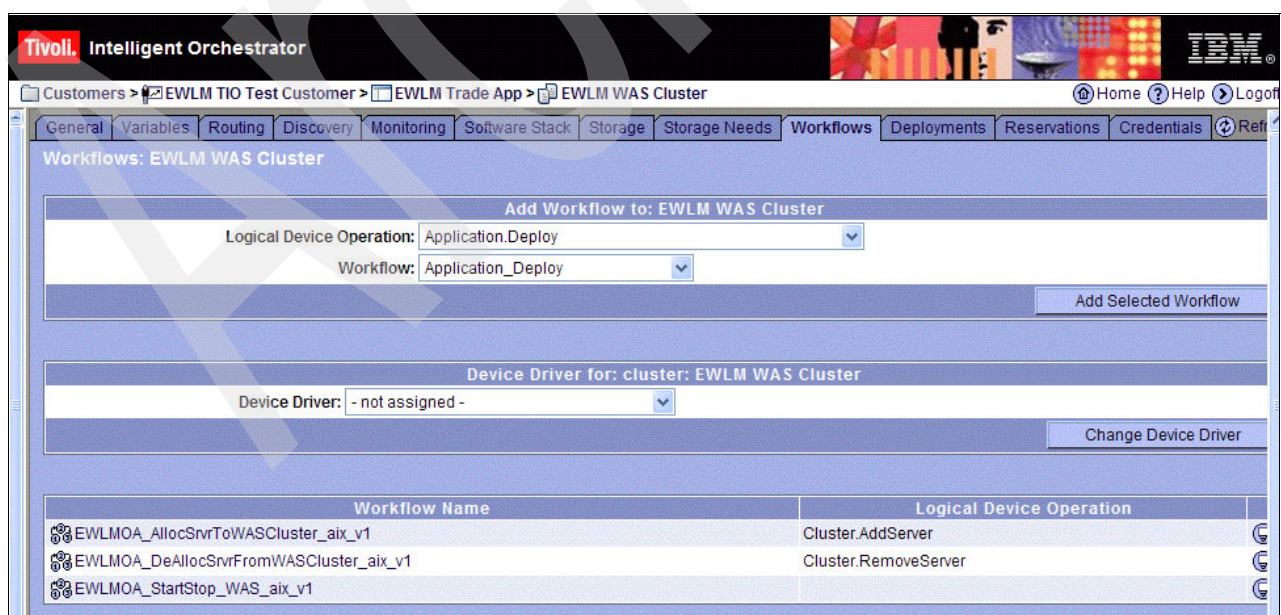


Figure 3-13 Workflows associated with Enterprise Workload Manager WebSphere cluster

If Cluster.AddServer is run, the EWLMOA\_AllocSrvToWASCluster\_aix\_v1 workflow that was defined for our environment starts. As part of this workflow, the server is added to the WebSphere Application Server cluster, and the Enterprise Workload Manager managed server is started. It also calls another workflow, called *EWLMOA\_StartStop\_WAS\_aix\_v1*, that starts the WebSphere Application Server.

## 3.4 Using objective analyzer to automate provisioning

The automation of the provisioning process of Enterprise Workload Manager and Tivoli Intelligent Orchestrator happens through the objective analyzer. At the time of this writing, the Enterprise Workload Manager Java™ Management Extensions (JMX™) APIs, which the objective analyzer invokes, can only be initiated locally. Therefore, it was necessary to install the objective analyzer driver on the same management server as our Enterprise Workload Manager domain manager.

### 3.4.1 Starting and stopping Tivoli Intelligent Orchestrator

Before you start Tivoli Intelligent Orchestrator, you must ensure that the following requirements are met:

- ▶ You have an appropriate Web browser.
- ▶ You have the fully qualified domain name (for example, hostname.domain.com) and port number for the Tivoli Intelligent ThinkDynamic Orchestrator server. (The default port number is 9080.)
- ▶ The Tivoli Intelligent Orchestrator WebSphere Application Server service is stopped.
- ▶ The LDAP server and database servers are started.
- ▶ You log in as *tioadmin* (the Tivoli Intelligent Orchestrator administrator).

To start Tivoli Intelligent Orchestrator:

1. From *tioadmin*, switch to the \$TIO\_HOME/tools directory.
2. Run the following command:

```
./tio.sh start
```

You are prompted for the WebSphere Application Server administrator user name and WebSphere Application Server administrator password. The default user name and password is *wasadmin*.

Alternatively, you can enter the following command:

```
./tio.sh wasadminID wasadminPW
```

Here, *wasadminID* is the WebSphere Application Server administrator ID for Tivoli Intelligent Orchestrator, and *wasadminPW* is the WebSphere Application Server administrator password.

3. Wait until all Tivoli Intelligent Orchestrator services, such as Tivoli Intelligent Orchestrator, Tivoli Provisioning Manager, and WebSphere Application Server V5-IBM\_TPM, start. This might take a few minutes. You should see messages similar to those in Example 3-1.

---

*Example 3-1 Tivoli Intelligent Orchestrator startup messages*

---

```
%TIO-I-START, Start TIO
TIO startup completed.
Review /opt/IBM/tivoli/thinkcontrol/logs/tio_start.log, deploymentengine_start.log,
policyengine_start.log for any startup errors.
```

---



**Note:** The `tio_start.log` shows the starting of the other services, such as the deployment engine, policy engine, and WebSphere Application Server. The contents of the log file look similar to the following example:

```
Starting WSWB Help System...
%TIO-I-CLEANCACHE, Cleaning out webapp cache...
%TIO-I-START, Start TIO
ADMU3000I: Server server1 open for e-business; process id is 16746
%TIO-I-STARTDE, Starting Deployment Engine...
%TIO-I-STARTENGINES, Starting Policy Engine...
TIO startup completed.
```

To stop Tivoli Intelligent Orchestrator, go the `$TIO_HOME/tools` directory and enter the following command:

```
./tio.sh stop wasadminID wasadminPW
```

Here, `wasadminID` is the WebSphere Application Server administrator ID for Tivoli Intelligent Orchestrator, and `wasadminPW` is the WebSphere Application Server administrator password. This command stops all Tivoli Intelligent Orchestrator services, such as the Deployment Engine, Policy Engine and WebSphere Application Server. After you run the command, you see the output in Example 3-2.

*Example 3-2 Output message after running the `./tio.sh stop` command*

---

```
This will take a few moments...
%TIO-I-STOP, stop TIO
java.lang.Exception: Timeout waiting for file
/opt/IBM/tivoli/thinkcontrol/help/eclipse/workspace/.metadata/.connection
./tio.sh: line 275: kill: (598) - No such process
./tio.sh: line 283: kill: (599) - No such process
./tio.sh: line 298: kill: (4877) - Operation not permitted
TIO shutdown completed.
Review /opt/IBM/tivoli/thinkcontrol/logs/tio_stop.log for any errors.
```

---

**Note:** Wait until you see the *TIO shutdown completed* message, which indicates that Tivoli Intelligent Orchestrator, Tivoli Provisioning Manager, and WebSphere Application Server are stopped.

### 3.4.2 Installing the objective analyzer driver

Before you install the objective analyzer driver, make sure that Tivoli Intelligent Orchestrator is stopped and follow these steps:

1. Extract the EWLMOA package to a temporary directory (`/tmpdir`). This creates a directory with several subdirectories.
2. Copy and install the driver:
  - a. Copy the `IBM-EWLMOA_version.tcdriver` file from the `/tmpdir/oa/ewlmoa` directory to the `$TIO_HOME/drivers` directory so that Tivoli Intelligent Orchestrator can locate the objective analyzer driver when it is started.
  - b. Change to the `$TIO_HOME/tools` directory.
  - c. The `tc-driver-manager.sh` command writes files into the `$TIO_HOME/lib`, `$TIO_HOME/config`, and `$TIO_HOME/repository` directories. Make sure that you have appropriate access permissions to these directories. Run the following command:

```
tc-driver-manager.sh installDriver IBM-EWLMOA_version
```

Here, *IBM-EWLMOA\_version* is the name of the tcdriver that you just copied.

**Note:** If you have to rerun the tc-driver-manager.sh command, remove the following files first because there might be some overwrite problems for them:

- ▶ \$TIO\_HOME/lib/ewlm\_interfaces.jar
- ▶ \$TIO\_HOME/lib/ewlm\_jmx.jar
- ▶ \$TIO\_HOME/lib/ewlmoa\_adapter.jar
- ▶ \$TIO\_HOME/lib/ewlmoa\_base.jar
- ▶ \$TIO\_HOME/config/wlmoaprops.xml

IBM-EWLMOA\_version.html under /doc in the .tcdriver file has information about the contents of the tcdriver package.

3. Copy the ewlm\_interfaces.jar from EWLMOA/classes to \$TIO\_HOME/drivers/lib to overwrite the \$TIO\_HOME/drivers/lib/ewlm\_interfaces.jar file.

**Note:** *EWLMOA* refers to the path where Enterprise Workload Manager domain manager is installed. On the Linux platform, by default, it refers to /opt/IBM/VE/EWLMOA.

4. Make a copy of the \$TIO\_HOME/config/log4j.prop file, such as \$TIO\_HOME/config/log4j-de.prop. Remove any sections that start with the comments #OAEWLM entries or #EWLM entries from the log4j-de.prop file.
5. Open \$TIO\_HOME/tools/run-deploymentengine.sh and change the configuration file to:  
-Dlog4j.configuration=file://\$TC\_CONFIG/log4j-de.prop.  
Then save the file.
6. Open \$TIO\_HOME/config/log4j.prop and remove any sections that start with the comments #OAEWLM entries or #EWLM entries. Then append all the entries from the /tmpdir/oa/ewlmoa/log4j.prop.update file to the \$TIO\_HOME/config/log4j.prop file.
7. Open \$TIO\_HOME/tools/run-pe.sh and add this flag:  
-Dwlmoa.props=\$TC\_CONFIG/wlmoaprops.xml.

### 3.4.3 Avoiding JMX conflicts

To avoid JMX conflicts, we moved the jmx.jar file to another directory that was not in the class path that was set up in *setupCmdLine.sh*. Therefore, we also modified the setupCmdLine.sh to exclude this file. To do this, follow these steps:

1. Make a copy of the setupCmdLine.sh file in \$WAS\_HOME/bin and name it *setupCmdLine1.sh*.
2. Edit the setupCmdLine1.sh file by changing the line starting with WAS\_EXT\_DIRS= and removing \$WAS\_HOME/lib/ext; from this line.
3. Save the setupCmdLine1.sh file.
4. Move the jmx.jar file from \$WAS\_HOME/lib to \$WAS\_HOME/lib/ext.
5. Make a backup copy of the \$TIO\_HOME/tools/engines.properties file.
6. The \$TIO\_HOME/tools/run-pe.cmd file runs \$TIO\_HOME/tools/setupCmdLine.sh. In \$TIO\_HOME/tools/setupCmdLine.sh, change the env1 setting to point to the modified file. An example is:  
env1=\$WAS\_HOME/bin/setupCmdLine1.sh
7. Save the \$TIO\_HOME/tools/setupCmdLine.sh file.

**Note:** In these steps, *\$WAS\_HOME* refers to the Tivoli Intelligent Orchestrator WebSphere Application Server home, which by default is /opt/WebSphere/AppServer on the Linux platform.

### 3.4.4 Configuring the JMX credentials

We completed the following steps to configure the JMX credentials at the Enterprise Workload Manager domain manager:

1. Set the EWLM\_ROOT environment variable. On the Linux platform, the default path to EWLMROOT is /opt/IBM/VE/EWLM.
2. Copy the CreateJMXUser.sh file to the \$EWLM\_ROOT/bin directory.
3. Run CreateJMXUser.sh as follows:

```
CreateJMXUser.sh $EWMWORKINGDIR/Interfaces/EWLMcred.db jmxuser jmxuser123
```

*\$EWMWORKINGDIR/Interfaces/EWLMcred.db* is the Enterprise Workload Manager authorization database file name, where *\$EWMWORKINGDIR* is the Enterprise Workload Manager working directory, *jmxuser* is the user name, and *jmxuser123* is the JMX password, which should be at least 10 characters in length.

4. Update the JMXUSER.policy at \$EWMWORKINGDIR /Interfaces by appending the following information:

```
grant Principal mx4j.beep.profile.sox.SoxPrincipal " jmxusername " {  
    permission javax.management.MBeanPermission "*", "*";  
};
```

Here, *jmxusername* is the user name of the JMX user was created at Enterprise Workload Manager domain manager.

### 3.4.5 Configuring JMX connect at Tivoli Intelligent Orchestrator

The objective analyzer has a properties file called wlmoaprops.xml. It contains settings, such as the Enterprise Workload Manager domain manager IP address and port, data collection parameters, and target performance. Example 3-3 shows some entries in wlmoaprops.xml.

*Example 3-3 Example entries in the wlmoaprops.xml file*

```
<wlm>  
  <domainmanager>  
    <ip>localhost</ip>  
    <port>9092</port>  
    <user>jmxuser</user>  
    <password>{am14dXN1cjEyMw==}</password>  
    <datacollectinterval>300000</datacollectinterval>  
    <datacollecttimeout>25000</datacollecttimeout>  
  </domainmanager>  
  <oa>  
    <uuidkey>ewlm.uuid</uuidkey>  
    <targetpb>0.5</targetpb>  
    <targetpiadd>0.85</targetpiadd>  
    <targetpiremove>0.50</targetpiremove>  
  </oa>  
</wlm>
```

Change the wlmoaprops.xml file in \$TIO\_HOME/config as follows:

- ▶ **jmx.ip:** The fully qualified host name or IP address of the Enterprise Workload Manager domain manager

**Note:** Because the Enterprise Workload Manager domain manager is on the same server as Tivoli Intelligent Orchestrator, jmx.ip can be a local host.

- ▶ **jmx.port:** The port used for communicating with the Enterprise Workload Manager domain manager
- ▶ **jmx.user:** The JMX user name that you created when you configured the JMX credentials
- ▶ **jmx.pwd:** The JMX password of the JMX user

### 3.4.6 Associating the objective analyzer with an application

The objective analyzer enables Tivoli Intelligent Orchestrator to make better provisioning decisions based on the performance information that it receives from Enterprise Workload Manager for the resources in the clusters for a given customer application. Therefore, it is necessary to associate the objective analyzer to the customer application by following these steps:

1. Copy the /tempdir/oa/ewlmoa/ewlmoasample.xml file into the \$TIO\_HOME/xml/EWLMOAEntries.xml file.

Tivoli Intelligent Orchestrator instantiates the objective analyzer and associates it with a Tivoli Intelligent Orchestrator-defined customer application. The file does not contain all of the Tivoli data center model definitions, such as for servers and software, that are needed.

2. Save the file as EWLMOAEntries.xml. Example 3-4 is a sample of the file.

*Example 3-4 ewlmoasample.xml file*

---

```
- <datacenter>
- <objective-analyzer-type name="EWLMOA"
  classname="com.ibm.tivoli.orchestrator.objectiveAnalyzer.ewlm.adapter.EwlmOAAAdapter"
  description="EWM Objective Analyzer">
  <objective-analyzer-parameter name="weighting" default-value="1"
description="weighting" />
  </objective-analyzer-type>
- <customer name="EWM Test Customer">
- <application name="EWM Trade App" locale="en_US" priority="10">
  <cluster name="Http Cluster" virtual-ipaddress="10.1.3.80" virtual-tcp-port="-1"
    min-servers="1" max-servers="6" pool="BladeWinPool" vlan="555"
fabric="SwitchFabric1" locale="en_US" />
  <cluster name="WAS Cluster" virtual-ipaddress="10.1.3.96" virtual-tcp-port="-1"
    min-servers="1" max-servers="6" pool="BladeWinPool" vlan="555"
fabric="SwitchFabric1" locale="en_US" />
  <cluster name="DB2 Cluster" virtual-ipaddress="10.1.7.16" virtual-tcp-port="-1"
    min-servers="1" max-servers="1" pool="BladeWinPool" vlan="555"
fabric="SwitchFabric1" locale="en_US" />
  <objective-analyzer type-name="EWLMOA" />
  </application>
</customer>
</datacenter>
```

---

3. Add the definitions of the network infrastructure and the complete customer relationship model to complete the data center model definition:
  - a. For the network infrastructure, add the definitions in the following order:
    - i. Switch fabric
    - ii. Subnet
    - iii. VLAN
    - iv. Switch

- v. Resource pools
  - vi. Servers
- b. For the customer relationship model, add the definitions in the following order:
    - i. Customers
    - ii. Application
    - iii. Application clusters and resource pool
    - iv. Servers
    - v. Software stacks
    - vi. Software products
  4. Run the following command as user tioadmin from the \$TIO\_HOME/xml directory to import the definitions into the DCM:
 

```
$TIO_HOME/tools/xmlimport.sh file://$TIO_HOME/xml/sample.xml
```

**Note:** You must specify the full directory path to sample.xml.

### 3.4.7 Configuring SSH

Before you run Enterprise Workload Manager provisioning workflows (or any workflow), you must set up SSH communication between the Tivoli Intelligent Orchestrator server and the target machines. This includes creating an SSH-based service access point in the Tivoli Intelligent Orchestrator data center model for the Tivoli Intelligent Orchestrator server and for all the target servers that might be provisioned or on which the workflows (required by Enterprise Workload Manager to retrieve server identity properties known as the UUID) might be applied.

1. Configure SSH between the Tivoli Intelligent Orchestrator server and the target servers in your DCM resource pool. For more details, refer to “Configuring OpenSSH” in *IBM Tivoli Provisioning Manager Installation Guide for Linux Version 2.1.0*, GC32-1616.
2. Set up the service access point for SSH:
  - a. Define the service access point for the target server (host) as follows:
    - i. From the Data Center Assets tab, select **Inventory** → **Servers**.
    - ii. Select the target server or servers for the credentials.
    - iii. Click the Credentials tab.
    - iv. Select **Edit** → **Add Access Point**.
    - v. Enter the name of the services access point, such as *SSH*.
    - vi. Select the type (mode or protocol used to communicate with ipv4/ssh). When you do this, it looks like “...SSH.”
    - vii. Select **Host**. This indicates that this server will act as a host.
    - viii. Select **Authentication** because communication with this host is authenticated.
    - ix. Specify the Locale, for example, *en\_us*.
    - x. Click **Save**.
  - b. Specify SSH as the default access point for device operations file transfer and issue the command as follows:
    - i. Select **SSH**.
    - ii. Select the Workflows tab.
    - iii. Select **Device Driver: SSH Service Access Point**.
    - iv. Click **Change Device Driver**.



**Note:** *SSH\_RSA\_Execute\_Command* is displayed as the workflow name. The *ServiceAccessPoint.ExecuteCommand* is displayed as the logical device operation.

- c. Define the credentials for the host as follows:
  - i. From the Data Center Assets tab, click **Inventory** → **Servers**.
  - ii. Select the target server or servers for the credentials.
  - iii. Click the Credentials tab.
  - iv. Select **SSH**.
  - v. Click the menu and select **Add RSA Credentials**.
  - vi. Enter a search key, such as *primaryx*.
  - vii. Enter a user name, such as *Administrator* or *root*.
  - viii. Leave the other entries blank.
  - ix. Click **Save**.

**Note:** For EWLMOA workflows to work for all target servers, we recommend that you keep the same value for the Search Key for all the target servers in the data center model that might be provisioned.

- d. Click **Set Default** to set *primaryx* (Figure 3-14) as the default credentials key for the host. The user that you define should be the user with administrator or root privileges for the target server.

General
Variables
Workflows
Deployments
Discovery
Monitoring
Storage Needs
Software
Software Stack
Credentials
Refresh
Edit

Service Access Points: d14v01

Name	Type	Port	Host	Domain	Context	Auth	
SSH	ipv4 / SSH	22	Yes		NOCONTEXT	Yes	
Credentials Type			Search Key		Default		
RSA			primaryx		Yes		
SCP	ipv4 / SCP	22	Yes		NOCONTEXT	Yes	

Device Operation	Description	Default Access Point
file-transfer	Transferring file	SCP
set-attribute	set system attribute	<no service points of this type configured>
execute-command	Executing command	SSH
get-attribute	Query system attribute	<no service points of this type configured>
ping	Return system status	<no service points of this type configured>

Figure 3-14 Default access point

- e. Define the service access point for the client (that is, the Tivoli Intelligent Orchestrator server):
  - i. From the Data Center Assets tab, select **Inventory** → **Servers**.
  - ii. Select the TIO server or servers to which you want to add the credentials.
  - iii. Click the Credentials tab.
  - iv. Select **Edit** → **Add Access Point**.
  - v. Enter the name of the service access point (for example, *SSH*).
  - vi. Select the same protocol as the host protocol (that is, **ipv4/ssh**).
  - vii. Clear **Host**.
  - viii. Select **Authentication**.
  - ix. Click **Save**.
  - x. Select **client-sap1** as the default access point for device operations file transfer.
  - xi. Issue command.

- f. Define the credentials for the client:
  - i. From the Data Center Assets tab, click **Inventory** → **Servers**.
  - ii. Select the TIO server or servers to which you want to add the credentials.
  - iii. Click the Credentials tab.
  - iv. Select **client-sap1**.
  - v. Click **Add RSA Credentials**.
  - vi. Enter a search key (for example, *primaryx*).
  - vii. Enter a user name, such as *tioadmin* or a user name under which Tivoli Intelligent Orchestrator was installed.
  - viii. Leave the other entries blank.
  - ix. Click **Save**.
  - x. Click **Set Default** to set *primaryx* as the default credentials key for the client.

### 3.4.8 Starting the Enterprise Workload Manager domain manager

To start Enterprise Workload Manager domain manager:

1. Change to the \$EWMROOT/bin path
2. Enter the following command:

```
./startDM.sh EWLWORKINGDIR
```

### 3.4.9 Starting the Enterprise Workload Manager Control Center

To start the Enterprise Workload Control Center, enter the following command:

```
./startWAS.sh EWLWORKINGDIR
```

### 3.4.10 Starting Tivoli Intelligent Orchestrator

To make sure that Tivoli Intelligent Orchestrator is started correctly:

1. Follow the instructions in 3.4.1, “Starting and stopping Tivoli Intelligent Orchestrator” on page 25.
2. Review the logs in \$TIO\_HOME/ logs directory, specifically:
  - tio\_start.log
  - deploymentengine\_start.log
  - policyengine\_start.log
3. Review the logs in the \$TIO\_LOGS/ directory, specifically:
  - \$TIO\_LOGS/install\_customer\_drivers.log

Check to see if the objective analyzer was installed successfully.

  - \$TIO\_LOGS/policyengine/oawlmconsole.log
  - \$TIO\_LOGS/policyengine/console.log

This is a log of the objective analyzer decision cycle.

### 3.4.11 Sample EWLMOA workflows

Calling the *Cluster.AddServer* logical device operation invokes *WLMOA\_AllocSrvToWASCluster\_aix\_v1*, which obtains the attribute of the cluster and adds the server to the cluster. Then, it calls *EWLMOA\_Assoc\_EwlmUUID\_aix\_v1* and *EWLMOA\_StartStop\_WAS\_aix\_v1*.

*EWLMOA\_Assoc\_EwlmUUID\_aix\_v1* obtains the Enterprise Workload Manager managed server identity and creates a variable for the server in the data center model. The objective analyzer uses this variable to associate the data center model definition of the server to the Enterprise Workload Manager definition of the server.

*EWLMOA\_StartStop\_WAS\_aix\_v1* either starts or stops the WebSphere Application Server, depending on the workflow that called it. If it is *EWLMOA\_AllocSrvToWASCluster\_aix\_v1*, then WebSphere is started; if it is *EWLMOA\_DeAllocSrvFromWASCluster\_aix\_v1*, then WebSphere is stopped.

Calling the *Cluster.RemoveServer* device operation invokes *EWLMOA\_DeAllocSrvFromWASCluster\_aix\_v1*, which obtains the attribute of the cluster and removes the server from the cluster. Then it calls *EWLMOA\_StartStop\_EWLM\_aix\_v1* and *EWLMOA\_StartStop\_WAS\_aix\_v1*.

### 3.4.12 Using objective analyzer workflows

The *EWLMOA\_Assoc\_EwlmUUID\_win\_v1* and *EWLMOA\_Assoc\_EwlmUUID\_aix\_v1* workflows require the service access point ID of the target server whose Enterprise Workload Manager UUID is to be retrieved to be passed as a parameter. Either of these workflows can be called in other provisioning workflows based on the operating system of the server that is allocated to a cluster.

The `$TIO_HOME/logs/engines/oawlmconsole.log` file logs the runtime messages from the objective analyzer.

### 3.4.13 Changes required in the objective analyzer workflows

For all the target servers, Tivoli Intelligent Orchestrator needs to obtain the Enterprise Workload Manager ID. This is done by using the *EWLMOA\_Assoc\_EwlmUUID\_aix\_v1* workflow. Therefore, we ran this workflow in the directory where Enterprise Workload Manager was installed. To do this, follow these steps:

1. Log in to the Tivoli Intelligent Orchestrator administrative console as *tioappadmin*:  
`http://hostname:9080/tcWebUI/`
2. Click the System configuration and Workflow Management tab (second tab).
3. For Workflow, select **EWLMOA\_Assoc\_EwlmUUID\_aix\_v1** (for the AIX platform).
4. Change the value for `wlmpath` to the Enterprise Workload Manager managed server (EWLM MS) root path of your target server. (The default is `/usr/IBM/VE/EWLMMS`.) To change the value:
  - a. Select the variable.
  - b. Edit the text.
  - c. Click **Save**.
5. Change the value for `wlmworkdir` to the Enterprise Workload Manager MS working directory of your target server.
6. Change value for `hostcredkey` to your target server credentials key, which was defined in the data center model. (This is the value that you previously set up in 3.4.7, “Configuring SSH” on page 30.)
7. Change the value for `clientcredkey` to the credentials key for your Tivoli Intelligent Orchestrator server, which was defined in the data center model. (This is the value that you previously set up in 3.4.7, “Configuring SSH” on page 30.)

8. For objective analyzer workflows to work for all target servers, we recommend that you keep the same value as Search Key (for the RSA credentials) for all the target servers in the data center model that might be provisioned. If this is not the case, make sure that the RSA credentials created for target servers and the Tivoli Intelligent Orchestrator server are set as the default. Also in steps 6 and 7, either leave the `hostcredkey` and `clientcredkey` variable values blank or enter the value default for both.
9. Select **Compile** → **Compile** to save and verify that the workflow has no syntax errors.
10. Repeat steps 4 through 9 for the `EWLMOA_Assoc_EwlmUUID_win_v1` workflow (for the target Windows platform).

### 3.4.14 Sample policyengine console.log

From an excerpt of the console log in the `$TIO_LOGS/policyengine/` directory in Example 3-5, you can see that a decision cycle took place for three clusters with cluster IDs: 2941 (HTTP), 2942 (WebSphere Application Server), and 2943 (DB2). It was determined that cluster ID 2942 had a probability of breaching its goals as denoted by `initialProbabilityOfBreach=1.0`. It was determined that cluster 2942 had two servers, but it required three servers as denoted by `action=Add Server numServers=1.0`. It was determined that the other clusters had one server and required one server as denoted by `Action='0'`. Therefore, they did not need additional servers.

*Example 3-5 Excerpt of the console log in the \$TIO\_LOGS/policyengine/ directory*

---

```

date time,663 DEBUG [Thread-8] ( ??:?) adapter.EwlmPlot: Probe 15: P(B): 0.5
. . .
date time,613 DEBUG [Thread-8] ( ??:?) fitnessfunction.SuggestedServers: clusterID=2943
numServers=1 neededServers=1.0 initialProbabilityOfBreach=0.5. . .
date time,663 DEBUG [Thread-8] ( ??:?) adapter.EwlmPlot: Probe 5:
getProbabilityOfBreach(int, int): numServers: 1 time: 0
. . .
date time,665 DEBUG [Thread-8] ( ??:?) fitnessfunction.SuggestedServers: clusterID=2941
numServers=1 neededServers=1.0 initialProbabilityOfBreach=0.5
date time,666 DEBUG [Thread-8] ( ??:?) adapter.EwlmPlot: Probe 5:
getProbabilityOfBreach(int, int): numServers: 2 time: 0
date time,666 DEBUG [Thread-8] ( ??:?) adapter.EwlmPlot: Probe 15: P(B): 1.0
date time,666 DEBUG [Thread-8] ( ??:?) adapter.EwlmPlot: Probe 5:
getProbabilityOfBreach(int, int): numServers: 3 time: 0
. . .
date time,668 DEBUG [Thread-8] ( ??:?) adapter.EwlmPlot: Probe 15: P(B): 1.0
date time,669 DEBUG [Thread-8] ( ??:?) fitnessfunction.SuggestedServers: clusterID=2942
numServers=2 neededServers=3.0 initialProbabilityOfBreach=1.0
date time,678 DEBUG [Thread-8] ( ??:?) optimizer.PoolOptimizerBean: Currently 1 idle
servers in pool ID 2037
date time,713 DEBUG [Thread-8] ( ??:?) adapter.EwlmPlot: Probe 5:
getProbabilityOfBreach(int, int): numServers: 1 time: 0
. . .
date time,720 DEBUG [Thread-8] ( ??:?) adapter.EwlmPlot: Probe 15: P(B): 1.0
date time,725 DEBUG [Thread-8] ( ??:?) optimizer.PoolOptimizerBean: traversed 2 nodes
date time,726 DEBUG [Thread-8] ( ??:?) optimizer.PoolOptimizerBean: === BEST NODE ===
date time,726 DEBUG [Thread-8] ( ??:?) optimizer.PoolOptimizerBean: Fitness value=-0.1
NetInfrastructureChange (clusterId, delta): (2942,1)
. . .
date time,741 DEBUG [Thread-8] ( ??:?) adapter.EwlmOAAAdapter: endDecisionCycle(2943):
Invoking ewlmOA.endDecisionCycle()
date time,742 INFO [Thread-8] ( ??:?) resourcebroker.ResourceBrokerBean: Optimizer :
clusterId=2942 action=Add Server numServers=1.0
date time,743 DEBUG [Thread-8] ( ??:?) reservation.ReservationAgent:
filterRecommendations() 3

```

```

date time,744 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: optneeds[0]=id =
`2943';Action = '0';numberOfServers = '0.0';clusterName = 'EWLM DB2 Cluster';applicationId
= '2939';clusterId = '2943';
date time,744 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: appResInfo=null
date time,745 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: optneeds[1]=id =
`2941';Action = '0';numberOfServers = '0.0';clusterName = 'EWLM HTTP Cluster';applicationId
= '2939';clusterId = '2941';
date time,745 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: appResInfo=null
date time,745 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: optneeds[2]=id =
`2942';Action = '1';numberOfServers = '1.0';clusterName = 'EWLM WAS Cluster';applicationId
= '2939';clusterId = '2942';
. . .

```

---

When the workload was reduced, it was determined the third server was no longer needed and should be removed from cluster 2942, as noted by *Action='1';numberOfServers = '1.0'* (Example 3-6).

#### *Example 3-6 Third server not needed by cluster 2942*

```

. . .
date time,369 DEBUG [Thread-8] ( ??) adapter.EwlmOAAAdapter: endDecisionCycle was called
for Cluster: 2941
date time,370 DEBUG [Thread-8] ( ??) adapter.EwlmOAAAdapter: endDecisionCycle(2941):
Cluster was in started mode
date time,370 DEBUG [Thread-8] ( ??) adapter.EwlmOAAAdapter: endDecisionCycle was called
for Cluster: 2942
date time,371 DEBUG [Thread-8] ( ??) adapter.EwlmOAAAdapter: endDecisionCycle(2942):
Cluster was in started mode
date time,371 DEBUG [Thread-8] ( ??) adapter.EwlmOAAAdapter: endDecisionCycle was called
for Cluster: 2943
date time,372 DEBUG [Thread-8] ( ??) adapter.EwlmOAAAdapter: endDecisionCycle(2943):
Cluster was in started mode
date time,372 DEBUG [Thread-8] ( ??) adapter.EwlmOAAAdapter: endDecisionCycle(2943):
Invoking ewlmOA.endDecisionCycle()
date time,375 INFO [Thread-8] ( ??) resourcebroker.ResourceBrokerBean: Optimizer :
clusterId=2942 action=Remove Server numServers=1.0
date time,375 DEBUG [Thread-8] ( ??) reservation.ReservationAgent:
filterRecommendations() 3
date time,376 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: optneeds[0]=id =
`2943';Action = '0';numberOfServers = '0.0';clusterName = 'EWLM DB2 Cluster';applicationId
= '2939';clusterId = '2943';
date time,377 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: appResInfo=null
date time,377 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: optneeds[1]=id =
`2941';Action = '0';numberOfServers = '0.0';clusterName = 'EWLM HTTP Cluster';applicationId
= '2939';clusterId = '2941';
date time,378 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: appResInfo=null
date time,378 DEBUG [Thread-8] ( ??) reservation.ReservationAgent: optneeds[2]=id =
`2942';Action = '1';numberOfServers = '1.0';clusterName = 'EWLM WAS Cluster';applicationId
= '2939';clusterId = '2942';

```

---

### 3.4.15 Objective analyzer logs

The objective analyzer uses the performance information that it obtains from the Enterprise Workload Manager domain manager on the servers and the service classes. It maps the information to the application clusters defined for the Enterprise Workload Manager TIO Test Customer. Some of this information is in the objective analyzer log (Example 3-7).

#### *Example 3-7 Objective analyzer log*

```

compute.OAWlmComputeMgr: clusterId: 2942 minServer: 2 maxServer: 4
compute.OAWlmComputeMgr: Found and mapped: b70n06:b70n06:
compute.OAWlmComputeMgr: b70n06 Utilization: 26.07359

```

```

common.WLMReportDataAdder: - <WLMDataApplEnvServer name="b70n06" index="2">
<WLMDataRtStats avgRt="2224" avgacttime="2224" successcnt="24" failedCnt="0" abortedcnt="0"
unknowncnt="0" totRt="53394" totblocked="0">
</WLMDataRtStats>
<WLMDataResourceStats cputime="2483544" resptime="0" queueTime="0" pagefault="0"
cpuusing="10" cpudelay="0" pagedelay="0" idlesamples="0" othersamples="637">
</WLMDataResourceStats>
</WLMDataApplEnvServer>
. . .
compute.OAWlmComputeMgr: SC_Trade3_portfolio
compute.OAWlmComputeMgr: AveRT = 0 PI: 0.4 GOAL: 140

```

---

Note that clusterID 2942 is the Enterprise Workload Manager WebSphere Application Server cluster.

### 3.4.16 Objective analyzer workflows

Based on the information that the objective analyzer receives, as explained in Section 2.3, “Objective analyzer” on page 10 (see Example 3-7), a new server can either be provisioned into a particular cluster or deprovisioned from a particular cluster. If the objective analyzer decides that a server must be provisioned into an application cluster, it sends a recommendation to Tivoli Intelligent Orchestrator, which arbitrates these incoming requests and determines the appropriate action. If it accepts the recommendation, Cluster.AddServer is called. Therefore, we had to add a workflow to the Cluster.AddServer in each of the application clusters for provisioning a server and start Enterprise Workload Manager and the needed middleware product.

We wanted our workflow to include:

- ▶ Obtaining the Enterprise Workload Manager UUID and associating it with the data center model server ID so that the information from Enterprise Workload Manager can be associated with Tivoli Intelligent Orchestrator customer relationship definitions
- ▶ Starting the Enterprise Workload Manager managed server code on the provisioned server
- ▶ Starting either HTTP or WebSphere Application Server, based on which application cluster into which it was to be provisioned

If the objective analyzer decides that a server needs to be deprovisioned, it sends a recommendation to Tivoli Intelligent Orchestrator. If Tivoli Intelligent Orchestrator accepts the recommendation, it triggers Tivoli Provisioning Manager (the deployment engine) to run Cluster.RemoveServer. Therefore, we had to add a workflow to Cluster.RemoveServer in each of the application clusters for deprovisioning a server and stopping Enterprise Workload Manager and the middleware product.

We wanted our workflow to include:

- ▶ Removing the server from the application cluster
- ▶ Stopping the Enterprise Workload Manager managed server code
- ▶ Stopping HTTP or WebSphere Application Server, based on the application cluster

Figure 3-15 on page 37 shows three workflows associated with the WebSphere Application Server cluster. EWLMOA\_AllocSrvToWASCluster\_aix\_v1 makes the association of how Enterprise Workload Manager and Tivoli Intelligent Orchestrator know the server. It then starts the Enterprise Workload Manager managed server code and WebSphere Application Server. EWLMOA\_DeAllocSrvFromWASCluster\_aix\_v1 takes the server out of the application cluster and stops the managed server and the WebSphere Application Server. Both workflows call EWLMOA\_StartStop\_WAS\_aix\_v1.



```

- <cluster name="EWLM WAS Cluster" locale="en_US" managed="true" virtual-tcp-port="-1" min-servers="2" max-
  servers="4" in-maintenance="false" pool="SparePool" vlan="2" fabric="DesignCenter_Fabric" tier="1">
+ <server name="b70n06" locale="en_US" is-device-model="SSH Service Access Point" in-maintenance="false"
  failed="false" tcp-port="0">
+ <server name="b70n03" locale="en_US" is-device-model="SSH Service Access Point" in-maintenance="false"
  failed="false" tcp-port="0">
  <use-workflow workflow-id="EWLMOA_AllocSrvrToWASCluster_aix_v1" />
  <use-workflow workflow-id="EWLMOA_DeAllocSrvrFromWASCluster_aix_v1" />
  <use-workflow workflow-id="EWLMOA_StartStop_WAS_aix_v1" />
  <property component="RESOURCE_BROKER" name="OperationsMode-current" value="automatic" />
</cluster>

```

Figure 3-15 Workflow association with a cluster

### 3.4.17 The provisioning test

To run our workload, we used LoadRunner to generate the trade3 scripts to drive the work to the HTTP server, which forwarded the requests to the two WebSphere Application Server profiles. The provisioning was automatic, but we followed the progression with Enterprise Workload Manager Control Center. We looked at the views (for examples, see Figure 3-2 on page 17 through Figure 3-5 on page 18) and everything seemed fine. We then added more users to increase the workload. We saw from the Managed Server that the CPU was increasing. The Service Class view showed that the PIs were increasing and we were starting to miss our goals (Figure 3-16).

Select	Service class	PI	Importance	Performance	Goal
<input type="radio"/>	SC_Calc_batch	0.00	Medium	Slowest velocity	Slowest velocity
<input type="radio"/>	SC_InternetExplorer	0.00	Medium	Slowest velocity	Fast velocity
<input type="radio"/>	SC_Plants_default	0.00	Medium	00.000 average	00.100 average
<input type="radio"/>	SC_Plants_login	0.00	Medium	0% in 00.150	90% in 00.150
<input type="radio"/>	SC_Plants_shopping	0.00	Medium	0% in 00.200	80% in 00.200
<input type="radio"/>	SC_Trade3_account	1.45	Medium	00.234 average	00.150 average
<input type="radio"/>	SC_Trade3_default	0.88	Medium	00.168 average	00.180 average
<input type="radio"/>	SC_Trade3_portfolio	1.05	Medium	00.158 average	00.140 average
<input type="radio"/>	SC_Trade3_profile	0.32	Medium	00.032 average	00.100 average
<input type="radio"/>	SC_Trade3_quotes	1.01	Medium	00.153 average	00.140 average
<input type="radio"/>	SC_Trade3_shopping	0.50	Medium	80% in 00.160	63% in 00.160
<input type="radio"/>	EWLM Service Class	<None>	<None>	<None>	Discretionary
<input type="radio"/>	SC_Plants_general	<None>	<None>	<None>	Discretionary
<input type="radio"/>	SC_Trade3_general	<None>	<None>	<None>	Discretionary

Page 1 of 1      Total: 14   Filtered: 14   Displayed: 14

Figure 3-16 PIs approaching 1.0

We looked at the Tivoli Intelligent Orchestrator administrative console and checked the Enterprise Workload Manager WebSphere Application Server cluster deployments. We saw that EWLMOA\_AllocSrvrToWASCluster\_aix\_v1 deployed successfully (Figure 3-17 on page 38).

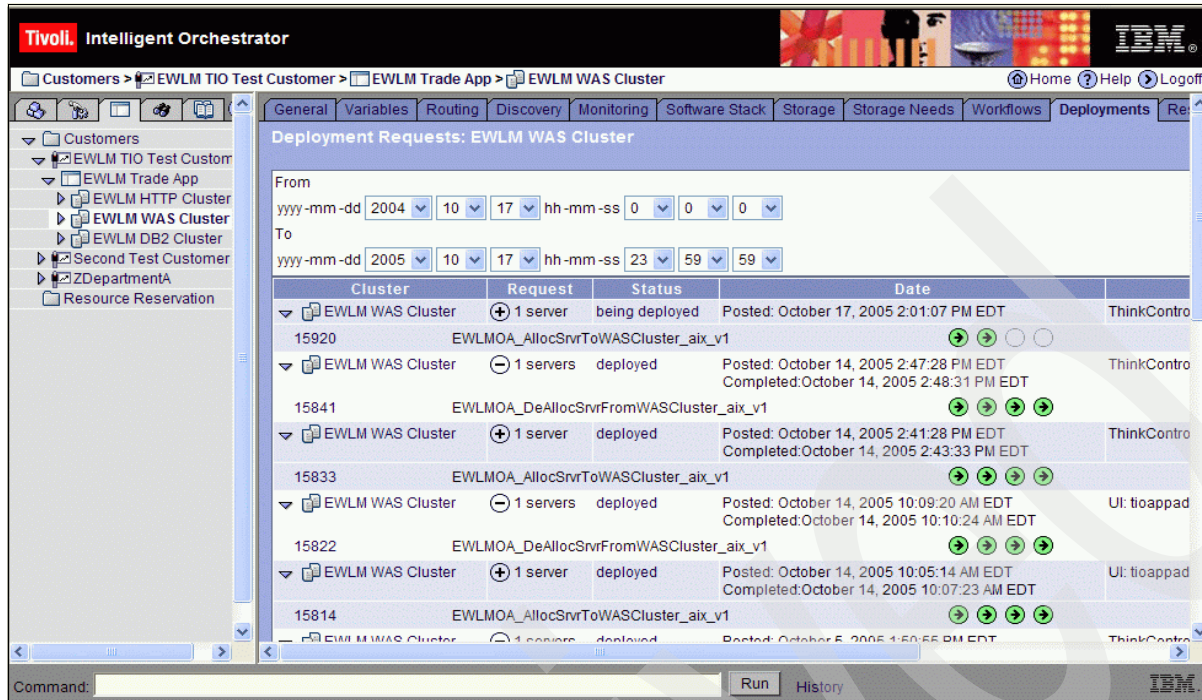


Figure 3-17 Enterprise Workload Manager WebSphere Application Server cluster provisioning

Next, we looked at the Managed Server view and saw that d14v01 was active (Figure 3-18).

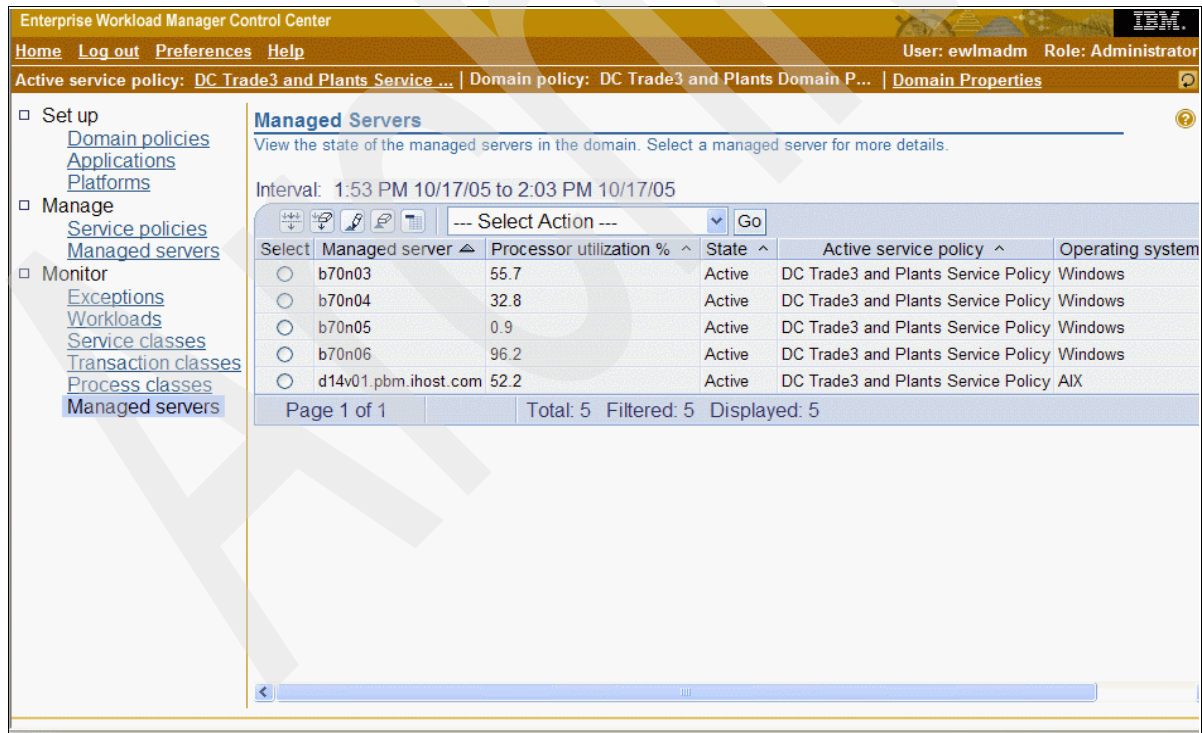


Figure 3-18 d14v01 server active

Next, we looked at the Enterprise Workload Manager Application topology - Service Class view and saw that the HTTP server was routing transactions to three WebSphere Application Server profiles, including d14v01 (Figure 3-19 on page 39).



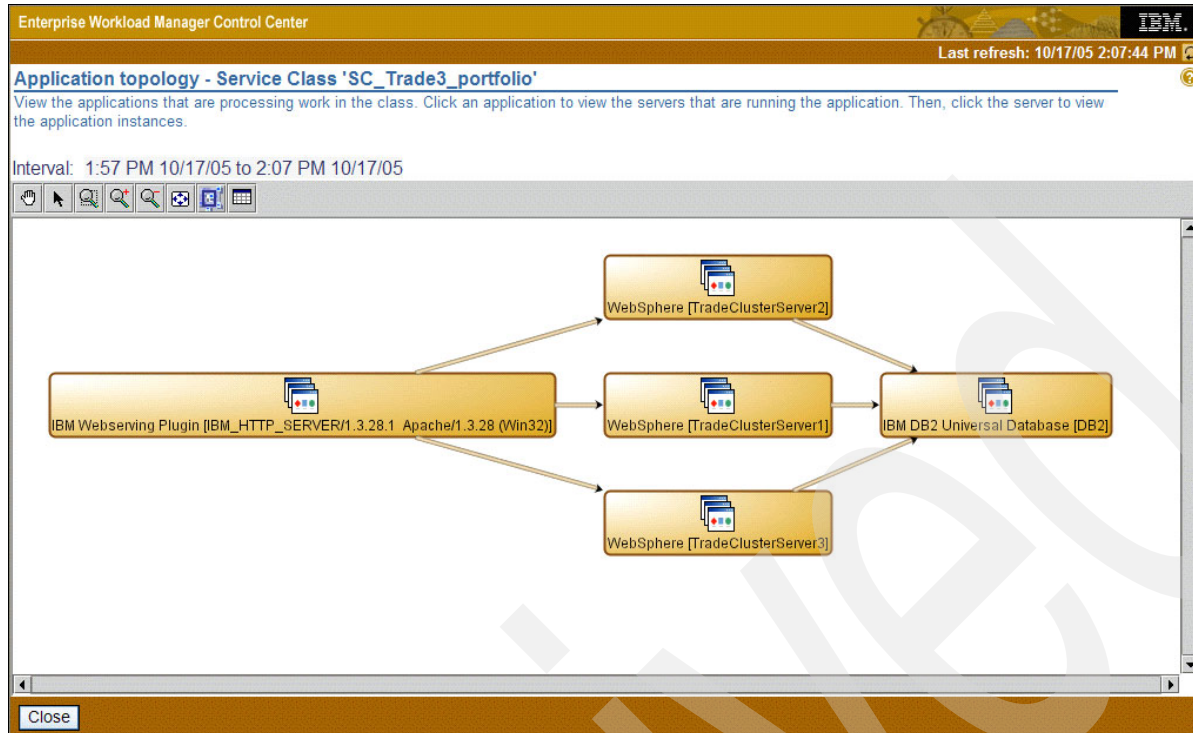


Figure 3-19 Application topology view after provisioning

Then from the Service Classes view, we saw PI numbers decreasing, and our goals were once again being met (Figure 3-20).

Enterprise Workload Manager Control Center

Home Log out Preferences Help User: ewlmdm Role: Administrator

Active service policy: DC Trade3 and Plants Service... | Domain policy: DC Trade3 and Plants Domain P... | 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.

Interval: 1:53 PM 10/17/05 to 2:03 PM 10/17/05

Select	Service class	PI	Importance	Performance	Goal
<input type="radio"/>	SC_Calc_batch	0.00	Medium	Slowest velocity	Slowest velocity
<input type="radio"/>	SC_InternetExplorer	0.00	Medium	Slowest velocity	Fast velocity
<input type="radio"/>	SC_Plants_default	0.00	Medium	00.000 average	00.100 average
<input type="radio"/>	SC_Plants_login	0.00	Medium	0% in 00.150	90% in 00.150
<input type="radio"/>	SC_Plants_shopping	0.00	Medium	0% in 00.200	80% in 00.200
<input type="radio"/>	SC_Trade3_account	0.93	Medium	00.139 average	00.150 average
<input type="radio"/>	SC_Trade3_default	0.81	Medium	00.145 average	00.180 average
<input type="radio"/>	SC_Trade3_portfolio	0.94	Medium	00.131 average	00.140 average
<input type="radio"/>	SC_Trade3_profile	0.32	Medium	00.032 average	00.100 average
<input type="radio"/>	SC_Trade3_quotes	0.96	Medium	00.134 average	00.140 average
<input type="radio"/>	SC_Trade3_shopping	0.50	Medium	85% in 00.160	63% in 00.160
<input type="radio"/>	EWLM Service Class	<None>	<None>	<None>	Discretionary
<input type="radio"/>	SC_Plants_general	<None>	<None>	<None>	Discretionary
<input type="radio"/>	SC_Trade3_general	<None>	<None>	<None>	Discretionary

Page 1 of 1 Total: 14 Filtered: 14 Displayed: 14

Figure 3-20 After provisioning

## 3.5 Summary

In this chapter, we showed how the integration of Enterprise Workload Manager and Tivoli Intelligent Orchestrator provisioned our systems and application middleware proficiently and automatically. We were able to continue to achieve our Enterprise Workload Manager-defined business goals, regardless of varying transaction rates. While achieving our business goals, we made efficient use of the system resources in our environment.

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this IBM Redpaper.

## IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 42. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *IBM Enterprise Workload Manager V2.1*, SG24-6785
- ▶ *Enterprise Workload Manager - Interpreting Control Center Performance Reports*, REDP-3963

## Other publications

The publication *IBM Tivoli Provisioning Manager Installation Guide for Linux Version 2.1.0*, GC32-1616, is also relevant as further information source.

## Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ IBM eServer Software Information Center  
[http://publib.boulder.ibm.com/infocenter/eserver/v1r1/en\\_US/index.htm?info/icmain.htm](http://publib.boulder.ibm.com/infocenter/eserver/v1r1/en_US/index.htm?info/icmain.htm)
- ▶ IBM Tivoli Software Information Center  
<http://publib.boulder.ibm.com/tividd/td/IBMTivoliIntelligentThinkDynamicOrchestrator2.1.html>
- ▶ IBM Tivoli Intelligent Orchestrator  
<http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliIntelligentOrchestrator.html>

## 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](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)





# Automated Provisioning using Tivoli Intelligent Orchestrator and Enterprise Workload Manager



**Redpaper**

**Gain an overview of  
the provisioning  
concepts**

**Learn about the  
provisioning solution  
components**

**Follow an example of  
a provisioning  
solution**

IBM Tivoli Provisioning Manager dynamically provisions, and deprovisions, server instances. It does so based on the end-to-end service-level achievement that is provided by the IBM Enterprise Workload Manager and coordinated by IBM Tivoli Intelligent Orchestrator.

This IBM Redpaper explains how the integration of Enterprise Workload Manager and Tivoli Intelligent Orchestrator can proficiently and automatically provision systems and application middleware. In doing so, we demonstrate how business goals continue to be achieved, regardless of varying transaction rates, with efficient use of the system resources in an environment. The paper begins with an overview of the products followed by the Enterprise Workload Manager and Tivoli Intelligent Orchestrator provisioning solution components. Then it describes our test environment and the installation and customization that we performed.

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)