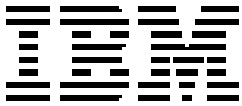
IBM Tivoli Intelligent Orchestrator and
IBM Tivoli Provisioning Manager

Overview Guide

Version 1.1.1

SC32-1419-01

IBM Tivoli Intelligent Orchestrator



Overview Guide

Version 1.1.1

SC32-1419-01

Note
Before using this information and the product it supports, read the information under "Notices".

Second Edition (December 2003)

Contents

Figures

Tables

Preface

This guide provides an architectural overview of the IBM Tivoli Intelligent Orchestrator, describes its main components and their interaction, and provides the guidelines for authoring workflow standards. Also provided is a reference to the database tables used in mapping information to the Tivoli Intelligent Orchestrator's Data Center Model database.

Who should read this guide

This guide is intended for anyone who want to learn more about the Tivoli Intelligent Orchestrator.

It is assumed that you have a certain degree of familiarity with a number of data center-specific terms. Familiarity with n-tier applications architecture and data center infrastructure may be helpful but is not required.

What this guide contains

This guide contains the following chapters:

- Chapter 1, "Introduction to the IBM Tivoli Intelligent Orchestrator", describes the context that triggered the inception of the Tivoli Intelligent Orchestrator, and provides general information on the system architecture and functionality.
- Chapter 2, "System architecture", provides an architectural overview of the Tivoli Intelligent Orchestrator, provides details for each system component, and describes the interactions between all of the system components.

Publications

This section lists publications in the Tivoli Intelligent Orchestrator library and related documents. It also describes how to access Tivoli publications online and how to order Tivoli publications.

Tivoli Intelligent Orchestrator library

The following manuals are available in the Tivoli Intelligent Orchestrator library for version 1.1.1:

- This manual, *IBM Tivoli Intelligent Orchestrator: Overview Guide* provides an architectural overview of the Tivoli Intelligent Orchestrator, describes its main components and their interaction.
- The *IBM Tivoli Intelligent Orchestrator: Operator's Guide*, contains a general overview of the architecture and functionality of the Tivoli Intelligent Orchestrator and describes the procedures to configure and operate the system.
- The *IBM Tivoli Intelligent Orchestrator: Installation Guide* provides all the necessary information to install the Tivoli Intelligent Orchestrator and its third-party components on Windows 2000, AIX, Solaris and RedHat Linux.

Accessing publications online

IBM posts the *Release Notes* for the IBM Tivoli Intelligent Orchestrator product along with publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli Software Information Center Web site. The Tivoli Software Information Center is located by using the following Web address:

http://www.ibm.com/software/tivoli/library/

Click the Product manuals link to access the Tivoli Software Information Center.

Click the IBM Tivoli Intelligent Orchestrator link to access the product library.

Note: If you print PDF documents on other than letter-sized paper, select the **Fit to page** check box in the Adobe Acrobat Print dialog. This option is available when you click **File —> Print**. **Fit to page** ensures that the full dimensions of a letter-sized page print on the paper that you are using.

Contacting software support

If you have a problem with any Tivoli product, refer to the following IBM Software Support Web site:

http:\\www.ibm.com\software\sysmgmt\products\support\

If you want to contact software support, see the IBM Software Support Guide at the following Web site:

http:\\techsupport.services.ibm.com\guides\handbook.html

The guide provides information about how to contact IBM Software Support, depending on the severity of your problem, and the following information:

- Registration and eligibility
- Telephone numbers and e-mail addresses, depending on the country in which you are located
- Information you must have before contacting IBM Software Support

Conventions used in this book

This guide uses the following typeface conventions:

Bold

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as Tip:, and Operating system considerations:)
- Column headings in a table
- Keywords and parameters in text

Italic

- Citations (titles of books, diskettes, and CDs)
- Words defined in text
- Emphasis of words (words as words)

- Letters as letters
- New terms in text
- Variables and values you must provide

`Monospace`

- Examples and code examples
- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

<text>
Indicates a variable in a path name. For example, in the path *<INSTALL-DIR>*`\tlm`, `INSTALL-DIR` depends on the location where you have installed the component, while `\tlm` is constant.

Operating system-dependent variables and paths

This guide uses the Windows convention for specifying environment variables and for directory notation.

When using the UNIX command line, replace *%variable%* with *$variable* for environment variables and replace each backslash (\) with a forward slash (/) in directory paths.

Note: If you are using the bash shell on a Windows® system, you can use the UNIX® conventions.

Chapter 1. Introduction to the IBM Tivoli Intelligent Orchestrator

This chapter describes the context that triggered the inception of the IBM Tivoli Intelligent Orchestrator, and provides general information on the Tivoli Intelligent

Orchestrator's architecture and main components. The following information is included:

- Overview
- General architecture
- Application environments
- Data center infrastructure

Overview

The increased use of information, as well as the increased need of technology to organize and take advantage of this information led to an increase in the demand on data centers. As a result, data centers have encountered problems with managing resources and providing appropriate levels of service for hosted applications.

Data centers host business applications according to certain expected execution service levels and based on various factors, such as operational responsiveness and application performance, availability, and security. These expectations are often satisfied via *isolation* and *over-provisioning*.

- *Isolation* means separating unrelated applications, and dedicating each application to a certain execution environment consisting of network and server infrastructure. The goal is to ensure that high application demand, faults, or security breaches do not affect the performance, availability, and security of another application.
- *Over-provisioning* means oversupplying server power in order to meet anticipated peak application demands, and prevent poor response times in case the application encounters unexpected demands.

When isolation is used, and each application is over-provisioned within each isolated application environment, the result is a "trapped" capacity, that cannot be used by other applications during times of high demand. The use of isolation and over-provisioning to meet expected service levels results in a low aggregated resource utilization and optimization.

The Tivoli Intelligent Orchestrator is an automated resource management solution for corporate and Internet data centers. The system is the first and only robust software solution that creates a direct, real-time correlation between application specific service level commitments and the computing resources required to meet these commitments. The Tivoli Intelligent Orchestrator proactively configures resources among applications in a multi-application environment to balance end-user traffic demands, excess capacity, and the contractual obligations of service level agreements. Using an adaptive control technology, the system accurately predicts capacity fluctuations, and facilitates dynamic infrastructure reallocation.

The Tivoli Intelligent Orchestrator offers a powerful system that can:

- Gather information about the performance of all your application clusters and build a workload model that can predict resource requirements going forward

- Manage resources across all your application clusters to optimize business-aligned service level delivery
- Automate the deployment of the optimal computing resources to each application environment

The Tivoli Intelligent Orchestrator automates the management of data center resources, allowing automatic policy-based management of computing resources and service level objectives by automating three key data center processes: infrastructure provisioning, capacity management, and service level management.

The Tivoli Intelligent Orchestrator enables you to create, customize, and store for further use a large variety of workflows that automate all data center processes, and makes it possible to build a powerful library of processes that can be assembled to meet any data center process requirement. The Tivoli Intelligent Orchestrator's workflows automate various data center processes, from configuring and allocating servers, to installing, configuring, and patching software, and can be either large and complex or as simple as a single command.

Tivoli Intelligent Orchestrator requires an abstraction of the logical structure of an application, as well as the physical application deployment. The same application is deployed into different environments, such as testing, staging, or production environments. Application software, as well as installing, configuring, patching software are very complex. By modeling all the software dependencies and identifying software device drivers, the relationship between workflows and software products is very much simplified. Deployment Engine workflows become device drivers as part of Tivoli Intelligent Orchestrator's support for various hardware and software products. The system extends the existing Data Center Model and Deployment Engine components, enabling a new way of viewing workflows.

The Provisioning Manager

Within the IBM® Tivoli® Intelligent Orchestrator, the IBM® Tivoli® Provisioning Manager is a stand-alone product that can be purchased separately, based on your data center business needs. The Provisioning Manager automates the manual provisioning process of infrastructure deployment. It captures and automates the execution of your best practices for managing data center resources by building a framework to compile and then repeatedly execute these best practices consistently and efficiently. The Provisioning Manager comes with predefined best practices for standard products from all major infrastructure vendors.

General architecture

Tivoli Intelligent Orchestrator's architecture includes the following main components:

Deployment Engine
: This component is responsible for the creation, the storage, and the execution of repeatable workflows that automate the server configuration and allocation in the system. A workflow can represent either an entire reconfiguration process affecting multiple servers, or a single step in a larger reconfiguration process.

Data Center Model
: This component includes a representation of all of the *physical* and *logical assets* under Tivoli Intelligent Orchestrator's management, such as servers, switches, load balancers, application software, VLANs, security policies, service level

agreements, etc. It keeps track of the data center hardware and associated allocations to customer sites.

Data Acquisition Engine
This component is responsible for acquiring and pre-processing performance data from each managed application environment. Data is captured from the application, operating system, and infrastructure layers. This component uses a subscribing mechanism to distribute signals to other components of the Tivoli Intelligent Orchestrator, and performs filtering of raw signals.

Application Controller
An instance of the Application Controller is created for each application environment under management. Based on the application's workload model and predictions, as well as on real-time performance data, this component determines the resource requirements of the application.

Global Resource Manager
This component receives requirements for servers or network devices from all the application controllers, and manages the overall optimization. It has two primary responsibilities: makes optimal resource allocation decisions, and ensures a stable control over the application infrastructure. Considering the different server requirements for each application environment, it determines where the servers are to be allocated.

Management Interface
This component provides an overview of the state of all *physical* and *logical assets* in the data center infrastructure, offering information about the servers and their allocation, and generating configurations and allocations. It can also be used to create application environments. It includes two user interfaces: a *web-based interface*, and a *command-line interface*.

Note: For additional information on the Tivoli Intelligent Orchestrator's general architecture, refer to the next chapter, "System architecture".

Application environments

Tivoli Intelligent Orchestrator's management system proactively configures servers and network devices in a multi-application environment, so as to balance the demand for computing resources, excess computing resources, and predetermined levels of service for each application. Each application managed by the system is part of an application environment. Based on a service level agreement, each application environment has a predetermined level of service that must be met. Tivoli Intelligent Orchestrator has an all-around control over the application environments. Each application environment includes the following layers:

Networking infrastructure layer

This layer includes the connections between servers in the application environment and servers outside the application environment, and describes the way the application environments are interconnected.
It contains various networking infrastructure items, such as switches, routers, load balancers, and firewalls.

Server infrastructure layer

This layer contains all of the servers allocated to the current application environment, based on which both the application layer and the operating system layer function.

Operating system layer

This layer provides a base operating level for the server infrastructure layer. It acts as an intermediary between the application layer and the server infrastructure layer, assisting the application layer in performing various functions. The operating system layer is based on a Microsoft Windows™, Unix, or Linux operating system.

Application layer

This layer contains the application hosted by the data center infrastructure and managed by Tivoli Intelligent Orchestrator, and defines the functionality of the application environment. The available application types are:

Type	Description
Web	Processes incoming http requests for applications.
Application	Provides business transaction functionality.
Database	Ensures the interface with the database, managing information via creating, reading, updating, or removing records.

System administrators can create application environments by using the user interface that communicates with all Tivoli Intelligent Orchestrator's components.
Alternately, the Deployment Engine that communicates with the data center infrastructure, can automatically create an application environment.

Data center infrastructure

The data center infrastructure contains computing resources that are configured and managed by Tivoli Intelligent Orchestrator, such as:

- **Servers and network devices**, from the server infrastructure and networking infrastructure layers of the application environment.
- **External interfaces**, that ensure the interaction between Tivoli Intelligent Orchestrator and the external systems that provide value-added services, such as other operational support systems.

Servers and network devices

The server infrastructure layer includes all of the servers allocated to the current application environment. For management purposes, these servers are pooled. The unallocated servers are grouped, to allow them to be allocated together to an application environment. An application environment can contain multiple groups of servers.

The network devices in the networking infrastructure layer include network switches, routers, firewalls, load balancers, etc. When switches are present in the data center infrastructure, virtual local area networks (VLANs) are possible, and they are controlled and configured by the Tivoli Intelligent Orchestrator.

External interfaces

An external interface in the data center infrastructure ensures the interaction between Tivoli Intelligent Orchestrator and external systems that provide other value-added services. For example, the external interface can have a billing interface, connected to a billing system, for providing billing functions.

The billing interface is invoked by the Deployment Engine, and provides all the resource allocation information required for billing purposes. For example, the billing interface provides information on how many overflow servers have been allocated to a client, and for how long.

Other external systems that have an interface with Tivoli Intelligent Orchestrator's external interface include various other operational support systems, such as system management applications, content management applications, fault management applications, and customer portals.

Chapter 2. System architecture

Tivoli Intelligent Orchestrator is a professional data center infrastructure management system that proactively configures servers and network devices in a multi-application environment, so as to balance end-user traffic demands, excess capacity, and the predetermined levels of service for each application. Each application managed by the system is part of an application environment. Based on a service level agreement, each application environment has a predetermined level of service that must be met. Tivoli Intelligent Orchestrator has an all-around control over the application environments.

Tivoli Intelligent Orchestrator's architecture includes the following main components:

- Deployment Engine
- Data Center Model
- Data Acquisition Engine
- Application Controller
- Global Resource Manager
- Management Interface

The figure below illustrates Tivoli Intelligent Orchestrator's general architecture, and shows the interaction between the main components of the system.

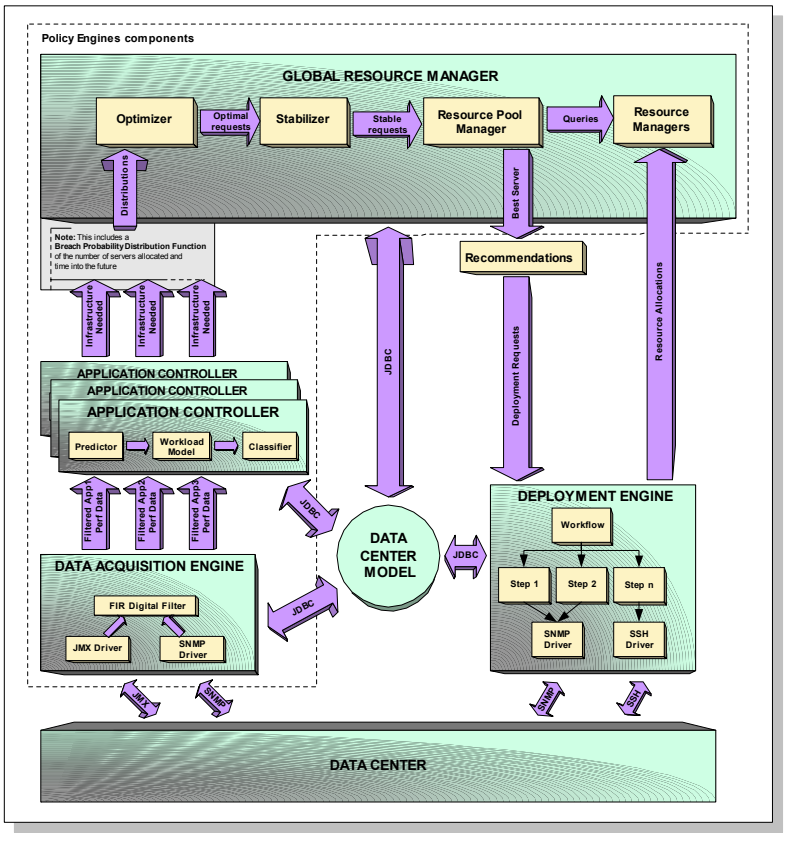


Figure 1. The high-level architecture of the Tivoli Intelligent Orchestrator

Deployment Engine

The Deployment Engine is responsible for the creation, the storage, and the execution of repeatable workflows that automate the server configuration and allocation in the system. A workflow can represent either an entire reconfiguration process affecting multiple servers, or a single step in a larger reconfiguration process. The Deployment Engine processes high-level configuration requests by executing various low-level configuration commands that configure any device in the environment. These commands are sequentially passed onto device-dependent drivers. Multiple processes and physical servers independently process entire commands, which are further delegated to external systems for system management. The Deployment Engine receives a reconfiguration command through a *deployment request interface*, which forwards the reconfiguration commands to a *deployment engine controller*. The controller includes the following components:

- Workflow assembly component
- Workflow execution component

Workflow assembly component

This component receives the reconfiguration command and coordinates the translation of the reconfiguration command into an executable workflow. This mechanism searches a workflow database to determine if the reconfiguration command, or parts of it, can be represented by workflows that have been previously created and stored in the database.

Workflow execution component

After a workflow has been determined for the reconfiguration command, this component receives the workflow for execution, and determines which server pertains to which each step in the workflow. It passes a command corresponding to each step to a *servers interface*. A *workflow execution controller* controls the workflow execution, and provides multiple working threads to allow simultaneous execution of multiple workflows.

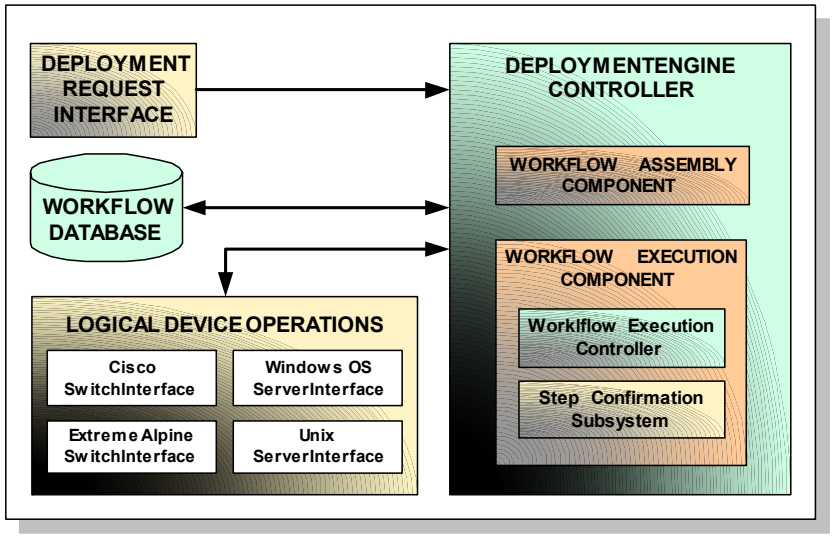


Figure 2. The Deployment Engine components

A *step confirmation subsystem* awaits for confirmation of successful implementation of the step on the destination servers. If the confirmation does not arrive in a predetermined time interval, the step confirmation subsystem informs the *workflow execution controller*, which awaits for a predetermined time before executing the next step. If the previous step was not successfully implemented, the workflow execution controller may re-execute that previous step. This ensures that each step in the workflow has been successfully implemented before subsequent steps are executed. Alternatively, the workflow execution controller can decide to continue execution of the workflow, regardless of the successful or unsuccessful implementation of each step in the workflow.

Deploying commands to servers

The logical device operations interface contains an interface to each type of device that can be configured in the data center infrastructure. For example, it can contain different interfaces for different types of switches, different interfaces for different types of servers, etc.

The workflow execution mechanism forwards a command to the appropriate interface, which formats the command messages so as to make them recognizable and easy to handle by the destination server. The interface sends the formatted command to this server and awaits the confirmation of the subsequent change in configuration.

When a reconfiguration command is received, the workflow database is searched to determine if there are any existing workflows that have been previously created for performing the same function. If the reconfiguration command can be completely represented by an existing workflow, the workflow is executed. If sufficiently detailed, the smaller workflow steps are executed by the server or network device for which they are destined.

Executing the workflow steps

When the level of detail of the steps in the main workflow is appropriate, the workflow is executed, following these steps:

1. The step is examined to identify the server or network device on which it will be executed.
2. A command corresponding to the step is created.
3. The command is formatted to be recognizable by the specific server.
4. The formatted command is sent to the appropriate server.
5. A receipt of the command implementation is received from the server.
6. If all steps in the workflow have been executed, the status of the resource is saved in the Data Center Model database.

The workflows can create new application environments, create new application clusters, and add or remove servers to or from running application clusters. The workflows can also be created to perform any of the following functions:

- Deploy an application environment, or any part of it, on a server
- Reboot a server
- Configure network communications on a server
- Communicate with switching devices to reconfigure servers on a VLAN
- Communicate with load balancers to reconfigure a cluster
- Reconfigure an application environment
- Inform the fault management system of an invalid action

- Raise a billing event

Data Center Model

The Data Center Model component includes a representation of all of the *physical* and *logical assets* under Tivoli Intelligent Orchestrator's management, such as servers, switches, load balancers, application software, VLANs, security policies, service level agreements, etc. It keeps track of the data center hardware and associated allocations to customer sites.

The database facilitates the information transfer between all of the other Tivoli Intelligent Orchestrator components, and makes it possible for these components to read and update this information according to any resource manipulations that have been performed. Also stored in the central database is information on various server groups, allocated or unallocated. For a group of servers, this information can include, for example, server identifiers, the group size, the number of active and idle servers, server priority within that group, etc.

Data Acquisition Engine

The Data Acquisition Engine component is responsible for acquiring and pre-processing performance data from each managed application environment. Data is captured from the application, operating system, and infrastructure layers. This component uses a subscribing mechanism to distribute signals to other Tivoli Intelligent Orchestrator components, and performs filtering of raw signals.

The Data Acquisition Engine contains the following components that gather information from their respective application environment layers:

- Application layer acquisition component
- Operating system acquisition component
- Server infrastructure acquisition component
- Networking infrastructure acquisition component

All of these acquisition components pass the obtained information to the *acquisition controller*, which passes it on to an *application controller interface*, which then forwards it to the Application Controller.

The figure below illustrates the interaction between all of these components.

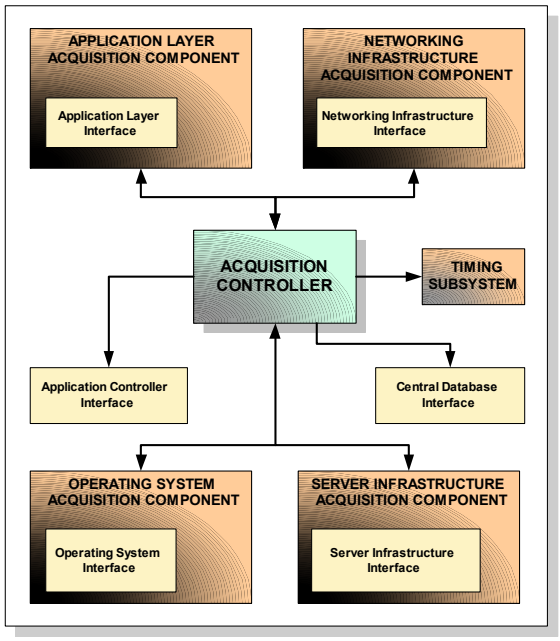


Figure 3. Data Acquisition Engine components

A *timing subsystem* controls the timing of the data acquisition, and keeps a record of the last set of data obtained from each layer in each application environment. At predetermined intervals, the timing subsystem informs the acquisition controller that a predetermined time period has elapsed. In response, the acquisition controller sends a command to one of the acquisition components, to obtain data from a specific application environment. The acquisition of data from each layer in the application environment can be performed simultaneously or can be staggered.

Application layer acquisition component

This component gathers information from the application layer through its *application layer interface*, that obtains information from all of the web, application, and database servers in the system. For example, the obtained information could be the processing speed of requests in the application environment, and the response time to requests.

Operating system acquisition component

This component gathers information from the operating system layer, through its *operating system interface*. Involved in this process are mechanisms that are used by both the infrastructure and the application layers to expose performance data.

Server infrastructure acquisition component

This component gathers information from the server infrastructure layer through its *server infrastructure interface*. The obtained information shows how much of the total processing power or how much of the total memory is currently being used. The information is based on the server groups that have been allocated to a specific application environment.

Networking infrastructure acquisition component

This component gathers information from the networking infrastructure layer, through its *networking infrastructure interface*. The information is obtained from switches, routers, firewalls, and load balancers, and shows how much of the bandwidth allocated to an application environment is being used, and the transaction rates at a protocol level.

Application Controller

An instance of the Application Controller is created for each application environment under management. Based on the application's workload model and predictions, and using the real-time performance data provided by the Data Acquisition Engine, the Application Controller determines the resource requirements of the application, which are then sent to the Resource Broker component, which manages the overall optimization. The Application Controller incorporates prediction into an adaptive controller that suggests resource requirements to the Resource Broker.

The Application Controller comprises the following components that interact to monitor the performance and predict the future demand levels for the application environment, so as to maintain the predetermined level of service:

- A Prediction component
- A Workload modeling component
- A Classification component

Prediction component

The prediction component receives demand information from the Data Acquisition Engine through the *monitoring data interface*, and predicts the future demand for the system's resources, for example, the arrival rate (hits/sec.) measurements for a web cluster.
The demand information for a particular server or network device is used by the predictor to determine both *stationary* and *non-stationary* trends. By definition, the *stationary* (time-serial) trends in the demand information are random, while the *non-stationary* (time-varying) trends are periodic, occuring at regular intervals, such as every day at a particular time, a particular day of the week, etc.

From the monitoring data interface, the demand information is sent to the following subsystems:

- Autocorrelation subsystem
- Time-varying trends detection subsystem
- Time-varying trends removal subsystem

The figure below illustrates the interaction between the prediction component's subsystems:

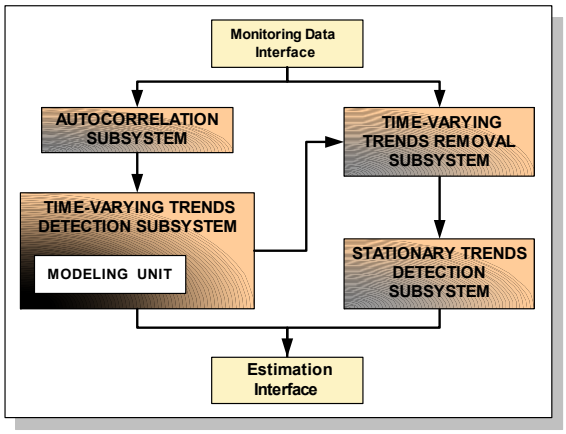


Figure 4. Application Controller's prediction component

Autocorrelation subsystem

This subsystem detects non-randomness in the demand information received from the monitoring data interface. If the demand information is not random, this subsystem creates a periodic time series model (or autocorrelation function) of the demand information. This autocorrelation function is a measurement of similarity between distributions of sample data.

Time-varying trends detection subsystem

This subsystem receives the demand information and the autocorrelation function from the *autocorrelation subsystem*. It analyzes this information to determine *time-varying* trends in the demand information for a particular server. The *time-varying* trends are those patterns of server demand that occur at regular intervals, such as every day at a particular time, a particular day of the week, etc.

A *modeling unit* in the time-varying trends detection subsystem creates a model of the time-varying information, based on a two factor full factorial design without replication algorithm. The model created by the modeling unit is periodic, and its periodicity can be leveraged to extend the model beyond the results provided to predict the future demand based on the recurring periodic nature of the current demand. The assumption with this extension is that there will be no large singular demands, and that demand levels will remain relatively consistent.

Time-varying trends removal subsystem

This subsystem removes the time varying components from the demand information received from the Time-varying trends detection subsystem. The removed time varying components are provided to a *stationary trends detection subsystem*, which creates a model of time-serial trends in the demand information, based on a linear autoregressive model.

The time-serial trend model is periodic, and its periodicity can be leveraged to extend the model beyond the results provided to predict the future demand based on the recurring periodic nature of the current demand. The assumption with this extension is that there will be no large singular demands, and that demand levels will remain relatively consistent.

The models created by the time-varying and stationary trends detection subsystems are passed to an *estimation interface*, where they are combined to provide a prediction for future demand levels, based on the server from which the demand information was taken. This combination is an aggregation of the time-varying model and the stationary model.

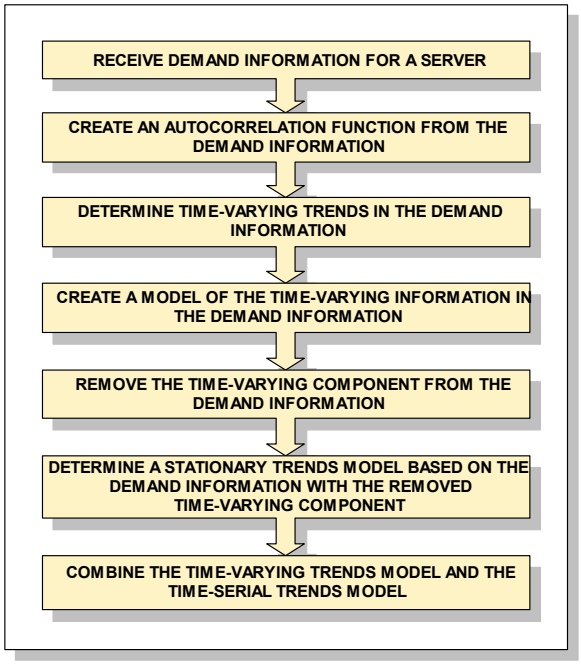


Figure 5. Predicting a future demand level for a server

Prediction algorithm

By modeling the traffic as a composition of two subproblems, one addressing non-stationary traffic, and another addressing stationary time series traffic, the system can achieve high probabilistic prediction rates.

Typically, the system uses the fine-grained non-stationary patterns according to the *time-of-day* and *day-of-week* factors. The stationary traffic is obtained from approximating this pattern and removing it from the original signal, and can be modeled to accommodate both the random and the growth aspects of the expected traffic.

Workload modeling component

This Application Controller component estimates each application environment's response to incoming traffic. It obtains the predicted future demand load from the Prediction component, and current performance information for a particular application environment from the Data Acquisition Engine. The performance information contains data describing the current demand loads for that application environment, as well as server performance data for the application environment under that demand. For example, the performance data can be the utilization of the servers allocated to that application environment. Based on this information, the servers in that application environment can be determined by considering the changing levels of demand. Based on the predicted demand, the workload modeling component estimates how the application environment will perform under the future load conditions, by using a linear regression model. The determined performance and demand parameters are: service time, service rate, variance, etc.

The workload modeling component also generates statistics for the critical CPU and critical arrival rate, which are then used to calculate the probability breach.
Based on the predetermined level of service for the application environment, the demand rate obtained from the Prediction component, and the performance and demand parameters, the workload modeling component uses, for example, a single-station queuing model, such as M/M/1 (M-distribution of interarrival times is exponential/M-distribution of service demand times is exponential/1-number of servers). This produces the demand rate at which the current servers in the application environment will not be able to maintain the predetermined level of service under the predicted demand. Separate queuing models are used for each workload class.

Classification component

This Application Controller component uses the information obtained from the Workload modeling component to determine how each application environment meets the predetermined level of service, and what changes should be made to maintain it. A *server requirements detection subsystem* determines the server demand for that application environment, to maintain the predetermined level of service.

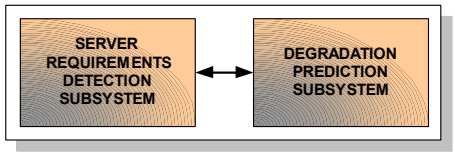


Figure 6. Application Controller's classification component

The *server requirements detection subsystem* uses real environment performance data indicating the used servers, the level of demand, and the performance of the application environment under these conditions. Considering the current servers used by the application environment and its performance, the server requirements detection subsystem determines the operating requirements of the application environment. The server requirements for the application environment can be extrapolated from the current performance.

A *degradation prediction subsystem* predicts the way the application environment performance will degrade if the determined server requirements are not implemented. This subsystem compares the predicted degradation of performance with the predetermined service level for the application environment, to determine any discrepancies.

Global Resource Manager

This component receives requirements for servers or network devices from all the application controllers, and manages the overall optimization. This component has two primary responsibilities:

- Makes optimal server allocation decisions
- Ensures a stable control over the application infrastructure

Considering the different server requirements for each application environment, the Global Resource Manager determines where the servers are to be allocated. The server reconfiguration and allocation data is then passed on to the Deployment Engine, which breaks down the configuration changes into commands that are formatted and sent to the involved servers and network devices to implement the changes.

The Global Resource Manager is comprised of the following components that all work together to determine the configuration and allocation for each server in the system:

- A Resource Broker component
- An Optimization component
- A Stabilization component
- A Resource pool manager component

The *Resource Broker component* uses real-time optimization algorithms, and manages the overall data center servers to optimally meet individual system demands. The *optimization component* receives the requirements for servers and network devices from the Application Controller, and determines the server configuration that best meets the anticipated needs of the application environment. The new server configuration is analyzed by a *stabilization component*, to ensure that the requirements for each application environment have been determined under stable conditions, without erratic fluctuations in the demand, and that each application environment will remain stable after the changes. After the new server configuration has been deemed stable, a *resource pool manager* determines the individual servers that must be added or removed from each application environment, in order to meet the anticipated demand changes.

Resource Broker component

This component manages all the resource pools, and attempts to service server requests from each of the application controllers. Using real-time optimization algorithms, it manages the overall data center servers to optimally meet individual system demands. The Resource Broker attempts to maintain some surplus servers for quick deployment into needy sites. When the overall demand on the data center is near its capacity, this component weighs individual site's needs with their particular service level agreements, to decide which sites get additional capacity and which sites are left underpowered.

This is the central component that manages all requests to optimize the use of available servers. When a decision requiring a change to the infrastructure is made, the Resource Broker package issues commands to the Deployment Engine.

Optimization component

This component uses the anticipated server requirements provided by the application controllers to determine a balance between requirements for different application environments that are competing for the same servers. The anticipated server requirements are received through a *requirements interface*, which passes this information to a *request type sorting component*, which sorts the requests according to the type of server requested. For example, the requests for an additional server with a Unix operating system are sorted from the requests for an additional server with a Windows operating system.

The optimization component contains a *server optimizer* for each type of server in the system, as exemplified in the following figure. Each server optimizer contains the following subsystems:

- *Creation subsystem*
- *Trimming/Pruning subsystem*
- *Search subsystem*
- *Editing subsystem*

The server optimizers create decision trees based on the received server request, the available servers, and the predetermined level of service for the current application environment and for all other application environments.

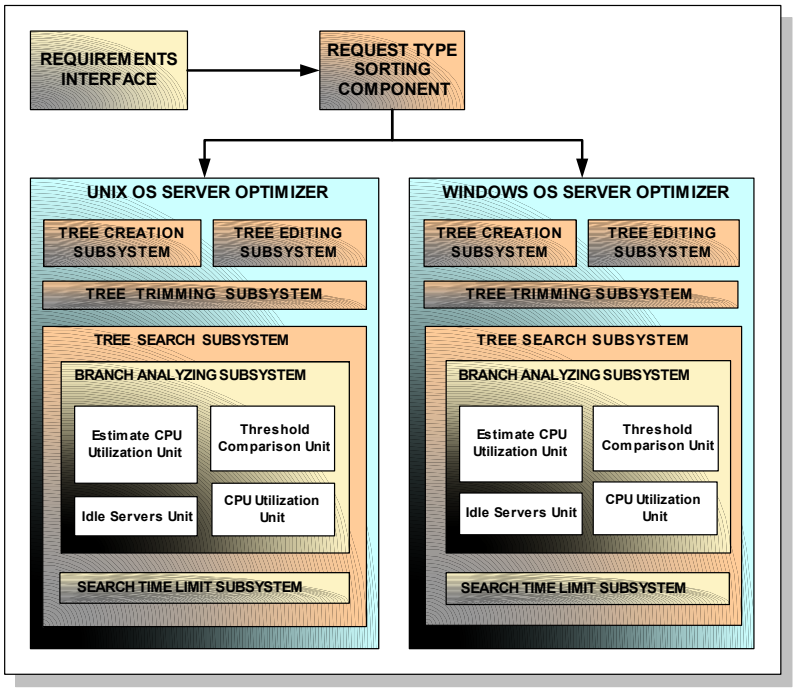


Figure 7. The Global Resource Manager's optimization component

The *request type sorting compunent* passes the server request to the appropriate server optimizer. Within the optimizer, the *tree creation subsystem* receives the server request, and creates a decision tree. Each branch in the decision tree specifies the number of servers to be added or removed from each unallocated or allocated server group.

The decision tree is created by determining every possible combination, change, or permutation in the server allocation, and then determining every possible second step combination, etc. for both current moment in time and throughout the future predicted time horizon. Starting with the current configuration of each server in the application environment, new branches are created for each possible configuration change that results in a new configuration.

Stabilization component

This component analyzes and filters the server changes indicated by the Optimization component, to determine whether similar changes were recently implemented. It maintains a stability in the server allocation and configuration, to prevent servers from being reconfigured and reallocated as a result of erratic changes in demand or performance of an application environment. The stabilization component can, for example, apply a time-based filter that prevents multiple opposing changes for a specified period of time.

Resource pool manager component

This component is responsible for the preliminary configuration and allocation of currently unallocated groups of servers. The preliminary configuration hastens the allocation of a server to an application environment. The resource pool manager determines the most used general configuration (for example, operating system), and configures groups of unallocated servers accordingly, to ensure that more servers that have the mostly used configuration are available.

Management Interface

This component provides an overview of the state of all *physical* and *logical assets* in the data center infrastructure, offering information about the servers and their allocation, and generating configurations and allocations. It can also be used to create application environments.

The Management Interface component includes two distinct user interfaces that can be used to manage and configure the application resources:

- A Web-based interface, that offers an intuitive way to display information about applications and components
- A Command-line interface, that can be used by operators who prefer to access the system's internal properties and operations using the command line

The management interface interacts with the Data Center Model database to allow user access to its information. The user interface also communicates with the Data Acquisition Engine, the Deployment Engine, the Application Controller, and the Global Resource Manager components.

Note: For additional information on how to configure and operate the Tivoli Intelligent Orchestrator using these two user interfaces, refer to the *IBM Tivoli Intelligent Orchestrator: Operator's Guide*.

The following sections provide details on each of these interfaces.

Web-based interface

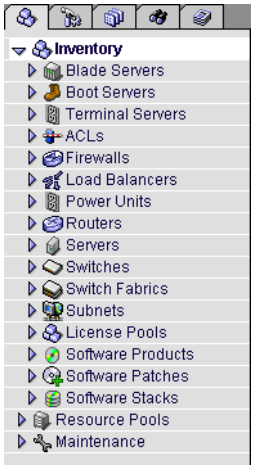
The Web-based user interface offers a real-time access to the Tivoli Intelligent Orchestrator, as well as an intuitive way to display information about the deployed application and components. It allows you to easily monitor and control the resource management and application performance.

Note: For additional information on the specific screen elements, refer to the introductory chapter in the *IBM Tivoli Intelligent Orchestrator: Operator's Guide*.

Using the navigation trees, you can easily configure and manage typical n-tier application architectures, as well as hardware assets and other system resources. You can define, configure, customize, and execute various workflows that meet any data center process requirement, and you can also create and configure Java plug-ins, simple commands, and logical device operations to be included in workflow configurations. Also provided is the ability to generate and output various reports that illustrate the Tivoli Intelligent Orchestrator's functionality.

A brief description of the Tivoli Intelligent Orchestrator's navigation trees is provided in the following.

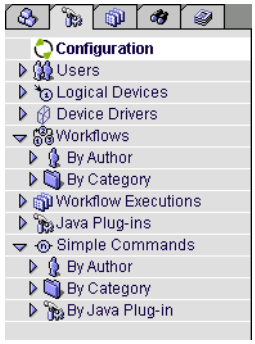
Data center assets and resources



This navigation tree enables you to configure and manage all of your system's hardware assets and resources. From this tree, you can:

- Configure blade servers, boot servers, terminal servers, ACLs, firewalls, load balancers, power units, routers, servers, switches, switch fabrics, subnetworks, license pools, software products, software patches, and software stacks.
- Manage and configure all the available resource pools, and the resources that are currently under maintenance (applications, overflow servers, hardware resources).

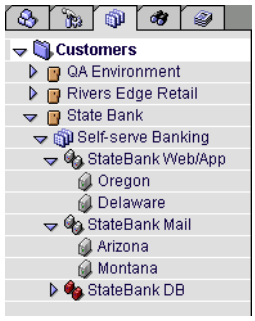
System configuration and workflow management



This navigation tree enables you to configure the global operating mode, manage the Deployment Engine, and also create, configure, customize, and execute various workflows. Workflows are grouped by author and category. You can also manage logical device operations that are included in workflow configurations, device drivers, Java plug-ins, and simple commands. From this tree, you can:

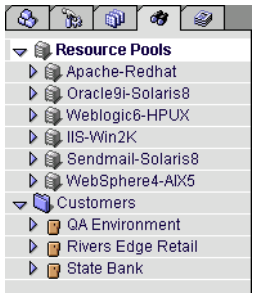
- Manage and configure all of the users that are currently registered with the system.
- Configure all logical device operations and device drivers defined in the system, as well as all available workflows, Java plug-ins, and simple commands.
- Add new workflows, copy or customize, execute, or delete any of the existing workflows, manage variables and transitions within workflows.
- Determine the current state of a data center component configuration by reviewing the execution history of the workflows applied to it.

Customer applications



This navigation tree enables you to configure and manage typical n-tier application architectures. The in-depth structure of the **Customer applications** navigation tree includes *customer accounts*, *applications*, *clusters*, and *servers*, and makes it possible to define, configure, and manage all of these objects.

Realtime performance monitoring



This navigation tree enables you to monitor the system's health. From this tree, you can:

- List all the servers that are available in the resource pools.
- Monitor the overall performance of the customers, applications, clusters, and servers defined in the system, and also evaluate the performance of individual servers.

Reports

This navigation tree enables you to generate various reports that illustrate the Tivoli Intelligent Orchestrator's functionality, and show the benefits of using the system. From this tree, you can:

- Generate reports on the availability of resources: which resources are mostly used, the utilization over a specified period of time.
- Predict future shortfalls of physical resources.
- Report such status as the current allocation of resources, the current status of resource pools.
- Obtain usage reports per cluster, application or customer account, detailing the type of resources, number of times used, total time used, etc.
- Generate SLM reports showing the measured adherence to SLOs of each managed application.
- Generate network administration reports, showing all the commands executed on various network devices.

Command-line interface

The command-line interface is designed to be mainly used by data center operators who prefer to access some of the system's internal properties and operations using the command line.

Note: For additional information on the SOAP commands that can be executed using the Tivoli Intelligent Orchestrator's command-line interface, refer to the "*SOAP commands available through command line*" chapter, in the *IBM Tivoli Intelligent Orchestrator Operator's Guide*.

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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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 might 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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
DB2
DB2 Universal Database
IBM
The IBM logo
Netfinity
RS/6000
Tivoli
Tivoli Enterprise
WebSphere

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

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks, of Microsoft Corporation in the U.S. and other countries.

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

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

Index

Program Number: 5724-F75