Tivoli Management Framework

Planning for Deployment Guide

Version 4.3.1

GC32-0803-02

Tivoli Management Framework

Planning for Deployment Guide

Version 4.3.1

GC32-0803-02

Note

Before using this information and the product it supports, read the information in "Notices," on page 227.

Contents

Preface

The *Tivoli® Management Framework Planning for Deployment Guide* explains the architecture provided by Tivoli Management Framework and its services as well as the choices to consider while planning the deployment of a Tivoli environment.

Who should read this guide

This guide is intended for system architects, system administrators, and technical specialists who are responsible for installing Tivoli Management Framework in their environment. Users of this guide should have some knowledge of the following:

- Operating systems to be managed
- Shell programming
- The Motif or Microsoft® Windows® environment

Prerequisite and related documents

In addition to this guide, you can review the following documentation for Tivoli Management Framework:

- *Tivoli Enterprise Installation Guide*

  Explains how to install and upgrade Tivoli Enterprise software within your Tivoli region using the available installation mechanisms provided by Tivoli Software Installation Service and Tivoli Management Framework. Tivoli Enterprise software includes the Tivoli server, managed nodes, gateways, endpoints, and RDBMS Interface Module (RIM) objects. This guide also provides information about troubleshooting installation problems.
- *Tivoli Management Framework User's Guide*

  Describes the concepts and procedures for using Tivoli Management Framework services. It provides instructions for performing tasks from the Tivoli desktop and from the command line.
- *Tivoli Management Framework Maintenance and Troubleshooting Guide*

  Explains how to maintain a Tivoli environment and troubleshoot problems that can arise during normal operations.
- *Tivoli Management Framework Reference Manual*

  Provides in-depth information about Tivoli Management Framework commands. This manual is helpful when writing scripts that are later run as Tivoli tasks. This manual also documents default and validation policy scripts used by Tivoli Management Framework.
- *Tivoli Management Framework Maintenance and Troubleshooting Guide*

  Explains how to maintain a Tivoli environment and troubleshoot problems that can arise during normal operations.

References to the interpreter type for a particular client are located throughout this guide. Interpreter types for supported operating systems are located in the *Tivoli Management Framework Release Notes*.

What this guide contains

The *Tivoli Management Framework Planning for Deployment Guide* contains three parts:

- **Part I—Introduction to Tivoli Management Framework**
  - Chapter 1, "Introduction to Tivoli Management Framework," on page 3

    Describes general information about Tivoli Management Framework and system management and provides an introduction to managing systems with Tivoli products.
  - Chapter 2, "Overview of the Tivoli architecture," on page 9

    Provides an overview of Tivoli architecture and Tivoli region connections.
  - Chapter 3, "Overview," on page 19

    Describes Tivoli Management Framework services.
  - Chapter 4, "Profiles and profile managers," on page 35

    Describes profiles and profile managers, their subscriptions, and distributions.
  - Chapter 5, "Endpoints and gateways," on page 47

    Discusses endpoints, the endpoint manager, the gateway, and endpoint communication.
  - Chapter 6, "Multiplexed distribution services," on page 63

    Describes the multiplexed distribution services in the Tivoli environment.
- **Part II—Creating a Deployment Plan**
  - Chapter 7, "Design guidelines," on page 85

    Discusses what is required to build a deployment plan, including how to plan for the physical and logical design of the Tivoli environment. It includes checklists based on administrative responsibilities, installed products, and physical systems.
  - Chapter 8, "Hardware and network considerations," on page 99

    Provides recommendations for hardware, software, and network infrastructure for the successful deployment of a Tivoli environment. It includes a discussion of DHCP, slow speed networks, network address translation, VSAT, and multihomed hosts.
  - Chapter 9, "Physical design," on page 113

    Describes the design of the underlying physical infrastructure on which the Tivoli solution will operate. It includes the network topology, performance considerations, security impacts, reliability concerns, and placement of servers.
  - Chapter 10, "Logical design," on page 135

    Describes how the logical structure of a Tivoli environment should be configured to meet the operational requirements of your organization. Discusses appropriate naming conventions for objects, the role of administrators, and the power of policy regions and profile managers.
  - Chapter 11, "Disaster recovery," on page 159

    Provides information about planning for high-availability systems and disaster recovery in a Tivoli environment.
- **Part III—Reference**
  - Chapter 12, "Planning scenario 1," on page 173

    Provides a design scenario of a single region deployment, using the principles described in the second part of the book.
  - Chapter 13, "Planning scenario 2," on page 191

Provides a design scenario of a multiple region deployment, using the principles described in the second part of the book.

– Chapter 14, “Authorization roles,” on page 205

  Describes the authorization roles needed to perform activities in a Tivoli environment.

– Chapter 15, “Directory structure and system variables,” on page 213

  Describes the directory structures and system variables created during the installation of a Tivoli server, managed nodes, gateways, and endpoints.

Accessing publications online

The documentation CD contains the publications that are in the product library. The format of the publications is PDF, HTML, or both.

IBM posts 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. Access the Tivoli software information center by first going to the Tivoli software library at the following Web address:

http://publib.boulder.ibm.com/tividd/td/tdprodlist.html

Note: If you print PDF documents on other than letter-sized paper, set the option in the **File → Print** window that allows Adobe Reader to print letter-sized pages on your local paper.

Ordering publications

You can order many Tivoli publications online at the following Web site:

http://www.elink.ibmlink.ibm.com

From this Web page, select **Publications** and follow the instructions.

You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, see the following Web site for a list of telephone numbers:

http://www.ibm.com/software/tivoli/order-lit

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

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, depending on the country in which you are located
- Information you must have before contacting IBM Software Support

Conventions used in this guide

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
- Keywords and parameters in text

Italic

- Words defined in text
- Emphasis of words (words as words)
- New terms in text (except in a definition list)
- Variables and values you must provide

`Monospace`

- Examples and code examples
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

This guide uses the UNIX convention for specifying environment variables and for directory notation:

- When using the Windows command line, replace $*variable* with %*variable*% for environment variables and replace each forward slash (/) with a backslash (\) in directory paths.
- When using the bash shell on Windows operating systems, use the UNIX conventions.

Part 1. Introduction to Tivoli Management Framework

Chapter 1. Introduction to Tivoli Management Framework

Managing heterogeneous distributed computer systems is a complex task that can involve various operating systems, distributed network services, and system management tasks. Tivoli Management Framework, and the products that can be installed on top of it, simplify the management of distributed systems.

This chapter introduces the benefits of a Tivoli environment and defines the management services used within that environment:

- "Benefits of Tivoli Management Framework"
- "Tivoli Management Framework and related products"
- "Managing Tivoli resources" on page 4
- "Using Tivoli software products" on page 6

The following chapters describe Tivoli concepts. If you are already familiar with these concepts, proceed to Chapter 7, "Design guidelines," on page 85.

Benefits of Tivoli Management Framework

Tivoli Management Framework is the software infrastructure for many Tivoli software products. Using Tivoli Management Framework and a combination of Tivoli Enterprise products, you can manage large distributed networks with multiple operating systems, various network services, diverse system tasks, and constantly changing nodes and users. Tivoli Management Framework provides management services that are used by the installed Tivoli Enterprise products.

Tivoli Management Framework provides centralized control of a distributed environment, which can include mainframes, UNIX® operating systems, or Microsoft Windows operating systems. Using Tivoli Enterprise products, a single system administrator can perform the following tasks for thousands of networked systems:

- Manage user and group accounts
- Deploy new or upgrade existing software
- Inventory existing system configurations
- Monitor the resources of systems either inside or outside the Tivoli environment
- Manage Internet and intranet access and control
- Manage third-party applications

Tivoli Management Framework lets you securely delegate system administration tasks to other administrators, giving you control over which systems an administrator can manage and what tasks that administrator can perform. Tivoli Management Framework includes the base infrastructure and base set of services that its related products use to provide direct control over specific resources in a distributed computing environment. Tivoli Management Framework provides a simple, consistent interface to diverse operating systems, applications, and distributed services.

Tivoli Management Framework and related products

The following describes Tivoli Management Framework and its related products:

- Tivoli Management Framework, based on an industry standard framework, provides basic system administration capabilities, and management services for Tivoli software products. These capabilities and services include a Tivoli administrator facility, an administrative privilege facility, a system policy facility, and a notification facility. Tivoli Management Framework includes the Tivoli desktop, which provides a graphical user interface (GUI), and a command line interface (CLI).
- *Application Extension Facility* enables you to dynamically customize Tivoli software products by adding site-specific behavior or values to standard, off-the-shelf products.
- *Event Integration Facility* enables you to build event adapters to map events from a product, resource, or component into a format compatible with several Tivoli Enterprise products.
- *Application Development Environment* includes programming tools for creating new custom management applications on top of Tivoli Management Framework.

Tivoli software products provide tools for managing specific system resources and services. These products are, among others, products for software auditing and distribution, remote monitoring, user administration, and Internet and intranet management. Refer to "Using Tivoli software products" on page 6 for more information.

The relationship among the Tivoli components is shown in Figure 1.

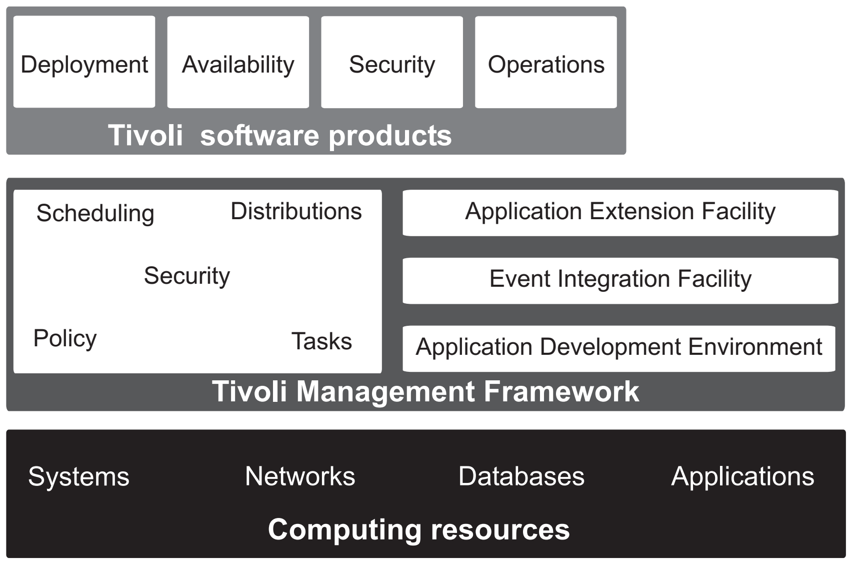


Figure 1. Relationship among components in a Tivoli environment

Managing Tivoli resources

Tivoli Management Framework provides the foundation for managing resources in a distributed environment in the following ways:

- Provides an administrative view into the network, as well as a set of tools that can be used across functions and applications
- Provides the run-time platform for Tivoli software products

Tivoli Management Framework provides a set of system management services that enable you to install Tivoli Management Framework and selected products on multiple, heterogeneous systems. After deployment, it provides a foundation for managing Tivoli resources, policies, and policy regions:

- Tivoli *resources* correspond to the systems, devices, services, and facilities in a distributed system. Each Tivoli resource is classified as a resource type. Examples of Tivoli resources are workstations, software, and administrators. After a Tivoli software product is installed, the underlying infrastructure provides you with a logical view of all resources managed by the product. Tivoli software products enable you to make local and global system changes to these resources. With Tivoli software products, you can automatically update the actual system resources when the corresponding Tivoli resources are created, modified, or deleted.
- A *policy* is a rule that you put into effect for a system and that is enforced as management operations are performed by administrators. Traditionally, these rules take the form of software, shell scripts, written procedures, and guidelines, some of which are known only to the system administrators. Refer to "Policy and policy regions" on page 22 for more information.
- A *policy region* is a collection of Tivoli resources that are governed by a common set of policies. A policy region is often created to represent a management domain or area of influence for one or more system administrators. Refer to "Policy and policy regions" on page 22 for more information.

The following is a list of Tivoli Management Framework components that enable you to perform management tasks:

- *Tivoli administrators*, which enable you to assign different authorization roles to different administrators and to securely delegate various system management tasks to these same administrators. Unlike conventional system administrators, Tivoli administrators can be assigned specific system management tasks without providing them full privileged user access, such as root on UNIX operating systems or Administrator on Windows operating systems. Tivoli administrators are described in more detail in "Administrators" on page 19.
- A *region connection facility*, which includes support for multiple Tivoli regions that can be connected across different networks. A region consists of a Tivoli server and its clients (managed nodes, gateways, and endpoints). Regions are described in more detail in "Regions" on page 9.
- *Multiplexed distribution services*, which enable you to efficiently distribute large amounts of data, such as configuration data in Tivoli profile-based products, across complex networks. The distribution services are described in Chapter 6, "Multiplexed distribution services," on page 63.
- Hierarchical *profiles* and *profile managers*, which enable you to combine and distribute configuration data that is common to several resources. These resources are managed by the Tivoli software products. Profiles and profile managers are described in Chapter 4, "Profiles and profile managers," on page 35.
- *Task libraries*, which serve as a collection point for common system management tasks and activities. Task libraries are described in "Task libraries" on page 27.
- *Dynamic IP addressing* support for clients. Refer to "Dynamic Host Configuration Protocol (DHCP)" on page 109 for more information.
- The *RDBMS Interface Module* (RIM), which enables some Tivoli software products to write product-specific information to supported relational databases. RIM is described in "RDBMS Interface Module" on page 31.

- A *query facility*, which searches and retrieves information from a RIM database. The query facility is described in "Query facility" on page 32.
- A *scheduler facility*, which enables you to control the timing and automation of regular system operations. The scheduler is described in "Scheduler" on page 29.
- A *notification facility*, which provides an audit trail of system management tasks. The notification facility is described in "Notification" on page 30.

Using Tivoli software products

You can use your Tivoli environment to manage a wide variety of system resources. Tivoli Management Framework itself does not affect system configurations; you must install the Tivoli products to affect system resources and configurations.

Depending on the Tivoli software products you install, you can perform many complex management tasks:

- You might, for example, create user accounts, modify group accounts, and update an e-mail alias file for 50 new employees, and then distribute these changes to all computer systems in the network (which could be thousands) or to selected computer systems.
- You can manage third-party products, such as Lotus Notes® or Catia, by loading Tivoli tools with management data. This management data describes characteristics of an application—name, version, source location, dependencies, software signatures—that the tools use to distribute and install the application. Management data can also include monitors and task libraries, which the tools use to maintain the application in the Tivoli environment.
- You can scan all the computer systems in a policy region for a list of installed software, identify systems with old releases, and automatically distribute, install, and configure the upgraded software on the appropriate computer systems.
- If your primary file server crashes on Saturday morning, Tivoli monitors can identify the problem and possibly trigger corrective action. Otherwise, the appropriate system administrator or Information Technology (IT) personnel can be paged. Your server can be back online in a matter of minutes or hours instead of days.
- A sales representative who travels with a laptop system running Dynamic Host Configuration Protocol (DHCP) can dial in from an airport or hotel room, receive an IP address, connect to the Tivoli environment, and get the latest copy of the software to present to customers.

Tivoli software products bring together several areas for IT management. These areas include the following:

Asset management
Automates and expedites the processes that track and protect company assets.

Change management
Ensures safe and efficient implementation of changes, software updates, and the like across the entire network .

Operations management
Centralizes control of all back office operations to a single console.

Security management
: Allows all security processes for a company's mainframe, distributed, and desktop environments to be managed and monitored from a central location.

Storage management
: Allows you to deploy, access, share, and protect your company's vital business information.

When planning your Tivoli environment, it is important to consider the various products you plan to install and where you will install each. Each product makes its own demands on the Tivoli environment. Therefore, carefully consider the number and placement of managed nodes and the Tivoli server in relation to the Tivoli software products being installed. For a better understanding of some of the possible implications of installing multiple Tivoli software products, refer to "Application considerations" on page 118.

Chapter 2. Overview of the Tivoli architecture

When creating a network, you must consider the overall architecture supported by Tivoli Management Framework and used by the Tivoli software products. This architecture can be comprised of multiple Tivoli regions. Each Tivoli region contains a Tivoli server and various clients, such as managed nodes, gateways, and endpoints. Tivoli regions can be connected together to take advantage of the basic network structure.

The chapter contains the following sections:
- "Components of the basic Tivoli architecture"
- "Name Registry" on page 11
- "Connecting multiple Tivoli regions" on page 12

Components of the basic Tivoli architecture

Each management component has a purpose in a Tivoli environment. Determining which computer system in your network is best suited to provide these components requires that you consider a number of factors, such as the following:
- Operating system
- Disk space requirement
- Hardware requirements
- Network topology and communication
- Tivoli management function

The following components comprise the basic Tivoli architecture:
- Tivoli servers
- Managed nodes
- Gateways
- Endpoints

Regions

A Tivoli region is an entity that contains the Tivoli server and its clients. A Tivoli region contains three tiers of resources, the Tivoli server, managed nodes and gateways, and endpoints, as shown in Figure 2 on page 10.

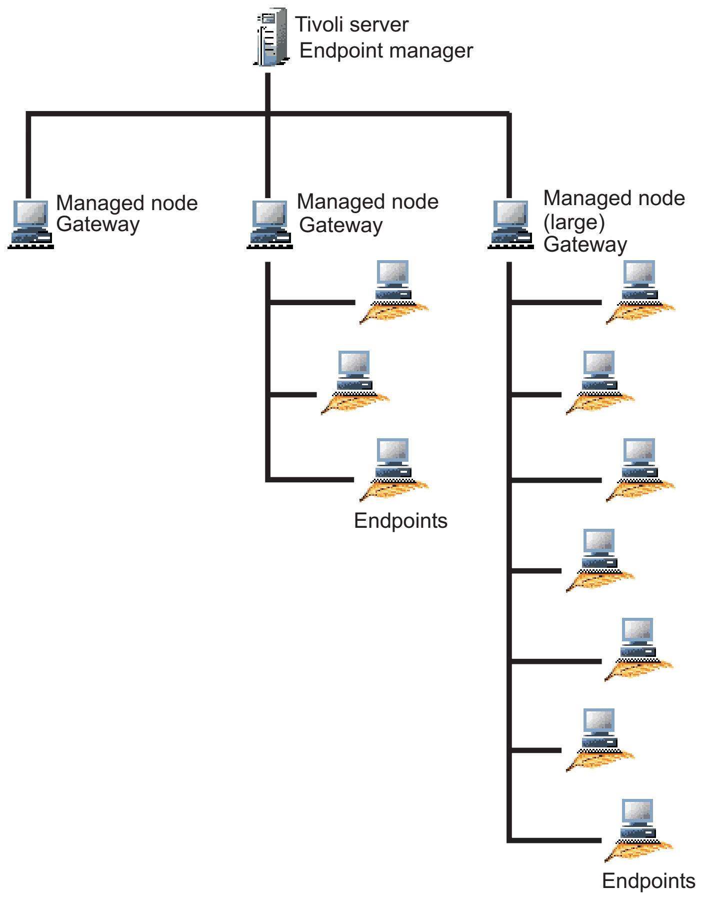


Figure 2. A Tivoli region is three-tiered with the Tivoli server, managed nodes and gateways, and endpoints

Using this three-tiered hierarchy, the amount of communication with the Tivoli server is reduced. Endpoints do not communicate with the Tivoli server, except during the initial login process. All endpoint communication goes through the gateway. In most cases, the gateway provides all the support an endpoint needs without requiring communication with the Tivoli server.

In a smaller workgroup-size installation, you can create the gateway on the Tivoli server. The server can handle communication requirements when fewer computer systems are involved. This is not an acceptable option in large deployments. The Tivoli server in a large installation will be overloaded if it also serves as a gateway. Refer to Chapter 5, "Endpoints and gateways," on page 47 for more information about endpoint communication.

Tivoli servers

The *Tivoli server* includes the libraries, binaries, data files, and the graphical user interface (GUI) (the Tivoli desktop) needed to install and manage your Tivoli environment. The Tivoli server performs all authentication and verification necessary to ensure the security of Tivoli data. The following components comprise a Tivoli server:

- An *object database*, which maintains all object data for the entire Tivoli region.

- An *object dispatcher*, which coordinates all communication with managed nodes and gateways. The object dispatcher process is the oserv, which is controlled by the **oserv** command.
- An endpoint manager, which is responsible for managing all of the endpoints in the Tivoli region.

When you install the Tivoli server on a UNIX operating system, the Tivoli desktop is automatically installed. When you install the Tivoli server on a Windows operating system, you must install Tivoli Desktop for Windows separately to use the Tivoli desktop.

Managed nodes

A *managed node* runs the same software that runs on a Tivoli server. Managed nodes maintain their own object databases that can be accessed by the Tivoli server. When managed nodes communicate directly with other managed nodes, they perform the same communication or security operations that are performed by the Tivoli server.

The difference between a Tivoli server and a managed node is that the Tivoli server object database is global to the entire region including all managed nodes. In contrast, the managed node database is local to the particular managed node.

To manage a computer system that hosts the managed node, install an endpoint on that managed node.

Gateways

A *gateway* controls communication with and operations on endpoints. Each gateway can support thousands of endpoints. A gateway can launch methods on an endpoint or run methods on behalf of the endpoint.

A gateway is generally created on an existing managed node. This managed node provides access to the endpoint methods and provides the communication with the Tivoli server that the endpoints occasionally require. Refer to Chapter 5, "Endpoints and gateways," on page 47 for more information about gateways.

Endpoints

An *endpoint* provides the primary interface for system management. An endpoint is any system that runs the lcfd service (or daemon), which is configured using the **lcfd** command.

Typically, an endpoint is installed on a computer system that is not used for daily management operations. Endpoints run a very small amount of software and do not maintain an object database. The majority of systems in a Tivoli environment should be endpoints. The Tivoli desktop is not installed with the endpoint software. If you choose to run a desktop on an endpoint, you must install Tivoli Desktop for Windows or telnet to a UNIX managed node. Refer to Chapter 5, "Endpoints and gateways," on page 47 for more information about endpoints.

Name Registry

Each region contains a *name registry*, which serves as a region-wide name service. It essentially maps resource names to resource identifiers and the corresponding information.

The Tivoli name registry contains *resource types*, which contain *instances* of that resource type. For example, ProfileManager is a resource type, and the Admin profile manager is an instance of that resource type.

When a lookup for a resource occurs, it looks for a resource instance by name and resource type (for example, it looks up the moria instance of the ManagedNode resource type).

Generally, instances do not contain other instances, with the exception of managed nodes. An instance of the ManagedNode resource type might contain nested instances, which are resources required by the managed node but unique to each managed node. For example, each managed node has a TaskExecute resource, which is used as the context for running all tasks. To avoid the management and resource overhead required to maintain these resources separately from the managed node instance, the nested instances are contained in the managed node instance.

All resources in a Tivoli region that need to be looked up should be registered in the name registry when created, unregistered when deleted, and updated when their label is changed. For most resources in Tivoli Management Framework, these actions are handled automatically.

Connecting multiple Tivoli regions

To meet the needs and demands of managing thousands of resources that are geographically dispersed across networks, Tivoli Management Framework enables you to logically partition your managed resources into a series of connected Tivoli regions. Each region has its own server for managing local clients and a set of distributed replicated services for performing management operations. Regions can be connected to coordinate activities across the network, enabling large-scale systems management and offering the ability for remote site management.

During the connection process, each region is supplied with the following information about the region to which it is being connected:

- Server name
- Region number
- Encryption level and password in use in the other region

This information is used when the remote region is registered in the local region. When the regions are connected, an exchange of resource information can be performed to make known the types and values of resources in the remote region. After the initial information exchange, the information should be updated on a regular basis. For additional information, see "Exchanging resources between Tivoli regions" on page 16.

Region connections are not graphically represented as icons on the desktop. However, region connections are persistent; that is, region connections are maintained until manually disconnected.

To connect two regions, each region must have a name that is unique among all regions. If you attempt to connect a region that has the same name as another region, the connection fails.

Any Tivoli product installed in two connected regions should be installed in compatible versions in each region. Incompatible versions of a product do not cause a connection to fail, but can cause operation problems at a later time. However, you can connect two regions that do not have the same products installed.

For procedures for connecting, updating, and disconnecting Tivoli regions, refer to the *Tivoli Management Framework User's Guide*.

Types of region connections

With the ability to selectively connect regions, Tivoli administrators can control the scope and effect of local configuration changes. At the same time, Tivoli administrators can enable centralized management and propagation of new policy rules, configuration changes, and operations to all connected regions. Region connections are *directed*, meaning that they are not necessarily symmetric with respect to the two regions involved. Connections can be either one-way or two-way.

One-way region connections

In a *one-way connection*, only one region has knowledge of the other, so information is passed from the managing system only. One-way connections are useful where a central site is responsible for administering several remote sites, but none of the remote sites need to manage resources at the central site or at other remote sites. Each remote site could also have its own local operator who might be responsible for managing day-to-day operations on local resources, while the connection from the central site is used for more global updates across the company, such as a new version of an application.

Although one-way connections are feasible, two-way connections are recommended. With one-way connections, operations that need to report status cannot return the status to the managing Tivoli region. For example, a distribution from a managing Tivoli region to a target on the other side of a one-way connection cannot report back the results. Therefore, the distribution might appear to hang or fail.

Two-way region connections

Each Tivoli region involved in a *two-way connection* is aware of the existence of the other. Information exchanges about system resources occur in both directions. Two-way connections are useful in a variety of situations, such as a very large local area network that is logically partitioned. By using two-way connections, the management load is spread across multiple Tivoli servers. In addition, two-way connections are needed to access and manage resources in other regions.

Multiple region architectures

When connecting multiple Tivoli regions, consider the following architectures:

- Hub and spoke
- Master-remote
- Fully interconnected

Hub and spoke connections

The hub and spoke architecture improves performance by distributing server load. In this architecture, the Tivoli environment is segmented into several regions, each responsible for directly managing a different physical segment of the enterprise. This architecture supports a centralized management paradigm by having all management operations performed through the hub region. Remote, or spoke, regions are placed and configured to optimize network utilization and system performance and to distribute Tivoli server loads.

The *hub region* provides a centralized Tivoli server in its own dedicated region. The hub server directly manages a limited number of resources so that it can be dedicated primarily to the maintenance and administration of the Tivoli object database and Tivoli environment. It is also the focal point for hub-wide activities as required. For example, the hub Tivoli server is responsible for configuring and distributing monitoring profiles to any servers in the environment. The hub server is not responsible for directly managing resources.

The *spoke regions* directly control the endpoints in the Tivoli environment. Spoke regions are used to group managed nodes by physical location in the network and to localize Tivoli functions to that physical location, improving network and system performance. Generally, spoke regions are not used as entry points for administrators. With a properly implemented logical design, virtually all Tivoli operations can be performed from the hub region without concern for where the object being manipulated exists in the spoke environment.

The systems configured in the hub region should be enterprise systems dedicated to the operation of Tivoli products and should not be systems that should otherwise be managed by a spoke.

A logical layer of policy regions and profile managers combined with the proper connection of regions placed on top of this physical architecture provides a simple, consistent interface to manage portions of the enterprise directly from the hub region.

Two types of hub-and-spoke configurations can be implemented: *standalone configuration* or *combined configuration*.

Standalone configuration: In the standalone configuration, two separate hub and spoke solutions are put in place—one for monitoring the availability of server systems and one for deploying files to user desktop systems (see Figure 3). In this example, a standalone Tivoli environment for the server-monitoring requirement is completely separate from the user desktop Tivoli environment.

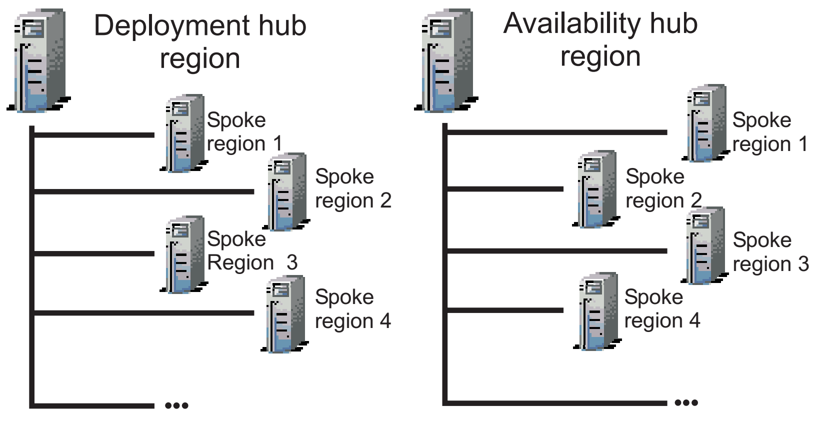


Figure 3. Hub and spoke region connection using a standalone configuration

The standalone configuration increases hardware costs by requiring a separate gateway infrastructure (and in some cases, based on the size of the environment, a separate spoke region infrastructure) be put in place for server availability. However, by providing the standalone hubs, the load on the deployment Tivoli server impacts the availability response time because there is a separate server for availability management.

Combined configuration: In the combined configuration, servers and user desktops are managed by one hub and spoke configuration. This is the configuration illustrated in Figure 4. Both servers and user desktops are hosted off gateways in the spoke regions. Systems are spread out geographically and connected to the spoke region most appropriate to that system given the load on regions and the physical network. Each spoke region, and gateway within a spoke region, can have a mix of user desktops and servers assigned to them.

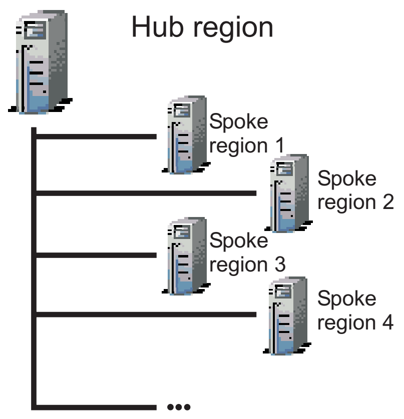


Figure 4. Hub and spoke connection, combined configuration

This configuration is the most inexpensive to implement from both a hardware standpoint and configuration complexity standpoint. However, because availability monitoring and deployment services are preformed in the same environment, the availability and performance of the Tivoli server is affected.

Master-remote connections

The *master-remote* architecture supports a distributed management structure by having the managed function operations being performed by administrators from their own regions. The different regions are connected to each other through a two-way connection, allowing the resources to be managed by any administrator in the managed environment.

Fully interconnected connections

The *fully interconnected* architecture has regions that are created for use within business units or for some subgroup within the enterprise. The regions are connected on an as-needed basis, which often leads to fully interconnected regions.

Connecting regions securely

When you make a connection between Tivoli regions, you must specify the appropriate login information, which includes password information. To specify the connection without sending these passwords across the network, make the connection locally on each Tivoli server you are connecting together. To specify the connection remotely without sending the passwords across the network, each

Tivoli server must have a remote command or remote shell capability and have trusted host access. For additional information, see the *Tivoli Management Framework User's Guide*.

Exchanging resources between Tivoli regions

Connecting Tivoli regions implies an initial exchange and periodic update of names and object identifiers contained in the Tivoli name registry. To reduce the number of cross-region messages that must be sent during name lookups, the resource information from one region is maintained in the name registry of all connected regions.

Updating resources across regions is a pull operation from the remote region to the local region; it is not a push or an exchange. Regions pull only those resource types that are managed resources within the region. Sometimes, a local region might not have the same set of resource types as a remote region, and operations performed on these resources will behave differently when performed across region boundaries. In some cases, it might not be possible to perform the operation remotely.

Resources and resource types are created by Tivoli Management Framework, Tivoli software products, and third-party vendors. Also, applications might exchange resource names. Therefore, applications can exhibit different behavior when crossing a region boundary.

Not all resources recorded in the Tivoli name registry are exchanged. Therefore, a number of restrictions exist on what can be done transparently across region boundaries. See "Basic resources and exchangeability" on page 25 for more information about resources that can be exchanged.

Resource exchange categories

Associated with each resource in the name registry is one of the following categories that affects how the resource is exchanged with other connected Tivoli regions:

exchangeable
A resource that can be updated between regions. This is the default for all resource types.

non-exchangeable
A resource that cannot be exchanged, or updated, between connected Tivoli regions. This category is for resources that need to be visible only within a single region.

custom
A resource that must define for itself what happens when it is exchanged between Tivoli regions.

For more information on these resource types, refer to "Basic resources and exchangeability" on page 25.

Resource visibility

Because resource names are updated between the name registries at discrete intervals, the names of remote resources in the local region can become outdated. A new resource created in the remote region is not registered in the local name registry of a connected region until the next update is performed. Therefore, this resource cannot be used in local operations until the next update.

For example, suppose region A and region B are connected and new managed nodes are added to region B. Until a resource update occurs between region A and region B, region A will not have current information on the managed nodes in region B and will not be able to effectively manage the new resources.

Likewise, if a resource is changed (for example, its name is changed) or deleted in the remote region, this change is not visible in the local name registry until the next update. Therefore, the scheduling and timing of resource updates must be balanced—often enough to guarantee current data about resources in remote regions, but not so often that system performance is degraded.

One exception to remote resource visibility is the visibility of resources in collections. When an administrator opens a collection, she or he sees the names of the various resources contained in that collection. When regions are connected, one or more of these collections can reside in the remote region and contain resources that also reside in the remote region.

If a resource is added to a collection in a remote region and the name of that collection has previously been pulled into the local name registry, the newly created resource might be visible in the local region as soon as it is created, even though the name of the resource itself is not entered into the local name registry until the next update of resources.

It is important to remember that resource updates copy only the *names* of remote resources into the local name registry, not the resources themselves. All operations on remote resources actually take place where the resource really resides—in the remote region.

Chapter 3. Overview

With Tivoli Management Framework, you can install and create several services that enable you to manage the resources in your network. This chapter provides an overview of the following Tivoli Management Framework components:

- "User interfaces"
- "Components and services"
- "Security and authentication" on page 32

For information about the object-oriented environment and management database based on the Common Object Request Broker Architecture (CORBA) specifications, refer to the *TME 10™ ADE Framework Services Manual* available through your Tivoli support provider.

User interfaces

Tivoli Management Framework provides two primary user interfaces: the desktop graphical user interface (GUI) and the command line interface (CLI). The GUI is often referred to as the *Tivoli desktop*. In addition to the desktop and the CLI, there are Web-based views of certain Tivoli management functions.

Depending on the operations you are performing, you use either the desktop or the CLI. For example, to view the relationships of policy regions and profile managers, use the desktop to visually display those relationships. The CLI can be more useful to perform repetitive actions, such as encapsulating Tivoli commands in a script.

Components and services

Tivoli Management Framework provides the following components and services:

- Administrators
- Policy and policy regions
- Resource types
- Scheduler
- Notification
- Task library
- RDBMS Interface Module (RIM)

Tivoli software products use these components and services much like software applications use the services offered by an operating system.

In addition to the components and services described in the following sections, Chapter 4, "Profiles and profile managers," on page 35 and Chapter 6, "Multiplexed distribution services," on page 63 describe these specific components in more detail.

Administrators

Tivoli software products use administrators to delegate the use of the system root account without giving those administrators the system password or complete control. Most system administrators have a Tivoli administrator that maps to their system account. However, users on multiple systems can use the same Tivoli administrator. A *Tivoli administrator* is the person or group of people each with a user account to access a computer system where Tivoli software products are installed.

Tivoli administrators can perform system management tasks and manage policy regions in one or more networks. When the Tivoli server is installed, an initial administrator, or *root administrator*, is created. A root administrator has root authority on UNIX operating systems and Administrator privileges on Windows operating systems. A root administrator can delegate system management tasks to other administrators by doing the following:

- Creating new administrators and assigning authorization roles
- Moving or copying resources and policy regions between desktops
- Granting root authority to other administrators

Each system administrator defined as a Tivoli administrator has a Tivoli desktop. When the administrator logs into the Tivoli environment, the desktop associated with that login is displayed. Each desktop contains the bulletin board where the administrator can read messages and contains the resources that the administrator can manage. From the desktop, an administrator can perform assigned system management operations on resources providing that the administrator has the required authorization roles.

Authorization roles

When root administrators create other administrators, they specify the resources and authorization roles for the new administrator. The authorization roles assigned to an administrator determine the management operations that the administrator can perform. Authorization roles provide a set of privileges to Tivoli resources. Authorization roles are mutually exclusive. Each role provides its own set of privileges—one role does not provide privileges of any other role.

The possible authorization roles for administrators are as follows:

super
Allows an administrator to connect and disconnect regions; perform maintenance operations on the region; install managed nodes, products, and patches; and so on. The **super** role is normally required for high-security operations and is not typically assigned to many administrators.

senior
Allows an administrator to create and define all Tivoli resources. The **senior** role is required for configuration and policy operations, such as creating an administrator, setting policy, or expiring notices.

admin
Allows an administrator to perform day-to-day tasks and system configurations. The **admin** role is required for general system management operations, such as adding an item to a profile, distributing a profile, or changing the message of the day.

user
Allows an administrator to view information and resources with read-only capability. The **user** role is required to bring up a Tivoli desktop and allows limited operations that do not affect configuration information, such as displaying configuration information.

backup
Allows an administrator to create backups of Tivoli object databases. An administrator must have the **backup** role in the region that contains the object databases for the Tivoli server and managed nodes to be backed up.

restore
: Allows an administrator to restore Tivoli object databases from backup. The administrator must have the **restore** role in the region that contains the object databases for the Tivoli server and managed nodes to be restored.

install-product
: Allows an administrator to install new applications and products in the local Tivoli region.

install-client
: Allows an administrator to install managed nodes within policy regions that allow the ManagedNode resource type.

policy
: Allows an administrator with both the **policy** and **senior** roles to create policy regions. In addition, the administrator can add resource types to policy regions and set up the policies that govern the policy region.

Dist_control
: Allows an administrator to control multiplexed distribution 2 (MDist 2) distributions.

Note: Depending on the Tivoli products installed, other authorization roles might be available. Refer to the product documentation for more information.

Scope of authorization roles

For example, some operation requires the **admin** role. If an administrator has only the **super** role, this administrator cannot perform these operations unless this administrator also is granted the **admin** role.

Note: To ensure that senior system administrators can perform operations at their authorization level and below, assign these administrators all authorization roles below their current level. For example, to ensure that an administrator with the **senior** role can perform all operations at the **senior** level and below, assign this administrator the **senior**, **admin**, and **user** roles.

Roles should be assigned based on the management operations specific to an administrator. You should carefully plan how you are going to delegate system management tasks. Tivoli Management Framework provides control and flexibility in assigning management authority to various personnel. Carefully consider and review the areas of responsibility for each Tivoli administrator when assigning roles for various resources and in various Tivoli regions. For example, Juan is to manage Documentation policy region. He should be assigned the **senior**, **admin**, and **user** roles for this policy region. If Juan has administrative needs for other policy regions or resources outside of his policy region, these can be granted later.

Authorization roles can be granted at the resource-level or region-level.

Resource authorization roles: Granting roles at the resource-level provides an administrator with authority to resources across policy regions, the Scheduler, or the Administrators collection.

Administrators can have different roles for different resources. Resource authorization roles can be assigned for many types of resources that appear as icons on a Tivoli desktop. If an administrator has a resource role, that role applies for all objects contained in that resource. For example, if John has the **senior** role in the Marketing policy region, he also has the **senior** role for all resources in that policy region.

Region authorization roles: Granting roles at the region-level provides an administrator with authority over all resources in the Tivoli region. If an administrator has an authorization role in a Tivoli region, that same role applies for all resources in that region. For example, if Isabella has the role **admin** in the region, she also has the **admin** role for all resources in that region.

An administrator with the **super** or **senior** role in the Administrator collection can create other administrators and assign them authorization roles. For security reasons, use extreme care when assigning the **super** role in a Tivoli region.

Roles across region boundaries: Most roles map across region boundaries. However some roles do not.

A region role enables an administrator to perform operations that can affect resources anywhere in the local region. If an administrator has any Tivoli role other than **super**, **install-client**, or **install-product**, the role maps across region boundaries. The **super**, **install-client**, or **install-product** role maps to **user** when crossing a region boundary.

Administrators with the **super** role can perform tasks that require the **super** role only in the region where the administrator was created. For an administrator to have the **super** role in more than one region, this administrator must be created in each region.

Notes:

- Although an administrator has the **super** role in more than one region, operations that require the super role do not complete successfully across boundaries.
- The **super** to **user** mapping is a security precaution that prevents an administrator from accidentally getting the **super** role in all regions.

Administrators and creating administrators

The root administrator is granted the **super** and **senior** roles during the installation of the Tivoli server. By default, when a new administrator is created, the new administrator has no authorization roles. If the new administrator needs to perform system management tasks, the necessary roles must be granted. The root administrator can change other administrators to root administrators, but those root administrators have only the roles they had before they became root administrators. The following applies when granting roles:

- The root administrator created during the installation of the Tivoli server can grant any role to any administrator.
- Non-root administrators can grant only roles that they have.
- Non-root administrators cannot grant roles to root administrators.
- Non-root administrators cannot grant roles to administrators that have any differing roles.

  For example, Dana and Jonobie are non-root administrators. Dana has **senior**, **admin**, and **user**. Jonobie has **super**, **senior**, and **admin**. Because Dana and Jonobie do not have the exact same roles, they cannot grant roles to each other.

Policy and policy regions

You can use policies to customize Tivoli products to fit your needs. A *policy* is a rule that you put into effect for a system and that is enforced as management operations are performed by administrators. For each policy, there are actually two policies: a default policy and a validation policy. A *default policy* is a set of resource properties assigned to a resource when it the resource is created. A *validation policy* ensures that all resources in a policy region comply with the established default policy for the policy region. For example, you can implement a policy that determines where a particular task or job can run or where user home directories are located. For many environments, policies are simply a set of conventions that may or may not always be followed.

Tivoli Management Framework provides an advanced, multilevel policy facility that enables you to encode acceptable policies and values for individual resources. As a result, the rules for managing network and system management resources can be enforced. Using multiple levels of policies, you can securely delegate routine system management tasks to other administrators with the knowledge that they can only make changes within the constraints of your policy and rules.

You can make it easier to customize products by using policies to make Tivoli products reflect the way your systems are managed. If you have the **senior** role and the **policy** role over a resource or Tivoli region, you can implement organization-specific system administration rules to ensure that management operations are only performed within the bounds of your rules and procedures.

Example of policies in an organization

Suppose a new employee, Thea Garza, joins the accounting department in the New York office. She needs to be added as a new user on an accounting department system.

You have implemented a set of management policies for computer systems in the accounting department. These policies define the requirements for users of accounting workstations. Each accounting user must have the following characteristics:

- A password that is not null
- Membership in the UNIX bookkeeping group, bkkpng
- Privileges in the Windows bookkeeping group, bkkpng
- A home directory mounted from a file server
- A UNIX user ID (UID) between 2000 and 5000
- A valid Windows login ID in the Bookkeeping domain

When an administrator creates an account for Thea, the user requirements specified through your policies are enforced.

There is a greater potential for policy conflicts in a distributed environment. For example, Thea is promoted to the position of head of accounting. She now needs to log in to systems in the San Francisco, Chicago, and Miami offices, in addition to her own New York system. If the system administrator in Chicago has implemented a unique set of validation policies for the Chicago systems, Thea might not be able to log in to computer systems in Chicago. To prevent such a problem, the administrators at each office (New York, San Francisco, Chicago, and Miami) need to coordinate their policies so that users can log in to the required systems.

Policy regions

A *policy region* is a collection of resources that share one or more common policies. Policy regions are abstract entities that enable you to construct the management and organizational model of your distributed computing environment. They are analogous to directories in a file system in that they enable you to construct a model of the organization through which system management tasks and operations are performed. They can also be arbitrarily nested and can contain any desired set of managed resources.

For example, you can establish a policy region hierarchy that reflects the organization of a company, with top-level policy regions named Sales, Marketing, and Manufacturing. Each of these policy regions might contain subregions to further represent the organization; for example, Accounts Receivable and Accounts Payable under Sales. Different administrators might have different roles in these regions, giving them the ability to manage appropriate sets of resources.

Policy regions contain managed resources. The set of managed resources in a policy region depends on the applications and products in your Tivoli environment. The Tivoli managed resources include managed nodes, task libraries, and profile managers. A managed resource belongs to only one policy region—initially, the policy region in which it was created. Although you can move managed resources from one policy region to another, typically to reflect changes in organization or management, you cannot copy managed resources. Refer to "Managed resources" for more information.

Note: Although endpoints are not created by default in a policy region, you can add them to a policy region and they can be moved between policy regions in the same way as other managed resources.

In a policy region, the administrator can do the following:
- Check for policy conflicts before resources interact with each other
- Supply default resource property values to be used when a resource is created
- Enforce policy when a resource is added or changed

Policy subregions

A *policy subregion* is a policy region that is located in another policy region. When a policy subregion is created, it initially uses the resource and policy properties of the parent policy region. These can later be changed or customized to reflect specific needs and differences of the subregion.

For example, suppose a policy region allows a task to be run with any effective UID. If you want to allow some specific tasks to be run as root when created by a specific administrator, you can create a policy subregion for these tasks and set the task library validation policy for the subregion so that a task must be run as root.

Managed resources

A *managed resource* is a specific instance of a resource type that has a default policy defined in the policy region. To create a managed resource, a default policy is required; only resources that are managed resources can be created in a policy region. Managed resources represent system or network resources that you manage with Tivoli software products. They are used to effect changes that conform to the policy of a policy region. For example, managed nodes and task libraries are managed resources. To manage a resource, you must first install a managed resource application.

Within the context of a policy region, each resource type is listed as either a directly or indirectly managed resource. A *directly managed resource* is one that can be created and managed within the policy region. An *indirectly managed resource* is one that is created and managed within a profile manager. Refer to Chapter 4, “Profiles and profile managers,” on page 35 for details on policy in profile-based applications.

The list of managed resource types is a policy region property. Tivoli resources can only be created in or moved into a policy region in which the type is a managed resource. You can modify the managed resource types for a policy region to control the resource types managed in the region.

Tivoli Management Framework provides a default set of managed resource applications that can be used to manage resources. You can change the policy objects associated with a managed resource. You can also create policy objects and edit their methods to control the values or attributes set on the resource. However, to create new managed resources, you must use Application Development Environment.

While a managed resource type can be valid in several policy regions, each specific instance of a managed resource type belongs to only one policy region. A managed resource is a member of the policy region in which it is currently located, regardless of where the resource was created.

For example, the printer managed resource type can be valid in several policy regions, but a specific printer can be a managed resource in only one policy region. If you create the printer resource Mitchell in the Lancaster policy region, Mitchell is a member of the Lancaster policy region. If you later move Mitchell to the Bearcat policy region, Mitchell becomes a member of the Bearcat policy region.

Basic resources and exchangeability

Associated with each registered resource there is a type and exchangeability characteristic.

Types of basic resources

Each resource in the name registry belongs is one of the following categories that affects whether the resource can be exchanged with other connected Tivoli regions:

exchangeable

An exchangeable resource can be updated between regions. This is the default for all resource types.

Note: In some situations, it might be necessary to create a new resource type using the **wregister** command; the new resource will be exchangeable. Refer to the *Tivoli Management Framework Reference Manual* for usage and examples of the **wregister** command.

When an instance of an exchangeable resource type is created in a remote region and then updated in the local region, the name of that newly created instance gets entered into the name registry of the local region. Thus, the newly created instance becomes visible in the local region.

non-exchangeable

A non-exchangeable resource cannot be exchanged, or updated, between connected Tivoli regions. This category is for resources that need to be visible only within a single region.

Applications can have their own list of nonexchangeable resource types as well.

custom

A custom resource type must define for itself what happens when it is exchanged between Tivoli regions. When you update a resource type that is custom, the **Update_Resource** method is invoked on each instance of the resource type. The **Update_Resource** method performs any necessary customized operations as part of the update operation. Custom resources cannot use the region timestamp mechanism. Therefore, updating custom resources is generally more time-consuming than updating other resource types. The following platform resource types use the custom category:

- AdministratorCollection—To make administrators from a remote region visible in the Administrators collection in the local region, you must first update the AdministratorCollection resource type in the local region. Updating only the Administrator resource type puts the remote administrators in the local name registry, but does not make them visible in the local Administrators collection.
- TopLevelPolicyRegion—To make the top-level policy regions from a remote region visible in the top-level policy regions window in the local region, you must update the TopLevelPolicyRegion resource type from the remote region. Updating only the PolicyRegion resource type puts the policy regions from the remote region into the local name registry, but does not make them visible in the local top-level policy region window.
- TaskRepository—To perform application-specific update operations.

Description and exchangeability of basic resources

Table 1 contains the resources in a Tivoli environment provided by Tivoli Management Framework. With each resource is an description and an indication of whether it is exchanged between connected Tivoli regions.

Table 1. Tivoli Management Framework resources and their exchangeability across region boundaries

Resource Type	**Description**	**Exchangeable**
ActiveDesktopList	A list of currently active desktops in the local region	No
Administrator	Name registry list of administrators and logins	Yes
AdministratorCollection	List of administrators as seen from the Administrator collection view	Yes
Classes	A list of classes with easily identifiable names in the local region	No
distinguished	A list of distinguished (one-of-a-kind) objects in the local region	No
Endpoint	Name registry list of endpoints	Yes
EndpointManager	Name registry list of all endpoint manager	Yes
Gateway	Name registry list of gateways	Yes
Job	Name registry list of task library jobs	Yes
LCF_NtLcfInstall	Name registry list of the Windows endpoint used by Tivoli Management Framework to install Windows other endpoints	Yes

Table 1. Tivoli Management Framework resources and their exchangeability across region boundaries (continued)

Resource Type	**Description**	**Exchangeable**
ManagedNode	Name registry list of managed nodes	Yes
PatchInfo	A list of patch information objects in the local region	No
PolicyRegion	Name registry list of policy regions	Yes
Presentation	A list of presentation objects in the local region	No
ProductInfo	A list of product information objects in the local region	No
ProfileEndpoint	An abstract resource type that contains no instances but is used by the **wgetallinst** command to look up resources that are profile endpoints (targets)	No
ProfileManager	Name registry list of profile managers	Yes
Query	Name registry list of queries	Yes
QueryLibrary	Name registry list of query libraries	Yes
RemoveNode	Name registry list of managed nodes that are removed from the administrator view but still remain in the object database	Yes
RIM	Name registry list of RIM objects in the local region	No
Repeater	Name registry list of repeaters	Yes
Scheduler	The scheduler object for the local region	No
SisLcfInstall	Name registry list of of the endpoint used by Tivoli Software Installation Service to install other endpoints	Yes
TaskLibrary	Name registry list of task libraries	Yes
TaskRepository	Name registry list of the definition of task library tasks	Yes
TMF_Notice	Name registry list of notice groups	Yes
TopLevelPolicyRegion	List of top-level policy regions as seen from the Top Level Policy Regions collection view	Yes

Task libraries

You can use Tivoli Management Framework to create a task library in which to store tasks and jobs. Tasks and jobs allow you to define a procedure and run it on multiple computer systems whenever necessary. Using tasks and jobs in this way hides the complexity of managing large, diverse networks from more junior operators.

A *task library* enables you to create and store tasks and jobs that can be run on one or more managed resources in a network. You can also use the task library to store executable files that are used by Tivoli applications. These executable files are invoked only by the application, not by the task library. In this situation, a task library provides the application with a known location for binaries, scripts, or programs that the application uses. The task library also provides information about options required to run each task it contains.

You can create multiple task libraries within each policy region and have different task libraries in different policy regions. Organizing task libraries by policy region is useful when you want to create sets of tasks that are specific for a particular group of resources or administrators. For example, you can create a separate task library in each policy region that contains all the tasks that must be performed on that policy region. This task library might include system backup tasks, printer operation tasks, and system monitoring tasks.

You can also create a separate task library for each group of related tasks and jobs. For example, you might create a task library named System Backup to store all tasks and jobs that perform system backup functions. Another task library might be called Printer Tasks to store all tasks and jobs that perform printer operations. You might perform the tasks contained in these task libraries on all the managed nodes in the policy region, in the Tivoli region, or in two or more connected regions.

A task library does not have to be exchanged in a resource exchange before its tasks can be performed across region boundaries. You can run a task on any *target* (endpoint, managed node, or profile manager subscriber) in any connected region. Depending on the Tivoli applications you are using, a task endpoint could be a managed node or an SQL server.

Tasks and jobs are subject to policy and, like other managed resources, this policy is enforced by the policy region that contains the task library in which the task or job is defined. The task library policy options include both default and validation policy for controlling such aspects as the allowable user and group IDs, the set of available task endpoints and profile managers on which a set of tasks may be run, and the task distribution mode.

Tasks

A *task* is a definition of something that needs to be done on a routine basis, such as clearing the printer queue, restarting a license daemon, or performing some other supported operation. Each of these operations is routinely performed on various computer systems throughout a network. In a diverse network, these operations must also be performed on a variety of computer systems and operating systems.

Tasks enable you to securely define a set of complex operations that you can easily run on any computer system, as necessary, without regard to platform type. Tasks are stored in a task library, so they can be used repeatedly. You do not have to redefine the operation each time a situation arises.

A task defines the executable files to be run when the task is executed, the role required to execute the task, and the user ID or group ID under which the task will run. A task does not contain the detailed information needed to execute the task, such as where the task will execute, what output type you would like, or the execution mode or parameters. You must specify this information when you execute the task.

When you create a task, you can specify a different executable file for each platform in your network. For example, the command to reset a printer queue varies between operating systems, but you can create a task that specifies the appropriate command to use for each platform. A single icon, therefore, can work for many different kinds of computer systems and operating systems.

Note: The platform executable files must exist before you create the task.

You control who is authorized to run a task by specifying the required authorization roles. After a task is created, only a Tivoli administrator with the required role in the policy region of the task endpoint can run the task. For more information about tasks and the task library, refer to the *Tivoli Management Framework User's Guide*.

Jobs

Jobs are tasks that are executed on specific managed resources. When you create a job, you select the task to be executed and define the execution information required to run the task. After a job is created, you can run the job on the specified resources without supplying additional information.

The execution information you specify when you create a job includes a list of managed resources on which the associated task will run and where the output of a task will be displayed. You can also specify the following information about a job:

- That it runs serially on each computer system
- That it runs in parallel on all computer systems
- That it is staged in groups of computer systems

When the execution mode is serial or staged, the task is executed on the managed resources in alphabetical order. Staging the execution is useful if you need to perform an operation on a large number of computer systems, but running the operation on all of the computer systems, either as a group or one after another, would tie up too many system resources. If you stage a job, limit the number of computer systems to 150 or less.

A job is defined by an application or component of Tivoli Management Framework. For example, a job created in a task library can be scheduled. Similarly, an application can define an operation that can be scheduled. The scheduler by itself does not provide a means to create new jobs, but rather offers a mechanism for running a job in the future.

Scheduler

The scheduler does not define jobs but does allow you to schedule previously created jobs. For example, you can schedule a job that has been created in the task library or a profile distribution.

The scheduler enables you to do the following:

- Schedule jobs to occur at specific times and within a specified time frame
- Schedule jobs to repeat a specified number of times at specified time intervals
- Schedule jobs to repeat indefinitely
- Restrict scheduled jobs to run only during the day, at night, during the week, or on weekends

You can browse scheduled jobs if you have the **user** authorization role, but most scheduler operations require at least the **admin** role, and some require the **senior** role. Any Tivoli administrator with at least the **admin** role in the scheduler can edit, disable, enable, or remove a job.

If you schedule a job, you are listed as the job owner until the job is deleted from the scheduler. This means that each time the job is run, it runs under your identity.

When the scheduler invokes a scheduled job, the role required to run the job is compared to information about the administrator who initially scheduled the job. If the administrator who initially scheduled the job no longer exists or no longer has the authorization necessary to run the job, the job execution fails. The scheduler notifies you of the status for the job. To avoid such job failures and the resulting job status notifications, you should regularly update the scheduler to remove unauthorized scheduled jobs. For example, you can sort jobs by administrator and remove any jobs that were scheduled by previous administrators.

Notification

Tivoli Management Framework provides a notification facility that tracks system administration activity. This facility informs Tivoli administrators of system management operations and reports which administrator performed a particular operation. The notification facility is especially useful in large installations that have many Tivoli administrators, because it can serve as an audit trail to allow for later determination of who performed certain actions in the system.

The following notification mechanisms are used to track system management activities:
- Notices
- Notice groups

Notices

A *notice* is a message concerning some operation or change in the Tivoli environment. Notices are generated by Tivoli operations. Only one copy of a notice is generated for a particular system management operation.

Message catalogs are used to store the message text in a language-neutral format. Message catalogs allow an administrator to view messages in a different language, if desired, based on the locale in which the desktop is run. This means that one administrator can view a notice in English, while another administrator can view the same notice in French.

When you view a notice, it is read from the notice database and displayed in the language you have chosen. The notice view manager keeps track of both read and unread notices. You can save a notice in a text file or forward it to other individuals by using the mail facility.

You can use notices to examine the types and numbers of resource operations. You can also use notices to determine how often system management tasks are performed by a given system administrator or for a given resource.

Notice groups

When notices are generated , they are sent to application-specific or operation-specific notice groups. A *notice group* stores and distributes messages pertaining to specified Tivoli operations. For example, the TME Administration notice group receives notices from such operations as creating a new administrator and changing the set of managed resources supported by a policy region.

Applications can also generate application-specific notices to report the success, failure, or specific details of an application operation. Applications typically create one or more application-specific notice groups to which application-specific notices are sent.

Tivoli Management Framework provides the following default set of notice groups:

TME Scheduler
Contains notices related to the operation of the scheduler.

TME Administration
Contains notices related to general Tivoli functions, such as the installation of applications and managed nodes, the addition of new administrators, and the creation and removal of policy regions and resources.

TME Authorization
Contains notices related to authorization errors, changes in administrator roles, and the creation of new administrators with authorization roles.

TME Diagnostics
Contains notices generated from running the **wchkdb** command or other maintenance operations. This notice group is generally used by senior administrators to aid in resolving problems.

Notices and multiple Tivoli regions

The notification facility operates in a slightly different manner if your environment has multiple connected Tivoli regions. Each Tivoli region has the same default notice groups. When two regions are connected, the local region is not automatically updated with the available notice groups in the remote region.

If you want to receive notices from a remote region, you must subscribe to the appropriate notice groups in the remote region. You can subscribe to a notice group in a remote region without having to subscribe to the corresponding notice group in the local region.

If you delete your subscription to a notice group in the local region, your subscription to the corresponding notice group in any remote region remains intact.

RDBMS Interface Module

The *RDBMS Interface Module* (RIM) provides a common interface that Tivoli applications can use to store and retrieve information from a number of relational databases. By using RIM, a Tivoli application can access any supported relational database in a common, database-independent manner. Storing information in an external database enables an application to take advantage of the power of relational database technology, such as SQL queries, to store and retrieve information.

Several Tivoli applications use RIM to hold their application data. In addition to Tivoli applications, the MDist 2 service also uses RIM to hold distribution data. Each RIM-supported application requires the administrator to do some configuration, such as creating a RIM object and setting passwords, when the application is installed.

Query facility

The *query facility* enables you to use SQL functions to access information in a RIM repository. A *RIM repository* is a relational database management system (RDBMS) database that a Tivoli application accesses through RIM.

The query facility consists of queries and query libraries. *Queries* specify which RIM repository to search and what information to retrieve. *Query libraries* reside in policy regions and are created to contain queries.

Security and authentication

Tivoli Management Framework provides a robust authentication model for system management tasks using encryption levels and provides the ability to substitute these encryptions using a Secure Socket Layer (SSL). This section provides information about security using the one provided by Tivoli Management Framework and using SSL.

Note: Tivoli Management Framework does not contain any **setuid** instructions, and therefore does not cause a security compromise in this manner.

Encryption levels

Tivoli Management Framework provides a facility for encrypting Tivoli security credentials that include sensitive data. These security credentials include sensitive data such as interregion passwords, Tivoli administrator logins and roles, and authentication requests and data. An encryption service is available for use when applications are designed to allow for the protection of sensitive application data.

Tivoli Management Framework provides three levels of encryption for Tivoli security credentials: **None**, **Simple**, and **DES**. Encryption is specified during the installation of the Tivoli server. The encryption provided by each level is as follows:

None Provides no encryption for Tivoli security data. Do not use this encryption level unless your network is completely secure; for example, by using SSL. There should not be any programs or programmers that can perform unauthorized network intrusions or tap into the network cabling. If such programs or programmers are present on the network, they might be able to break into Tivoli Management Framework as an imposter.

Simple

Provides an XOR-based encryption scheme that protects Tivoli security credentials from casual viewing. The impact of this encryption on performance is minimal. This encryption scheme is not unbreakable, but a nontrivial amount of time and effort is required to crack the encryption and view any Tivoli security data. The **Simple** encryption level is useful if your network is physically secure and there is not a threat of internal security breaches.

DES (Data Encryption Standard)

Provides a high level of security. **DES** is not completely unbreakable, but **DES** encryption is widely considered to be one of the most secure encryption schemes available. Tivoli Management Framework includes an implementation of **DES** for use in those environments where a high degree of security and authentication control is required. The **DES** encryption scheme protects Tivoli security credentials from being compromised by unauthorized network intrusions.

Notes:

- C2 security requirements do not impact or impair the ability to use Tivoli Management Framework. If your network operates under C2-level security, you can use Tivoli Management Framework to manage your network without compromising C2-level security, unless you set the encryption level to **None**.
- When you select the encryption level of **Simple** or **DES**, you must also provide an encryption password. The encryption password is used during the encryption process as a user-supplied key and is also required if you want to change the encryption level. You should protect encryption passwords as you would any password; like any compromised password, a compromised encryption password causes a risk of a security breach.

Intraregion and interregion encryption

You can set different encryption levels for Tivoli operations within a Tivoli region (intraregion) and Tivoli operations between connected regions (interregion). You also can specify different encryption passwords for intra- and interregion operations, even if both types use the same encryption level.

You can also specify a different encryption password for installing Tivoli Management Framework within a Tivoli region, thereby controlling who has the ability to create managed resources and install Tivoli applications.

Tivoli Management Framework allows you to establish different encryption levels for Tivoli operations within a Tivoli regionand between regions. You can also specify different encryption passwords for intraregion and interregion operations, even if both types of operations use the same encryption level. This allows you to have, for example, **Simple** encryption for communication within a Tivoli region and **DES** encryption for communication across regions. If you have two secure local networks but the connection between them is not completely trusted, using a higher encryption level for requests between the two regions can increase your level of confidence in the integrity of the installation.

You can use any mix of encryption levels and passwords for Tivoli intraregion, interregion, and intra-installation operations.

Secure sockets layer data encryption

Secure Sockets Layer (SSL) provides for secure communication through the use of public-key cryptography and digital signatures to provide SSL authentication, data integrity, and encryption. To protect the privacy of network traffic in the Tivoli environment, Tivoli Management Framework supports SSL encryption.

To implement SSL in your Tivoli environment, you need to have a working knowledge of the SSL protocol, key exchange (public and private), digital signatures, cryptographic algorithms, and certificate authorities. Although the SSL protocol supports built-in peer authentication, Tivoli Management Framework uses SSL to encrypt data only. Current authentication forms, such as shared secret keys, continue to function within the SSL data channel.

Note: It is possible to provide SSL authentication by manually manipulating the keystore contents using the iKeyman (**gsk4ikm**) key management utility on non-Linux managed nodes. On Linux managed nodes, use the OpenSSL key management utility.

Digital certificates and keys are used by SSL to authenticate participants and encrypt payload data. By default, Tivoli Management Framework uses SSL to encrypt data only. The SSL handshake still authenticates the participants as usual, but no additional trust can be associated with the keys used in the default Tivoli installation. However, you can replace the keys and certificates with your own, thus permitting authentication.

Tivoli Management Framework ships with the following default keystores. These files permit access by Tivoli services that have root permissions and user applications that might not have root privileges.

$DBDIR/Tivoli.kdb
Contains the private key and trusted certificates on a managed node. Because this keystore contains a private key, you should protect it so that it cannot be stolen and used to impersonate a managed node.

$DBDIR/TivoliCert.kdb
Contains only the trusted certificates.

Both files contain the certificate of the Tivoli certificate authority as a trusted signer. This means that the keystores trust your managed nodes throughout the world. In addition, Tivoli.kdb contains a Tivoli global signing key, which is signed by the Tivoli certificate authority. When a managed node starts for the first time, the Tivoli global signing key automatically generates and signs a unique private key that is trusted by all Tivoli product installations.

The advantage to using the default support provided by Tivoli Management Framework is to enable SSL encryption between managed nodes easily. One concern, however, is that SSL is not being used to provide additional trust, which poses the possibility of an imposter generating a private key that impersonates a managed node using the default keystores. Remember that Tivoli Management Framework still authenticates all operations as usual.

If you decide that you want to implement peer-to-peer authentication, you can remove the default Tivoli certificates and keys and replace them with your own.

Chapter 4. Profiles and profile managers

In large distributed networks, computer systems are frequently grouped according to the type of work for which they are used. For example, computer systems in an engineering group might be used to produce CAD drawings, while those in an accounting group might be used to produce tax documents. With Tivoli Management Framework, you can place common configuration information for computer systems used for similar purposes in a centralized area. Doing so makes it easier to access, manage, and duplicate resources. Profiles and profile managers enable you to do this.

This chapter defines profiles and profile mangers and describes the following topics:

- "Profiles, profile managers, and targets"
- "Types of profile managers" on page 37
- "Profile manager hierarchies" on page 39
- "Subscription and distribution of profiles" on page 41

Profiles, profile managers, and targets

A *profile* is a collection of specific information about a Tivoli software product. Each item in a profile contains system configuration information. The information in a profile is specific to the particular profile type. Profile records are stored in a platform-independent format that allows the same profile records to be distributed across an environment that contains multiple platforms.

A *profile manager* is a container for individual profiles. It provides a place to create and organize groups of profiles and to link *subscribers*, or recipients, to them. A profile manager can contain multiple profiles of the same type, or it can contain profiles of more than one type. Profile managers control the distribution of profiles to subscribers. Profile managers are created within a policy region. Subscribers to a profile manager can be in the same policy region as their profile manager or in other policy regions.

A profile manager can be viewed logically as having two sections. One section contains profiles and the other section contains subscribers. The set of profiles contained in a profile manager might be of different types. For example in Figure 5, the profile manager has a monitoring profile (M) and a user profile (U). The profile manager in this figure has two subscribers, managed node MN A and managed node MN B.

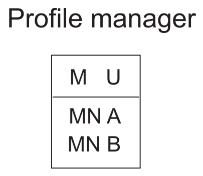


Figure 5. Profile manager with two subscribers and two profiles

Note: The illustration depicted in Figure 5 on page 35 is used throughout this chapter as a representation of a profile manager and its associated profiles and subscribers.

A *target* is a system resource, such as a managed node, endpoint, or Network Information Services (NIS) domain, that is the final destination of a profile. Tivoli Management Framework views each target as a subscriber to a profile.

It is important to distinguish between endpoints and targets:

endpoint
A computer system that is running the endpoint service (lcfd).

target Any managed resource that represents the final destination for a profile distribution.

After a profile reaches a target, the profile data is distributed to an application-specific component on the target, which interprets the profile data and adds, overwrites, or modifies the appropriate system files and profile databases on the target. In addition, profiles sometimes cause targets to perform an action or send data, or both.

Note: All endpoints do not accept all profile types. For example, NIS domain endpoints that are subscribed to a profile manager containing both user profiles and monitoring profiles only accept the records in the user profile; all records in the monitoring profile are ignored.

For the profile manager, the subscriber section defines those computer systems that are candidates to receive the profiles. At the time of distribution, the administrator can choose to select a subset of subscribers for the particular distribution (as well as a subset of the profiles) and can also choose various options related to the distribution. A computer system, such as a managed node or an endpoint, can appear on numerous subscriber lists for multiple profile managers. For example, a system in use by a secretary in Austin will be a part of the Secretary and Austin subscription lists. When a computer system subscribes to a profile manager, it is equivalent to subscribing to all profiles contained in the profile manager.

Profile managers can be subscribers to other profile managers (see Figure 6 on page 37). This provides a mechanism for building hierarchical distribution chains. For example, you can define a profile manager (PM B) that contains no profiles, but only a list of subscribers (MN A and MN B). This profile manager (PM B) could be subscribed to another profile manager (PM A). Profiles distributed from the parent profile manager (PM A) are distributed to the subscribed profile manager (PM B) and then to its subscribers (MN A and MN B). In this way, the subscribed profile manager (PM B) acts like a distribution list in an e-mail system.

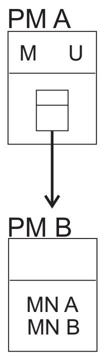


Figure 6. Subscribing a profile manager as the target for a profile distribution

Types of profile managers

Profile managers can operate in two modes: database or dataless.

database mode
: Enables a profile manager to distribute to any profile manager (dataless or database), managed nodes, and NIS domains, but not to endpoints.

dataless mode
: Enables a profile manager to distribute to both managed nodes, endpoints, and NIS domains, but not to other profile managers.

You select the mode for a profile manager when you create it. You can also change the mode after it is created.

Profile managers in database Mode

In database mode, a profile manager distributes profiles to the profile database of its subscribers. The *profile database* is the part of the Tivoli object database that maintains profile and subscriber information. If the subscriber is a target, the profile is stored in the profile database (as a local copy) and data is written to the computer system either during the same operation or during a later distribution.

Note: The *database* of database profile manager refers to the *target* of the distribution, not the database where the profile manager keeps its own information.

Because endpoints do not have a profile database to write to, database profile managers cannot distribute to endpoints.

Some profiles have multiple records contained in them and are designed to modify one or more system files when distributed. In this scenario, when you distribute two or more profiles of the same type to the same system, it is feasible to merge the records. This merging is carried out in the profile database of the managed node or profile manager. To illustrate profile merging, consider a profile that contains the phone lists. This phone list profile simply updates a file called phone.out on the target. The contents of the file are phone records that are contained in the phone list profile. To distribute profile P1 and profile P2, which contain different data, to the same system, first distribute P1 to the target managed node, MN A. The target file contains the three records that were in P1. See Figure 7 on page 38 on page 38.

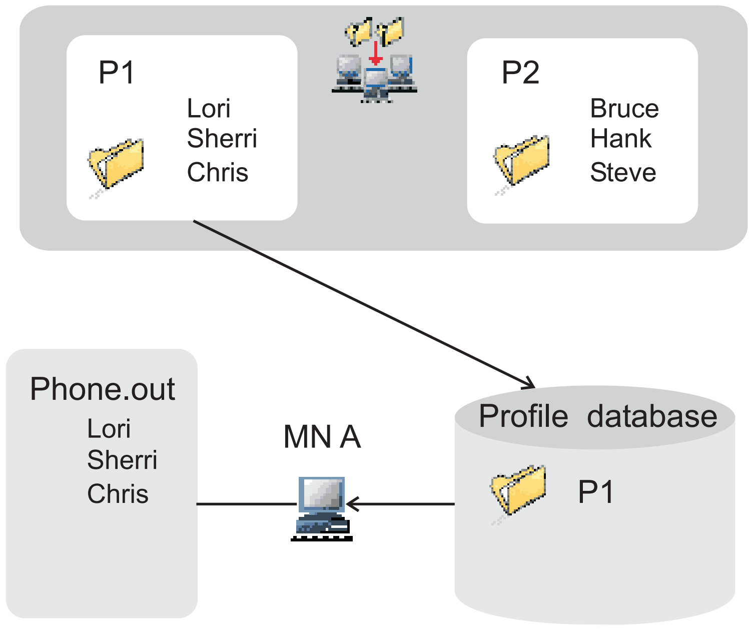


Figure 7. Distributing a profile from a database profile manager

Next, distribute profile P2 to the same target system. Rather than replacing phone.out with only the contents of P2, Tivoli Management Framework merges the records from both profiles, as shown in Figure 8.

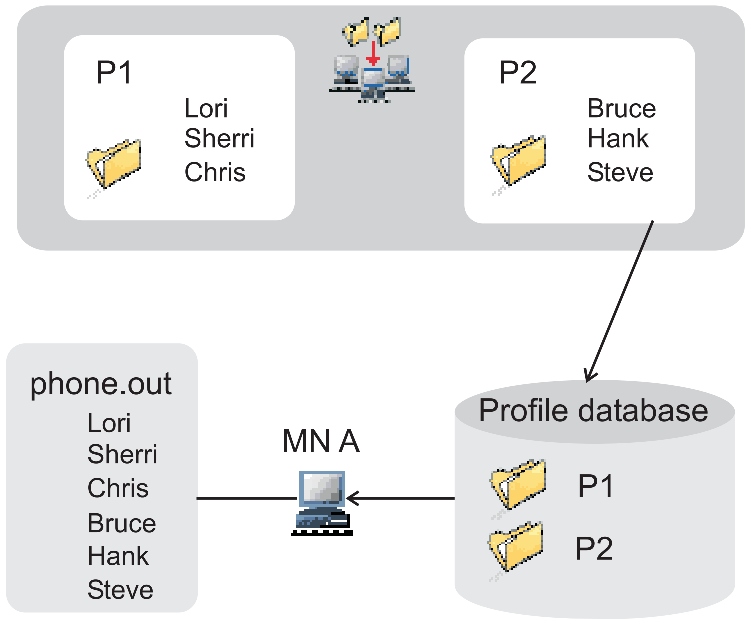


Figure 8. Merging a profile from a database profile manager

Profile managers in dataless mode

In dataless mode, a profile manager distributes profile to its subscribers, but does not store local copies of the profiles. Instead, it transfers the profile data contained directly to the application methods that applies the changes to the computer system. Therefore, you cannot subscribe another profile manager to a profile manager in dataless mode, because that would require the profile manager to attempt to write the profiles to a nonexistent profile database.

Note: A dataless profile manager has a profile database of its own, in which it keeps profiles and subscriber information. It is called dataless because it distributes profiles without regard to the existence of a database on its subscribers.

Although dataless profile managers are for endpoints, you can subscribe other managed system types. The profile data, however, is applied directly to the system files and profile is not stored as a local copy. For example, you can subscribe a managed node to a dataless profile manager and distribute a profile to it. In this scenario, the managed node functions like an endpoint, processing the profile data without copying the profile to its profile database.

Because profiles in dataless profile managers are not retained on the target system after distribution, the target system cannot track previous profile distributions. This means that the target cannot merge profiles before changing system files. If merging profiles is required, it must be done in a database profile manager.

Profile manager hierarchies

Creating profile manager hierarchies helps to organize the distribution of profiles to the appropriate targets. In general, a profile manager should not contain a mixture of targets and other profile managers.

Subscriber profile managers

Subscriber profile managers, profile managers that do not explicitly contain profiles other than by distribution from other profile managers, can be used to group subscribers for organizational purposes and to create profile manager hierarchies. For example, you can create a profile manager for each type of operating system that you maintain, or by department, location, and so on. Figure 9 shows groupings of IBM® OS/2® (PM OS/2), AIX® (PM AIX), Hewlett-Packard (PM HP), Windows 2000 (PM 2K), and Windows NT® (PM NT) operating systems.

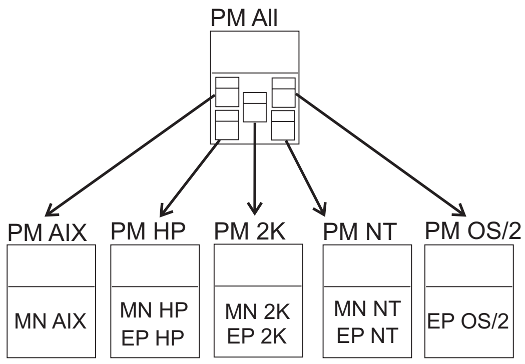


Figure 9. Profile manager hierarchy for operating systems

When you add a new system, ensure that the new system is a subscriber to all appropriate profile managers. Also, if you add a new profile manager that must have all systems as subscribers, you must individually select each system as a subscriber to the new profile manager.

With a subscriber profile manager, when you add a new AIX system to your Tivoli region and want it to receive all profiles that your existing AIX systems receive, you can simply subscribe the new system to the AIX subscriber profile manager. This automatically subscribes the system to all profile managers to which the AIX subscriber profile manager is subscribed. Also, when you create a new profile manager that must have all AIX systems as subscribers, you subscribe the AIX subscriber profile manager to the new profile manager, and all AIX systems are added.

Recommendations for creating profile manager hierarchies

Build profile manager hierarchies with the following considerations:

- Higher level profile managers are running in database mode.
- Higher level profile managers contain only profile managers as subscribers.
- Only the lowest level profile managers have targets as subscribers.

Consider the hierarchy in Figure 10.

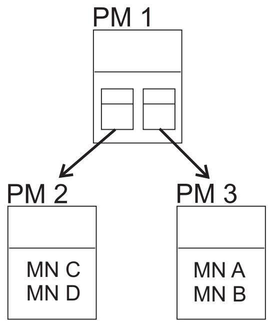


Figure 10. Profile manager hierarchy with targets subscribed at lowest level

In this profile manager hierarchy:

- Profile manager PM 1 has profile managers PM 2 and PM 3 as subscribers.
- PM 2 has managed nodes MN C and MN D as subscribers.
- PM 3 has managed nodes MN A and MN B as subscribers.
- PM 1 must be in database mode, because it has profile managers as subscribers.
- PM 2 and PM 3 are dataless profile managers. They contain only managed nodes as subscribers, hence they can be either in dataless or database mode.

Figure 11 on page 41 represents a profile manager hierarchy that makes it easier to add endpoints to your environment. Targets in the figure are managed system types, such as managed nodes and endpoints, with a monitoring profile (M) and a user profile (U).

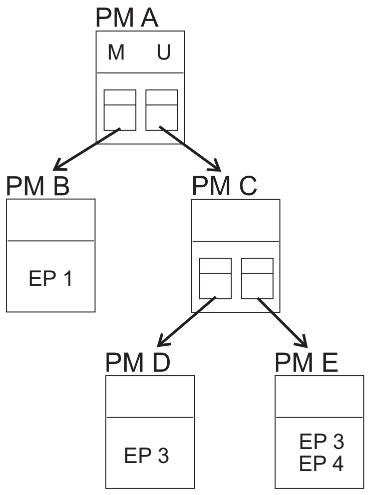


Figure 11. Profile manager hierarchy practical for adding endpoints

Subscription and distribution of profiles

For each level in a profile manager hierarchy, the profile manager maintains a record of settings for each of its profiles. The records indicate which profile items and fields were set locally and which were received from a source profile. This record-keeping is performed for each profile database. When a profile is distributed, the profile can be marked either **Preserve Modification** or **Make Exact Copy**.

When **Preserve Modification** is specified, the profile manager controls the reconciliation of all differences between local profile records and the original (source) profile records. The *original profile* is defined in the top-level profile manager. The *local copy of the profile* is the copy in the profile database of the subscribers. When a profile manager distributes a profile copy to a subscriber, the source data is merged with the local profile on the subscriber. The resulting local profile data is distributed down the hierarchy. This process continues recursively until a target is reached.

However, if you perform a distribution and specify **Make Exact Copy**, all profile records are distributed and subscriber profile records are completely replaced by the source profile records. This is necessary to update all profile subscribers so that they have an exact copy of the distributed profiles.

Distributing profiles with local modifications

Targets that subscribe to a profile manager are eligible to receive all profiles contained in the profile manager. However, targets that subscribe to a particular profile manager do not have to have identical configurations. When profile data is distributed, a copy of the profile data is created in each profile database. You can make local modifications to either an individual computer system or a portion of the profile manager hierarchy. You can also specify whether to preserve local profile modifications or replace the local profile copy with an exact copy of the profile distribution. For information about adding profile items, modifying the properties of existing profile items, and deleting profile items from a profile, see the appropriate Tivoli software product documentation.

Note: For ease of profile maintenance and accountability, it is recommended that you do not make and preserve many local modifications.

In Figure 12, profile manager PM A contains profiles P1, P2, and P3 and can distribute to it subscribers, managed node MN A and profile manager PM B. Profile manager PM B contains profiles P4, P5, and P6 and can distribute to its subscriber, managed node MN B. When profile manager PM A distributes it profiles, managed node MN A and profile manager PM B receive profiles P1, P2, and P3. Because profile manager PM B contains profiles P4, P5, P6, managed node MN B receives all profiles (P1 through P6) where profiles P1, P2, and P3 are local copies that can be modified.

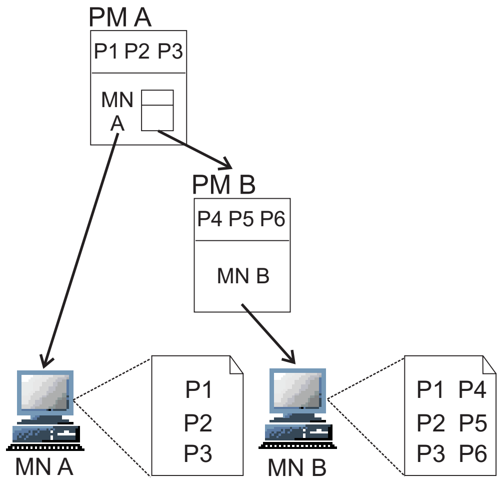


Figure 12. Distributing profiles

In Figure 13 on page 43, profile manager PM A contains profiles P1, P2, and P3 and can distribute to it subscribers, managed node MN A and profile manager PM B. Profile manager PM B contains no profiles and can distribute to its subscriber, managed nodes MN B and MN C. When profile manager PM A distributes its profiles, managed node A and profile manager PM B receive profiles P1, P2, and P3. You make modification to the local copies of profiles P2 and P3, which stores them in the local profile database as P2a and P3a, respectively. When PM B distributes it profiles, managed nodes MN B and MN C receive profile P1 and the modified profile copies P2a and P3a.

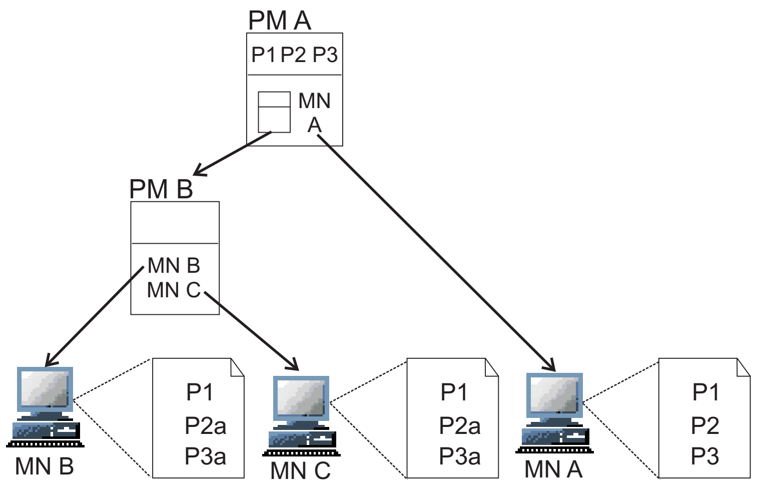


Figure 13. Distributing profiles with local modifications

Distributing exact copies of profiles

When you distribute a profile, you can specify that you want an exact copy of the profile distributed to the targets. That is, any profile setting that already exists on the target that are not part of the profile are deleted during the distribution.

When a profile is distributed with the exact copy option, Tivoli Management Framework deletes any entries in the system or application file that are not in the profile.

For example, assume a user profile (U) contains records 1, 2, and 3 and the distribution endpoint contains records 1, 2, 5, and 7. Following an exact copy distribution, record 3 is added to the files on the distribution endpoint and records 5 and 7 are deleted. Figure 14 illustrates this example.

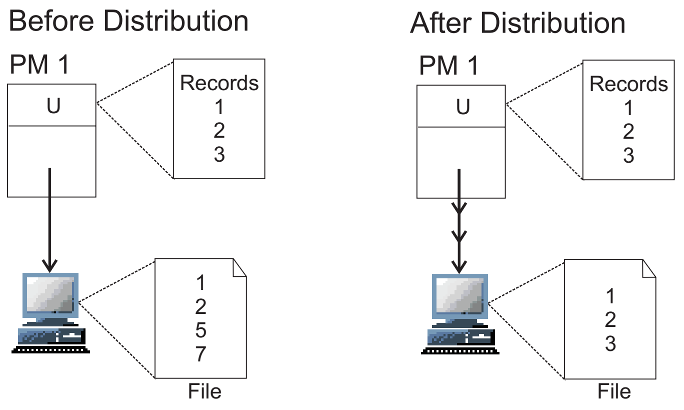


Figure 14. Exact copy distribution

The **Make Exact Copy** option works differently for database profile managers and dataless profile managers. The hierarchy of the profile managers, along with the type of the profile determine how the exact copy works.

For database profile managers, the records are merged. With the exact copy option specified and when the profiles are distributed to the target, the exact copy works as shown in Figure 14 on page 43.

For dataless profile managers, which distribute the profiles directly to the targets, the records in the last profile distributed with the **Make Exact Copy** option are the only ones that remain on the target. To successfully merge records in profiles in database profile managers, create another dataless profile manager to act as the converging profile manager.

The next example describes the **Make Exact Copy** option. After profile A is distributed to endpoint fido with the **Make Exact Copy** option specified, the system file on fido contains only the records in profile A (records 1 and 2). Profile B has not been distributed, so profile manager PM 1 does not contain its records. Records 3, 4, 8, and 9 are deleted. The result is illustrated in Figure 15.

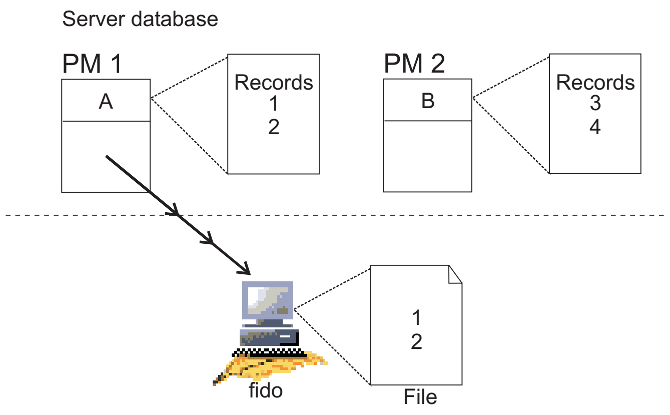


Figure 15. Distributing an exact copy of profile A in a dataless profile manager without a converging profile manager

If profile B is subsequently distributed to endpoint fido with the **Make Exact Copy** option, the system file on fido contains only the records in profile B (records 3 and 4), as shown in Figure 16.

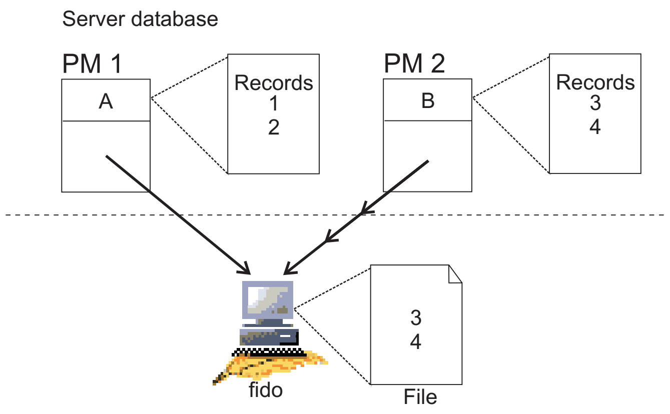


Figure 16. Distributing an exact copy of Profile B in a dataless profile manager without a converging profile manager

If you later distribute either profile A or B to managed node fido using the **Make Exact Copy** option, the records not contained in the merged profile (in this example, records 8 and 9) are deleted from the system file on fido. This is illustrated by Figure 17.

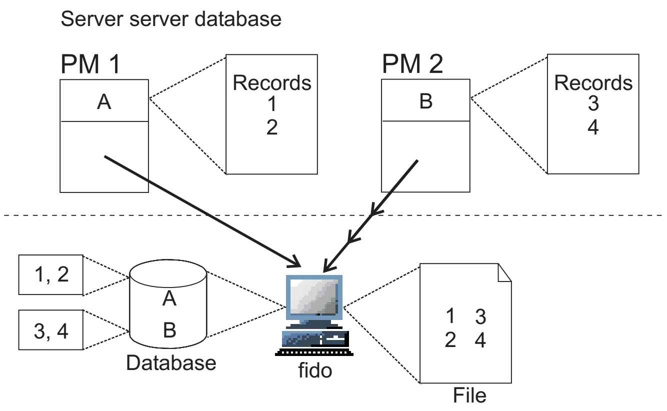


Figure 17. Using a converging profile manager to merge records and distribute the appropriate exact copy of the records to the target

Using the Preserve Modifications and Make Exact Copy options effectively

The profile manager hierarchy, the type of profile manager, and the type of modifications you specify all affect the system or application files. As profiles are distributed with the **Make Exact Copy** option or the **Preserve Modifications** option, files are created, deleted, and recreated.

This is important because deleting and recreating records in certain system or application files can be problematic for some operating systems, such as Windows 2000 and Hewlett-Packard trusted systems. To avoid unnecessary deletions, consider doing one of the following:

- Distribute all but the last profile to all levels using the **Preserve Modifications** option. This merges the records from those profiles in the profile database (either on the managed node or in a converging profile manager). Then distribute the last profile to all levels using **Make Exact Copy**, which deletes records on the endpoint that are not in any of the merged profiles.
- Distribute all profiles to the next level of subscribers only. This enables the converging profile manager (or the profile database on a managed node) to receive all the profiles and create a complete set of records. You can then distribute from the converging profile manager to all levels using **Make Exact Copy**.

Chapter 5. Endpoints and gateways

An *endpoint* is a computer system running the Tivoli endpoint (lcfd) service. Endpoints receive profile distributions and run operations. Typically, computer systems installed as endpoints are not used for the daily management operations in a network. Instead, an endpoint is a computer system being managed from other managed nodes. You cannot, therefore, create, modify, or delete Tivoli objects from an endpoint. These functions are performed from a managed node, including the Tivoli server.

The main advantage of endpoints is scalability. You can gather required management information from thousands of endpoints and remotely manage those computer systems with very little overhead. Another advantage of endpoints is the comparatively small demand they make on computer resources. An endpoint installation requires only 1 to 2 megabytes (MB) of disk space as compared to the 85 MB required for a managed node or about 110 MB for a Tivoli server. Actual disk space requirements depend on the type and number of Tivoli applications installed.

Endpoints communicate with the remainder of the Tivoli region through its assigned gateways. A *gateway* is a Tivoli resource that is hosted on a managed node and provides the communication and methods required by the endpoint.

This chapter discusses the following topics:

- "Endpoint manager, gateway, and endpoint interaction"
- "Endpoint logins" on page 48
- "Endpoint configuration" on page 55
- "Sharing endpoint resources across Tivoli regions" on page 58
- "Endpoint migration" on page 59
- "Preferred gateways" on page 59
- "Method storage and implementation" on page 60
- "Considerations for NetWare and OS/2 gateways" on page 61

Endpoint manager, gateway, and endpoint interaction

Before an endpoint becomes a managed system, it must communicate and interact with the endpoint manager and gateways. The *endpoint manager* establishes and maintains the relationship between an endpoint and its assigned gateway. The endpoint manager is automatically created on the Tivoli server during installation. The primary role of an endpoint manager is to assign an endpoint to a gateway when the endpoint first logs in. If its assigned gateway stops responding to the endpoint for some reason, the endpoint manager would again be involved in assigning the endpoint to a new gateway. The endpoint manager is also responsible for identifying the gateways that an endpoint is assigned to when applications are trying to contact an endpoint.

Gateways assume some of the function of a Tivoli server. By shifting a share of the management processes to the gateway, the Tivoli server is freed to manage more clients. Each gateway can support thousands of endpoints. Gateways reside on managed nodes in most cases (see "Considerations for NetWare and OS/2 gateways" on page 61 for more information about gateways that reside on systems that do not have all the functionality of a managed node).

The endpoint performs all communications with its assigned gateway without requiring additional communications with the Tivoli server. The gateway invokes endpoint methods on the endpoints or runs gateway methods for the endpoint. To ensure communication between multiple endpoints and a gateway, make sure that more free processes are available on the gateway than assigned endpoints.

When a gateway is created or restarted, it contacts the endpoint manager for information about its assigned endpoints. As endpoints are assigned to the gateway, the endpoint information is passed to the gateway and stored in its endpoint list.

Gateways must be installed and running before you install and start endpoints. All gateways in a Tivoli region should listen on the same port. The default listening port is 9495.

Endpoint logins

An endpoint performs four types of logins: *initial*, *normal*, *isolated*, and *orphan*. First, the initial login is performed, and then the normal login is performed.

For a normal login sequence, the endpoint logs in to its assigned gateway and the gateway acknowledges it. The initial login process is more complex. This process establishes the endpoint as an active member of its Tivoli region by assigning it to a gateway.

An endpoint is isolated when it cannot contact its assigned gateway. When that occurs, it initiates an isolated login. In certain cases, an endpoint can become orphaned when the endpoint manager no longer has an entry in its database for the endpoint. Isolated and orphan logins are discussed in "Isolated login" on page 54 and "Orphan login" on page 54, respectively.

If the endpoint is unable to contact its assigned gateway, additional gateways are provided through a list of login interfaces or gateway addresses. This list is compiled by the endpoint manager, defined in the **select_gateway_policy** script, or configured with **lcfd** command options.

Note: Depending on how your environment supports network address translation (NAT), you might need to define the host names for gateways in the **select_gateway_policy** script. The gateways defined through **select_gateway_policy** or the **lcfd** command can be host names instead of object identifiers (OIDs).

To facilitate the login process and endpoint communication, configure login parameters during endpoint creation or with the **lcfd** command after installation. For more information about the **lcfd** command and the **select_gateway_policy** script, refer to the *Tivoli Management Framework Reference Manual*.

Initial login without a select_gateway_policy script

The following are the phases of the initial login process of an endpoint:

- The endpoint establishes communication with a Tivoli region.
- The endpoint manager selects a gateway to which the endpoint is assigned.

- The endpoint receives its gateway assignment and performs a normal login to the assigned gateway.

The first step in the initial login process for an endpoint is finding a gateway in the Tivoli region. The endpoint cannot be managed until it becomes associated with a gateway. The endpoint start by sending an initial login packet to a gateway, which acts as an intercepting gateway. An *intercepting gateway* handles communication and login for the endpoint until it has an identity and is assigned to a gateway. If no configuration options are set, either during installation or during the startup of the lcfd service, the endpoint broadcasts for a gateway. Because of network load, do not use broadcast as a means for endpoint login. Instead, use **lcfd –D bcast_disable=1** to disable broadcast and use **lcfd –D lcs.login_interfaces=***address* to specify a gateway.

Note: If your environment supports NAT devices that share IP address assignments through dynamic timeslicing, ensure that the IP address assignment timeout value is greater than the **login_timeout** and **udp_interval** settings on the endpoint.

Figure 18 illustrates the initial login process of an endpoint.

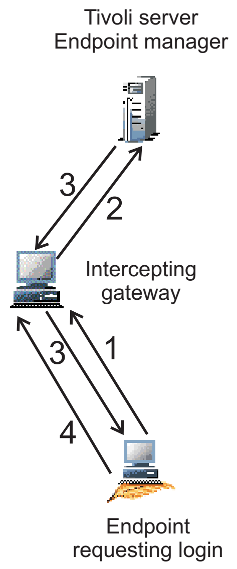


Figure 18. Initial login process of an endpoint

1. The endpoint login request is intercepted by a gateway.
2. The gateway forwards the login request to the endpoint manager.
3. Having no defined policy, the endpoint manager assigns the endpoint to the intercepting gateway and sends the new login information to the gateway. The endpoint manager adds up to five alternate gateways to the endpoint alternate gateway list. The gateway relays the information to the endpoint.
4. The endpoint logs in to its assigned gateway.

To summarize the initial login, the following are general parameters and default timeouts:

1. The endpoint uses a set of gateway addresses configured during installation to establish a gateway to receive the endpoint login request.

- These gateway addresses are specified by the **lcs.login_interfaces** option.
- By default, the endpoint attempts to contact each of the gateways three times with 5 minutes between each attempt. The attempts are specified by the **login_attempts** option. The time between each attempt is specified by the **login_timeout** option.
- If the endpoint is successful in contacting one of the alternate gateways, endpoint policy scripts run.

2. If login fails to all the addresses in the **lcs.login_interfaces** list, the endpoint broadcasts a request to log in (if broadcast is enabled).
3. If no gateways in the **lcs.login_interfaces** list respond or the broadcast login request fails, the endpoint waits 30 minutes (the **login_interval** default of the **lcfd** command) before beginning the login process again with step 1.
4. If the login is successful, the endpoint receives its identity in the Tivoli region from the intercepting gateway along with the name of its assigned gateway.
5. When logged in, the endpoint performs a normal login to its assigned gateway.

Initial login with a select_gateway_policy script

Defining the **select_gateway_policy** script provides the endpoint manager with an ordered list of gateways. The endpoint manager attempts to contact each listed gateway until it makes a valid connection. The first gateway to respond receives the endpoint assignment. For more information about endpoint policy, refer to "Implementing Policy Scripts" on page 56.

The endpoint manager assigns a gateway and adds the endpoint information and gateway assignment to its endpoint list. The endpoint manager establishes a unique identity for the endpoint. The endpoint information is sent back to the intercepting gateway. The intercepting gateway relays the assignment information to the endpoint. The endpoint performs a normal login to its assigned gateway.

Figure 19 on page 51 illustrates the initial login process for an endpoint, where a **select_gateway_policy** script has been defined listing two gateway options. The endpoint manager tries to contact the gateways in the order listed in **select_gateway_policy**. In this example, gateway A is listed first and gateway B is listed second. Gateway B is the first to respond to the endpoint manager.

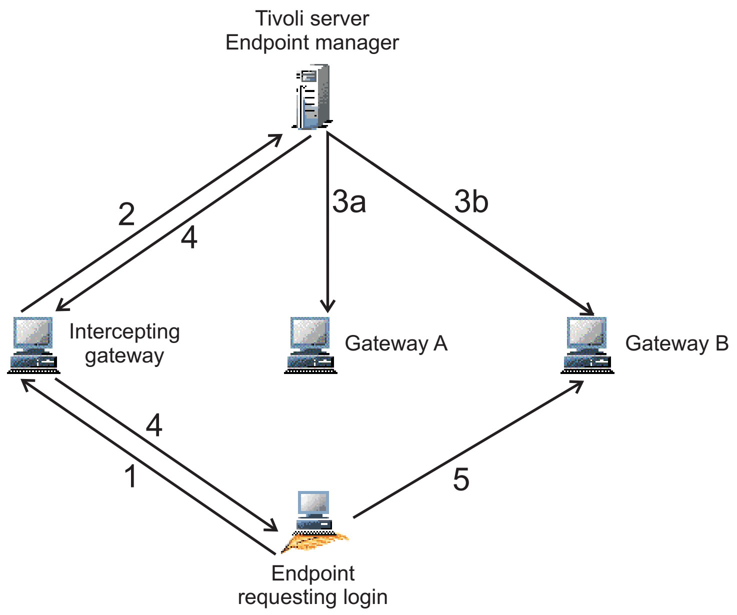


Figure 19. Endpoint initial login with select_gateway_policy defined

1. The endpoint login request is intercepted by a gateway.
2. The gateway forwards the login request to the endpoint manager.
3. The endpoint manager refers to the **select_gateway_policy** script and attempts to contact gateway A and then gateway B.
   a. Connection with gateway A fails.
   b. Contact with gateway B is successful. Gateway B becomes the assigned gateway for the endpoint. Gateway A and gateway B are added to the **lcs.login_interfaces** list.
4. The endpoint manager returns the login assignment information to the intercepting gateway. The intercepting gateway then relays the information to the endpoint.
5. The endpoint logs in to its assigned gateway.

In Figure 20 on page 52, an interconnected Tivoli region was created to redirect endpoint initial logins to their endpoint manager on the Tivoli server in the local Tivoli region. This scenario can be common in large, multisite enterprises where thousands of endpoints are logging in to multiple regions. A Tivoli region dedicated to redirecting endpoint logins can ensure that endpoints log in to gateways in their local region. Tivoli server A is the redirecting Tivoli region and has the **select_gateway_policy** script defined. Gateway A is local to Tivoli server B and is listed in the **select_gateway_policy** script.

Note: Endpoints must be managed in their local Tivoli region.

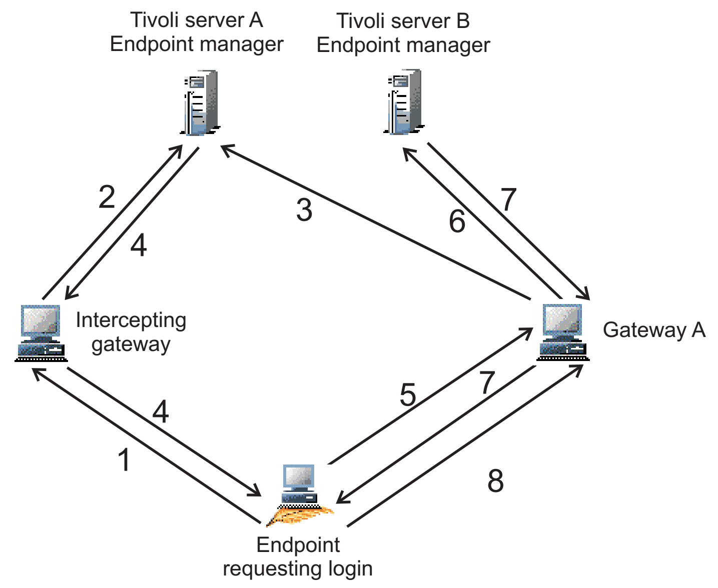


Figure 20. Endpoint redirection to another Tivoli region

1. The endpoint login request is intercepted by a gateway.
2. The gateway forwards the login request to the endpoint manager.
3. The endpoint manager refers to the **select_gateway_policy** script and finds gateway A in Tivoli region B first on the list. The endpoint manager retrieves gateway A's interface information.
4. The endpoint manager sends the network interface information of gateway A with directions to use it as the intercepting gateway to the original intercepting gateway. The gateway relays the information to the endpoint.
5. The endpoint begins the initial login process again with the new information. The login request is intercepted by gateway A, as designated in the policy script in region A. Gateway A is acting as an intercepting gateway.
6. The request is forwarded to the endpoint manager in region B.
7. Having no defined policy in region B, the endpoint manager assigns the endpoint to gateway A and sends the new login information to the gateway. The gateway relays the information to the endpoint.
8. The endpoint logs in to its assigned gateway.

Normal Login

After an endpoint is established as a member of its Tivoli region through the initial login process, subsequent or normal logins occur. During a normal login, communication takes place between the endpoint and gateway only. The endpoint sends a login packet directly to its assigned gateway stored in the lcf.dat file. See Figure 21 on page 53 for an illustration of normal login.

Because the gateway has the endpoint in its endpoint list, communication is established immediately without contacting the Tivoli server or the endpoint manager. The endpoint then performs the following operations:

- Writes current configuration information to the last.cfg file. This file is overwritten each time the configuration changes.
- Stores connection information in the encrypted lcf.dat file.
- Listens for method calls. The endpoint is now fully managed by Tivoli Management Framework.

Note: Refer to "Endpoint configuration" on page 55 for more information about the last.cfg and lcf.dat files.

Figure 21 illustrates the flow of data for logins after the first login. A normal login occurs when the endpoint service starts. After the normal login is successful, each communication to or from the endpoint is done as an upcall or downcall with the exception of an isolated login.

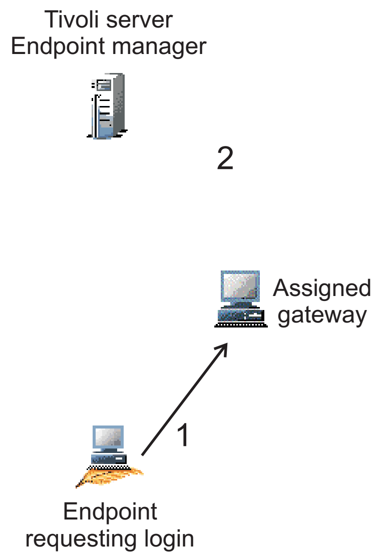


Figure 21. Endpoint normal login

1. The endpoint logs in to the assigned gateway. The endpoint is immediately established as a communicating member of the Tivoli network.
2. The endpoint manager is *not* contacted.

To summarize the normal login, the following are the general parameters and default timeouts:

1. Using the gateway list, which is given to the gateway by the endpoint manager, the endpoint attempts to contact its assigned gateway. If this fails, the endpoint attempts to contact any aliases of the assigned gateway. The attempts are specified by the **login_attempts** option. The time between each attempt is specified by the **login_timeout** option.
2. When the endpoint cannot contact its assigned gateway or aliases, the endpoint enters isolation mode and uses a set of gateway addresses (configured during installation) to contact a gateway to receive the endpoint login request:
   - These gateway addresses are specified by the **lcs.login_interfaces** option.
   - By default, the endpoint attempts to contact each of the gateways three times with 5 minutes between each attempt. The attempts are specified by the **login_attempts** option. The time between each attempt is specified by the **login_timeout** option.
   - If the endpoint is successful in contacting one of the alternate gateways, endpoint policy scripts run.
3. If login fails for all the addresses in the **lcs.login_interfaces** list, the endpoint broadcasts a request to log in (if broadcast is enabled).

4. If no gateways in the **lcs.login_interfaces** list respond or the broadcast login request fails, the endpoint waits 30 minutes (the **login_interval** default of the **lcfd** command) before beginning the login process again with step 1.

Isolated login

When an endpoint cannot contact its assigned gateway, the endpoint is considered isolated. The following are some examples of how an endpoint can become isolated:

- The gateway is down.
- There are problems with network connectivity to the gateway.

For the endpoint to be assigned quickly to a new gateway, each endpoint receives a list of alternate gateways when it receives its initial gateway assignment information. The list of alternate gateways can be defined in and provided by the **select_gateway_policy** script. If the **select_gateway_policy** script is not defined, the endpoint manager sends a list of up to five gateway addresses to try.

If the endpoint loses contact with its assigned gateway, the endpoint goes through a list of alternate gateways (and associated aliases) in its attempts to log in. If the endpoint fails to log in to any of the alternate gateways, the endpoint sends another isolated login packet. The login process is similar to the initial login process described in "Initial login without a select_gateway_policy script" on page 48 in that the gateway selection process is triggered. Also, the **lcs.login_interfaces** list is replaced with a new list of gateways, instead of appended with new gateways (as in the initial login).

To summarize the isolated login, the following are the general parameters and default timeouts:

1. When the endpoint cannot reach its assigned gateway, the endpoint uses a set of gateway addresses configured during installation to contact a gateway to receive the endpoint login request.
   - These gateway addresses are specified by the **lcs.login_interfaces** option.
   - By default, the endpoint attempts to contact each of the gateways three times with 5 minutes between each attempt. The attempts are specified by the **login_attempts** option. The time between each attempt is specified by the **login_timeout** option.
   - If the endpoint is successful in contacting one of the alternate gateways, endpoint policy scripts run.
2. If login fails for all the addresses in the **lcs.login_interfaces** list, the endpoint broadcasts a request to log in (if broadcast is enabled).
3. If no gateways in the **lcs.login_interfaces** list respond or the broadcast login request fails, the endpoint waits 30 minutes (the **login_interval** default of the **lcfd** command) before attempting to contact its assigned gateway again.

Orphan login

An endpoint is an orphan when the endpoint considers itself a member of a Tivoli region; however, the endpoint manager and Tivoli name registry do not recognize that the endpoint ever logged in. Thus, you will not be able to perform any actions on those endpoints, such as software distributions or inventory scans, until it rejoins the Tivoli region. Endpoints can become orphaned in the following cases:

- When restoring the endpoint manager database where endpoints have joined the region since the last backup was made

- By unintentionally deleting an endpoint from the endpoint manager database using the **wdelep** command

The first case occurs when you have to restore the Tivoli server from a backup. In most cases, backups are done on a regularly scheduled basis. After one of these backups, it is likely that new endpoints will log in to the region for the first time. These new endpoints, therefore, are recorded in the endpoint manager, but do not appear in the backup until the next scheduled backup occurs. When the database is restored from the backup, the new endpoints are no longer represented in the endpoint manager. The endpoints, however, still have the endpoint service running.

The second case occurs when the **wdelep** command is run inadvertently, and the endpoint service is not stopped and removed from the endpoint. Because the endpoint service runs independently from the endpoint manager, the endpoint does not know that the endpoint manager no longer knows about the endpoint. Thus, the endpoint still considers itself part of the region. Until the endpoint attempts an action that affects the endpoint manager, such as an upcall, the endpoint will not know it is an orphan. If the endpoint attempts to log in to its assigned gateway, it fails and enters isolation mode.

You can also use **allow_install_policy** and **select_gateway_policy** scripts to control how orphan endpoints are added back to the endpoint manager database. For security of individual regions, the endpoint cannot be redirected to another Tivoli region from its original one during the orphan login. Also, the **lcs.login_interfaces** list is replaced with a new list of gateways (as in the isolated login), instead of appended with new gateways (as in the initial login).

To summarize the orphan login, the following are the general parameters and default timeouts:

1. When the endpoint cannot reach its assigned gateway, the endpoint uses a set of gateway addresses configured during installation to contact a gateway to receive the endpoint login request:
   - These gateway addresses are specified by the **lcs.login_interfaces** option.
   - By default, the endpoint attempts to contact each of the gateways three times with 5 minutes between each attempt. The attempts are specified by the **login_attempts** option. The time between each attempt is specified by the **login_timeout** option.
   - If the endpoint is successful in contacting one of the alternate gateways, endpoint policy scripts run including any orphan endpoint parameters you have specified.
2. If login fails for all the addresses in the **lcs.login_interfaces** list, the endpoint broadcasts a request to log in (if broadcast is enabled).
3. If no gateways in the **lcs.login_interfaces** list respond or the broadcast login request fails, the endpoint waits 30 minutes (the **login_interval** default of the **lcfd** command) before attempting to contact its assigned gateway again.

Endpoint configuration

You can configure an endpoint in the following ways:

- Implementing policy scripts for use by gateways and the endpoint manager
- Using **lcfd** command options and their values in the last.cfg file
- Specifying **lcfd** command options during endpoint installation, which is explained in the *Tivoli Enterprise Installation Guide*

For information about configuring multi-NIC endpoints, see *Tivoli Enterprise Installation Guide*.

Implementing Policy Scripts

The endpoint manager and gateway include the ability to use policy scripts to perform certain actions at various stages of the endpoint login process. Endpoint policy differs from default and validation policy in that policy objects are not associated with the endpoint scripts. Refer to the section about endpoint policy scripts in the *Tivoli Management Framework Reference Manual* for option information and instructions for editing.

The run time of these policy scripts affects the number and efficiency of logins that the gateway and the endpoint manager can process at one time. For an environment with a large number of endpoints, limit the number of commands that are placed in the policy scripts, because commands might required long periods of time and large amounts of processing resources to run. In certain cases, the endpoint can become isolated after waiting too long for the gateway to respond, which can impact endpoint manager performance.

For example, you have 1,000 endpoints logging in to a gateway at approximately 9:00 a.m. Because the run time of the policy scripts take longer to complete for each login, additional logins have to wait for the preceding logins to complete. When the preceding logins complete, the gateway and the endpoint manager are available to process additional login requests. If you need to run Tivoli commands in this context, use endpoint policy scripts to trigger tasks after login. See the section about the task library in the *Tivoli Management Framework User's Guide* for information about how to create tasks.

Table 2 describes the origin (where the script runs) and the trigger (when the script runs) for each endpoint policy script:

Table 2. When and where endpoint policy scripts run

Policy script name	Origin	Trigger
allow_install_policy	Run by the endpoint manager	Run when the endpoint installation begins
select_gateway_policy	Run by the endpoint manager	Run each time an endpoint needs to be assigned to a gateway
after_install_policy	Run by the endpoint manager	Run directly following the endpoint installation and initial login
login_policy	Run by the gateway	Run each time the endpoint logs in

Note: For NetWare gateways, **login_policy** scripts are run on the gateway proxy.

allow_install_policy policy script

This policy script controls which endpoints are allowed to log in to the Tivoli region. For example, you might not want endpoints from subnet 26 on this Tivoli region. The default behavior of this policy allows endpoints to log in unconditionally. You can also use this policy to perform any pre-login actions you might need. For example, this policy can help filter duplicate logins to the endpoint manager when the endpoint manager is overloaded with activity or policy scripts are taking a long time to run.

The **allow_install_policy** script is run by the endpoint manager as soon as it receives an endpoint initial login packet from an intercepting gateway. If the policy script exits with a nonzero value, the login process is terminated immediately. If the policy exits with a zero value, the login process continues.

select_gateway_policy policy script

This policy script, run by the endpoint manager, provides an ordered list of gateways that should be assigned to an endpoint. The **select_gateway_policy** script is run each time an endpoint login packet is forwarded to the endpoint manager. The **select_gateway_policy** script overrides the default selection process and should be used for Tivoli environment with multiple gateways. If an endpoint is isolated, the endpoint uses the list of alternate gateways, which were provided by this policy script. This list is sent to the endpoint with the initial login assignment information and after a migration or normal login.

The endpoint manager tries to contact each gateway in the order listed in the policy script until it successfully contacts a gateway. The first gateway contacted is the gateway to which the endpoint is assigned. The intercepting gateway is also added to the end of the list in the policy script to ensure that the endpoint has at least one definite contact. If the gateways listed in the script cannot be contacted, the endpoint manager assigns the intercepting gateway to the endpoint.

after_install_policy policy script

This policy script is run by the endpoint manager after the endpoint has successfully been created. Because the script runs before the first normal login of an endpoint, you cannot use it to run downcalls.

The policy is run after the initial login only; it is not run on subsequent logins of an endpoint.

login_policy policy script

This policy script is run by the gateway and performs any action you need each time an endpoint logs in. For example, this script can be configured to automatically upgrade the endpoint software. The script is also useful for checking changes to IP addresses and port numbers. When the gateway detects changes, it notifies the endpoint manager. When the policy script exits with a nonzero value, the endpoint login will not fail.

Note: The same **login_policy** policy script is run on all the gateways in a Tivoli region.

Configuration files

An endpoint uses files to store configuration information. These configuration files reside in the \dat subdirectory of the endpoint installation:

lcf.dat Contains login information related to the endpoint. You cannot edit this binary file. However, certain information in this file can be overwritten by using command options when starting the endpoint service.

last.cfg
Contains the most recent configuration information. You configure this file with **lcfd** command line options. For information, see the **lcfd** command in the *Tivoli Management Framework Reference Manual*.

lcf.id Contains a unique ID number to represent the endpoint. Do *not* copy this file from one computer system to another (for example, during mass installations), because each endpoint requires a unique identifier.

After the endpoint connects to its assigned gateway, the gateway address, port number, and any network aliases for the assigned gateway and alternate gateways are written to the lcf.dat file. All other configuration information is written to the last.cfg file. On subsequent startups, the startup commands (**lcfd** or **lcfd.sh**) read the configuration information from the lcf.dat file and the last.cfg file. As with the initial login, after the endpoint and gateway are connected, the configuration information is written to the last.cfg file.

To change the configuration of an endpoint, either edit the last.cfg file or restart the endpoint using one of the startup commands (**lcfd** or **lcfd.sh**) with the appropriate options. If you choose to edit the last.cfg file, the new configuration information is used when you restart the endpoint. When connected, the information is again written to the last.cfg file.

The lcfd command

If you start an endpoint from the command line using either **lcfd** or **lcfd.sh**, the options you specify override the equivalent entries in the last.cfg file. The endpoint restarts with the new configuration, which is written to the last.cfg file when the connection is complete. This new configuration is used in all future startups.

The **lcfd** command can also be used to set the way an endpoint sends its initial login information. Specifically, the **–g** option of the **lcfd** command enables you to specify the gateway or gateways that an endpoint will try to contact. The following is an example of this parameter:

```
lcfd -g elm_gateway+9494:ash_gateway+9494
```

This example restarts the local endpoint and specifies two gateways the endpoint will try to log in to. The endpoint tries to connect to elm_gateway first and then tries ash_gateway on default port 9494.

Note: This option does not specify the gateway to which the endpoint is ultimately assigned. Only the endpoint manager can assign a gateway to an endpoint.

Sharing endpoint resources across Tivoli regions

The Endpoint and EndpointManager resources can be exchanged across connected Tivoli regions. These resources can be shared to enable profile distribution and running tasks supported by the endpoint. Direct management of the endpoint (such as migrating endpoints to new gateways or listing the endpoints assigned to a gateway) must, however, be performed within the Tivoli region local to the endpoint. Management commands, such as **wgateway**, **wep**, and **wdelep**, must be run within the Tivoli region. For more information about these commands, refer to the *Tivoli Management Framework Reference Manual*.

If you share the Endpoint resource, you must also share the EndpointManager resource. The Gateway resource can also be shared, but there is little benefit in doing so. Therefore, do not share the Gateway resource. Refer to “Basic resources and exchangeability” on page 25 for more information about shared resources.

Endpoint migration

You can change the gateway assigned to an endpoint by migrating the endpoint to a new gateway. The **wep migrate** command prompts the endpoint manager to update both the new and old gateways. Because the endpoint might not be reachable at the time its gateway assignment is changed, the endpoint is not directly notified. When the endpoint-to-gateway communication is established, the migration is complete.

Note: Endpoints cannot be migrated across Tivoli region boundaries.

The following processes complete the migration of an endpoint:

downcall
Gateways automatically receive migration information from the endpoint manager. Therefore, a downcall sent by the newly assigned gateway reaches the endpoint without a problem. The endpoint reads the new gateway address and updates its lcf.dat file. Subsequent communication is sent to the new gateway.

Note: This process is not supported when gateways and endpoints are separated by a NAT device. NAT environments must rely on upcall and login to complete the migration process.

upcall If a migrated endpoint sends an upcall, its formerly assigned gateway recognizes that it no longer manages the endpoint and intercepts the request. The intercepting gateway obtains the new gateway assignment for the endpoint from the endpoint manager. This information is forwarded to the endpoint. The endpoint then logs in to its new gateway and resends the upcall.

login The treatment of a login request from a migrated endpoint to its formerly assigned gateway is similar to the upcall process. The former gateway intercepts the endpoint login request. It then obtains the new gateway assignment for the endpoint from the endpoint manager. This information is forwarded to the endpoint. The endpoint then logs in to its new gateway.

If the former gateway is unreachable, the endpoint proceeds as if isolated. The endpoint uses its list of alternate gateways (either from the endpoint manager or **select_gateway_policy** script) to try to log in to another gateway. If the endpoint fails to log in to any of the alternate gateways, the endpoint sends a broadcast packet (if configured). The login process is similar to the isolation login process in that the gateway selection process is triggered.

Note: The entries in the **select_gateway_policy** script take precedence over the gateway specified by the **wep migrate** command if the endpoint is unable to contact its former gateway.

Preferred gateways

Tivoli endpoints are automatically assigned to another gateway if they cannot contact their assigned gateways. However, an administrator can set a *preferred gateway* for an endpoint. When the preferred gateway of an endpoint becomes available, the endpoint can be configured to migrate back to its preferred gateway. Refer to the **wep** command in the *Tivoli Management Framework Reference Manual* for more information about setting preferred gateways.

The preferred gateway field is set to the first gateway returned by the **select_gateway_policy** script anytime this script is run for the endpoint. If the **select_gateway_policy** script does not return a list of gateways, the preferred gateway field is set to the assigned gateway during the initial login of the endpoint.

Movement of endpoints to their preferred gateways can take place in the following ways:

- An administrator can use the **wepmgr** command to set the **automigrate** option to **on** or **nonmobile**. This option defaults to **off**, so there is no automatic migration unless an administrator enables this feature. Setting **automigrate** causes endpoints to migrate back to their preferred gateways when that gateway becomes available. Mobile endpoints are excluded from this migration if the attribute is set to **nonmobile**.
- An administrator can use the **wep migrate_to_pref** command to manually migrate endpoints to their preferred gateway. The **migrate_to_pref** option migrates endpoints to their preferred gateways if the preferred gateway is available. As with the **wep migrate** command, **wep migrate_to_pref** does not cause any communication with the endpoint. Only endpoint manager and gateway data are updated.

Endpoints can be excluded from automatic migration by clearing their preferred gateway attribute.

If your deployment contains endpoints and gateways with multiple network interface cards (NICs), you can configure endpoints to log in to specific NICs on each gateway. For more information, see the chapter about configuring multi-NIC gateways and endpoints in *Tivoli Enterprise Installation Guide*.

Method storage and implementation

A subset of Tivoli operations, called *endpoint methods*, run directly on the endpoint. The endpoint passes the results of these operations back to the gateway, which forwards them to the calling managed node. Other Tivoli operations run on the gateway proxy (the hosting managed node) on behalf of the endpoint. Results of these *gateway methods* are passed to either the endpoint or to the calling managed node.

An endpoint is not installed with any methods. Instead, the endpoint maintains a method cache. Before a method is invoked on the endpoint, the endpoint cache is checked to determine if the method is already available.

If not, the method is downloaded from the endpoint gateway and added to the cache. If the method is already in the cache, the endpoint and its gateway determine if the cache contains the most recent version of the method. If not, an updated method is downloaded. If a current version of the method exists in the endpoint cache, the endpoint runs the method and returns all results to the gateway.

The cache enables the endpoint to build up a set of methods that it can execute quickly. It also allows the endpoint to be easily updated by downloading newer methods as needed. The default maximum size of the cache is 20 MB. You can change the maximum cache size with the **lcfd** (or **lcfd.sh**) command using the **–D cache_limit**=*max_size* option.

Considerations for NetWare and OS/2 gateways

Tivoli Management Framework provides standalone gateways; that is, they cannot be considered as managed nodes. These gateways are available for Novell NetWare and OS/2 operating systems. In general, they provide the same functionality of other gateways:

- Endpoint login and communication
- Invocation of methods to be run on the endpoint and gateway

Refer to the *Tivoli Management Framework Release Notes* for details on the functionality available for NetWare and OS/2 gateways. This section describes some of the special considerations of these gateways.

NetWare gateways

For the Novell NetWare operating system, you need to consider the gateway proxy and IPX support.

Gateway proxies for NetWare

NetWare gateways require a gateway proxy to run endpoint policy scripts. Any managed node can act as a gateway proxy. Using the **wgateway** command, you define a list of managed nodes that will act as a gateway proxy. The NetWare gateway contacts each of the managed nodes in the list, one at a time, until it finds a managed node available to run the script. If all the managed nodes in the list are down or unavailable, the NetWare gateway is unable to run the endpoint policy scripts. By default, no gateway proxies are defined. Refer to the **wgateway** command in the *Tivoli Management Framework Reference Manual* for information about modifying gateway proxies.

IPX support

The NetWare gateway allows you to connect to endpoints with Transmission Control Protocol/Internet Protocol (TCP/IP) or IPX. To connect to an endpoint in IPX, you can use the IPX address, name resolution, or IPX broadcast. IPX/SPX name resolution allows login by the server name of the NetWare gateway. Thus, the endpoint does not have to know the IPX address of the NetWare gateway for login. It is important to note that this name resolution uses Routing Information Protocol (RIP) packets. Refer to the Novell documentation for more information about routing RIP packets.

For endpoint logins in Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX) environments, you can explicitly specify the IPX address of the NetWare gateway where to log in. Alternatively, if the requested gateway is located within a five-hops radius from the endpoint, you can specify the gateway's server name instead of its IPX address. If you do not specify any gateway and the endpoint broadcast is enabled, the IPX login packet is sent to all the servers located within a five hop radius.

Both the IPX/SPX name resolution for the gateway and the IPX extended broadcast login make use of the Routing Information Protocol (RIP). Specific RIP routing configurations can affect their functionality. The use of IPX/SPX name resolution for the gateway and IPX extended broadcast perceptibly slows down the endpoint initial login.

OS/2 gateways

You must install and configure the OS/2 security enablement services for the OS/2 gateway to function correctly. OS/2 gateways must be installed on disks that are formatted using the high performance file system (HPFS).

Chapter 6. Multiplexed distribution services

Tivoli Management Framework provides multiplexed distribution services to enable distributions of large amounts of data to multiple targets in an enterprise. These services are used by a number of profile-based Tivoli applications so that they can maximize data throughput across large, complex networks.

Multiplexed distribution services use a hierarchy of repeaters to fan out to a large number of endpoints. Each service limits its own use of the network, as configured through repeater parameters, to help prevent intense network activity that can stress network bandwidth for periods of time.

This chapter includes the following topics:
- "Distribution services"
- "Network communication" on page 76
- "Default repeater configuration" on page 77
- "Multicast distributions" on page 77

Distribution services

Tivoli Management Framework provides two multiplexed distribution services—MDist and MDist 2. Both services are available to Tivoli applications and can be active at the same time. Therefore, you should consider this when planning distributions that potentially could exceed memory and network bandwidth limitations.

Note: To use MDist or MDist 2 across connected Tivoli regions, you must use a two-way connection between the regions.

MDist service

The MDist service performs a synchronous transfer of data through a configured hierarchy of repeaters. You cannot modify the data stream as it flows through the repeater hierarchy, so each target receives an exact copy of the original input stream. Because input and output occur almost simultaneously, there is no need for the repeater to store a distribution. This enables the repeater to handle distributions of unlimited size, unless the number of targets in a distribution exceeds the maximum number of simultaneous connections (**wrpt max_conn** command) of the repeater. If this occurs, the size of the segment being distributed is limited to 2 gigabytes (GB), similar to the MDist 2 service.

The distribution finishes only after each target receives the entire distribution or encounters an error in the process. Because MDist does not store data within a repeater, if the distribution fails, you must check the log file to determine which targets were successful, update your subscriber list, and redistribute the data to revised subscribers.

Figure 22 on page 64 illustrates how the MDist service sends data to its targets. Repeaters on managed nodes can send distributions to other repeaters and managed nodes. Repeaters on gateways can send distributions to other repeaters, managed nodes, and endpoints.

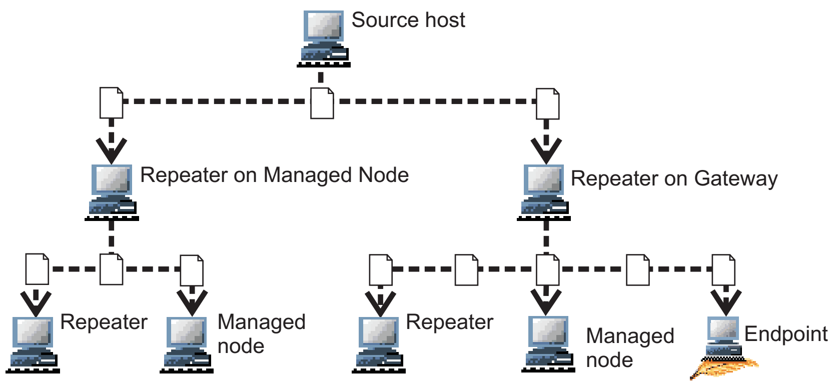


Figure 22. MDist internal structure

MDist 2 service

MDist 2 extends the functionality of MDist to handle large-scale distribution needs of Tivoli applications. Unlike MDist, MDist 2 provides asynchronous transfer of data that enables you to submit multiple distributions from the same host without tying up your system. MDist 2 also ensures successful delivery of a distribution through the use of *repeater depots*, which are data repositories that store distribution data temporarily or permanently. If a connection to a target cannot be established, the distribution is safely maintained in a repeater depot until the connection is reestablished and the distribution delivered. The distribution resumes from the last successful checkpoint. A handshake occurs at each connecting system. If a failure within the distribution hierarchy occurs, you do not need to resend the distribution.

Note: Only gateways and endpoints can be targets of a MDist 2 distribution. Managed nodes are not supported as targets.

MDist 2 supports the following functionality:

Distribution monitoring and control
Enables you to monitor the status of a distribution and take action on it (pause, resume, or cancel) through the use of the Distribution Status console. You can see which targets received the distribution, see which experienced errors, and estimate when the distribution will finish. You also can delete distributions manually or set an interval at which completed distributions are removed from the database.

Mobile computing support
Enables users to control distributions to their workstation using the Mobile Computing console. Users can download and install distributions at their convenience. If the workstation disconnects at any time (such as in the middle of a download), when the user reconnects, operations resume where they left off. Moreover, administrators can retain control of distributions by sending hidden or mandatory distributions.

Disconnected endpoint support
Enables users to download and save distributions in a local storage directory for installation at a later date. When the user reconnects to the network, results of the distribution are returned to the gateway and the state of the target is changed to Successful.

Roaming endpoint support

Enables endpoints to receive a distribution when an endpoint migrates to a different gateway during the distribution. When MDist 2 detects that an endpoint has migrated, it updates the route of the distribution and forwards it to the new gateway. By default, all endpoints are roaming endpoints.

Note: Distributions can only be rerouted to a new gateway if the if the endpoint's old gateway is still functioning (running). The old gateway must be up and running before it can reroute the distribution to the new gateway.

Installation from CD or file server

Enables administrators to create a distribution and specify that the distribution is on a CD or file server. When the dataless distribution arrives at the endpoint, the user retrieves the distribution from the specified source instead of a repeater. For more information, refer to "Repeater-to-endpoint distributions" on page 77.

Wake on LAN (WOL) support

Enables administrators to send distributions to powered off computers. If a repeater is unable to contact an endpoint, it checks to see whether the distribution has enabled Wake on LAN® support and attempts to start the computer. Regardless of whether the endpoint is mobile, a Wake on LAN packet is sent when the mandatory date for the distribution has passed or when the distribution is hidden. The target remains up and running after the distribution completes.

Note: For WOL support, the routers and switches must be configured to allow Wake on LAN traffic. Tivoli Management Framework supports subnet-directed Wake on LAN. Subnet-directed Wake on LAN means that the repeater sends a broadcast packet that is targeted at the subnet for the endpoint, which is determined by the subnet mask for the endpoint. Routers and switches must be configured to pass these broadcast packets.

Multicast support

Enables administrators to send distributions using multicast technology instead of unicast technology. Multicast support enables a single profile to be simultaneously distributed for source to target in parallel. For example, if the distribution from a repeater to three other repeaters were done as a multicast distribution, only one distribution occurs. In contrast, if this distribution were done as a unicast distribution, three independent parallel distributions would occur.

Note: For multicast support, the routers and switches must be configured to allow multicast traffic.

Adaptive bandwidth feature

The adaptive bandwidth feature is new in this release (Version 4.3.1) of the Tivoli Management Framework. It provides another means to control how much bandwidth is consumed when performing software distributions. See "Using the adaptive bandwidth feature" for information.

Using the adaptive bandwidth feature

Traditionally, a gateway uses a static amount of bandwidth, measured in Kbps, when sending a software distribution to an endpoint. The static amount of bandwidth that is used to perform a distribution does not change, whether the network is busy (bandwidth is being consumed by multiple applications) or whether the network is idle (little or no other traffic is present). In busy networks where distribution speeds are static, one way to mitigate the impact of distributions would be to schedule them to take place during off-hours or during predetermined maintenance periods where traffic is expected to be light. Given the distributed nature of many businesses, finding such opportunities can be difficult because the network can be in use 24 hours a day, 7 days a week.

When the adaptive bandwidth feature is used, it monitors network conditions and an algorithm automatically adjusts the amount of bandwidth that is consumed when MDist2 distributions are sent across a network. When the network is not congested, distributions are allowed to consume more resources to quickly get them to their destinations. If the network is congested, the adaptive bandwidth feature automatically reduces the bandwidth used to perform distributions so other traffic on the network is not slowed down even more when the distribution occurs.

Notes:
- If your network uses IPSec services to protect data as it travels over the public internet, Authentication Header (AH) protection does not interfere with the adaptive bandwidth routines, but using tunnel mode and Encapsulating Security Payload (ESP) together can interfere with the adaptive algorithm's ability to determine network load. This will cause software distributions to be sent very slowly, even when the network is idle.
- Adaptive Bandwidth Control will use additional CPU resources on the gateway proportional to the speed and amount of data being sent.
- Adaptive Bandwidth works best when there are less than 20 receivers simultaneously receiving a distribution behind a single slow link.
- Gateways that have adaptive bandwidth enabled should have the network card operating system option that offloads segmentation to the network card ("Offload TCP segmentation") must be disabled. Similarly, receivers of distributions, should have "TCP ACK Delays" disabled. The documentation for your operating system, as well as the documentation for your network interface cards, contains descriptions fo how to disable offloading of TCP segmentation and TCP ACK delays.

As a convenience, examples of how to disable Offload TCP segmentation and TCP ACK Delays on Linux systems are provided, below. For other supported platforms, refer to your operating system and network interface card documentation for the instructions to disable these options.

Adaptive sender and adaptive receiver settings: In the paragraphs that follow, the term "adaptive sender" refers to gateways that distribute packages to other gateways, other repeaters, or to endpoints.

1. Determine if TCP Segmentation is enabled. Type the following at a shell prompt on the local console:
   ```
   -bash-3.00# entstat -d ent0 | grep Segmentation
   ```
2. If the output from step 1 shows **TCP Segmentation Offload: Enabled**, you must disable it. Type the following commands on the local console; note that `state=detach` will bring down the network card, but the `state=up` command will restart it:
   ```
   chdev -l en0 -a state=detach
   chdev -l ent0 -a large_send=no
   chdev -l en0 -a state=up /etc/rc.net 1
   /etc/rc.net
   ```

Changing AIX adaptive receiver settings: To disable the AIX adapter receiver setting, type the following at a shell prompt on the local console:

```
no -o tcp_nodelayack=1
```

Changing Solaris adaptive receiver settings: To disable the Solaris adapter receiver setting, type the following at a shell prompt on the local console:

```
ndd -set /dev/tcp tcp_deferred_ack_interval 1
```

Changing Windows adaptive sender settings: To disable the Windows adaptive sender settings, complete the following steps. Consult the documentation for your network adapter for specific instructions:

1. Right click **My Computer**.
2. Click **Manage**.
3. Click **Device Manager**.
4. Click the plus (**+**) sign next to **Network Adapters** to expand the adapter list.
5. Double click the Network Adapter you are configuring, or right click an adapter and then click **Properties**.
6. In the **Advanced** tab in the **Properties** window, select **Offload TCP Segmentation** and change its value to **Off**.

Enabling, configuring and using adaptive bandwidth: When the adaptive bandwidth feature is enabled, software distributions are sent using high, medium, or low priority. The amount of impact a distribution is allowed to have on network performance is specified as both a base priority and as a subjective number between 0 and 100.

The base priority (**adaptive_priority_high**, for example), indicates the general importance of distribution speed and the degree of impact it can have on available bandwidth. High priority distributions are performed faster than medium and low priority distributions and the adaptive feature allows high priority distributions to consume more of the available bandwidth than the other priority settings.

The adaptive bandwidth feature dynamically decides how much of the available bandwidth that an adaptive distribution, given its base priority, should consume and still not bring the entire network to a stand still. So, it is not accurate to say that an adaptive high priority distribution will take a fixed percentage of all available bandwidth. Instead, high priority distributions can take most, but not all of the bandwidth that is available when the distribution is started. **Adaptive_priority_medium** distributions consume less bandwidth than high priority distributions, and **adaptive_priority_low** distributions consume the least bandwidth of all of the adaptive priorities.

Within each priority, you can specify a subjective number, from 0 to 100 to indicate how aggressively you want a distribution to consume bandwidth, given the amount that is possible to be consumed by its base priority. The subjective number that you specify determine how much of the total bandwidth allowed for a given priority should be consumed. So, if **adaptive_priority_high=100** is specified, the adaptive feature will ensure that your high priority distributions are delivered as fast as possible by consuming most of the available bandwidth. If you specify **adaptive_priority_high=50**, the adaptive feature will limit itself to only consuming about ½ of the bandwidth that it could otherwise consume if the subjective number was set to 100.

If the adaptive priority is set too low, distributions may go into the interrupted state when other network traffic is present because adaptive will refuse to send data.

As is with all distributions, the netload parameters for gateways and endpoints acts as a governor for adaptive bandwidth consumption, and distributions will never send data faster than the current **netload** setting.

Adaptive bandwidth control is not used during multicast distributions.

When **adaptive=false**, the gateway's **net_load** parameter will be used to statically control bandwidth (see **wmdist -s netload** in the Tivoli Management Framework Reference Manual).

Note: If you prefer that all distributions have the same impact on the network, regardless of their priority, set **adaptive_priority_high=adaptive_priority_medium=adaptive_priortiy_low**.

The following lists the parameters that enable and configure the adaptive bandwidth feature. These options are specified on the **wmdist** command, as the `keyword=value` pairs that follow the **-s** option.

adaptive=[true | false]
: **true** indicates adaptive measures will control how much bandwidth a distribution can use. **false** indicates bandwidth consumption is statically controlled by the repeater’s **wmdist net_load** parameter.

adaptive_priority_high=*int*
: *int* is a subjective integer from 0 to 100. This integer indicates how much of the available bandwidth that a high priority distribution can consume. The higher the number, the more impact a distribution can have on the network, in terms of the bandwidth it is allowed to consume.

adaptive_priority_medium=*int*
: *int* is a subjective integer from 0 to 100. This integer indicates how much of the available bandwidth that a medium priority distribution can consume. The higher the number, the more impact a distribution can have on the network, in terms of the bandwidth it is allowed to consume.

adaptive_priority_low=*int*
: *int* is a subjective integer from 0 to 100. This integer indicates how much much of the available bandwidth that a low priority distribution can consume. The higher the number, the more impact a distribution can have on the network, in terms of the bandwidth it is allowed to consume.

adaptive_interface=[auto | *interface_identifier***]**
: Adaptive distributions can be directed to use a particular network interface. The default is **auto**, which indicates that the network interface used will be selected by the adaptive bandwidth feature. The default is recommended for all scenarios, even when **odadmin set_force_bind** is set to **true**. If **auto** is not used, then the Network Interface Card that is specified must match the interface that is enforced by the **odadmin set_force_bind** option. To override this, specify the identifier of the network interface adapter to use for all adaptive distributions.

adaptive_log_level=*int*
: *int* is the logging detail level for adaptive distributions. Specify an integer from 0 to 4. 0 turns off the logging of messages generated during adaptive distributions. 1 logs the fewest messages, and 2, 3, and 4 create

progressively more logged messages. Settings of 3 and 4 should only be used to troubleshoot adaptive distribution problems with a single target because the volume of messages logged for multiple targets could overwhelm the gateway's resources.

If you specify a given priority, and you find it to be too slow, you can either raise the priority (for example, from **adaptive_priority_low** to **adaptive_priority_medium**, and provide a new relative qualifier (0 to 100). Other customers may find that the **adaptive_priority_low** value is too slow, and it takes too long for distributions to complete because of other network traffic. They may decide to simply increase the current value by 10. For example, it you initially set **adaptive_priority_low=10**, and distributions are taking too long, try raising the relative qualifier from 10 to 20 (**adaptive_priority_low=20**), or send the distribution using the next highest priority (medium) that will use the quantifier, for example, **adaptive_priority_medium=60**.

MDist 2 repeater components

MDist 2 repeaters can store, prioritize, and disperse distribution data through the use of queues and depots. These components are created when you create a repeater.

Repeater queues: A *repeater queue* regulates how distributions flow between a repeater and its targets and enables a repeater to handle a large number of concurrent, active distributions through the use of priority connections. These queues automatically retry distributions to disconnected or unavailable targets.

Within queues, distributions are sorted according to their priority level (high, medium, or low). Within each priority level, the distributions are ranked by when the repeater received the distributions (not by the order in which the distributions were started). For example, suppose that a high-priority application, which is divided into three data segments (or files), arrives in a repeater queue. The data segments associated with the distribution can use all available high-priority connections and any available connections for lower priority levels.

Note: If a distribution is separated into multiple data segments, all segments use the same connection. The segments are sent serially, starting with the first segment.

Figure 23 on page 70 illustrates how connections are allocated from the specified priority level downward. For example, a medium-priority distribution first uses medium-priority connections and then uses any available low-priority connections.

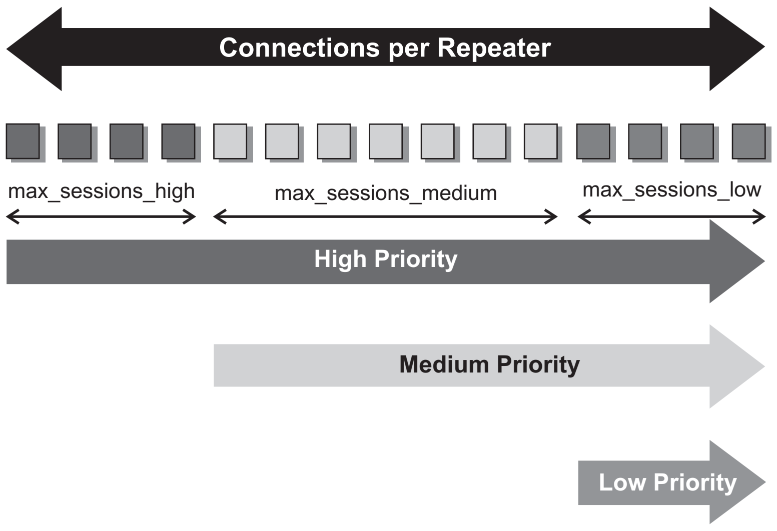


Figure 23. Setting maximum priority connections

Consider the case of a repeater configured with the following maximum priority connections:

- 5 high-priority connections
- 20 medium-priority connections
- 5 low-priority connections

In this example, the maximum number of connection sessions that a distribution can use is as follows:

- High-priority distributions can use a maximum of 30 sessions.
- Medium-priority distributions can use a maximum of 25 sessions.
- Low-priority distributions can use a maximum of 5 sessions.

A medium-priority distribution would first use the 20 medium-priority connections before using the low-priority connections, and so on.

Repeater depots: A *repeater depot* is a directory on the repeater that enables you to temporarily or permanently store data segments associated with distributions. For example, you need to resume a distribution that was interrupted. Instead of resending the entire distribution, MDist 2 can send the needed files from the depot of the last repeater that received the files. If a connection to a receiver cannot be established or is broken, the files are safely maintained in the depot until the connection is reestablished and the distribution delivered.

A depot is used to store all distribution data segments. Each segment consists of two files with .toc and .dat extensions. The .toc file contains the properties of the data segment. The .dat file contains the data to be distributed.

A depot is located in the following directory on each repeater system (assuming default installation):

- On Windows operating systems: $DBDIR/tmp/depot
- On UNIX operating systems: /var/tmp/depot

The /tmp directory also contains a /states directory, which includes a database with the following two tables:

- The first table contains a list of distributions on which the repeater is working. This queue is sorted first by priority and then on a first in, first out (FIFO) basis. Each distribution entry in this table contains a *distribution message*, which helps MDist 2 track the distribution properties. The message is a sequence of key and value pairs.
- The second table tracks the targets associated with each distribution. When a distribution completes or when the notification interval is reached for each repeater, MDist 2 updates its status in the database. Therefore, if a repeater is restarted, MDist 2 knows which targets received the distribution.

When the repeater is running, it has a copy of the queue in memory. If the repeater is restarted, it can read the persistent queue from disk and regenerate the queue that it holds in memory. The repeater can then resume where the interruption occurred.

Note: The default disk capacity of each depot is 500 MB.

Distributing profiles with MDist 2

Figure 24 illustrates how a profile, such as a software distribution, interacts with MDist 2 components to distribute data to multiple targets.

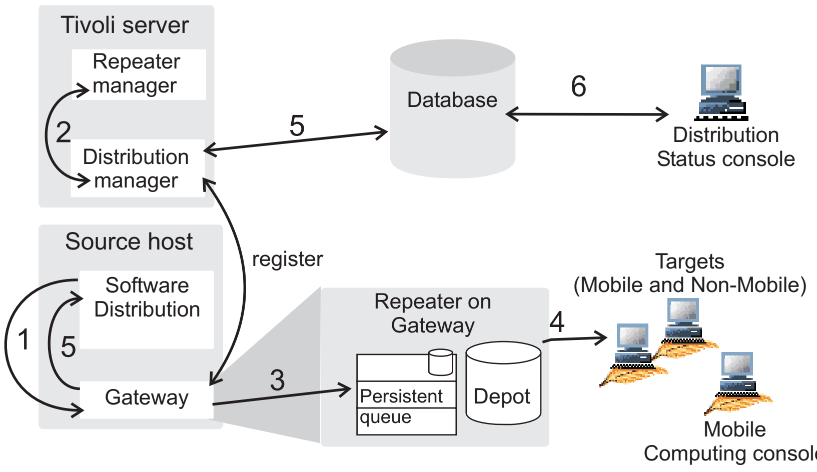


Figure 24. MDist 2 internal structure

The following steps describe the internal structure illustrated in Figure 24:

1. A software package profile is submitted to the source host repeater (either on a managed node or gateway). The source host repeater registers the distribution with the distribution manager on the Tivoli server. The distribution manager stores status information about a distribution in the database tables. There is one distribution manager for each Tivoli region. Therefore, each Tivoli region tracks distributions started in it.

   Note: If the distribution spans multiple regions, all status information is kept in the distribution manager of the Tivoli region in which the source host resides.
2. The distribution manager obtains the distribution route from the repeater manager on the Tivoli server and initializes all the entries in the database. The

repeater manager maintains information about your Tivoli environment, including targets and repeaters. Each repeater manager determines which path a distribution takes. There is one repeater manager for each Tivoli region.

3. The distribution manager returns the route to the source host repeater, which queues the distribution information according to its priority. The queue runs the job at the proper time, in accordance with the resources available on the repeater.
4. The distribution is sent to its targets or retained in the depots to be sent at a later time. When the distribution reaches a target, the repeater checks if the target is configured as a mobile endpoint. If it is not a mobile endpoint, the distribution is downloaded and installed when the endpoint logs in. If it is a mobile endpoint, the repeater first checks to see if the distribution is hidden or if it has passed its installation date. If so, the distribution is downloaded when the endpoint logs in. If not, the distribution is paused on the gateway until the distribution expires or is resumed by the user through the Mobile Computing console.

   Note: For more information about the Mobile Computing console, refer to *Tivoli Management Framework User's Guide*.
5. The application is notified of the results of the distribution. Results are returned from the repeaters to the application through the same distribution channel that the application used to send the data (in reverse order). The source repeater then passes this status to the distribution manager, which updates the database information.

   Note: The database is only updated when a distribution completes or when the notification interval is reached for each repeater.
6. Administrators can monitor and control the distribution through the Distribution Status console, command line, or a custom SQL script. The data that is received is obtained from the database.

Distribution hierarchy

By default, the Tivoli server is a repeater distribution server for all targets in the region. You can configure managed nodes to serve as additional repeaters. A repeater is automatically created when you create a gateway and is capable of distributing data in parallel to a large number of clients. The following examples illustrate possible deployment scenarios. Each region consists of a Tivoli server and a set of clients.

In the simplest installation, the Tivoli region consists of a server and a set of endpoints, as shown in Figure 25 on page 73.

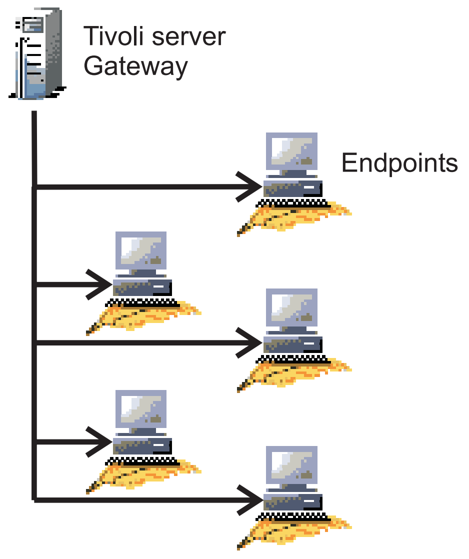


Figure 25. MDist or MDist 2 distribution hierarchy to endpoints with Tivoli server as gateway

In this situation, the Tivoli server is also a gateway, which provides all communication services between the endpoints and the rest of the Tivoli environment. A gateway is automatically configured with services and can serve as the fanout point for all distributions to its endpoints. The gateway distributes information to the endpoints in parallel.

Another simple configuration connects a Tivoli server with a group of managed nodes, as shown in Figure 26.

Note: This configuration is for the MDist service. MDist 2 cannot distribute to managed nodes.

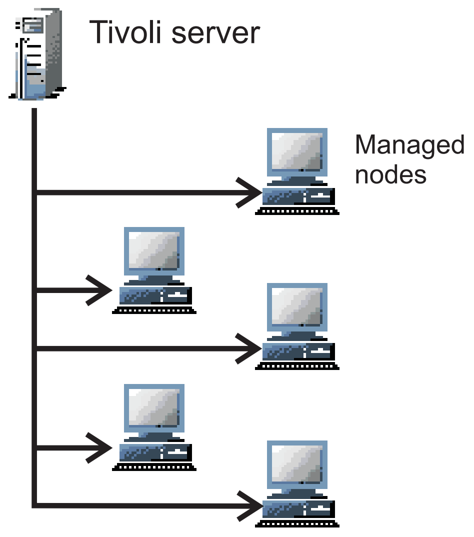


Figure 26. MDist only distribution hierarchy to managed nodes with Tivoli server

In this installation, the Tivoli server also distributes information to each client in parallel.

Such simple network scenarios, however, do not represent most network enterprises. The majority of Tivoli deployments require a Tivoli server to handle deployment across a large and complex network, which can consist of a Tivoli server that is connected to gateways and managed nodes, and a very large number of target clients. To maximize throughput, multiplexed distribution services spread the distribution load across a network tree. Dispersing the distribution load in this manner limits resource contention that can arise when one server is responsible for distributing to many clients.

Figure 26 on page 73 gives a more accurate representation of this type of complex network environment in which a Tivoli server deploys Tivoli profiles to gateways that, in turn, deploy the data to a very large number of endpoints.

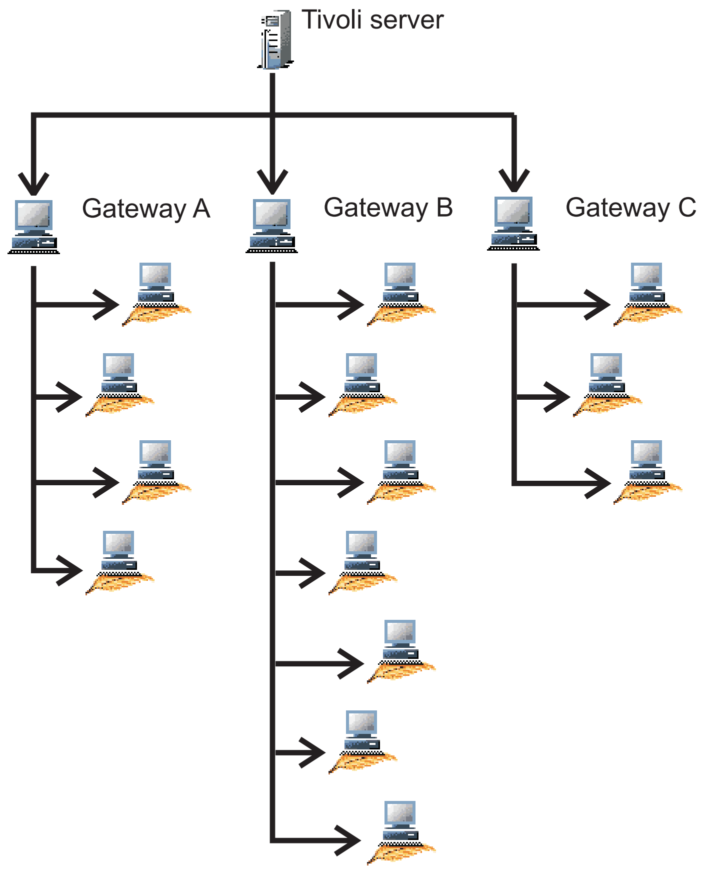


Figure 27. MDist or MDist 2 distribution hierarchy with multiple gateways

In this network scenario, the Tivoli server distributes to the three gateways (gateway A, gateway B, and gateway C) in parallel. Each gateway then distributes in parallel to each of its assigned endpoints. After the data transfer, each gateway collects results from its clients and returns distribution results to the Tivoli server.

While this installation scenario greatly enhances distribution, many network environments require a more complex, scalable distribution solution, as illustrated in the MDist scenario shown in Figure 28 on page 75.

Note: The following scenario does not apply to MDist 2 distributions because managed nodes cannot be targets of a distribution.

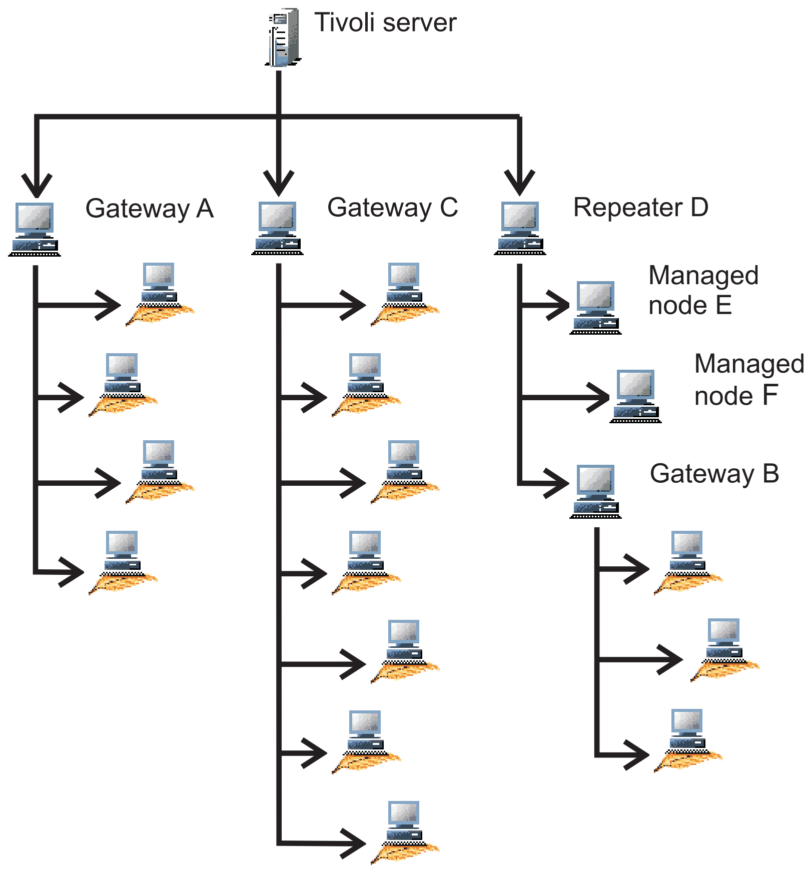


Figure 28. MDist distribution hierarchy with gateways and repeaters

In this scenario, an additional distribution layer is added to the distribution hierarchy—repeater D now serves as a distribution node for all distributions to gateway B and its clients. Repeater D also serves as a distribution node for managed node E and managed node F, which can be any Tivoli client—managed node or endpoint.

The Tivoli server again begins parallel distribution to the three computer systems that are connected to it: gateway A, gateway C, and repeater D. Gateway A and gateway C distribute the data to their assigned endpoints. Repeater D distributes the data to its three clients: gateway B, managed node E, and managed node F. And finally, gateway B distributes to its assigned endpoints.

In the preceding scenario, gateway A, gateway B, gateway C, and repeater D all serve as repeaters for the distribution. In a Tivoli region, a repeater is a gateway or managed node that supports the MDist service. As illustrated, the repeater receives a single copy of data and distributes it to the next tier of clients, which can include endpoints, managed nodes, or other repeaters. For the MDist service, this distribution can be described as a data pipeline.

A repeater can be used to receive and distribute data, and it can also be a target. In the preceding scenario, the Tivoli server distributes a copy of data to repeater D. As a repeater, repeater D passes data through to the next tier of clients. However, repeater D can also be a target that receives the distributed data from the Tivoli server. Unlike the MDist service, the MDist 2 service requires that an endpoint must be installed to receive a distribution.

A repeater range is the tier of Tivoli clients that receive data from the repeater. In Figure 28 on page 75, the repeater range of the Tivoli server contains three clients: gateway A, gateway C, and repeater D. The repeater range of repeater D includes gateway B, managed node E, and managed node F.

After the data transfer, repeaters collect the distribution results from each client and send the results back up the distribution chain. The distribution hierarchy defined by the ranges of all the repeaters is used for all multiplexed distribution transfers. The data distribution hierarchy can be changed at any time by redefining the repeater range. However, the changes apply only to future distributions—those that are in progress follow the hierarchy as defined at the initiation of the process.

Network communication

During distributions, Tivoli Management Framework employs two models for network communication depending on the type of distribution you are performing. A distribution either can go from a repeater to another repeater or the distribution can go from a repeater to endpoints.

Always design a repeater hierarchy that meets the networking needs of your Tivoli environment. For example, consider the case of a slow WAN connected to a fast LAN containing multiple targets. A distribution across a WAN, which communicates with each target directly, sends duplicate data across the slow link. However, if you install a repeater to receive distributions from the WAN and then transmit the data to the LAN, network traffic across the WAN is reduced to a single copy of the distribution.

Repeater-to-repeater distributions

Tivoli Management Framework provides an Inter-Object Message (IOM) communication service that enables the rapid and reliable transfer of large quantities of data between computer systems in a Tivoli region or between regions in a network. During deployment, the IOM service provides a means of transferring large blocks of data without involving object dispatcher, or oserv service, communication between source and target computer systems.

Note: The Bulk Data Transfer (BDT) service that supports IOM data transfers enables you to limit the number of target ports that are required for the Tivoli environment to operate. For more information about BDT, refer to 107.

The object dispatcher coordinates communication between all managed nodes in a Tivoli region. These resources continuously keep port 94 open to receive information from other object dispatchers or Tivoli processes. Distributions from a source computer system to a target system can occur through object dispatcher communication over port 94.

However, using port 94 for object dispatcher communication during large distributions can lead to network overload, resulting in a slow and inefficient deployment scheme. To resolve some of this load, Tivoli Management Framework establishes IOM channels for MDist distributions that are larger than 16 kilobytes (KB). For the MDist 2 service, there are no limits—IOM channels are opened each time.

The following summarizes the sequence of communication events that occur when distributing a data packet that is larger than 16 KB:

1. The source contacts the target. The source passes the access key, host name, and port to the target. The target uses the key, host name, and port to connect to the source.
2. The target contacts the source and gives it the access key to authenticate itself.
3. The source begins transfer of data to the target.

Repeater-to-endpoint distributions

Repeater-to-endpoint distributions use a slightly different model of network communication than the repeater-to-repeater distribution. The repeater makes a connection to the endpoint and sends data down to the endpoint. Then the endpoint authenticates the repeater. For distribution purposes, endpoints do not connect back to the repeater. IOM communication service is only between repeaters. By default, all data flows on endpoint port 9494.

A variation of this model involves separating data from a distribution before sending it. This feature, available to distributions using MDist 2 only, is advantageous in network topologies where you might need to send a distribution to hundreds of small groups of endpoints; for example, groups of branch offices across the country. The administrator must still submit a dataless distribution to its endpoints; however, the contents of the distribution can reside either on a CD or on a file server. On reaching the target, MDist 2 locates the data and the endpoint is able to download the distribution data from its source (either file server or CD). For more information about installing from a file server or CD, refer to *Tivoli Management Framework User's Guide*.

Default repeater configuration

When you install Tivoli Management Framework, the Tivoli server is created as a repeater. In interregion distributions, the configuration of each MDist service provides for a single transfer of data from one region to another region. When the data reaches the repeater on the Tivoli server in the destination region, the repeater distributes in parallel to the targets in that region. This default configuration allows Tivoli profile-based applications to distribute data in a consistent manner. The default configuration is particularly suited to repeaters that have fast connections between all clients within a given region. In addition, gateways are automatically configured with the MDist service, when they are created on a managed node.

If your Tivoli environment has interconnected Tivoli regions, exchange the repeater resource each time the repeater hierarchy changes (including the creation of each gateway). This resource exchange enables a repeater manager to route a distribution to another Tivoli region.

For information about how to create and configure repeaters, refer to *Tivoli Management Framework User's Guide*.

Multicast distributions

Tivoli Management Framework supports *multicast distributions*. The multicast technology works with the MDist 2 service to provide high-speed, simultaneous distributions to repeaters and endpoints. By definition, multicast sends one distribution to a group of targets.

Before multicast, distributions were done in unicast, where the distribution was transmitted in parallel to each target. Unicast is a viable solution in many situations. However, because a new copy of the package gets distributed to each target, unicast can consume considerable network resources for large distributions to many targets.

The multicast technology, when coupled with the MDist 2 service, provides for simultaneous distributions where each target that is running a multicast receiver is updated at the same time from a single distribution packet. Distribution times are nearly independent of the number of targets. To do this, multicast uses User Datagram Protocol (UDP) broadcasts to send the files simultaneously to multiple targets. Because UDP does not provide end-to-end integrity for the transmitted data, Tivoli Management Framework multicast uses the Reliable Multicast Transport Protocol (RMTP) to ensure each target receives the entire distribution without lost data. At the end of the distribution, the RMTP protocol allows clients to notify the server of packets they missed. The server will then resent these missing packets.

Multicast can be used in the following scenarios:

Preloading depots
: This approach allows an administrator to send a distribution from a source host to multiple gateways (preloading MDist 2 depots with packages for later distribution).

  To load a depot using multicast, the distribution must be multicast-enabled. For multicast-enabled distributions, the repeater determines whether it should send this multicast-enabled distribution based on its own **repeater_multicast** option. If a distribution is multicast-enabled and the repeater option **repeater_multicast=true**, the repeater attempts to send the distribution using multicast. Additionally, the destination repeater must also have the repeater option **repeater_multicast=true** to receive the distribution. Therefore, the **repeater_multicast** option must be set on both the source and destination repeater.

  The **fail_unavailable** option does not apply to loading depot.

Distributing from repeater to endpoint
: This approach allows an administrator to send a distribution from a single gateway to multiple endpoints. You cannot use multicast distribution to send a distribution from a gateway to a single endpoint. When a gateway has only a single endpoint, the distribution is always a unicast distribution.

  For multicast-enabled distributions the gateway determines whether it should use a multicast distribution based on its **endpoint_multicast** option. If a distribution is multicast-enabled and the gateway option is **endpoint_multicast=true**, the gateway attempts to send the distribution using multicast. If either of these conditions is false, the gateway sends the distribution using unicast. If the distribution was sent multicast and some of the endpoints failed, the distribution service checks the **fail_unavailable** option to determine whether it should retry those endpoints using unicast.

The following figures show how multicast can be used to significantly reduce the number of copies of a distribution. In Figure 29 on page 79, a unicast scenario, eight copies of the distribution package are involved in a distribution.

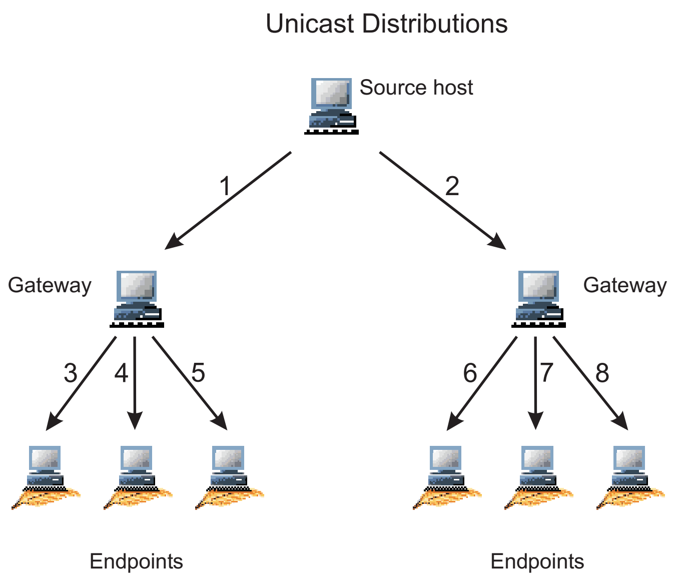


Figure 29. Unicast distributions

In Figure 30, a multicast scenario, only three copies are needed.

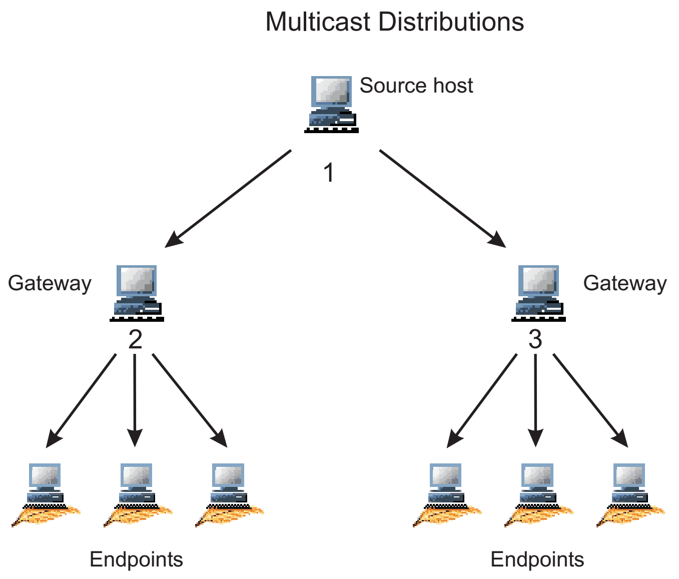


Figure 30. Multicast distributions

Both unicast and multicast can be used for distributions. For example, a distribution can be sent in unicast from the source host to repeaters, and then the repeaters can send the distribution in multicast to their endpoints. Similarly, a distribution can be sent in multicast from the source host to repeaters, and then the repeater can send the distribution in unicast to its endpoints.

To use multicast, the routers between the source host and a receiving node must be configured to support multicast traffic. When setting the multicast settings for a repeater, the time to live (TTL) option must be configured to include the number of hops (routers and switches) between the multicast sender and the receiver.

Using multicast to load depots

In a typical business scenario there might be many branch offices distributed around the country, or the world. If the IT department in the company's headquarters needs to distribute software to each branch office, multicast can be employed to keep the distribution time to a minimum.

If each branch office has a managed node in it, the source host (the Tivoli server in the headquarters building that initiates the distribution) can multicast the distribution by way of a high speed link (for example, a satellite link) to MDist 2 depots on managed nodes at each branch office.

As with unicast, managed nodes that multicast to endpoints must be configured as gateways. The Depot Services running on the gateway in each branch office listens for multicast broadcasts. When a multicast broadcast is received by a Depot Service, the service writes the distribution to an existing MDist 2 depot on the gateway. After the data is in the MDist 2 depot, it can be multicast or unicast to all of its endpoints over the LAN in the branch office. The Depot Service sends a multicast message, an UDP broadcast, to advertise the distribution. The advertisement contains a description of the data being sent and the endpoints that should receive it. Multicast receivers listen for these advertisements to determine which distributions they should receive.

Each distribution requires a separate distribution process; you cannot load MDist 2 depots and deliver to endpoints in a single step. The first distribution sends the package from the source host to the gateways (or repeaters). The second distribution sends the package from the gateway to one or more endpoints (or in the case of a repeater, to another gateway). If an MDist 2 depot is not functioning when the distribution is sent to it, there is currently no provision to retry the multicast distribution. Instead, retries can be manually performed using a unicast distribution to the endpoints that did not receive the original multicast distribution.

Multicasting from repeaters to endpoints

Each gateway in a Tivoli region can receive multicast distributions from its source host and then use multicast to send the package to multicast receivers that run on each of the gateway's endpoints. In this scenario, your network must support multicast traffic from each gateway to their endpoints. When a gateway receives a distribution, it uses multicast to simultaneously fanout the distribution to each of its endpoints that are targets of the distribution. If a gateway has endpoints that are connected on both fast and slow network links, multicast must be configured to send data at the speed of the slowest link. Tuning for a faster link can cause lost packets on the slower links.

Note: Mobile endpoints cannot receive multicast distributions unless they are marked as mandatory targets or unless the distribution is marked as hidden.

A client receiving a multicast distribution stores the entire distribution in a local file. The endpoint must have enough free disk space to receive and install the distribution package. The endpoint should have twice the size of the distribution package free on disk. Twice as much free space is needed because some of the space is used as a cache for the distribution itself and some is used for the data as it is being installed. After the distribution completes, the cached copy is deleted.

Multicast distributions cannot be cancelled or paused. A unicast downcall is performed to start the application to process the data. If you do not want an application to be updated, you can refuse the update when the application is started.

Part 2. Creating a deployment plan

Chapter 7. Design guidelines

You need to create an architecture design document to use as a road map when you install Tivoli Management Framework and other Tivoli products. This chapter provides a checklist to assist you in creating that document. The chapter has two main topics:

- "Overview of the design process"
- "Assessing your existing environment" on page 86
- "Planning the physical design of your network" on page 91
- "Planning the logical design of your network" on page 96

Overview of the design process

To create an architecture design document, you must know how your environment is organized and the current status of platforms, networks, applications, procedures, and processes. You need to know how you want to implement the Tivoli environment, and you need to know what Tivoli Enterprise requires. This information becomes the input for the design document. The design consists primarily of physical and logical designs.

The tasks to design a Tivoli deployment follow. The details of these tasks are discussed throughout this guide, and cross-references to the details are provided.

Physical design tasks

The physical design is for developing the underlying physical infrastructure on which the Tivoli solution will operate. The implementation of the physical design consists of the following tasks:

1. Perform an assessment of the environment to be managed.
2. Determine the number of Tivoli management regions (Tivoli regions) required in the environment.
3. If multiple Tivoli regions are required, divide the managed environment into individual regions.
4. If you need to use multiple regions for scalability reasons, decide the connection topology.
5. Decide on the placement of Tivoli server.
6. Determine the number of gateways and their placement in the network.
7. Determine the number of repeaters and their placement in the network.
8. Determine the number of application servers and their placement in the network.
9. Determine the number of RDBMS Interface Modules (RIMs) and their placement in the network.
10. Generate a document for the physical design that specifies all the interconnections and placement of components.

For details on these tasks, refer to Chapter 9, "Physical design," on page 113.

Logical design tasks

The logical design is for identifying how the Tivoli solution will be configured to meet the operational requirements. The implementation of the logical design consists of the following tasks:

1. Decide on the naming conventions to be used for all your resources.
2. Design your policy region hierarchy schema.
3. Design your profile manager hierarchy schema.
4. Define the roles for each administrator according to your security policies.
5. Create the complete architecture design document that includes both the physical and logical design data.

For details on these tasks, refer to Chapter 10, "Logical design," on page 135.

Assessing your existing environment

Before you can select and install the Tivoli components and services, you must fully review and document the environment you plan to manage. Only through careful consideration of your organizational needs and limitations (for example, hardware availability) and an understanding of the technical capabilities and restrictions of the Tivoli software can an optimal design be created.

An understanding of the following conditions accelerates the design efforts and ensures that the Tivoli solution performs as expected:

- A thorough and accurate picture of your networking environment, of how the network is managed, and of how you expect it to be managed after the Tivoli deployment
- The behavioral characteristics and requirements of the Tivoli applications that are to be deployed
- The business goals for system management in your environment and how Tivoli applications will achieve those goals

The number of factors that need to be considered and their influence on the architecture varies for each organization. Viewing these factors from a limited number of perspectives, however, offers a way of arranging factors into manageable groups. The following sections consider your environment from these different perspectives:

Physical perspective
The network physical topology, such as available equipment and configuration of systems, in which the Tivoli applications must operate. The existing environment influences how and where you deploy systems that run the Tivoli applications. In many ways, the physical environment forms baseline constraints for the entire design. The relationship between physical factors and Tivoli applications is particularly close.

Application perspective
The Tivoli applications that will be deployed and third-party applications that will be managed or integrated with other Tivoli applications. Different Tivoli applications have different hardware and networking support requirements. How and how often an application is to be used can be an important factor in the design process.

Organizational perspective
The expectations and limitations derived from the way you do business

and division of authority and control of the systems to be managed. Organizational factors can influence the goals and means to deploy Tivoli environment.

The different factors have been broadly divided into the following categories for each perspective:

Critical considerations
Critical considerations include factors that have a direct impact on the Tivoli design process.

Important considerations
Important considerations include factors that have no effect on architecture, but could affect design in the sense of how to configure Tivoli Management Framework.

The factors, ordered by priority, are not a comprehensive list, and these vary for each organization. You might identify additional factors in your environment.

Physical perspective

The physical factors that require consideration during the design process include both network factors and computer system factors.

Network factors

Prepare a network diagram or sketch of the network topology, if one is not already available for your organization. This network design information is necessary for the design process and should at a minimum provide details about the following critical and important considerations. For a detailed discussion of critical network components and configuration for Tivoli Management Framework, refer to "Network considerations" on page 106.

Critical considerations:

- Line speed of each network connection, because it impacts the number of Tivoli regions and the placement of gateways, repeaters, and managed nodes
- Firewalls and Network Address Translation (NAT) devices in the network

  Provide the placement, configuration, and policies associated with each firewall and NAT device.
- Bandwidth restrictions

  Are there bandwidth restrictions that must be honored to share limited bandwidth with other applications? Also, does the available network bandwidth vary by time of day?
- Number of managed resources at remote locations

  Do they tend to be small branch offices, large distributed sites, or both?
- Number of subnets
- Number and placement of local area networks (LANs), routers, switches, hubs, and other network devices
- Network reliability
- Change control process for network changes

  Identify change control procedures for changes to the network such as subnet or IP addressing. Changes to the network need to be communicated to the IT administrators responsible for the maintenance of the Tivoli environment.

Important considerations:

- Systems to be managed by Tivoli applications that are not reachable by either IP or Internetwork Packet Exchange (IPX)
- Full forward and reverse name resolution support for domain name service (DNS) and DNS server redundancy
- Host and IP address naming conventions and schemes used to identify networking and computer equipment
- For IPX, server name, computer name, IPX address conventions and schemes used to identify networking and computer equipment
- Dynamic Host Configuration Protocol (DHCP) or Windows Internet Naming Service (WINS) server identification, redundancy of servers, and description of how these utilities are configured
- Socks protocol configuration
- Network monitoring
- Whether Transmission Control Protocol/Internet Protocol (TCP/IP) routing is structured static or dynamic
- Centralized or distributed server environment
- Windows domains, primary domain controllers (PDCs), and backup domain controllers (BDCs)

Computer systems factors

Prepare a list of computer systems along with their configuration that need to be managed by Tivoli Management Framework. This list should provide details about the following critical and important considerations.

Critical considerations:

- Number and location of servers and user desktops to be managed
- Types of computer systems and operating systems to be managed (for example, UNIX, Windows, Linux)

Important considerations:

- Hardware configuration of the computer systems to be managed

  The hardware configuration requirements for Tivoli Management Framework are described in "Recommended server and client configuration" on page 99.
- Software configuration of the computer systems to be managed

  The software configuration requirements for Tivoli Management Framework are described in "Recommended server and client configuration" on page 99.
- Disk space available on the systems to be managed
- Network interface cards and IP addresses
- TCP/IP stack used on the computer systems to be managed

  For a list of Tivoli-supported TCP/IP stacks, refer to the *Tivoli Management Framework Release Notes*.
- Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX) stack used on the computer systems to be managed

  For a list of Tivoli-supported IPX/SPX stacks, refer to the *Tivoli Management Framework Release Notes*.
- IP address and the host name of each computer system needed to be managed
- IPX address and server name of each computer system needed to be managed in IPX
- Whether Network Information System (NIS) or NIS+ is being used in the environment

If yes, identify the NIS domain master.

- If PDC domains are used for Windows systems, identify each domain and provide detailed information.
- Types of new systems to be deployed to support the Tivoli solution

  This question can be answered after your planning analysis, though in some cases, organizations might be predisposed to certain platforms.

Application perspective

Tivoli applications have varying environmental requirements; conversely, Tivoli applications have varying impacts on the environment in which they run. It is important to consider the behavioral characteristics of each of the Tivoli applications being deployed in your organization. Similarly, you need to consider the implications arising due to applications in use in your organization that will be managed or integrated with Tivoli Management Framework.

These factors must be considered before you can determine the number of Tivoli management regions to deploy, the repeater and gateway placement, the capacity required for servers and managed nodes, and so on.

Tivoli application factors

The following factors pertain to generally used Tivoli applications and are not a comprehensive list of factors. Some general implications for Tivoli application servers are discussed in "Application servers" on page 133. Refer to the specific application documentation to understand the full implications to your design.

Critical considerations:

- Applications and modules to be deployed
- Parts of the organization that require different Tivoli applications

  For example, will inventory be required for servers? Will monitoring or event management be required for user desktops?
- Your schedule for system monitoring (for example, 7 days a week and 24 hours a day or 5 days a week and 24 hours a day)

Important considerations:

- The Tivoli application most important to your organization, if you must stage the Tivoli deployment
- The system activities to be monitored and how the current monitors act on out-of-range conditions
- A change control procedure for applications and databases
- Source code control and method of control
- A problem management system and process
- System management applications currently in use

  Do you have plans to migrate these applications to new releases or to reinstall existing applications?
- Amount of customization required to write Tivoli monitors, create complex software packages, and so on

In-house application factors

In-house applications are all the non-Tivoli applications in use in your organization. Implications can arise if these applications will be managed or integrated with Tivoli Management Framework.

Important considerations:

- Applications, the servers on which these run, the clients who access these, and the application database name
- Your mission-critical applications
- The crucial network devices and computer systems for mission-critical applications
- The crucial daemons, processes, and services for your mission-critical applications
- Contingency plan for your mission-critical applications
- The type of support expected from Tivoli Management Framework for mission-critical applications (that is, monitoring, software distribution, inventory, or data distribution)

Organizational perspective

Prepare a list of organizational factors that might have an impact on the deployment. This list should provide details about the following critical and important considerations.

Critical considerations

- Organizational objectives, business processes, IT processes, business needs, and future plans

  This aspect has a great impact in the design process and needs to be looked at carefully from the physical and applications perspectives.
- Amount of growth anticipated for the environment

  The growth could be in terms of an increase in the number of managed resources within existing infrastructure or due to setting up of new offices.
- Internationalization considerations

  This could be the case when the organization spans different countries, and each country uses a different language on their resources. For example, how will an English language system understand a German language message?
- Support for managed resources separated by multiple time zones

  For example, if you have managed resources in Japan and England, the design process should consider the time when the profiles need to be distributed, when the managed environment will be available for maintenance tasks, and so on.
- The availability requirements for the management system to perform management tasks

  This will assist you in defining the maintenance schedule for Tivoli applications.
- What are your service level agreements for different management services?

  This will assist you in configuring your management servers and building up your redundancy in terms of hardware and software to meet such service level agreements.

Important considerations

- Primary risks to this project
- Number and responsibilities of existing system administrators
- Major impediments to the IT service your organization offers
- Unique Tivoli development, test, support, training, and production system environments
- The immediate problems you are trying to solve by deployment of Tivoli products

- Your immediate and long-term goals, specific to Tivoli products
- Change control process
- Will managed systems be restaged? What happens to managed resources such as systems that are taken off the network?

Planning the physical design of your network

Planning for the physical design of Tivoli Management Framework and its suite of applications in your network depends on a multitude of factors.

Determining the number of Tivoli regions

Determine the number of regions required in the environment. The following checklist provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	113	
Firewall implications	117	
Geographic and network considerations	122	
Slow network links	120	
Types of systems to be managed	120	
Number and types of Tivoli applications	118	
Nonproduction Tivoli regions for test, support, and training	113	
Performance considerations	113	
Growth and scalability of the Tivoli environment	115	
Number of endpoints and gateways	115 and 129	
Number of managed nodes	115	
Business organization considerations	116	
Centralized versus distributed administration	116	
Data encryption	117	
Limiting administrator access	117	
Failover and recovery of the Tivoli environment	160	

The following checklist provides page references to scenarios or illustrations.

Scenario or illustration	Explained on page...	Notes
Centralized administration scenario	116 and 142	
Distributed administration scenario	117 and 142	
Consolidating views of resources across Tivoli regions scenario	119	

Divide environment into Tivoli regions

If multiple regions are required, divide your managed environment by individual regions. The following checklist provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	113	
Isolated Tivoli regions versus connected regions	121	
Geographic and network considerations	122	
Business organization considerations	121	
Tivoli region connections	122	
Security concerns	117	
Types of managed systems	120	

The following checklist provides page references to scenarios or illustrations.

Scenario or Illustration	Explained on Page...	Notes
Hub and spoke illustration	153	
Hub and spoke: standalone configuration illustration	14	
Master-remote scenario	116	
Fully interconnected scenario	119	

Determining type of region connections

If you are using multiple regions for scalability reasons and need the regions to be connected, decide the connection topology. The following checklist provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	121	
Performance considerations	113	
Upper limits for connected Tivoli regions	116	
Encryption for Tivoli region connections	117	
Limiting administrator access to connected Tivoli management regions Tivoli regions	117	
Firewalls	117	
Limitations to viewing resources in connected Tivoli regions	119 and 158	
Maintenance requirements for connected Tivoli regions	163	

Key concept	**Explained on page...**	**Notes**
Slow network connections and Tivoli maintenance operations	121	

The following checklist provides page references to scenarios, and illustrations.

Scenario or illustration	**Explained on page...**	**Notes**
Fully interconnected hub and spoke illustration	153	
When not to make connections scenario	116	
Connections to support distributed administration scenario	116	
Viewing resources across Tivoli region connections scenario	120	

Determining the placement of the Tivoli server

After reviewing region connections, you must decide on the placement of the Tivoli server. The following provides page references to the key concepts.

Key concept	**Explained on page...**	**Notes**
Overview	122	
Improving network utilization and server load	13	
Slow network links	120	
Network bandwidth restrictions	114	
Firewall considerations	117	
Concerns for a wide area network (WAN)	123	
Geography implications	123	

The following checklist provides page references to scenarios, and illustrations.

Scenario or Illustration	**Explained on Page...**	**Notes**
Centralized administration scenario	116 and 142	
Distributed administration scenario	117 and 142	

Determining the number of gateways and their placement

After determining the placement of Tivoli servers, the next step is to size the number of gateways and their placement in the network. The following provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	128	
Scalability limits	115 and 129	
Failover support	129 and 164	
Network topology requirements	130	
Slow network links	132	
Number and location of endpoints	129	
Number and types of Tivoli applications	129	
Small workgroup configurations	10	
Impacts with the standalone configuration for hub and spoke connections	15	
Impacts with the combined configuration for hub and spoke connections	15	
Selecting an endpoint login method	104	
NAT environments	111	
Tivoli applications considerations	105	
NetWare requirements	61	

The following checklist provides page references to scenarios and illustrations.

Scenario or illustration	Explained on page...	Notes
Slow links and software distribution scenario and illustration	130	
Branch office scenario	132	
Small workgroup configuration illustration	10	

Determining the number of repeaters and their placement

After you have designed the gateway architecture, you can decide on the number of remaining repeaters and their placement in the network. The following provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	123	
Slow network links	120	
Network topology	124	
Number and location of computer systems to be managed	123	
Tivoli applications considerations	118	
Network bandwidth issues	114	

Key concept	Explained on page...	Notes
MDist 2 and RIM considerations	124	

The following checklist provides page references to scenarios and illustrations.

Scenario or Illustration	Explained on Page...	Notes
Placement illustration	126	
Breaking repeater distributions with improperly defined policy regions scenario	153	
Repeater depot scenario	128	

Determining the number of application servers and their placement

Do not wait until after installation to determine the number and placement of Tivoli application servers. The following provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	133	
Applications that require RIM	31	
Limiting administrator access to applications	139 and 147	
Application impact to the Tivoli server	99	
Performance considerations	110, 113, and 157	
Application placement and Tivoli region connections	153	
Applications and firewalls	117	
Number of event consoles	119	
Number of endpoints to be managed by an application	115	
Application failover considerations	159	
Inventory server considerations	118	
Software distribution considerations	99, 123, 124, 126, and 129	
IBM Tivoli Enterprise Console considerations	114, 119, and 129	
Using distributed monitoring with software distribution	120	

The following checklist provides page references to scenarios and illustrations.

Scenario or illustration	Explained on page...	Notes
Inventory scanning rate scenario	118	
IBM Tivoli Enterprise Console with software distribution scenario	15	
Software distribution and slow network links scenario	120	
User administration scenario	119	

Determining the number of RIM hosts and their placement

The number and placement of Tivoli application servers influence the number and placement of RIM hosts. The following provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	132	
RIM and Tivoli applications considerations	132	
RIM and MDist 2 considerations	124	

Planning the logical design of your network

Planning for the logical design of Tivoli Management Framework and its suite of applications in your network depends on a multitude of factors.

Deciding on the naming conventions

The first task in the logical design is to decide on the naming conventions to be used for all your resources. The following provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	135	
Host names and IP addressing conventions for networking and computer equipment	108	
Naming conventions for Tivoli objects and connected Tivoli management regions	119 and 158	

The following checklist provides page references to scenarios and illustrations.

Scenario or illustration	Explained on page...	Notes
Sample naming scheme for Tivoli objects	136	

Scenario or illustration	Explained on page...	Notes
Resource names and connected Tivoli regions scenario	119	

Designing the policy region hierarchy

Design the schema for the policy region hierarchy. The following provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	143	
Limiting administrator access	117	
Policy regions in relationship to profile managers	151	
Connected Tivoli regions considerations	153	

The following checklist provides page references to scenarios and illustrations.

Scenario or illustration	Explained on page...	Notes
Simple hierarchy scenario	23	
Extended hierarchy scenario and illustrations	143	
Division of roles with hierarchies scenario	139	
Hub and spoke Tivoli regions scenario and illustrations	153	

Designing the profile manager hierarchy

The next step in the logical design is to design the profile manager hierarchy schema. The following provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	151	
Using managed node versus endpoint-compatible profile managers (dataless versus database mode)	37	
Slow network links	120	
Connected Tivoli regions considerations	151 and 153	

The following checklist provides page references to scenarios and illustrations.

Scenario or illustration	Explained on page...	Notes
Simple hierarchy scenario	151	
Distribution scenario and illustration	43	
Extended hierarchy scenario and illustrations	143	
Hub and spoke Tivoli management regions scenario and illustrations	153	

Defining administrator roles

The final step in the logical design phase is to define the roles of the administrators according to your security policies. The following provides page references to the key concepts.

Key concept	Explained on page...	Notes
Overview	139	
Limiting access to resources	117	
Connected Tivoli regions	22	
Scheduler authorization	30	
Task authorization	28	
Centralized administration model versus a distributed administration model	141	
Failover	160	

The following checklist provides page references to scenarios and illustrations.

Scenario or illustration	Explained on page...	Notes
Examples of Tivoli roles of authority	20	
Limiting access in connected Tivoli regions scenario	117	
Profile creation and distribution scenario	147	
Software distribution security breaches scenario	139	
Distributed monitoring security breaches scenario	139	
Inventory security breaches scenario	139	

Chapter 8. Hardware and network considerations

This chapter discusses the hardware, software, and network infrastructure for successful deployment of Tivoli Management Framework. Using the architecture design document that you created (refer to "Overview of the design process" on page 85), prepare your systems and network as necessary. This could mean tuning the existing devices or procuring additional hardware. This chapter discusses the following topics:

- "Recommended server and client configuration"
- "Network considerations" on page 106

Recommended server and client configuration

This section describes the recommended hardware and software configuration for the Tivoli server, managed nodes, and endpoints.

The specific hardware platforms and versions of operating systems supported are described in the *Tivoli Management Framework Release Notes*. The requirements for other servers depend on the Tivoli applications being installed in your organization; therefore, they are not included in this discussion. Refer to the Tivoli product documentation and release notes for hardware and software recommendations for the servers of each product.

Tivoli server considerations

The Tivoli server is the foundation computer system in a Tivoli environment. The Tivoli server coordinates all communication with managed nodes and performs all authentication and verification necessary to ensure the security of data in your Tivoli environment. This server maintains the server database, which is larger than the databases maintained by Tivoli clients.

Hardware considerations for a Tivoli server

To select the computer system for a Tivoli server, consider the following factors:

- Use a computer system with good integer performance.

  Only integer operations are used for Tivoli operations; floating-point performance does not have any impact on Tivoli performance. The better the integer performance of the Tivoli server, the better the performance of Tivoli applications are.
- Dedicate the Tivoli server to Tivoli operations.

  The Tivoli server should not run a large number of non-Tivoli applications. Dedicating the Tivoli server to Tivoli operations provides optimal performance.
- Use a computer system of a size appropriate for your environment.

  Determining the size of the computer system for a Tivoli server depends primarily on the number of managed nodes in your Tivoli region and secondarily on the applications supported. If you expect to have only a small number of managed nodes, an entry-level computer system is acceptable. If you plan to have hundreds of managed nodes, use a server class computer system with fault tolerance, multiple CPUs, and a large amount of RAM.
- Use the recommended memory.

  The Tivoli server memory requirements are platform-specific. Generally, for a UNIX server, you should have 512 megabytes (MB) to 1 gigabyte (GB) of memory. Optimum performance is achieved by dedicating the computer system to Tivoli operations, because other applications might use up memory. For complete details, refer to the *Tivoli Management Framework Release Notes*.

- Ensure that you have the required disk space.

  Estimated disk space requirements for Tivoli Management Framework components are listed in the *Tivoli Management Framework Release Notes*. These do not include the space needed in the database directory for backups, each of which could be very large depending on the database size. The minimum recommendation for disk space is 1 GB for Tivoli Management Framework on the server.

  While planning the disk space requirements on your Tivoli server, you should ensure that there is enough room on the system for the software you are planning to install on it, both for the short term with your current installation and for the long term as you add computer systems and new architectures.
- Ensure that each Tivoli server has sufficient swap space.

  Ensure that enough process slots are available on your system to run Tivoli applications and any other applications. Tune the computer system to handle a large amount of processes. Use the information in the *Tivoli Management Framework Release Notes* to assess swap space and process slots in your environment.

  Depending on the size of your Tivoli environment, the amount of memory and swap space and number of process slots needed on the Tivoli server varies. Normally, the amount should be two to three times the amount of system RAM. Refer to the operating system documentation for information about how to tune the system for applications that have a large amount of processes.
- Tune the system to handle large amounts of input/output (I/O) activity.

  For example, if you have IBM Tivoli Enterprise Console installed, a large amount of disk input/output is required for its write processes.
- Ensure that your network uses TCP/IP for communication among managed nodes and Tivoli servers.

  The Tivoli distributed architecture is designed to work across a wide variety of LANs and WANs and network topologies. The minimum requirement is bidirectional TCP/IP lines for communication among managed nodes and Tivoli servers.
- Select hardware based on the Tivoli applications to be installed in your environment.

  Monitoring and event management requires a server class machine with multiple CPUs and a large amount of RAM. However, inventory scanning and software distribution do not require as powerful a machine as monitoring because you can distribute the activities that are performed by these two applications. These recommendations are dependent on how large your Tivoli environment is and how extensively you plan to use Tivoli applications.
- Plan for a backup device for your critical systems and the Tivoli database.

Contact your customer support provider to answer questions about the suitability of a particular system as a Tivoli server.

Software considerations for a Tivoli server

To configure the software environment, consider the following factors:

- Use the same port number for each object dispatcher in the Tivoli region.

The Tivoli server and its managed nodes each have their own object dispatcher, or oserv service, that runs continuously. The oserv service (or daemon) on the Tivoli server coordinates communication between the oserv service on each of its managed nodes.

- Install Tivoli Management Framework to create the Tivoli server before creating managed nodes and gateways and before installing any Tivoli applications.

  Installing the Tivoli server loads the libraries, binaries, and other files required to use Tivoli Management Framework. The server database is created and several system variables are defined.
- Enlarge file systems to allocate adequate space for the server database and other run-time data on UNIX Tivoli servers.

  Tivoli Management Framework installation software checks for adequate disk space before installation, but does not enlarge file systems.
- Create dedicated file systems for the server database directory, the installation directory, and the database backup directory.

  After the file systems are created, these file systems should be enlarged to allow for data expansion and installation of data. The size of the database backup directory depends on how many online database backups are required based on your service level agreement.
- Keep the server databases on file systems local to the Tivoli server.

  The server databases contain information central to the management of the Tivoli region. For Windows, databases must be kept on a local Windows NT File System (NTFS); these files should never be accessed through Network File System (NFS). If you are considering multiple Tivoli regions, the server in each region must have sufficient file system storage for these databases.
- Keep Tivoli binaries and libraries on file systems local to the Tivoli server.

  Storing the Tivoli binaries and libraries locally improves the ease of providing maintenance upgrades to the Tivoli environment.

Managed node considerations

Managed nodes use the same set of Tivoli binaries and libraries that are required by the Tivoli server. They also have a database similar to the one on the Tivoli server. From a managed node, you can run the Tivoli desktop and directly manage other Tivoli managed resources.

Use an endpoint, not the managed node, to manage the host system for the managed node. Endpoints require fewer system resources than a managed node, leaving more resources for the user or other processes. Reducing the number of managed nodes also reduces the workload on the Tivoli server.

You do not need to have a Windows operating system as a managed node solely to use it as an administrator desktop. You can run Tivoli Desktop for Windows on any supported Windows-based system, including endpoints and systems that are not part of your Tivoli deployment to manage Tivoli resources.

Install a managed node on a computer system that is responsible for system management functions. The managed node can be used as a host for the following (some of these items can be placed together on the same managed node):

- Gateways
- Gateway proxy to run policy scripts for NetWare gateways

Note: Although you can use the Tivoli server to host the gateway proxy, this configuration is not optimal because scripts take significantly longer to run.

- RDBMS Interface Module (RIM) hosts
- Any computer system used to run Tivoli commands
- Any computer system that runs Tivoli Software Installation Service
- Any computer system that requires a managed node, refer to the release notes for each Tivoli Enterprise software product to determine which require a managed node.
- Network Information Service (NIS) master server
- Any system used for Tivoli application development or extension

Hardware considerations for managed nodes

To select the computer systems for managed nodes, consider the following factors:

- Ensure that a managed node running the Tivoli desktop is a powerful computer system.

  The managed node needs to be capable of running a large Win32 or Motif application efficiently. It should provide good networking performance and should not be running a large number of non-Tivoli applications. Depending on the number of windows open at any one time, increase the amount of RAM in the managed node to improve performance.
- Use a computer system of a size appropriate for your environment.

  Managed nodes require more system resources than any other managed resource. This is seen primarily with the client database created on each managed node. As you install other applications on the managed node and perform management functions from that computer system, the size of the database grows.

  Depending on the applications installed, a client database typically requires 10 to 15 MB of disk space. Refer to the *Tivoli Management Framework Release Notes* for the Tivoli Management Framework database requirements. Refer to the appropriate application guides or release notes for their database disk-space requirements.
- Determine the size of managed nodes that will be gateways.

  The main factors to consider for sizing are the following:
  - Number of endpoints
  - Number of upcalls and downcalls
  - Size of the cache on endpoints
  - Number of endpoint platform types that must be supported
- Verify host name and IP address resolution for all systems that are to be installed as managed nodes before beginning the installation.

  Both the host name and IP address resolution is required for managed node installation for the following:
  - The managed node performing the installation
  - The system on which the managed software will be installed
- Use Tivoli Software Installation Service to install managed nodes.

If you are not running the Tivoli desktop from a particular managed node, you can reduce the RAM and swap-space requirements, depending on the computer system model and the applications to be run on the managed node. Tivoli Management Framework imposes no significant RAM or swap-space requirements on a managed node except when actual Tivoli operations are being performed. If you schedule Tivoli operations to occur during off-hours when the managed node is not busy, the RAM and swap-space requirements might not be as high as if you perform the operations when the managed node is also busy with other applications and processes.

Software considerations for managed nodes

To configure the software environment, consider the following factors:

- Keep databases on file systems local to the managed node for performance and database integrity reasons.

  The Tivoli database contains information central to the managed node. A managed node requires Administrator or root write access to its Tivoli database.
- Keep Tivoli binaries and libraries on file systems local to the managed node.

  Storing the Tivoli binaries and libraries locally improves the ease of providing maintenance upgrades to the Tivoli environment.
- Enable Administrator or root write access to the system where the Tivoli database and binaries will be installed.

  The managed node installation requires Administrator or root write access to the Tivoli database and the binaries. Write access to the Tivoli binaries is not required when the installation is complete.

Considerations for UNIX managed nodes

The following factors are additional considerations specific to UNIX managed nodes:

- Create dedicated file systems for the Tivoli database directory, the Tivoli installation directory, and the database backup directory.

  After the file systems are created, the three file systems should be enlarged to allow for data expansion and installation of data. The size of the database backup directory depends on how many online database backups are required based on your service level agreement.
- Enable the shell Internet service in the inetd.conf file.

  If the service is disabled for security reasons on a system, it must be enabled while the managed node is being installed.

Considerations for Windows managed nodes

The following factors are additional considerations specific to Windows managed nodes:

- Install the managed node on the system root drive formatted with NTFS.

  If the system drive cannot be converted to NTFS, the managed node must be installed on another logical drive that is formatted with NTFS.
- Install the Tivoli Remote Execution Service, a Windows version of the UNIX rshd daemon, before you install a managed node.

  Tivoli Remote Execution Service can be installed automatically on every Windows managed node, as long as at least one Windows repeater already exists in the Tivoli region (the first Windows managed node installed automatically is configured as a repeater). In some cases, Tivoli Remote Execution Service has to be installed on each Windows operating system manually or by using some other software distribution method before you install a managed node on the Windows operating system.
- Set the keyboard on the Windows target system to US/English.

  If you are using a non-US/English keyboard, verify that the kbdus.dll file exists in the WINNT\system32 directory. This file is the dynamic link library enabling the US/English keyboard. It is available from Microsoft.

- Keep the name of the initial Windows administrator account.

  Changing the name causes the installation of the managed node to fail, because the installation methods for Windows managed nodes assume that there is a Windows local administrator on the target system with the name Administrator.
- Ensure that Simple Mail Transfer Protocol (SMTP) is properly configured for Tivoli e-mail notification.

  Many Tivoli tools generate e-mail for alerts and other messages. For example, a software distribution could send e-mail after a distribution, and a monitor could send e-mail when a monitor engages. Tivoli Management Framework does not include a proprietary e-mail server, but instead relies on the UNIX **sendmail** command based on SMTP for sending these alerts and messages. If your environment includes Windows operating systems, Tivoli Management Framework is not able to send these messages, because Windows systems use the Messaging Applications Programming Interface (MAPI) instead of SMTP to route e-mail between a mail server and its clients.

  Use the **wmailhost** command during installation on all Windows Tivoli servers and managed nodes to properly configure Tivoli Management Framework to correctly forwarded e-mail notifications. Refer to the *Tivoli Enterprise Installation Guide* for more information.

Considerations for NetWare gateways

The following factors are additional considerations specific to NetWare gateways:

- Enable long name support on the NetWare system.
- You must install the gateway on the SYS volume.

  The gateway process uses SYS:public\Tivoli as the working directory.
- Ensure that the NetWare system can ping the Tivoli server.

Endpoint considerations

Endpoints do not maintain the Tivoli object database. This keeps the amount of resources required on the endpoint to a minimum, while simplifying the management of the database in the managed environment.

Selecting computer systems for endpoints

To select computer systems for endpoints, consider the following factors:

- Ensure that enough disk space is available on the endpoint for the endpoint method cache.

  By default, the method cache size is up to 20 MB, and the system drive must have sufficient space to accommodate this. This depends on which Tivoli applications run on the endpoint.
- Ensure that the computer system has at least 1 MB of memory (resident set size).
- Select any administrator to install the endpoint.

  Installing the endpoint does not require the Tivoli administrator authorization role that is required for the installation of other Tivoli Management Framework components.
- Specify the login interfaces list as the installation option.

  If multiple gateways are on the same subnet and the endpoint is allowed to perform the broadcast for the initial login, several of the gateways could receive the login request and process it. The login interfaces list directs the endpoint to contact a specific set of gateways instead of broadcasting. The broadcast operation will impact on the performance of the network.
- Use the endpoint default port (9495) for all endpoint installations.

Gateways by default are configured to use port 9494. By using different ports, both gateway services and endpoint services can run on the same system.

- As with Windows managed nodes, format the system drive with NTFS. If the system has no NTFS drives, create an NTFS drive for the installation of the endpoint.
- Install Tivoli applications that the endpoints will use on each gateway that the endpoint can log in to.

Choosing an endpoint installation mechanism

Installing the endpoints on a handful of systems can easily be accomplished on an individual system basis. If you are working with many endpoints, however, you need to be aware of the installation mechanisms available for managing efficient installation:

Tivoli Software Installation Service

With this service you can install endpoints onto multiple systems in parallel. It checks the installation prerequisites before installing each product. In addition, you can perform initial prerequisite checking without actually installing any products. You can also configure additional prerequisite checks that are specific to your environment. You can also use a response file to install Tivoli products from the command line. This feature is particularly useful when you install a product onto a large number of computer systems. Tivoli Software Installation Service creates an installation repository that stores the Tivoli product and patches images and imports these images before they can be installed.

Installation using Tivoli Software Installation Service can be performed on UNIX and Windows endpoints and is the recommended approach for these platforms. You can install many endpoints and other Tivoli products at the same time and monitor the status of these installations.

winstlcf command

This command installs and launches the endpoint on one or more systems. This command can install UNIX and Windows endpoints, except Windows 98. Because these Windows operating systems do not have a remote execution service, the first Windows endpoint in the domain must be manually installed using the local installation mechanism (described next) or you can use Tivoli Remote Execution Service as explained earlier. You can then use the endpoint as the proxy to install all other Windows endpoints within the same domain.

Local installation

You can locally install and start the endpoint on Windows and OS/2 systems. The installation mechanism provides silent installation capability to automate the installation.

Login scripts

A login script can be used to install and start the endpoint when the Windows or OS/2 user logs in to a Windows domain. You need to define the login script configuration on the Windows domain controller system before the endpoint installation. The login script is a powerful installation method in the Windows environment and automates the installation on new nodes without requiring administrator action. For Windows clients that cannot be network accessible at all times and that might not have a permanently assigned IP address, it is recommended that login scripts be used to install the endpoint from a remote share with a response file. This method is effective only if users log in on a regular basis and requires Windows domain credentials to access needed system services.

Other installation mechanisms

You can plan for an alternate method for installation of endpoints, such as users voluntarily downloading the endpoint installation files from an intranet server. Other methods can involve executing an agent in a Lotus Notes database or executing a script that links to the endpoint installation share and runs setup with a response file. These alternative methods require a significant resource expenditure for end user communication, tracking of compliance, and management follow-up.

For additional information about installing endpoints using these mechanisms, refer to the *Tivoli Enterprise Installation Guide*. For additional information about endpoints, refer to Chapter 5, "Endpoints and gateways," on page 47.

Network considerations

Correct network operation is critical to a successful Tivoli implementation. You must consider how to configure the following network components:

- Communication protocols
- Host naming conventions and resolution
  - DNS
  - DHCP
- Multihome hosts
- Very Small Aperture Terminal (VSAT)
- Network address translation (NAT)

Communication protocols

Tivoli Management Framework and its suite of Tivoli Enterprise software products primarily use TCP/IP. As a result, all Tivoli servers, managed nodes, and gateways in the managed environment must have TCP/IP installed, configured, and operational on their systems. Endpoints can use either TCP/IP or Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX). TCP/IP is supported for all endpoint platforms; IPX/SPX is supported only for endpoints on Windows and NetWare.

Understanding your TCP/IP addressing schema is essential to correct development of the gateway and repeater designs. For example, endpoints are easily assigned via TCP/IP address ranges. Understanding where a particular range of addresses is located relative to the endpoints assigned gateway can aid in optimizing network traffic flow.

The following describes different types of communication requirements and the ports used in the Tivoli environment for better planning in your environment:

Inter-object dispatcher communication

Communication between object dispatchers is the basic interaction mechanism with a Tivoli environment. The communication using TCP connections takes place whenever requests from one managed node invoke a remote method on another managed node. The inter-object dispatcher communication also manages dialogs between a managed node and a Tivoli server when a method request needs to be authorized and its implementation details resolved. This connectivity is also called the *object-call* or *objcall channel*.

The object dispatcher service listens on TCP port 94. This is a registered port, assigned by the Internet Assigned Numbers Authority (IANA). User

Datagram Protocol (UDP) port 94 is also issued by the IANA. All object dispatcher and interregional communication with the object dispatcher uses this port as the destination. These are sustained TCP connections (refer to Figure 31) and can be disrupted only when network faults occur or when a dispatcher is restarted.

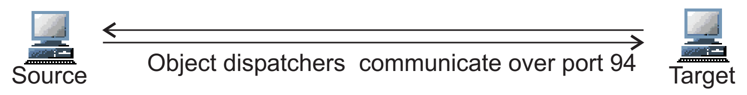


Figure 31. The object dispatcher communication over port 94

Inter-object messaging (IOM)

This is a communication channel that allows exchange of bulk data between two object implementations. This exchange takes place independently, or out-of-band (OOB), of the objcall channel. The creation of IOM channels occurs while the method runs on an on-demand basis. The IOM connection channels also connect via TCP. OOB communication typically runs over ephemeral TCP ports above 1023.

IOM channels are created by a method that wants to initiate bulk data transfers greater than 16 kilobytes (KB) with a remote method of an object. This connection requires use of OOB communication ports. Communication less than 16 KB runs over the existing TCP port 94 communication mechanism. The local method acts as an IOM-channel TCP server. The method selects a listening port, starts a network server, and passes the connection information to the remote method. This information, known as the DKEY, gets transferred securely through the objcall channel as an option to this remote method. The remote method at the other end, when invoked, uses the DKEY to establish the client end of the IOM TCP connection. When the bulk data is exchanged, the IOM connection is broken. Examples of IOM calls are operations such as a software distribution.

Bulk Data Transfer (BDT)

The BDT service is the underlying support for IOM and other large-scale data transfers. When used with an MDist 2 repeater, BDT can limit the number of target ports that are required for the Tivoli environment to operate, thereby increasing the effectiveness of firewalls. IOM runs on top of BDT so all firewall issues apply. The BDT service listens for BDT channels on behalf of all Tivoli Management Framework BDT clients on a given node. All BDT traffic for each managed node uses a single service port. Therefore, the port range is reduced to one configured port. Note that you can use the **odadmin set_bdt_port** command to set the port that the BDT service uses only if the **odadmin single_port_bdt** option is set to **TRUE**. If **single_port_bdt** is set to **FALSE** (the default), the BDT port is not used and does not apply to any other node operations.

In addition, you can use the BDT service with the SSL-A and SSL-B security packages. BDT channels are protected by the Secure Sockets Layer (SSL) protocol when the following conditions are met:

- Single port BDT mode is selected
- SSL is enabled on both managed nodes

For information about how to enable single-port BDT, refer to the **odadmin** command in the *Tivoli Management Framework Reference Manual*.

Endpoint and gateway communication using TCP/IP

An endpoint first uses UDP to advertise its presence in the enterprise. Gateways read UDP messages destined for the gateway port. After an endpoint is recognized as valid, it is assigned a permanent gateway. The endpoint then connects as a TCP client to the TCP server on the gateway service.

The gateway service can listen on any port chosen during the gateway creation. This port is not registered with the IANA. The gateway port selection is unrestricted. The default port is 9494.

The endpoint service also listens on a well-known port that is selected during installation. Like the gateway port, its selection is unrestricted and defaults to 9495. If the selected port is unavailable (this can happen if the endpoint service is terminated and restarted within the two-segment TCP lifetime), the endpoint service can be configured to either fail immediately or pick an arbitrary port that is available.

Endpoint and gateway communication using IPX/SPX

For IPX/SPX, the endpoint first uses an IPX packet (instead of UDP) to advertise its presence in the enterprise. When the endpoint receives its gateway assignment, it connects as an SPX client to the SPX server (instead of TCP) on the gateway service. Like the TCP/IP case, the default port for the gateway is 9494 and is not registered. If the selected port for the endpoint is not available, the endpoint can be configured to fail immediately or to pick an arbitrary, available port.

Host naming conventions and resolution

The managed environment should have a stringent host naming convention in place. Many environments generally use Domain Name Service (DNS), Windows Internet Naming Service (WINS), or host name files to resolve host names to TCP/IP addresses. In any case, there should be a one-to-one mapping of host names to IP addresses. As you develop your architecture, ensure that you understand how managed resources will be uniquely identified within the entire network.

Name resolution requirements are different for addressing managed nodes and endpoints. Managed nodes resolve the IP address of other managed nodes in the environment at creation time. When the managed node is created, the name of the managed node is used to resolve the IP address of the managed node. From that point on, the IP address is always used. The IP address is kept within Tivoli Management Framework and is used each time the managed node is started to connect to its Tivoli server. In addition, you should enable communication to always use the DNS name of a Tivoli server when interpreting an IOM key and making a connection. For example, DNS is useful for multihomed servers that are known by different IP addresses on different subnets. Set the **set_iom_by_name** option to **TRUE** using the **odadmin** command.

Tivoli endpoints use their gateways to do address resolution. When an endpoint logs in to the gateway, the IP address assigned to the endpoint is passed to the gateway. This information is maintained by the gateway and used for any downcalls made to the endpoint. There are no DNS, WINS, or other name resolution issues for gateways addressing their endpoints. This indirectly results in the gateway automatically supporting Dynamic Host Configuration Protocol (DHCP). If the endpoint is assigned a different IP address by DHCP at endpoint login time, this information is automatically passed to the gateway for use in performing downcalls to the endpoint. If the address has actually changed, the gateway passes this information back to the endpoint manager for the Tivoli region. Refer to"Dynamic Host Configuration Protocol (DHCP)" for more information.

In addition, endpoints can resolve gateways based on fully qualified host name or IP address of the gateway. Refer to the **select_gateway_policy** documentation in the *Tivoli Management Framework Reference Manual*.

Domain Name Service (DNS)

Tivoli Management Framework uses both the lookup and reverse lookup capabilities of DNS in communicating between Tivoli servers and managed nodes. In many environments, both these functions might not be available or might not be correctly maintained and updated on an ongoing basis. A poorly maintained DNS function might not directly affect the design of your architecture, but it will create problems during its implementation.

Most network-smart applications use the host name-to-IP address mapping, while other applications use IP address-to-host name mapping. Tivoli Management Framework uses both. You must have reverse mapping from Tivoli server to client and from client to Tivoli server. If you are using DNS and do not provide the IP address-to-host name mapping, the Tivoli client installation fails. The best way to determine if the correct mapping is available in your system is to run one the following commands:

- `nslookup` *nnn.nnn.nnn.nnn*
- `ping -a` *nnn.nnn.nnn.nnn* `(Windows only)`

If you run this command from the Tivoli server, *nnn.nnn.nnn.nnn* is the IP address of a client you plan to install. If you run the command from the client, *nnn.nnn.nnn.nnn* is the IP address of the Tivoli server. If this command provides the host name that corresponds to the IP address, DNS is configured for reverse mapping. If the host name is not provided, do one of the following:

- Add the IP address-to-host name maps to DNS.
- Stop using DNS on the Tivoli server.
- Use the LMHOST facility on Windows or add data to the /etc/hosts file and use /etc/hosts as a DNS fallback.

Dynamic Host Configuration Protocol (DHCP)

Dynamic Host Configuration Protocol (DHCP) allows an organization to dynamically assign IP addresses to computer systems in the network. When a computer system connects to the network, the DHCP server assigns the system an available IP address from a predefined range of addresses. Each system leases the address it receives. When the lease period expires, the IP address is disassociated with the system and returned to the list of available IP addresses. (Lease periods were specified when your organization configured DHCP. See your DHCP documentation for information about this and other aspects of DHCP.)

Tivoli Management Framework provides support to networks that use dynamic IP addressing. This support is provided for the following managed resources:

- Managed nodes
- Endpoints

If you have clients that require DHCP support, you must first configure DHCP in your environment. Tivoli Management Framework does not provide DHCP; it simply enables you to use DHCP and Tivoli Management Framework in your computing environment.

Note: For a Windows Tivoli server, ensure that it is running WINS and has a static IP address. Windows Tivoli servers must rely solely on WINS for name resolution; DNS cannot resolve addresses allocated for DHCP address leases. (You can use DNS for non-DHCP client name resolution.)

When a managed node connects with the Tivoli server, the managed node passes its current IP address to the server. If the IP address is different from the one that was previously known for the managed node, the server updates its address mappings. The Tivoli server can resolve IP address changes caused by DHCP as well as those caused by moving a computer system to a new subnet.

Use the **allow_dynamic_ipaddress** option set to **TRUE** on the **odadmin** command to enable IP address resolution on a Tivoli server. Refer to the *Tivoli Management Framework Reference Manual* for more information about the **odadmin** command.

DHCP is handled differently for endpoints. Endpoints notify their gateway of their IP address at login time. If the address has changed, the new IP address is sent to the endpoint manager.

Multihome hosts

Multihome hosts are computer systems that have multiple network interfaces on different TCP/IP subnets. Tivoli Management Framework supports multihome hosts through the **set_iom_by_name** option of the **odadmin** command. This option enables Tivoli communication to rely on the host name of the Tivoli server rather than an IP address. Refer to the **odadmin** command in the *Tivoli Management Framework Reference Manual* for a complete listing of options.

Mobile clients

A mobile computer system is a computer that is generally in the field, mostly disconnected, and that will usually connect from different geographical locations. This type of computer system is called a *mobile endpoint*. The user of such a computer system is referred to as mobile user.

Tivoli Management Framework provides the following management aspects for mobile endpoints in a Tivoli environment:

- Ability to deliver distributions to mobile endpoints without requiring them to be constantly connected
- Ability of the mobile user to perform some Tivoli operations while mobile endpoint is disconnected

Very Small Aperture Terminal (VSAT)

Satellite links create a technical challenge for a Tivoli Management Framework implementation. Many satellite links are relatively slow, for example, 9.6 KB or 19.2 KB. In addition to the link speed, there is also the transmission latency and the fact that many satellites use a “store and forward” transmission mechanism so that there is a delay within the satellite itself.

There are environmental parameters within the initial Tivoli Management Framework installation that can be adjusted to ensure that operations do not time out due to these delays. Your architecture might include some example calculations for how long certain operations might take over the link. These can be used to set the expectations early in the deployment. Example operations might include inventory scans of file distributions of an average size. As a general recommendation, increase all timeouts on the repeaters for VSAT environments. Refer to applicable Tivoli application documentation for additional VSAT impacts to the Tivoli environment.

Network Address Translation (NAT)

With the increase of Internet addresses worldwide, it is becoming more difficult for enterprises to assign globally unique IP addresses to each host. There simply are not enough unique IP addresses to ensure secure and accurate data transmission from site to site or company to company.

As a result, network address translation (NAT) devices are fast becoming a way for large, multisite enterprises to avoid IP address conflicts. These devices act as a conduit between two address spaces within an organization. NAT devices create virtual private networks (VPNs) by acting as a router between the spaces. Additionally, they transform IP addresses from private to public one way and from public to private in the reverse direction. In most cases, NAT devices share the public IP address space with a larger private address space.

Tivoli Management Framework supports the use of NAT in the following ways:

- Between a gateway and endpoints
- Between managed nodes

There are special considerations for using NAT in a Tivoli environment, and there are some other possible configurations, including the use of the IBM Tivoli Enterprise Console product. Refer to the Redbook *Tivoli Enterprise Management Across Firewalls* for using NAT in a Tivoli environment.

Basic NAT maps only the IP addresses of packets when they traverse through address space boundaries. Network Address Port Translation (NAPT) alters both the addresses and the ports. NAT is supported in Tivoli Management Framework; NAPT is not supported. Because of this implementation, do not use IP address during installation or in operational scripts when deploying Tivoli Enterprise software products in a NAT environment.

Tivoli Management Framework supports NAT computers systems in the following configurations. Before upgrading any systems in a NAT environment, ensure that the following NAT environment requirements are met:

- Each managed node or endpoint must have only one fully qualified domain name.
- A managed node or endpoint cannot have multiple interfaces mapping to a single host name. If a managed node has multiple IP addresses, each must resolve to a unique host name.
- For NAT support across a Tivoli region, each system must be able to be identified by a unique, fully qualified host name. Each host can use a different naming mechanism, such as Domain Name System (DNS) service or any related resource, such as Network Information Services (NIS) maps or a local /etc/hosts file. The naming mechanism provides a secure name resolution service that resolves host name addresses on both sides of the address resolution space. It is essential that the administrator ensures that all of these mechanisms resolve a given host name uniquely across the region.

Secure Shell (SSH)

Remote installations of managed nodes and endpoints require a connection to the machine on which the installation is being performed. When an rexec or rsh connection is used, data transmitted on the connection is not encrypted. Therefore, Tivoli Management Framework provides the option to use a secure shell (SSH) connection. An SSH connection encrypts all data sent over the connection.

You must install the SSH server on the machine on which the installation is being performed. To install managed nodes using SSH, you must install the SSH client on the Tivoli server. To install endpoints using SSH, you must install the SSH client on the managed node from which the SSH connection is made.

Before you perform an installation using the SSH option, you must configure SSH. For more information, see the section about configuring SSH in the *Tivoli Enterprise Installation Guide*.

For a list of versions of SSH that Tivoli Management Framework supports, see the *Tivoli Management Framework Release Notes*.

Chapter 9. Physical design

The physical design for the implementation of Tivoli Management Framework develops the underlying physical infrastructure on which the Tivoli solution will operate. When designed, deployed, tested, and operational, the physical design might be difficult to change without a disruption of the Tivoli environment and, therefore, the operational environment. Sufficient time needs to be allocated to ensure that the correct design has been developed and that the hardware that has been specified can be scaled up or down with solution requirements.

The physical design process includes the following high-level considerations:
- "Single or multiple Tivoli regions"
- "Multiple Tivoli region design considerations" on page 121
- "Rules for placement of a Tivoli server" on page 122

Single or multiple Tivoli regions

The Tivoli region is the physical and functional building block of the Tivoli solution. All Tivoli managed resources and management functions are centered in the region. Therefore, the foremost task to be performed in the physical design process is to decide the number of regions for your organization.

For each Tivoli region that you create, you must consider the physical resource requirement for the communication between each Tivoli region. Understanding the number and relationships of regions needed for the solution is critical to designing the overall solution, as well as in ordering the required hardware.

This section describes the following criteria and limitations you should consider when deciding whether to use a single region or multiple regions and where the region boundaries should occur:
- Performance considerations
- Scalability limits
- Organizational considerations
- Security considerations
- Application considerations
- Managing servers versus user desktops
- Slow network links
- Reliability concerns

While designing your Tivoli environment, you should identify regions for carrying out the development, test, support, and training functions. Ideally, these regions should be separate from your operational environment but should mirror or sample the resource content and configurations of the operational environment.

Performance considerations

One of the most important factors that determines the number of Tivoli regions needed to manage an enterprise is performance. Performance degradation in the managed environment could result due to various reasons, such as overloading the region and other management servers, network congestion, or the deployment of region-intensive applications.

During planning, it is difficult to predict the management environment performance because of circumstances that are apparent only after the deployment is complete. Analysis of your physical environment and applications being rolled out mitigates problems with performance. Performance is directly linked to the frequency and types of management operations being planned.

When multiple regions are created primarily for a management system performance goals, the regions are generally connected to each other. It is recommended that you use a hub and spoke architecture in connecting regions when multiple regions are created for performance issues. This architecture is described in detail in "Hub and spoke connections" on page 13.

From the physical planning perspective, consider the Tivoli server load and network limitations.

Server load

Tivoli Management Framework consumes some amount of the CPU processing power and memory of the host server system. The limited resources of the host represent a practical limit to the number of Tivoli clients that can be reasonably serviced by a single Tivoli server. One of the possibilities of decreasing the load on a Tivoli server is to create multiple regions. In essence, this distributes the management load of a set of clients across multiple servers.

The load on the server can occur in the form of file descriptors needed to maintain contact with the clients, the amount of network traffic between the server and the clients, and the CPU or memory demands of the server activity itself. Also consider future growth in the deployment of Tivoli products into an enterprise.

How you install and create Tivoli managed resources affects the load on the Tivoli server. Tivoli processes run on the computer system where the object is created. Therefore, if you create all Tivoli objects on the Tivoli server, all the processes run on the server. Depending on the size of your installation and the size of the Tivoli server, this could put too much of a load. To avoid this situation, install some objects (such as the event server) on lesser-used managed nodes.

Network limitations

In general, Tivoli Management Framework uses Transmission Control Protocol/Internet Protocol (TCP/IP) for communication, although it does support Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX) communication on PCs connected to NetWare gateways. The NetWare gateways communicate with the Tivoli server using TCP/IP. Therefore, the amount of network traffic that can be sustained by the networking interfaces on a Tivoli server represents a practical limit to the number of Tivoli clients that can be serviced by a single Tivoli server.

Consider the following ways to improve the Tivoli server networking performance:

- Introduce high-speed network adapters on the Tivoli server
- Design the network topology so that unnecessary traffic does not flow on the Tivoli server network segment
- Use local file system storage for Tivoli binaries and libraries
- Limit the network traffic from the Tivoli server to other clients and vice versa

The first two ways pertain to your physical network design, for which your network specialists should be consulted.

Using local file system storage for Tivoli binaries and libraries, and using multiplexed distribution repeaters for large file distributions (such as profile and software distributions) can greatly reduce the demand on local resources (network bandwidth, CPU, or memory). Endpoints also minimize this problem. Because the majority of clients in an installation are endpoints, fewer managed nodes need to communicate with the Tivoli server. Each gateway is enabled for multiplexed distribution, which further reduces the resource demand.

If you have managed nodes instead of endpoints, using local file storage and multiplexed distribution repeaters lessens the load on the network. At some point, however, the demand exceeds the capabilities of a single region.

Scalability limitations

One of the most important factors for deciding whether to use single or multiple Tivoli regions is the scalability limitations of the Tivoli environment:

Number of endpoints

The number of endpoints managed by a single region has been increased to tens of thousands. It has been shown in production environments that 20,000 endpoints can be managed by a single region. For organizations requiring more than 20,000 endpoints to be managed, multiple regions are required. The limit of 20,000 endpoints represents a threshold beyond which special performance and tuning requirements might be needed. Therefore, use multiple connected regions.

Number of managed nodes

Generally, a single Tivoli server can support a maximum of 200 managed nodes. However, use endpoints instead of managed nodes in most cases. Endpoints are the preferred mechanism for managing your environment. The introduction of endpoints greatly reduces the number of managed nodes in a single region. A gateway installed on a managed node can perform all communication and operations with thousands of endpoints. Endpoints therefore have no direct communication with a Tivoli server.

In addition, the ability to perform maintenance functions such as database checks are greatly inhibited by large numbers of managed nodes. If the network contains more than 200 managed nodes, create multiple regions and connect them.

One reason for the limit of 200 managed nodes per Tivoli server is based on how the server maintains contact with each managed node. The server opens a connection with a managed node when communication is needed. As a performance optimization, the server keeps the connection open even after the operation is complete, thus avoiding the overhead of opening and closing connections for each operation. This effect can be compounded with connected regions, which put the same connection load on a Tivoli server that a managed node does.

Many operating systems, however, impose limits on how many file descriptors (or file handles) a given process can have open at any one time. Limits include all local and remote connections and all database files. Consult the operating system documentation on these limits and how to modify the limits. In addition, any applications that run on the same operating system impact the number of file descriptors open. Thus, it is important to consider how an application uses the operating system.

Local connections are associated with methods and external clients (command line programs), and remote connections are associated with other client systems. All remote communication, including communication between the clients and Tivoli server within the local region and in other regions, is affected by these limits.

To avoid performance overhead, limit the number of managed nodes in a region to 200, even if specific operating systems do not have this limitation. If you use more than this limit, the performance can be impacted.

Number of active desktops
A region can support a maximum of 30 concurrently active desktops, regardless of how the desktop is to be used. A region can support 50 desktops if you use Tivoli Desktop for Windows and connect to a managed node that is not the Tivoli server. In this case, one managed node should be dedicated for each 10 desktops. For performance reasons, do not run any desktops off of the Tivoli server.

Note: When using Tivoli Desktop for Windows, each window opened creates two connections to the managed node.

Number of interconnected regions
Do not plan for more than 15 two-way interconnected regions in your Tivoli environment. This is not a built-in limit, but represents a threshold beyond which special performance and tuning issues might arise. Resource updating through the **wupdate** command and environment checking through the **wchkdb** command become more difficult to perform, thus requiring special attention after these limits are reached.

Organizational considerations

Organizational considerations can lead to the decision to use multiple Tivoli regions. Many organizations have organizational structures that require multiple management domains that are self-contained. These domains might not share information and management duties with a central or enterprise level domain. In some cases, these domains are divided along business unit lines or they may be divisions within the information technology organization itself. In either case, there is a requirement that one domain not be dependent on another domain for any management function. In these cases, each domain should have its own region. These regions can communicate information with each other, but this is not required for their operation. There might not be any connection between the regions, so they can be seen as isolated single region implementations.

Another factor that affects the number of regions is the need to perform local administration. If you use a centralized system administration approach (that is, administrators in a single location administering distributed systems), a single region might be the best solution. If, however, you use localized system administration (administrators at various operational sites), multiple regions may be more appropriate.

For example, ABC Instruments has a southern sales region with 150 sales offices throughout the southeastern United States. Each office has one managed node. The main sales office for the southern region is in Atlanta, and all the system administrators work out of this office. With all the system management resources centralized in this way, ABC Instruments would create one region for the entire southern region.

The northeastern region of ABC Instruments is quite different; New York state alone has 150 sales offices. The other states have another 100 offices. Because of the number of computer systems in New York, ABC Instruments has an administrator in one of the New York offices to manage only those computer systems. This means ABC Instruments is using local management over the New York systems and centralized management over the remaining systems in the sales region. In this situation, ABC Instruments should create one region for New York and a second region in another city, such as Boston, for the remainder of the northeast sales region. ABC Instruments can then connect the two regions using a two-way connection, allowing the resources to be managed by either administrator.

Security considerations

A few security considerations might require multiple Tivoli regions: encryption levels, limited access, and firewalls. Refer to Chapter 3, “Overview,” on page 19 for a description of the security features.

Encryption

The **Simple** encryption level is usually appropriate and recommended for most Tivoli environments. This encryption level protects Tivoli security credentials communicated between object dispatchers from being compromised by network-level intrusions.

A Tivoli environment cannot mix encryption levels within a single region. All nodes within a region must use the same encryption level (**None**, **Simple**, or **DES**). If you need different encryption levels between the clients, you must create multiple regions, although, this should not be the sole reason for considering multiple regions in your environment. With a slight performance overhead of encryption, you can make the entire environment use the highest level of encryption, **DES**, rather than using multiple regions.

Limited access

A security consideration when selecting a single region or multiple regions might be the desire to restrict what local administrators can do on computer systems elsewhere in the enterprise. For example, you might want to give a local administrator the ability to perform most administration tasks on a local set of computer systems but no visibility of a different set of computer systems. One way to achieve this goal is to create multiple regions and to use a one-way connection. The local administrator cannot see any of the resources in the other regions, although, this should not be the sole reason for considering multiple regions in your environment. Most security concerns can be addressed by carefully designing policy regions and Tivoli administrator definitions, unless your environment requires physical isolation of operators or resources.

Firewall

Another security consideration for considering multiple regions might be due to firewalls in your environment. The major architectural concern for Tivoli Management Framework operating across firewalls is one of port availability. You need about five times *N* (5*N*) ports on the firewall for the communication across it, where *N* is the number of managed nodes that reside on either side of the firewall and are part of the same region or connected regions. Refer to the redbook *Tivoli Enterprise Management Across Firewalls* for detailed information about firewalls.

Additionally, some Tivoli applications do not always use Tivoli Management Framework services for network communication, and therefore do not honor the port range that you set. For example, IBM Tivoli Enterprise Console nonsecure event logging (**postemsg**) does not use Tivoli Management Framework.

Verify your architecture to ensure that none of these communication channels can be run between resources on either side of a firewall.

In organizations that do not have many ports available or have significant security concerns about opening inbound or outbound ports to a large number of external systems, consider deploying a small region outside of the firewall. This adds some additional complexity to the deployment as sufficient system, disk, and memory requirements must be accommodated on this computer system. The advantage of placing a region externally is that only one external computer system needs to communicate to one internal system. If this is done, the internal region and the external region cannot exchange resources or the internal region attempts to communicate with the external computer systems directly.

Application considerations

Applications being deployed in your managed environment play a significant role in your decision of single or multiple regions. This section considers two applications, the Inventory component of IBM Tivoli Configuration Manager and IBM Tivoli Enterprise Console as examples of how applications affect your physical planning. Another section, "Consolidating views of resources across Tivoli regions" on page 119, is related to the behavior of different application resources in a multiple region environment.

Inventory considerations

RDBMS Interface Module (RIM) provides access to various types of relational database managers for applications such as IBM Tivoli Configuration Manager and IBM Tivoli Enterprise Console. The RIM database can have relatively CPU-intensive activity during full-scan inventory updates of the database. As a result, the limits for RIM performance establish the maximum number of endpoints per region and, therefore, the number of regions in your environment.

The Inventory component of IBM Tivoli Configuration Manager requires RIM to update the database with recently retrieved inventory scan data from the endpoints. This data funnels through the Tivoli server and to the RIM host for entry into the database.

Generally, assume that it takes 2 minutes per endpoint to collect scan data on a local area network (LAN). This guideline can help you forecast how many endpoints can be scanned and the data updated in the RIM database in a fixed period of time. Experience with Inventory in performing full scans and differential scans in a large environment has resulted in this number being an average scanning time for a full scan (not a differential scan) of an endpoint. This average takes into account failure rates with systems being down and needing to be rescanned, system failures that result in the loss of scan data, and other considerations in attempting to achieve a 100 percent success rate.

Using 2 minutes per endpoint, you can calculate the maximum number of endpoints per region. Assuming that you have 7 working days with 24 hours per day to scan, you then have a total of 10,080 minutes to scan your environment and to write the data to the RIM database. These calculations result in a maximum of 5,040 endpoints per region if you want to implement a full rolling inventory scan, collecting full scan data on all endpoints in the environment every 7 working days in a LAN environment. In the same type of environment, you can expect 10,080 differential scans in 7 working days with a 24-hour scanning period. Production environments have shown that these numbers in a hub and spoke region to be acceptable.

When differential scans are performed, experience shows that the average scan time per endpoint is less than 1 minute, which allows about 10,000 endpoints per region. These numbers provide planning data for sizing your Tivoli regions.

To allow for the scanning of larger environments, multiple RIMs are required. This is achieved by spreading out the endpoints across multiple regions with each region having its own RIM object pointed to the same database.

Event server considerations

If you are deploying the IBM Tivoli Enterprise Console product, you can have only one event server per region. When deciding the number of event servers, and therefore, the number of regions, consider the following:

- The event server can support a sustained rate of event processing of about 10 to 12 events per second, that is, 36,000 to 43,200 events per hour. This processing is highly dependent on the complexity and structure of the event rule base. Obviously the more complex the rule base, the more processing and input/output (I/O) overhead is required. Depending on the number of event sources, this might be a reasonable number of events to expect. Before additional event servers are put into the design solely due to expected event rate, a careful review of event criteria and the possibility of local event filtering at the resource or element manager level should be explored.
- An event server can support 10 (15 in optimal conditions) simultaneous users (event consoles). This is a user-interaction processing consideration. It is highly dependent on the number of events being presented to the operators and what types of interactions the operators have with event consoles. If the initial requirement is for more than 10 event consoles, you should explore why each person needs an event console. If it is for informational purposes only, can they be informed through another method? For example, can you send them an e-mail, a page, or provide a pop-up window. Also consider whether each designated operator is able to understand the content of the event in the form in which it is presented.

Because IBM Tivoli Enterprise Console is usually positioned as the center of the event management solution, the design should provide redundant function in the event of failure. A recommended approach for such a requirement is to use a high-availability hardware configuration rather than multiple event servers or multiple regions. In the high-availability systems, options are available from hardware manufacturers to allow access to applications and data in a shared mode, and the switching from one hardware system to another is done below the application level. Refer to “High-availability design” on page 164 for more information about Tivoli Management Framework and failover systems.

Consolidating views of resources across Tivoli regions

Connected regions imply an initial exchange and periodic update of names and object identifiers contained in each Tivoli name registry. Some resource types recorded in the Tivoli name registry cannot be exchanged between connected regions. You might, however, encounter times when you want a consistent or unified view of that resource in multiple regions. In this situation, you must create a region, in addition to the existing ones, to get a consolidated view of the resource. For more information, refer to “Name Registry” on page 11.

For example, consider the XYZ software company. Because of the number of managed nodes in the network, XYZ software company has three regions, all which are connected with two-way connections. XYZ software company has installed a user administration application for user and group management and has a policy that all user names must be unique.

This user administration application adds a resource class called UserProfile, which maintains a list of all user names in the region. Using validation policy, the organization can require unique user names. Each time a user is added in the region, the Tivoli name registry is checked to ensure that the name does not already exist.

XYZ software company needs a consistent view of existing user names in all its regions. However, the UserNameDB resource cannot be exchanged across connected regions. Therefore, the organization can require unique names within a region but cannot enforce unique user names across connected regions. When the user is added, the name registry of the local server is checked.

The other way to present a consistent view of this information is to use the data in another resource that is shared across Tivoli regions—in this case, in a user profile. XYZ software company needs to create a new Tivoli server where all user account information will reside. This Tivoli region might include only one managed node. From this managed node, system administrators maintain a user profile containing the unique user names. Another Tivoli server can subscribe to the profile manager in the new region. The user profile is distributed to each Tivoli server and then distributed to targets. The profile can be locked against changes at lower levels in the distribution hierarchy, thereby ensuring that changes are made only in the original profile.

In some cases, resource types that cannot be exchanged cause applications to behave differently across regions than they would behave within a region. In most situations, the solution may be to create another region and use other Tivoli tools to distribute the required information.

Differences in managing user desktops and managing servers

In a small to medium managed environment, one region can manage both the servers and user desktops. In an environment having a large number of servers and user desktops, managing both these entities through one region might not be the right approach because they can have differing management performance requirements.

If you have a difference in performance requirements between managing servers versus managing user desktops, consider having two regions: one for server systems and one for user desktop systems. Refer to “Standalone configuration” on page 14 for an illustration of managing servers and user desktops.

Slow network links

Slow network links can affect region configuration. In general, do not create a second region on the other side of a slow network link simply due to link performance. Better solutions can enable you to avoid the complexity of multiple regions for this reason. However, in situations described later in the section, consider a second region across the slow network link.

The impact of slow network links is most obvious when large amounts of data are being transferred between a server and its clients, as in the following examples:

- Profile pushes as a result of subscription

  When a profile is being pushed to a subscribing client, all the data contained in the profile is distributed to the client.
- Software distributions

When a software distribution is pushed, the server may distribute a large amount of data to a number of clients simultaneously.

In each of these cases, the potential exists to transmit a large amount of data. However, the following solutions to these problems avoid the overhead of creating a second region:

- Repeaters allow software and profile distributions to be transmitted once over a slow link and then distributed to all systems on the other side. In Chapter 6, "Multiplexed distribution services," on page 63 the repeater mechanism is more completely described.
- As described in "Network limitations" on page 114, local storage of the Tivoli binaries and libraries avoids the need to send large volumes of binary or library data across a slow link.

Consider a second region if you have a single Tivoli server managing a large number of managed nodes connected on the other side of the slow link. A region that spans slow links and uses utilities such as the **wchkdb** and **wbkupdb** commands can cause problems. Because of multiple managed nodes connected across the slow link, the complete consistency checking (**wchkdb**) can take a considerable amount of time to complete. Because of the existence of slow links, the backup operation (**wbkupdb**) can take a considerable amount of time to complete.

Reliability concerns

There is a perception that multiple regions provide a degree of reliability against Tivoli server host failures by dividing the management responsibility for a collection of clients across multiple Tivoli servers. While there is some benefit to multiple Tivoli servers in terms of redundant hardware availability, a two-way region connection between peer Tivoli servers is not a way to provide redundant Tivoli server function. Both Tivoli servers must be up and available for all interregional functions to work. If a Tivoli server is down, all clients owned by that Tivoli server are likewise unavailable to the Tivoli server in the connected region.

Do not create multiple regions solely to provide redundant Tivoli server function.

Multiple Tivoli region design considerations

When you choose to implement a multiple Tivoli region solution, you must divide the entire Tivoli environment into multiple regions and then configure them. The regions can be configured in these ways:

- **Isolated regions—**Each region has its own Tivoli server for managing local clients and a set of distributed replicated services for performing management operations. Because each region is a management solution by itself, there might be no region connections. Administrators defined and operating within the region can only manage the resources within a region.
- **Connected regions—**Regions can be connected to coordinate activities across the network, enabling large-scale systems management and offering the ability for remote site management and operation.

If the multiple regions are intended for self-contained management domains as mentioned in "Organizational considerations" on page 116, each domain should have its own region. In this situation, having standalone regions might be the best configuration for an organization. The administrators in each region can manage the resources within their region with no interdependencies on other regions.

For connected Tivoli regions, partition your environment based on the following factors:

- Network topology, such as regions for different LANs or according to different TCP/IP network addresses
- Organizational and management responsibilities, such as regions for different groups or departments such as Marketing, Accounting, and Sales departments
- Geography, such as regions for different cities, countries, or other locations
- Type of managed resource, that is, managing servers versus managing user desktops
- Security concerns, such as firewalls

It is recommended that you physically divide the regions by network topology or geographical location. Regions are physical entities. You need to consider the network topology as a primary consideration.

In general, region connections are highly flexible and can be set up to map onto any arbitrary set of connections. In many ways, determining the appropriate region configuration is similar to establishing Network Information Services (NIS) servers across subnets, Windows domains, or file servers and network traffic.

Generally, a two-way connection is preferred so that flexibility is built into the design from the beginning. Security concerns, if any, can be addressed through the use of policy regions and Tivoli administrator definitions.

Each type of region connection provides benefits and drawbacks in comparison to each other:

- **Hub and spoke**—The most common and flexible model for connecting regions is the hub and spoke architecture.
- **Master-remote**—The primary difference between hub and spoke and this model is that there are local management functions that can be performed directly from the remote region in this model. The administrators must be configured to run directly on the remote region. Also, the logical design must support management functions that originate from the remote region. The master region can drive some level of enterprise policy, but the management interface shifts from the hub region in the hub and spoke model to the remote region in the master-remote model.
- **Fully interconnected**—When fully interconnecting all regions, the regions are placed more for operation considerations than network and performance considerations as found in the hub and spoke model.

Chapter 2, "Overview of the Tivoli architecture," on page 9 describes these topologies in detail.

Rules for placement of a Tivoli server

Consider the types of servers as you create the physical design. Other than the Tivoli servers, the requirement for other servers depends on the Tivoli applications being installed, such as:

- Tivoli server
- Repeaters
- Gateways
- RIM hosts
- Application servers

You need to decide how many servers you need, their placement, and their relationships with each other.

Tivoli server

There is one Tivoli server per Tivoli region. When you have determined how many regions you need, you have effectively determined the number of Tivoli servers required. The Tivoli server is generally placed close to where the Tivoli administrators are located. This is true for cases when you have a single region, isolated regions, or a hub region. This discussion considers only the hub and spoke model of connected regions, and you can extrapolate from it according to your model of connection.

There is a trade-off between placing spoke regions close to the hub and placing the spoke regions farther away from the hub. For example, you might place the spoke regions across a wide area network (WAN) from the hub region, which makes the spoke regions closer to their managed nodes and endpoints. Keeping spoke regions on the other side of the WAN is generally a better configuration. This enables the spoke region to serve as the first repeater in a distribution hierarchy across the WAN. It also helps to minimize the expense of running region maintenance functions across the WAN. One typical function that takes a long time to perform across the WAN is the **wchkdb** command.

Another reason for putting spoke regions on the other side of the WAN is the flow of inventory data back to the database, if you are performing inventory scans. The flow of data does not follow the repeater structure and always flows through the Tivoli server. Having the Tivoli server on the other end of the WAN link provides for more flexibility in how you tune inventory to minimize network impacts for this flow of data.

This configuration, however, can frequently result in requiring more spoke regions than is otherwise required. The number of remote sites might exceed the maximum number of region connections. Or each remote site might be so small that the number of managed systems at that site does not warrant a separate spoke region.

Spoke regions should be geographically based, being placed at key data centers remote from the hub region. These spoke regions communicate with gateways at remote sites distributed through the spoke region geographies.

Repeaters

With gateways and endpoints, region installations greatly benefit from increased scalability without the need to configure additional repeaters. However, organizations with several managed nodes and endpoints require a well-configured repeater schema to balance the distribution load across the network. The number of repeaters in a region and the relative balancing of work between repeaters can have a significant impact on the overall performance and scalability of data distribution operations.

In most organizations, a typical distribution goal is to move data as quickly as possible across the network while using as little network load as possible. For many businesses with fast LANs between clients within a region, the default configuration—the Tivoli server connected to gateways to distribute data—is sufficient. However, in some network environments, the default configuration may not meet the data distribution needs of the organization, perhaps because of varying network topology and network speed. In addition, consider how to offload some work from the Tivoli server. In these situations, you can do the following:

- Add repeaters
- Adjust the flow of data at each repeater

To calculate the correct number of repeaters for the region, consider the following questions:
- How many system resources can be dedicated to support multiple simultaneous network connections for data distribution?
- How large will the largest data distribution operation be?
- Will certain types of network connections hinder a distribution (such as slow WAN connections)?
- Are there any time constraints under which the largest distribution must be completed to a single set of computer systems?
- What multiplexed distribution tuning parameters enable controlled use of network bandwidth, memory and disk paging space, simultaneous network connections, and so on?
- How many systems will simultaneously receive distributions?
- Are the systems located on multiple subnets that do not bottleneck through a common router?

Determining the file size of the distributed file, the number of target clients, the system resources of the repeater, and the bandwidth between a repeater and its targets can be a crucial step in correctly configuring a repeater. Tivoli profiles can range from a few kilobytes to several megabytes, so establishing this information before distribution can avoid time-consuming and costly hung distributions. Software distribution profiles are generally the largest files that you will distribute. If you plan to distribute software distribution profiles, refer to the *IBM Tivoli Configuration Manager for Software Distribution User's Guide* for detailed information about configuring repeaters to deploy software packages.

The following factors influence the speed and the efficiency of a distribution:
- Size of distribution
- Network topology and line speed
- Network bandwidth available for distributions
- Number of endpoints and number of endpoints per subnet
- Endpoint availability (is the endpoint on the network and powered on?)
- Repeater hierarchy, including fanout patterns
- Multiplexed distribution tuning parameters (bandwidth, number of connections, memory cache size, and so on)

Guidelines for configuring repeaters

Whether the repeaters are gateways or managed nodes, knowledge of the underlying network topology and network performance characteristics are required for proper configuration of repeaters. The following selection guidelines apply to most environments:
- If you have a slow network link, such as over subnets, install a repeater on each side of the connection with a local copy of the Tivoli binaries.
- If a client is often a source host for software distribution, make it a repeater to enhance performance.
- If a repeater range contains too many clients in multiple subnets, improve performance by adding a repeater to each subnet.
- Limit the range of a repeater to clients within a single subnet or other repeaters.

- If a repeater fails or is not available, the transfer fails for everything (repeaters and ranges) that falls below that repeater in the distribution hierarchy. Therefore, do not use an unreliable client as a repeater.
- If you plan to use the MDist 2 service, install and configure RIM.
- Because a repeater is essentially a managed node, you add complexity in your environment by having repeaters. Therefore, add repeaters judiciously.

A gateway repeater range is dynamic. Each time an endpoint is assigned to a gateway, the gateway repeater range is automatically modified. However, repeater ranges on managed nodes and Tivoli servers are static.

Additionally, the MDist service does not provide automatic rerouting for failed repeaters. Recovery attempts are defined by the application that initiates the transfer. Refer to the application documentation for details.

If a transfer is initiated from a client that is not a repeater, the initial transfer sends a single copy of the data to the client repeater. The transfer then follows the defined distribution path. For example, if managed node E initiates a distribution, the default behavior sends the data to the nearest repeater, which is repeater D. Repeater D then distributes the data to the other clients: managed node F, gateway B, gateway A, and gateway C, as shown in Figure 32.

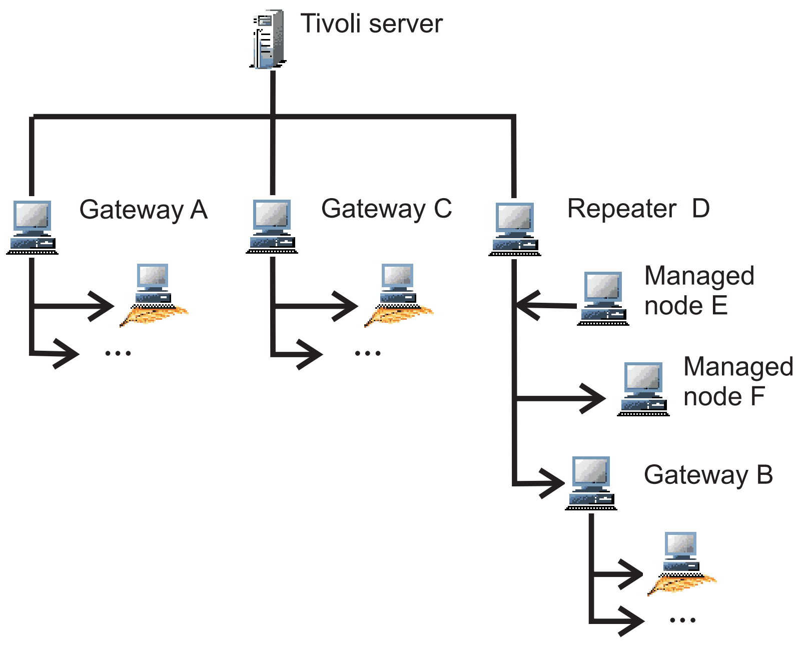


Figure 32. Distribution initiated from a host that is not a repeater

If a specific client frequently initiates transfers and is large enough to handle multiple network connections, the client is a likely candidate for a repeater.

If you configure managed node E as a repeater and initiate a distribution, data is distributed directly to the other repeaters: gateway B, gateway A, and gateway C. Because a client can be a target client as well as a repeater, you must put the client (which is also a repeater) in the range of another repeater to establish a distribution hierarchy. In other words, do not put a client that is a repeater in its own range.

You can also configure a client as a repeater with an empty range, which means that a repeater has no clients. In this situation, the repeater acts as a repeater only for transfers that it initiates.

To configure a repeater to efficiently operate as a distribution node, use the tuning parameters as described in the *Tivoli Management Framework User's Guide* to adjust the flow of data distribution. Each set of tuning parameters determines the amount of system resources the repeater can consume. You can adjust these parameters to set the network load, the maximum number of open connections, and the maximum memory and disk use. You can also configure the disk threshold, working directory, and disk usage rate for each repeater.

Rules for placing repeaters

To discuss the roles for placing repeaters, consider an organization having requirements to distribute a 1 MB file to 500 clients that are dispersed throughout several subnets. Figure 33 illustrates how this region is set up.

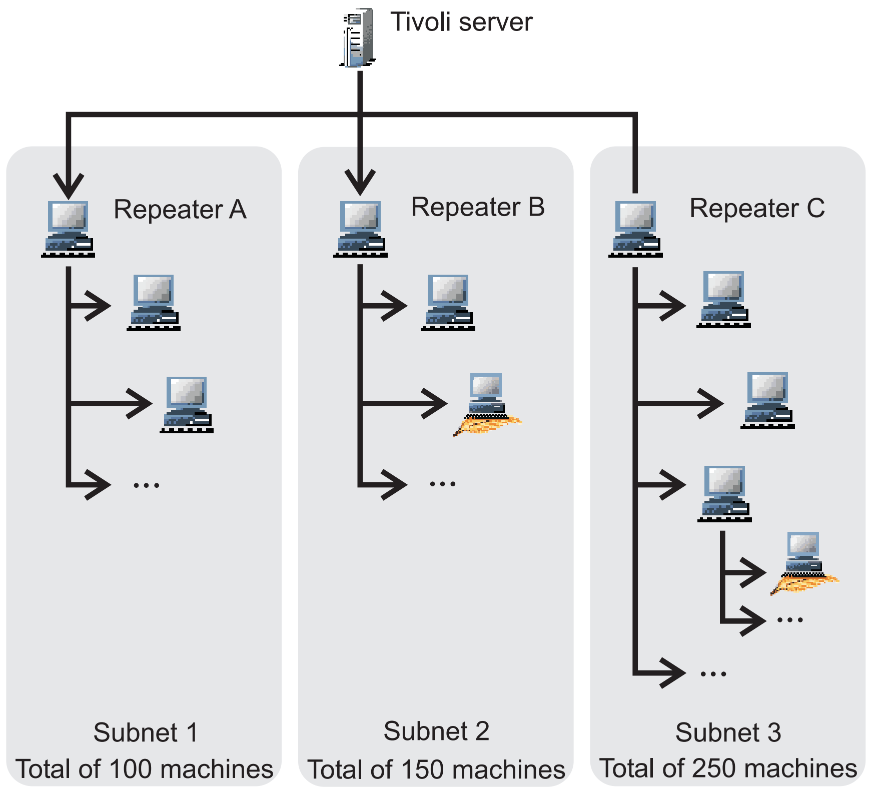


Figure 33. Distribution across several subnets

In this network, you can create a repeater on all managed nodes that serve as distribution points. Using repeater A, repeater B, and repeater C enables the Tivoli server to spread the distribution load across the network and, more importantly, across each subnet.

This approach is a more effective use of the network than sending a copy of the software across the network for each remote target. You must set up repeaters to route a single copy of the software across the network to a remote computer system and use that computer system as a local distribution or fanout point to multiple targets.

In addition to placement of servers for repeater functions in the network, the Tivoli servers need to be configured properly to serve as repeaters. These discussions, although pertaining to configuring a Tivoli server as a repeater, can be extrapolated for other environments as a general guideline for repeater placement. Consider the following cases to when configuring the Tivoli server as a repeater:

- The first case involves three managed nodes acting as repeaters, which have access speeds within their subnet that equal or exceed the interconnection speed with the Tivoli server. In this case, a simple repeater configuration from the Tivoli server to managed nodes with the Tivoli server configured with a positive network load (configured with the **net_load** tuning parameter) works best.

  A positive network load is used when the repeater is repeating to systems that have network access speeds that equal or exceed the speed of the repeater and where the network path speeds from the repeater to the destination also equal or exceed the network access speed of the repeater. This is typical of situations where all the systems reside on a high-speed backbone and the limiting factor is the I/O speed of the repeater.
- The second case involves three managed nodes that act as repeaters, which have access speeds within their subnet that are significantly lower than the interconnection speed with the Tivoli server and the bandwidth into the three subnets is relatively consistent across all subnets. In this case, a simple repeater configuration from the Tivoli server to the managed nodes with the Tivoli server configured with a negative network load works best.

  With a negative network load, the Tivoli server repeats data to each of the managed nodes at the same rate. A negative network load is used when the repeater is repeating to systems that have network access speeds that are significantly lower than the speed of the repeater or where the network path speeds from the repeater to the destination are significantly slower than the network access speed of the repeater. This is typical of the branch office configuration where the repeater resides on a high-speed backbone but repeats through a frame cloud or point-to-point links that provide slower network speeds into the branch offices.
- The third case is the more complex situation, but is the most common. In this case, the Tivoli server must repeat to the three repeaters where some of the repeaters conform to case one and some conform to case two. If you configure the environment for case one, you will likely flood network links to the slower repeaters. If you configure for case two, you will likely have unnecessarily slow distributions to the faster repeaters. For these cases, a designated managed node should be configured as the repeater entry point for distribution. In addition, this managed node should be placed on the same high-speed backbone as the Tivoli server. This managed node can then be configured with a network load appropriate for the link it must communicate across. The Tivoli server can be configured as in case one. This solution is illustrated in Figure 34 on page 128.

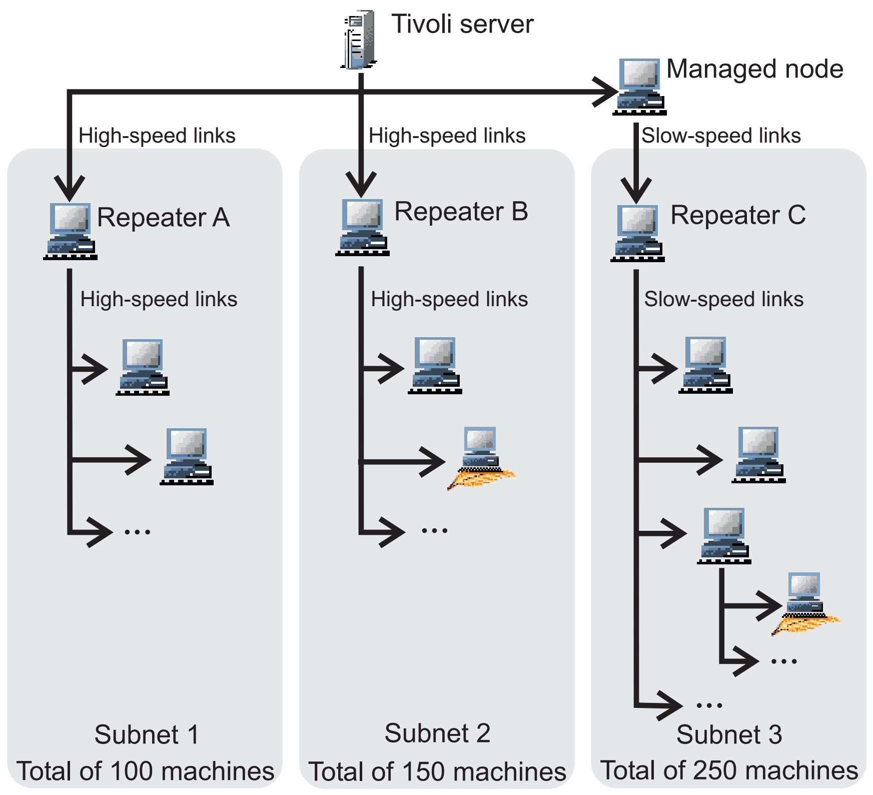


Figure 34. Distribution where network line speed varies

Guidelines for determining the number of repeater depots

If you are using MDist 2 in addition to determining the number and placement of repeaters, you need to determine the number of repeater depots. Every repeater can be configured to be a depot. Thus, when placing repeaters take into consideration where you need depots. Place depots on the other side of slow network links such as WANs to enable quicker distributions.

In addition to depot placement, consider the disk space available for storage of distributions. If you decide to store distributions after they complete, you need to calculate the expected size of distributions (for example, software packages) and select a computer system with adequate disk space.

For example, consider the ABC Instruments. They want to store files they distribute on the depot after the distribution completes. They need to gauge how much disk space the depot must have to hold those files. They plan to distribute Microsoft Office, Lotus Notes, and Norton AntiVirus. The sum of the file sizes is approximately 453 KB. Thus, the depot must have at least 453 KB available to store the distributions. When making these types of calculations, increase the estimates with some additional disk space.

Gateways

Assigning endpoints to gateways is a critical portion of the physical design exercise. Gateways should be located relatively close to their endpoints within the physical network. This minimizes network traffic as commands, status, management applications, and file packages are passed between the gateway and its endpoints.

Gateways are added to the environment in the following situations:
- When scalability limits require a new gateway
- When gateway failover support is required
- When network topologies require a new gateway

Gateway scalability limits

In developing the gateway design, consider the following scalability limits:
- There is a recommended maximum of 200 gateways per region, because in most cases the gateway is located on a managed node.
- There is a recommended maximum of 2,000 endpoints assigned per gateway. The number of endpoints supported by a particular gateway is highly dependent on the following factors:
  - Tivoli applications that are supported by the gateway
  - Gateway operating system
  - Processing power, memory size, and available disk space on the gateway computer system

  The number of endpoints are limited by specific applications, which depend on the following factors:
  - Usage of endpoint login policy
  - Network bandwidth to the Tivoli server

Gateway failover support

The number of available gateways for one endpoint is important in managing very large environments, because the endpoint is highly configurable for login. This means that the endpoint can log in to other available gateways that are defined in the endpoint login interfaces list, if the assigned gateway is unreachable. If you want to define the login interfaces list, you can configure the list when you install the endpoint, or you can change the list later with the **set interfaces** option of the **wep** command.

Regardless of the number of endpoints, each region needs to have a minimum of three management gateways to provide fault tolerance for logins and management functions in case one gateway fails. Three or more gateways are required in each region if your environment needs to protect against multiple gateway failures occurring at the same time.

If there are only two gateways and error failover occurs, the client receives only the address of the single alternate gateway as its list of available gateways. If the alternate gateway should fail later, the endpoint has no other gateway address to use and is isolated. If small remote sites connected by low-bandwidth links are configured to use local gateways only, providing multiple gateways for failover is probably not practical unless it is supporting a vital business process.

However, the use of three gateways per region for failover purposes should not be considered an essential requirement for endpoint support. In cases where the number of endpoints to be managed does not warrant three gateways, or where service levels for repairing gateway hardware fall within acceptable norms for availability of the management environment, three gateways might be considered unnecessary.

In any case, a load-balancing algorithm is required to return endpoints to their original gateway after the failed gateway is back up. Do not keep the endpoints on their failover gateway. To return the endpoints to their original gateway, implement a load-balancing algorithm. This results in the gateway selections lists returning to their state prior to the failover.

After a failed gateway is repaired, the endpoints come back online. During the outage, the endpoints are not managed. But if this falls within acceptable limits, there is no need for a second or third gateway. With two gateways, you can continue to provide service with gateway failover. Be sure to migrate the endpoints back to their original gateway, which results in the system going back to the state it was in prior to the failure. Each endpoint logs in to its primary gateway and has the failover gateway in its list of available gateways.

Network topology requirements for gateways

Because gateways automatically act as a distribution point for endpoints, it is important to take into account the network topology. When information is distributed through the gateway to endpoints, a single data stream is sent to the gateway. From there, multiple streams fan out to multiple endpoints. When you are choosing gateways, consider where in the network it makes sense to have only a single data stream and where you can better use multiple data streams. Install gateways at the end of WAN links.

If you use firewalls, place a gateway on the same side of the firewall as the endpoints it manages.

Note: Do not place a firewall between endpoints and gateways. Endpoints use a random source port for communication.

As with slow links, having a gateway inside the firewall enables you to send data through the firewall one time. From there it can fan out to as many clients as necessary without jeopardizing your network security.

For example, consider the ABC Instruments. The Tivoli environment for ABC Instruments is spread over multiple sites that are connected by 56 kilobit per second (Kbps) lines. The main office has all the IT staff. With the slow line between the sites, ABC Instruments has better performance if they send a single data stream between sites and then fan out the data within each site.

Assume that ABC Instruments wants to upgrade 100 computer systems at its remote sites with Microsoft Office. If ABC Instruments places a gateway (all gateways are repeaters) in their main office and attempts to distribute data to 100 systems located at remote branch offices, a very large data stream is pushed through a very slow line 100 times. This type of distribution could quickly saturate the network or take days to complete. See Figure 35 on page 131.

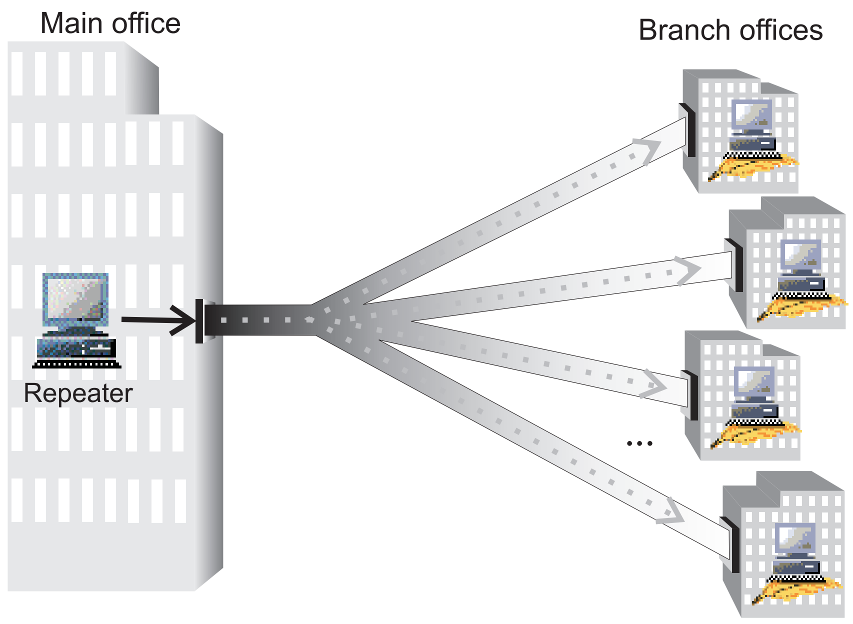


Figure 35. Distribution across a slow line

If, however, the gateway is placed at the remote site close to the endpoints, the data stream crosses the slow line only once and then is pushed to 100 endpoints using the intrasite network, which is much faster. Refer to Figure 36.

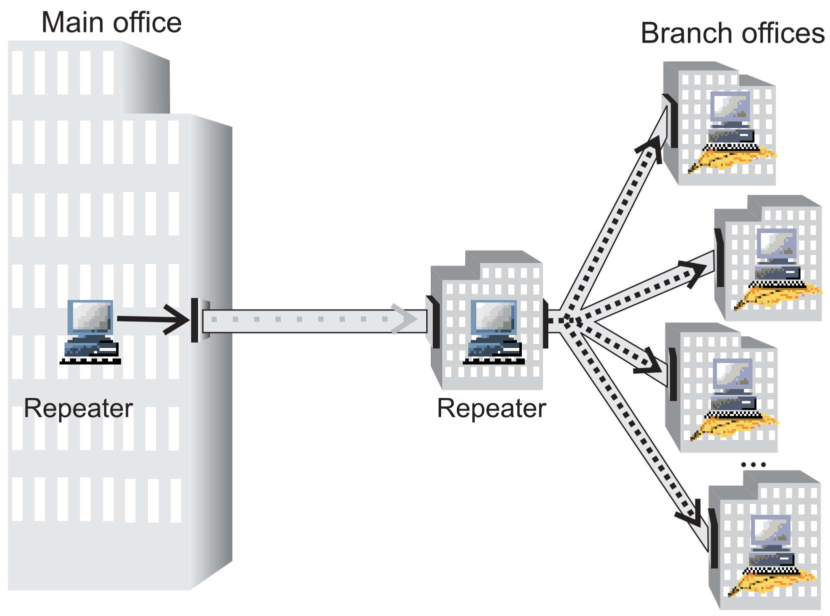


Figure 36. Distribution at the end of a slow line

For fanout purposes, gateways should be added to support the endpoints on the other end of the WAN environment if cost and performance issues justify the expense. This is especially true when there is a large endpoint to gateway ratio. If fanout is not going to be used in your environment, there are not necessarily any advantages in putting the gateway at the other end of the WAN. This could be true in cases where you are not going to perform a software distribution and in cases where there are different types of operating systems across the WAN, each requiring a separate file package for its usage.

The branch office environment provides an especially challenging gateway configuration problem in that there is a small endpoint to gateway ratio. A branch office environment is one where there are very few systems at the end of a slow network link. In this situation, adding gateways to the branch office itself helps with repeating functions when you distribute to multiple endpoints in one distribution. It does complicate maintenance situations where database check operations would have to be performed to a large number of gateways across slow network links.

If the calculations determine that more than one gateway is required per remote site, ensure that the gateways are placed in the correct positions. To avoid the network bandwidth contention, these gateways can be placed on different TCP/IP subnets and service endpoints that do not require the same network links or routers to access their respective gateways.

RIM hosts

RIM can be a relatively CPU intensive, and therefore selection and placement of the RIM host is critical to the performance of the region configuration.

Guidelines for selecting RIM hosts

There can be one or more RIM hosts per application per region. The application documentation contains details about RIM host configuration. Multiple applications can share a single RIM host. These RIM objects can be configured differently and can be configured to run off of two RIM host servers.

RIM database sizes are directly related to the level of detail you want to keep about your managed environment. This is especially true for inventory scans. For a successful Tivoli environment, an experienced database administrator should manage the RIM database.

Rules for placing RIM hosts

For inventory scans, you can run the RIM object on the inventory database server, the Tivoli server, or a separate managed node in the Tivoli region:

- **RIM on the inventory database server—**The RIM object should not be put on the inventory database server. For performance reasons, multiple RIM objects need to point to a single database instance or to multiple database instances running on a single system. Each of these RIM objects must reside in a separate region. Because a managed node can be a member of only one region, and because the inventory server can only be a managed node in only one region, the inventory server itself is not a good candidate computer system on which to run the RIM object.
- **RIM on the Tivoli server—**In cases where the Tivoli server has spare processing capability and the region is small, you can run the RIM object on the Tivoli server. This is a recommended configuration.
- **RIM on a managed node—**You can use a separate managed node running off of the Tivoli server to run the RIM object. This configuration is required in cases where the Tivoli server is a single processor system and does not have spare processing capability.

For IBM Tivoli Enterprise Console, RIM placement rules are simpler. Configure IBM Tivoli Enterprise Console to run with the event repository RIM object on the same computer system where the event server is. For performance reasons, the event repository database should be separate from, and running on different hardware from, the inventory database.

Application servers

In addition to selecting servers for repeaters and gateways, consider the selection and placement of application servers. For details about the selection and placement of application servers, see the application documentation.

Chapter 10. Logical design

The logical design for the implementation of Tivoli Management Framework identifies how the Tivoli Management Framework object database will be configured to meet the operational requirements.

Through the use of policy regions and profile managers, you can decide which Tivoli administrators can perform which functions and on what managed resources.

Although the logical design can be modified more easily after it is in place than the physical design, careful thought should be given in developing the logical design. The design becomes more complex over time and thus harder to modify. A good physical design can be easily invalidated if the logical design is not built to support the physical design.

The logical design process includes the following considerations:
- "Naming conventions for Tivoli objects"
- "Administrator considerations" on page 139
- "Policy region considerations" on page 143
- "Profile manager considerations" on page 151
- "Cross-region boundary considerations" on page 153

The logical design is based on a consolidation of your requirements. As requirements are gathered, you structure them to define management domains or to provide a rationale for separating management applications from each other, managed resources from each other, or applications from resources within the logical design.

The requirements might identify your needs for Tivoli software customization. The answers to these requirements form a foundation for the entire logical design process.

Naming conventions for Tivoli objects

It is extremely important that a naming convention for each element of the logical architecture be defined. It is also important that all personnel who create new managed resources, Tivoli administrators, or management functions within the Tivoli environment understand and adhere to this naming convention. If a naming convention is not followed, the Tivoli administrators will eventually lose track of the internal structure of the Tivoli environment. A benefit of following a well-constructed naming convention is that the environment becomes almost self-documenting, and limited additional documentation is required to understand the relationships of the different parts of the environment. Also, a good naming convention within the Tivoli environment lends itself to automation through scripting.

A new object can be created for each resource type. The Tivoli server guarantees that the object identifier for this resource is unique in the whole region even if other regions are connected (as long as resources are exchanged between connected regions). The name of objects represented by the label, however, can reside several times in the Tivoli environment. Therefore, it is possible to have various resources on the desktop with identical names. For example, it is possible to have a profile manager named Development, as well as a user administration group profile named Development. The type of object in use might or might not be obvious from the context of the desktop, but it might not be clear when working from the command line. For this reason, implement a naming convention that specifies the object type within the name, such as Development-PM, where PM indicates a profile manager.

Unique names make it much simpler to manage the environment. In a multiple region scenario, when an object name is not unique, Tivoli Management Framework appends a string of the format *`#TMR_name#`* to the object. This string forces the object to be unique within the Tivoli name registry. However, this name is not apparent to operators, and the wrong object could be selected for the operation or a naming conflict could occur and the operation would fail to execute. Unique names across the enterprise also enable the combining of regions if required by the organization in future.

At times, you will use Tivoli Management Framework through the command line or have to look at traces or log files. In these cases, it is essential to quickly identify the object type, which can be made easier by following a common naming convention.

The naming convention should cover all resources that might eventually be part of the Tivoli environment. This includes policy regions, profile managers, user profiles, jobs, task libraries, gateways, and so on. The following is an example of a naming convention. It is provided as an example and does not contain guidelines for all the preceding items.

Sample naming conventions for resources

For logical concepts such as policy regions, profile managers, and profiles, multiple levels of delimiters are generally used, separated by a delimiter. The resources that might be an exception to this suggested naming convention are given later in this section.

To more easily distinguish resource names, add some organizational information to a description of the function of the resource. This leads to the naming structure of the following format:

`usage.platform.tmr-name.resource-type`

This is similar to the Domain Name System (DNS) naming convention with an order that goes from most significant data to least significant data (left to right), with periods separating fields. This convention assists in Tivoli desktop navigation by enabling you to see the most significant data in the Tivoli desktop, which left justifies its contents. The following describes the variables in the naming convention in more detail:

usage
: Describes the function or task of the resource and can be one or more letters. The description of the management function or task of the resource should be as short as possible, while maintaining an obvious meaning for every administrator. Examples of its usage are as follows:
  - `nodes` for nodes
  - `lists` for subscriber lists
  - `mon` for monitors

You can qualify this further by adding more letters, such as nodes-bldgA for nodes in Building A. It is recommended that you use a delimiter such as a hyphen (-) or an underscore (_) instead of spaces when defining the naming convention. When spaces are included in names, it becomes necessary to surround parameters in quotation marks when referring to them on the command line.

platform

Specifies the operating system, if necessary, and can be denoted by two letters. If a management resource contains platform-specific information, you can add the code of the operating system as part of the resource name.

This convention helps administrators choose the correct targets for the distribution of profiles. For example, you can define user profile names for several operating systems, where the *usage*, *tmr-name*, and *resource-type* are the same. The following are examples of this convention:

- `usage.aix.tmr-name.resource-type`
- `usage.nt.tmr-name.resource-type`

If the resource is platform independent, consider using one of the following conventions:

- `usage.xx.tmr-name.resource-type` uses `xx` instead of an operating system code
- `usage.tmr-name.resource-type` omits the *platform* variable

Again, you must remain consistent—if no code is used and the resource is platform specific, this could cause problems. Administrators should create a table that contains the codes for all supported platforms. This table can be used to validate resource names or can be added as validation policy in Tivoli Management Framework.

tmr-name

Describes the Tivoli region where the resource belongs. This distinction is necessary when you have a different management policy for each region in your Tivoli environment. For example, the following defines a resource for several countries where the rest of the resource name is same:

- `usage.platform.canada.resource-type` for the resource in the region named for Canada
- `usage.platform.brazil.resource-type` for the resource in the region named for Brazil

In addition, you can use three letters in *tmr-name* and designate each letter to have the first letter code applicable to the geography containing a region (such as a country) and two remaining letters for the specific location or division. For example, `usage.platform.utx.resource-type` indicates that the resource is the region in the United States and is specific to Texas. Another example, `usage.platform.cac.resource-type` indicates that the resource is in the region in Canada and is specific to the accounting division. If a resource applies to all regions or you have a single region, you can still use these letters for further definition or omit *tmr-name* altogether—as long as you remain consistent. Similarly, if the resource has been defined for all regions of a country (in this example Brazil), you can use the code of the country together with the string `xx` (`usage.platform.bxx.resource-type`.

If you are sure that even after future expansion, you are going to have only one region, you can omit the name of the region from the naming convention. The naming convention structure then becomes `usage.platform.resource-type`.

resource-type

Specifies the type of resource. Some suggested two-letter codes for resource types are listed in Table 3.

Table 3. Two-letter codes for naming resources

Resource type	**Code**
Administrator	ad
File package	fp
Group profile	gp
Profile manager	pm
Print profile	pp
Policy region	pr
Query library	ql
Software package	sp
Task library	tl

Expand the table for the resources being used in your environment. For example, `usage.platform.tmr-name.up` defines a user profile. With this convention, the resource type is easily recognized.

A few examples of the `usage.platform.tmr-name.resource-type` sample naming convention follow:

- `nodes.xx.uxx.pr` for naming the nodes policy region in the U.S. region, which applies to multiple divisions in the U.S. region (refer to “Nodes policy region” on page 144).
- `clients.xx.bxx.pr` for the policy region containing all desktop or client subsystems in the Brazil region, which applies to multiple divisions in the Brazil region.
- `clients-bldga.xx.sing.pr` for a policy region for all desktops or client subsystems in Building A in the Singapore region. The subdivision of Building A is done to keep the number of systems per policy region small.
- `nt-mkt-svr-usa.acme.pm` follows the naming convention. This name is for a profile manager specific to Windows operating systems that are servers in the Marketing division located in the U.S. The profile manager is in the Acme region; Acme is the name of the organization.
- `usa-subscriber.acme.pm` also follows the naming convention. This name is for a profile manager in the subscribers policy region pertaining to all the nodes in the U.S. (refer to “Subscribers policy region” on page 148). This profile manager is in the Acme region.
- When you need to indicate the operating system but do not want to use the *platform* variable universally, you can prefix the operating system in the *usage* variable. For example, `nodes.usa.pr` indicates a nodes policy region in the U.S. region, and `nt-ny.usa.pm` indicates a profile manager specific to Windows NT systems in New York in the U.S. region.

This naming convention given above is just an example. You can add additional information to the resource names, for example, the name of the application that is managed with the Tivoli object.

Resource types that do not require naming conventions

The following are several resource types for which naming conventions of Tivoli resources are not recommended:

- Tivoli region name

  Every Tivoli server can either be accessed with the region number that is automatically assigned during the installation or with the Tivoli region name. You can think of this name as an alias for the server. The region name should be the same as the *tmr-name* variable in the `usage.platform.tmr-name.resource-type` format as described earlier.

  This approach for standardizing the Tivoli region name has an advantage when creating local resources. The Tivoli command **wtmrname** returns the name of the region and can be used in a shell script when assigning a resource name to a new Tivoli object by using **wtmrname**.*resource-type*. Tivoli region names must be unique.
- Administrator

  It is more convenient to use the full user name for the resource type administrator. One exception might be when multiple administrators use the same Tivoli administrator ID. For example, **adduser.xx.gmu.ad** indicates all administrators charged with adding users who use the same administrator name.

  Note: Shared administrator IDs are discouraged in Tivoli Management Framework for accountability reasons.
- Generic collection

  A generic collection is normally used by administrators to group resources on their own desktop. That name is relevant only for the owner of the generic collection, the administrator.
- Managed nodes and endpoints

  Managed node and endpoint resource names are frequently used in daily operations. Use a short and recognizable name. Assign the Transmission Control Protocol/Internet Protocol (TCP/IP) host name or NetBIOS computer name without the domain name suffix to the Tivoli resource name for these resources.

  When naming an endpoint installed on a managed node host (the recommended way to manage that host), suffix the host name with -mn or -ep. This naming scheme provides the required unique names for the endpoint and managed node.

Administrator considerations

Evaluate your administrative requirements and build a plan for assigning administrative roles.

You can use a combination of Tivoli region roles and resource roles for restricting the scope of authority of the administrator. For example, administrators can be granted **user** authority across the Tivoli region (this allows them to view resources but not perform management operations). In addition, administrators can be granted the **admin** or **senior** roles on specific policy regions. This allows them to initiate management actions for those specific policy regions.

For example, you might have one or more administrators who are responsible for creating application-specific profiles to be used to manage your environment. These administrators must be given the **senior** role in the appropriate policy region. Administrators responsible for distributing these profiles require only **admin** authority, as you might not want them to have the permissions to modify these profiles.

Continuing with the example, you might have other administrators who are responsible for installing and creating managed systems in the Tivoli region and others who are responsible for maintaining standard subscriber lists. How your scopes of authority are defined within your organization are a major factor in determining the policy region hierarchy and the roles that individual administrators are given within those policy regions.

Be careful when granting Tivoli region roles to an administrator because they automatically get applied to all Tivoli region resources, both existing resources and resources created in the future. In general, do not grant the **admin** and **senior** roles as Tivoli region roles and grant only the **user** role to an administrator. For example, administrators with the **admin** and **senior** roles can modify resources created by an administrator with the **super** role.

Grant the **senior** and **admin** roles (and other application-specific roles) to the administrators at the policy region level only. This ensures that as your environment grows and new applications and management functions are added, control is maintained over administrators that have management control over the various resources.

The following examples illustrate the Tivoli role and resource role further:

- Software distribution profiles often modify operating system resources. The before, after, remove, commit (BARC) configuration program must run with root or Administrator privileges when file packages are distributed, committed, or removed. With the **admin** role, administrators can create a file package with configuration programs containing any code they want. They can then distribute the file package to any system in the Tivoli region and the programs are executed with the highest system privilege.
- Custom monitors must run with root or Administrator privileges. With the **admin** and **senior** role authorization, administrators can create a custom monitor in a monitoring profile that can contain any code that they choose. The monitor can then be distributed to any system in the Tivoli region and is executed with the highest system privilege by the distributed monitoring engine. Monitoring response commands also run as root or Administrator because they often modify operating system resources to correct system problems. Response commands can also be added or changed by an administrator with **admin** authorization to circumvent host security.
- The custom scanning method attribute of inventory profiles can be used to circumvent host security, and only the **admin** role is required to do this.
- An administrator with the **admin** role can create task library tasks that allow host security to be circumvented if the task library policies permit tasks to run as root or Administrator, unless appropriate security precautions are taken.

Critical Considerations:

The following are critical considerations when defining administrators for the Tivoli environment:

- How are the administrators and operators grouped?

- Do certain administrators need to control all aspects of some resources and not others? For example, the workstation support team should have no access to servers.
- Do certain administrators need to access specific applications and not others? For example, people who distribute software should not be able to change user IDs and passwords.
- Do specific aspects of application usage need to be isolated? For example, isolate software file package creation, subscriber creation, and monitor creation from file package distribution.
- Do specific resources need to be isolated from all other resources? For example, isolate payroll servers from print servers.
- What are the remote control security requirements? How do you need to restrict access for operators of remote control and Tivoli administrators?
- Are there multiple concerns for how things are accessed (by servers, and then by company)?
- What special-consideration systems are there? For example, key business systems such as SAP servers.
- Is there a multitiered Tivoli region structure, and where do people interact with the design? For example, all administrators are defined at the top region or enterprise level, or some administrators are working at a regional level.
- Is there an obvious split between groups of systems and the applications that are used to manage them? This split can affect Tivoli region, policy region, profile manager, and administrator definitions, as well as gateway and repeater placement, capacity planning, and placement of Tivoli application servers.

Important Considerations:

The following are important considerations when defining administrators for the Tivoli environment:
- What is your priority in protecting resources? Identify the most important resource to be protected.
- What kind of security breaches have happened in your environment in the past?
- What is the physical security available for your computer systems and network?
- Is your organization prepared to make organizational changes to implement effective security?
- Who are the owners for every piece of software and hardware in your environment? These owners must have the authority to set rules and deal with violations for the specific resource.

Location of administrators

Whether you use centralized or distributed administration plays a major role in the overall design of the Tivoli environment. Which of the two approaches to use depends on the systems management strategy and the physical environment to be managed. You can use one approach or a combination of the two.

The distinction between the two approaches is simple—centralized administrative designs use an administrative team located at a single site (either physical or logical) who is responsible for managing the entire enterprise. Distributed designs position administrative authority near the sites being managed and develop hierarchies of administrators where duties are delegated hierarchically.

Centralized administration

In centralized administrative design, administrators who are physically placed in a central location manage all system resources, even if those resources are spread across multiple physical sites. Individual sites maintain few or no administrative responsibilities. This gives one administrator or a group of administrators responsibility for the entire management environment.

Administering everything from a single region can impose restrictions on the number of endpoints that can be handled. Single Tivoli region implementations can be made much larger through the use of endpoints. This centralized physical configuration is likely to be the simplest to implement.

Distributed administration

In distributed administrative design, administrators are spread throughout the organization and maintain authority over the operational sites at which they reside. In this arrangement, authority is usually organized and dispensed hierarchically.

Examples of administration designs

A centralized administration design is shown in Figure 37. In this example, all administrators are associated with TMR Central. Their authorization roles (**super**, **senior**, and **admin**) apply throughout the Tivoli environment, even across Tivoli regions in different geographic locations.

Note: If the administrator of TMR Central needs to modify any profile after it has been distributed to the other Tivoli regions, the TMR Central administrator needs the relevant roles at the other sites as well.

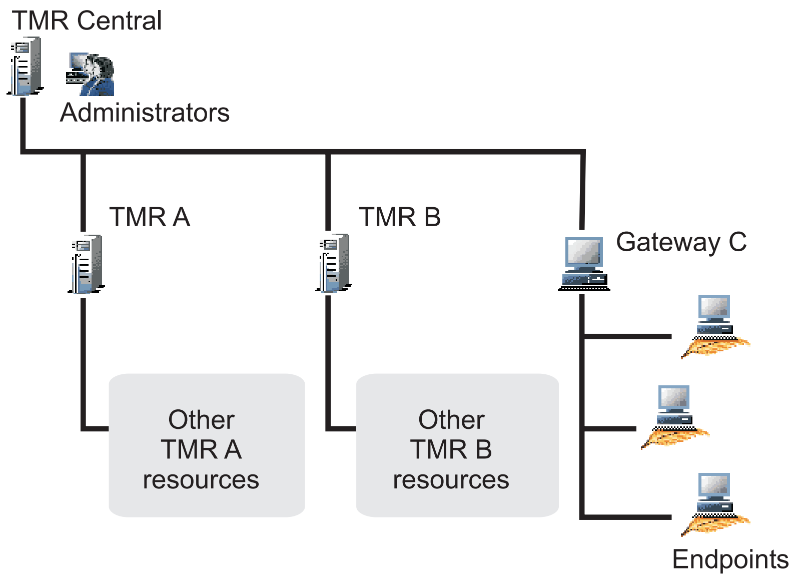


Figure 37. Centralized administration

An example of a hierarchical distribution of administrative authority appears in Figure 38 on page 143. In this example, the same physical configuration is deployed as in the previous example. The only difference is that additional administrators reside at or have administrative roles at region A and region B or both. This gives TMR A and TMR B a measure of autonomy that is not possible with centralized administrative models. Administrative roles assigned at TMR Central apply to all the resources of gateway C.

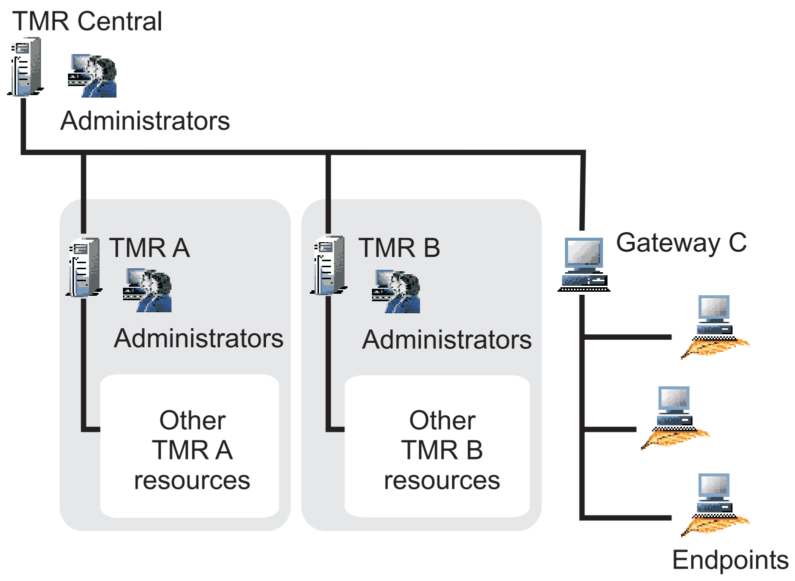


Figure 38. Distributed administration

Policy region considerations

Use policy regions to organize objects representing managed resources or management application resources into logical groups. You can also use policy regions to apply policy and restrict administrative access to these entities. The number and topological design of policy regions significantly affect their functionality and ease of use. By carefully organizing resources into appropriate regions, you can greatly simplify management operations and maintenance.

In general, you can make policy region hierarchy as simple or complex as you require. For example, it is possible to create all management objects within a single large policy region. This, however, becomes difficult to manage and provides little granularity regarding administrative roles and authorities.

Administrator authority to create, change, or delete objects and to perform management actions can be applied by policy region. Therefore, a well-organized set of policy regions enables you to ensure that administrators can perform operations only on the resources for which they are responsible and that the operations they do perform comply with policies set by management.

In the following sections, one methodology to organize resources across policy regions is described. This is just one example. The choice of how to implement a policy region hierarchy is up to you and can vary widely based on your organizational and business requirements.

Overview of a policy region hierarchy

With Tivoli Management Framework, a hierarchy of encapsulated policy regions can be built. The policy regions and their hierarchy are primarily created for one of the following reasons, ordered here according to their priority:

- **Control user access—**Every managed object in the Tivoli environment must be created in a policy region to ensure access control over that object.
- **Enforce unique policy—**Managed resource policy propagates down the hierarchy of the policy regions. Limiting the number of managed resource types in a policy region helps to ensure that the security of the environment is maintained.
- **Reasonably sized containers—**Having fewer than 2,000 objects in any policy region helps to keep your environment manageable. The more objects in a policy region, the more difficult or time consuming it is for you to find the object when you need to perform some action using the desktop.

The policy region structure should be flexible, open for expansion, and meet different security requirements. The following policy region schema can fit differing environmental situations. Review this schema and modify it, if necessary, to meet your requirements.

The schema has three top-level policy regions discussed in the next three sections: nodes, subscribers, and application. Each of these policy regions (and policy subregions) serves a unique purpose. This gives clarity of structure and complete predictability within the Tivoli environment. This predictability is required to ensure that the logical design adheres to your business security model.

nodes policy region
Provides an organized set of containers for the computer systems within your environment.

application policy region
Contains the profile managers that in turn contain the Tivoli application profiles. Administrators with the authority to invoke management operations use these profiles in conjunction with the subscriber lists to perform operations on the appropriate systems.

subscribers policy region
Provides a way to organize the managed resources into groups that normally have similar management requirements. In an analogy to e-mail, think of the subscriber section as a way of managing distribution lists. Policy regions help ensure that only authorized administrators can modify the various lists and use the lists for management operations.

Each policy region serves as the entry point for the top-level Tivoli administrator and acts as a container for all other policy regions in the system. How granular the design of the policy region hierarchy becomes depends on many factors, including the size of your environment, the number and authority of administrators, and so on. The rights to view or perform actions on specific resources can be specified at the policy region level.

Nodes policy region

The nodes policy region is a container for the objects representing managed systems within your environment. This includes managed nodes and endpoints. Managed nodes must be associated with some policy region when they are created. Though endpoints can be created without associating them with a policy region, it is likely that you will want or need to associate them with a policy region. For example, remote control software requires that an endpoint be associated with a policy region to be able to open a remote control session with that system.

Again, though you can place all managed systems within a single policy region, this is not recommended. Rather, the nodes policy region should represent the top of a hierarchy of subregions that provides a means for organizing all the managed systems in a way that makes sense for your organization. The nodes policy region hierarchy is generally driven by the following security considerations:

- What are the remote control security requirements?
- How are administrators and operators grouped?

Using remote control software, an operator can control specific desktops in the environment. You might require security checking on the ability of an operator to take the control of this desktop. This can be controlled by granting the administrator the **admin** role for the policy region that contains this desktop or the endpoint. You can logically group the endpoints in subregions according to the remote control takeover policy. For example, there can be administrators who can either control client desktops or control server desktops, or they control all the desktops.

The prerequisite design for nodes policy region hierarchy is to group the administrators by their type and function. The answers to the questions in "Administrator considerations" on page 139 will assist you in the task of grouping the administrators, and as a result, in deciding the nodes policy region hierarchy. With this information, you can minimize the number of places where you need to give specific administrators **admin** access to managed systems.

For example, if you have grouped administrators by department in your organization and have subgrouped them by management of servers (indicated by "svr") and user desktops (indicated by "dsk"), you can have the nodes policy region hierarchy as shown in Figure 39.

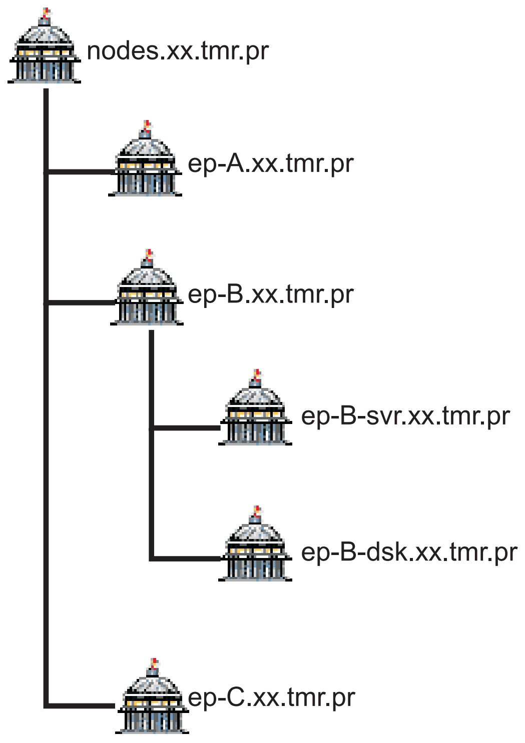


Figure 39. Nodes policy region hierarchy for servers and user desktops

When establishing policy regions, remember that policy regions, and the hierarchies in which they are arranged, also define the default scope and lineage of administrative authority. Each level of the hierarchy should be defined by a consistent criterion. In Figure 39 on page 145, the administrator for the ep-B.xx.tmr.pr policy region has access to both the servers and user desktops policy regions by default, because administrator access propagates down the hierarchy. Also the policy set at this level would automatically propagate down to the subregions.

In Figure 40, groupings in the first level represent continents or cultures. In the grouping beneath North America, the divisions are all by city. In this example, New York is represented by NY, and Los Angeles is represented by LA. In other groupings, the divisions might be made by country.

Most nodes policy region hierarchies also contain at least one level that is organized by hardware type, or operating system, or both. This organization can help an administrator to navigate to a particular system quickly and easily when trying to resolve a problem with that system.

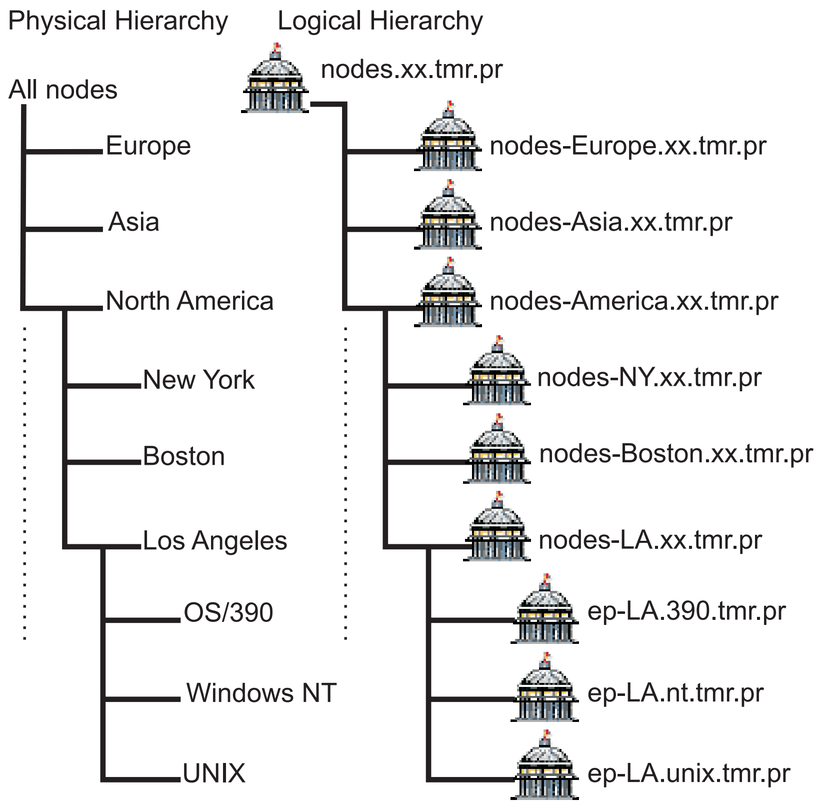


Figure 40. Nodes policy region hierarchy by location and with operating systems

Most management operations are performed through profile distributions or other application-specific actions. An appropriate node hierarchy is a good way to organize the various managed systems within your environment, but you use subscribers and application hierarchies more often in day-to-day operations. Subscribers and application hierarchies are described in more detail in “Application policy region” on page 147.

Organize the nodes policy region to maintain security of the environment by controlling user access and to find existing computer systems when needed. For performance reasons, policy region hierarchies that contain more objects and fewer levels are better than hierarchies that contain many levels and few objects per level.

Application policy region

The application policy region contains management application resources such as profiles. This policy region is sometimes called the “services” policy region. A policy region hierarchy is used to organize and control access to management operations. Administrators with the authority to perform management operations, such as the distribution of profiles or the execution of tasks and jobs, primarily use this policy region hierarchy. The typical contents of the individual regions within this hierarchy are profile managers and task libraries.

You can have unique policy regions for each application or discipline. For example, you might have one or more administrators who are responsible for software distribution. These administrators might not be authorized to perform other functions, such as monitoring or user administration. Likewise, you might want to limit other administrators from performing software distributions. In this case, you can create a policy region specifically for software distribution. This policy region would contain the profiles for software distribution. Other resource types (such as distributed monitoring profiles) would be restricted from existing within this policy region. Only administrators with software distribution authority would be given access to this policy region.

The hierarchy you define should be based on the administrative requirements mentioned in “Administrator considerations” on page 139. You might have separate regions for each and every management application, or you might decide to group applications together. For example, you can have a deployment region that includes both software distribution profiles and inventory profiles.

Consider access control and policy together when deciding the application policy region hierarchy to help secure the logical design. The major considerations while deciding the hierarchy due to access control are the following:

- What is your major division for access control and what are your secondary divisions for access control?
  - Are the administrators grouped by hardware, software, and applications?
  - Are they grouped by operating system?
- Are there any special considerations?
  - Business sensitive applications such as SAP or PeopleSoft
  - Key servers or administration requirements imposed by a data center organization

Figure 41 on page 148 shows an application structure separated into different policy regions per managing application. To have fewer levels of policy regions, it is recommended to make these policy regions top-level regions rather than making them subregions under the application policy region. The software distribution (sd), inventory (inv), and distributed monitoring (dm) policy regions are divided further according to the different user applications, such as Lotus Notes and SAP. The third level of subregion could be by operating system, divided by Windows NT and AIX platforms. Notice that the application policy region contains subregions and the subregion contains a profile manager. Such a hierarchy would provide flexibility and control in managing dynamic environments.

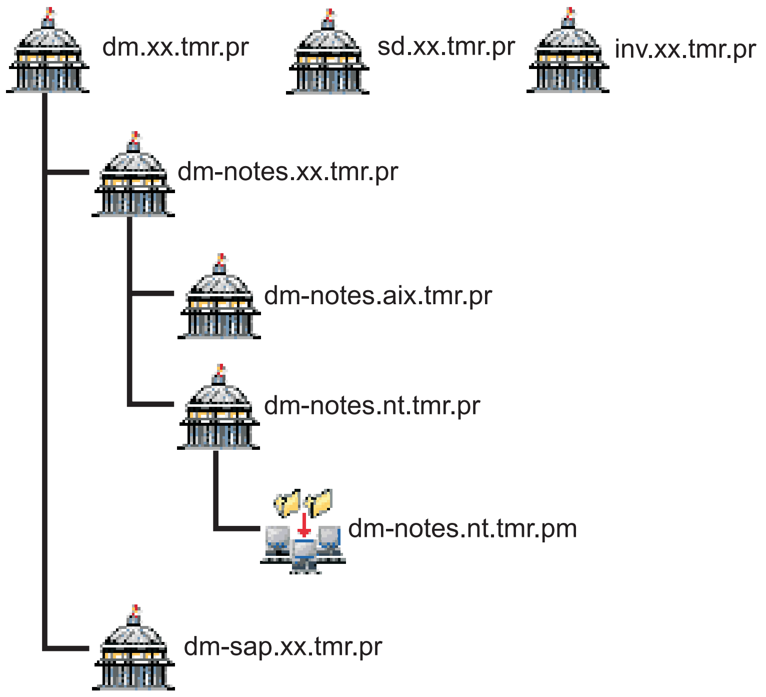


Figure 41. Application policy region hierarchy

As the logical design is defined, be aware that some applications, in particular the management modules, might create their own policy regions upon installation. These policy regions and their corresponding functions need to be taken into account in the overall architecture.

The guiding principle is to organize the application policy region so that there are clearly implied areas for adding new services and so that different variants of the same service are easy to locate and use.

Subscribers policy region

The subscribers policy region is an entry point for a hierarchy of policy regions that is used to organize managed systems into homogeneous groups. That is, these groups contain systems that likely require the same management operations to be performed on them.

Management operations are typically performed through the distribution of profiles or the execution of tasks. Profile managers are used to link profiles to a set of subscribers. A predefined set of subscribers can help the administrator perform an action on a set of systems. For example, a file distribution or inventory scan operation might need to be performed on all Lotus Notes servers, on all accounting systems in Tokyo, or on all systems running Windows NT.

To be able to manage groups of systems such as these, you generally define subscriber-only profile managers. That is, profile managers that contain no profiles themselves, but only a set of subscribers that are related in some way.

Creating such profile managers is similar to defining distribution lists in an e-mail system. Likewise, the maintenance of these lists (adding or deleting subscribers) can be performed by an administrator that is not the same one that initiates the action.

To perform a management operation on a set of systems, the subscriber profile manager can simply be subscribed to another profile manager that contains an application profile. When the application profile is distributed to the intermediate profile manager, the profile is in turn distributed to its subscribers. You could define arbitrarily large distribution hierarchies through this mechanism, though too complex of a hierarchy can be counterproductive.

By placing these subscriber-only profile managers in their own policy region hierarchy, you can define administrative roles by policy region to control which administrators are authorized to maintain these subscriber lists. See Figure 42.

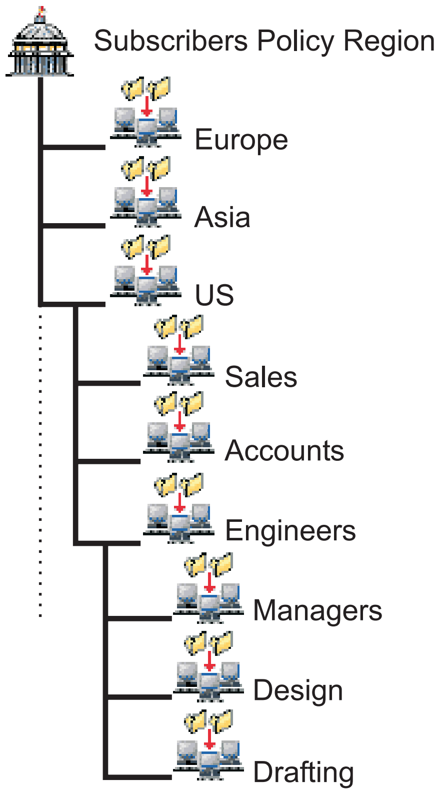


Figure 42. Subscribers policy region hierarchy

A comprehensive hierarchy of profile managers should be created to allow subscribers to be addressed anonymously by their membership in some generalized category of entity. Anonymity is essential to avoid the hard-coding of individual computer system identities into the distribution architecture.

In the subscriber region, another possible organization of profile managers could be as illustrated in Figure 43 on page 150.

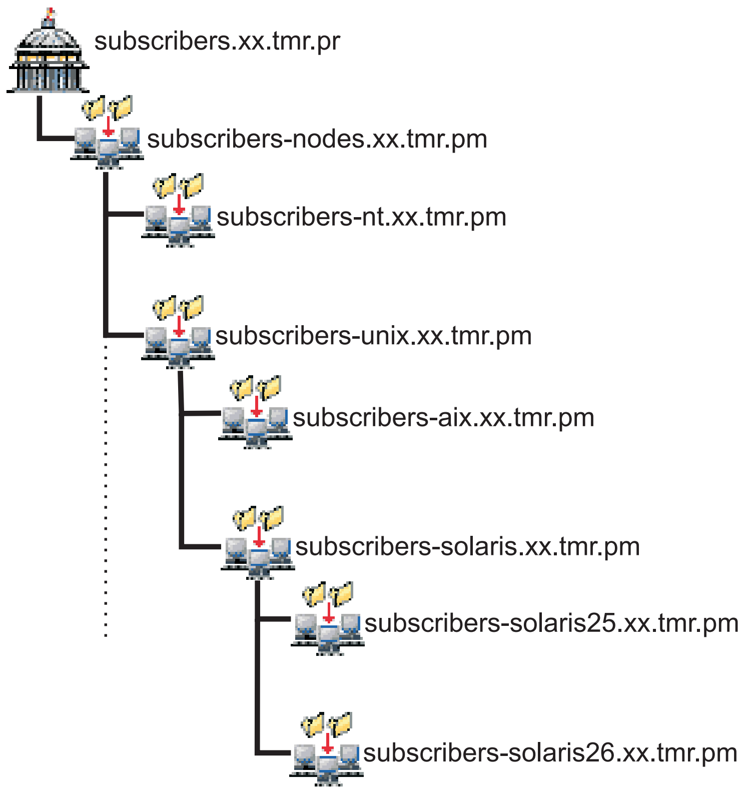

subscribers.xx.tmr.pr
subscribers-nodes.xx.tmr.pm
subscribers-nt.xx.tmr.pm
subscribers-unix.xx.tmr.pm
subscribers-aix.xx.tmr.pm
subscribers-solaris.xx.tmr.pm
subscribers-solaris25.xx.tmr.pm
subscribers-solaris26.xx.tmr.pm

Figure 43. Subscribers policy region hierarchy with operating systems

In Figure 43, these profile managers eventually resolve to AIX, Solaris, and Windows NT managed node subscribers. That is, whenever a new node is created in the relevant node subregion, it must be recorded as a member of the correct subscriber lists.

Now, application profile managers can reference the new node without further administrative intervention.

In the subscribers policy region, a system can be subscribed to more than one profile manager. For example, a system in use by an accountant in Austin is a part of the accounting and Austin profile managers. In general, you should include only profile managers in a subscriber region.

Policy region summary

In the preceding policy region structure, each computer system in the enterprise must be represented as a managed resource in the nodes policy region and as subscribers to profile managers in the subscribers policy region. Also, each Tivoli application profile type should be added as a managed resource to the appropriate policy regions within the application policy region hierarchy.

When you add new computer systems, you add them to the appropriate nodes policy region and to each subscription list depending on the scope of authority assigned to the administrator. Because these subscriber lists are used by the application policy region, no further action is required to enable the distribution of services to the new computer system.

1. Computer systems subscribe to the profile manager.
2. The profile manager in the subscribers policy region subscribes to the profile manager in the application policy region.
3. Application services are provided to the correct computer systems.

Figure 44 illustrates how it works.

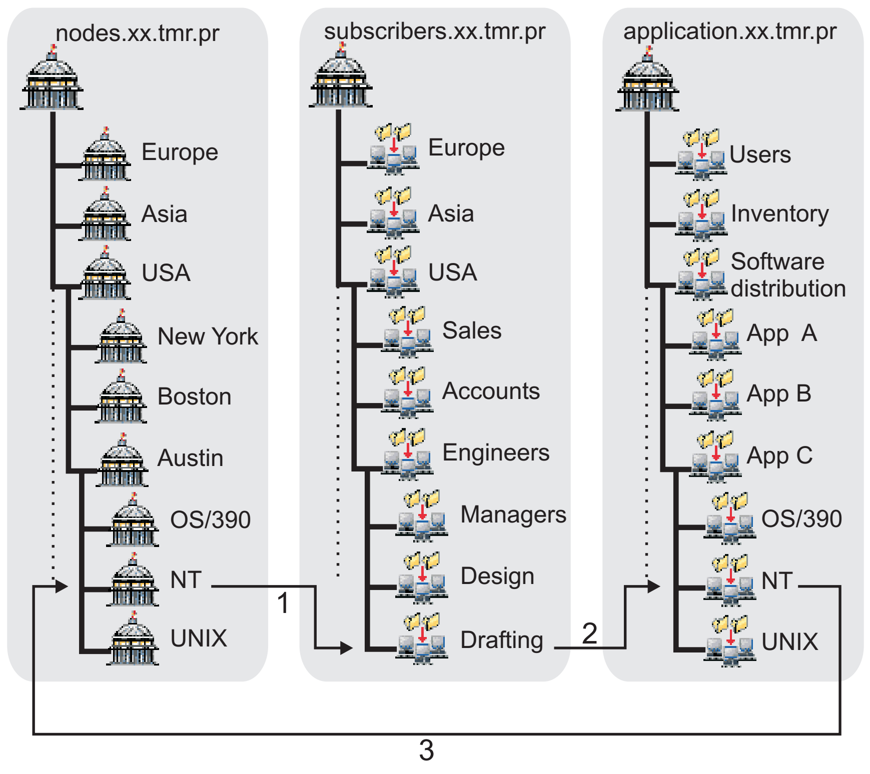


Figure 44. Summary of application, subscribers, and nodes policy regions

The three policy region structure is extremely flexible. Any group, from any position in the network, can be added to subscription lists at any position in the subscription hierarchy. Subscription lists are readily subscribed to any service available. By carefully arranging the three hierarchies, you can exert simple but powerful control over the scope and depth of every management service in the system.

Profile manager considerations

As with most of the Tivoli architecture, you have a great deal of freedom in designing your own profile manager schema based on your environment. Certain design patterns, however, apply to many environments and have been successfully implemented by many customers. Though you could place all profile managers in a single policy region, you might want to control administrative authority over profile managers and subscription lists by placing them in appropriate policy regions as described in "Subscribers policy region" on page 148.

Profile managers should be built at the proper locations within the hierarchy of policy regions to support the functions for the administrators at those levels. For example, you can create a profile manager for endpoints in a policy region that represents a geographical state for use by a senior administrator. Then you would subscribe all the profile managers that represent endpoints in a specific city within that state. These lower-level profile managers would be created in the city-based policy regions and would be used by junior administrators who would have a more limited view of the enterprise. Profile managers can be configured to group endpoints by operating system types, job functions, or possibly by the applications they run. The groupings are not to be considered fixed and should be expected to change over time as new computer system groupings are found to be necessary.

Depending on how you have used policy regions to group resources, the appropriate profile managers need to be developed to match resources with the appropriate profiles and tasks. In addition, Chapter 4, “Profiles and profile managers,” on page 35 describes examples of how to design the profile manager hierarchy in detail. Some guidelines for developing a profile manager schema are the following:

- Limit the number of direct subscribers in any profile manager. The icons representing the subscribed resources are stored as an attribute in the profile manager object itself. Every time the profile manager is opened, the status of these resources is checked. This status checking operation, especially across Tivoli region boundaries, can take a lot of time. So with more endpoints subscribed directly to a dataless profile manager, the time to open the profile manager on a Tivoli desktop increases.

  The subscriber dataless profile manager might have 2,000 endpoints in it, but the profile manager where the direct subscription takes place might have only one, or only a few, icons representing the subscriber profile managers.
- Avoid direct profile endpoint subscriptions. This minimizes the required maintenance when a resource is deleted or its function is changed. Endpoints should be placed under subscriber profile managers. These subscriber profile managers are then subscribed to all appropriate profiles for the group. Thus, by changing the subscription profile manager that an endpoint is in, you can easily unsubscribe it from one set of profiles and subscribe it to another set. You also do not need to keep track of all the places the endpoint might have been subscribed.
- Do not subscribe managed nodes to dataless profile managers. This is supported, but can lead to inconsistencies that can result in incorrect system configuration. Dataless profile managers should have only endpoints as subscribers. Only database profile managers should be used to contain the original instance of profiles. This allows the profile to be distributed to profile managers, managed nodes, and endpoints.
- Create profile managers in separate policy regions from the managed resources to which they are distributed. This allows the administration for the profiles to be defined separately from the administration for the managed resources. Such an approach is described in the “Policy region considerations” on page 143.
- Avoid subscription of managed nodes and endpoints across Tivoli regions. Region authentication operations need to be carried out across region boundaries. The excessive authentication reduces the performance of the system and tends to bypass the multiple region structure that helps with these types of performance issues. Also, certain applications might not be able to resolve administrator and endpoint policy across a Tivoli region boundary. For further details, refer to “Resource exchanges and updates” on page 158.

Cross-region boundary considerations

When you have multiple Tivoli regions, you must consider additional issues for policy region hierarchies and profile manager structure. In particular, consider how the nodes, application, and subscribers policy region as described in “Policy region considerations” on page 143. To illustrate the policy region and profile manager schema in a multiple region environment, an organization with a hub and spoke architecture is used as an example (refer to Figure 45).

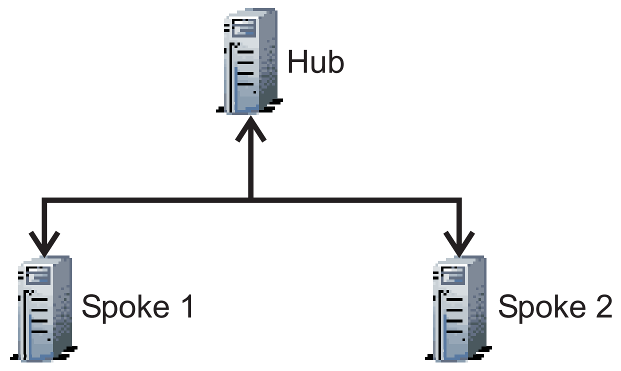


Figure 45. Hub and spoke region connection

Considerations for policy regions and profile managers

A nodes policy region is created in each of the three Tivoli regions, as illustrated in Figure 46 on page 154, Figure 47 on page 154, and Figure 48 on page 155. The hierarchy of the policy regions within each region could be different from each other depending on your requirements.

While creating the managed resources, be sure that they are created in a policy region inside of the Tivoli region closest to the managed resource (where closest is in terms of the network topology). If this is not done properly, over time resources are distributed randomly throughout the Tivoli environment, hampering the use of repeaters and fanout servers.

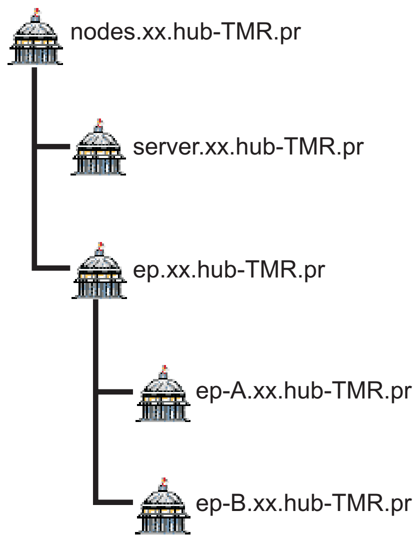


Figure 46. Nodes policy region for hub Tivoli region

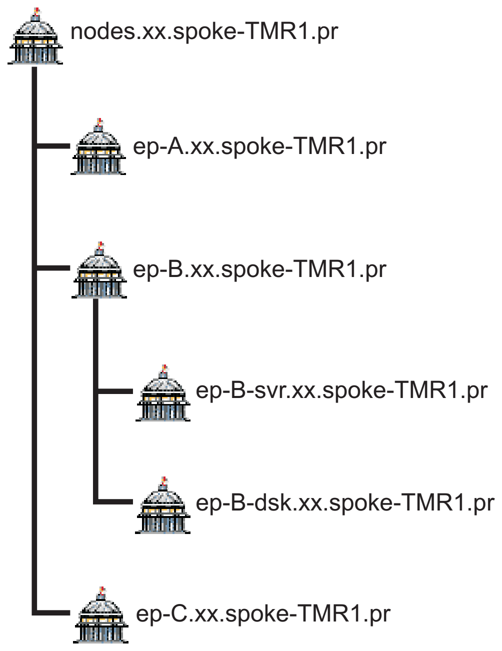


Figure 47. Node policy region for spoke 1 Tivoli region

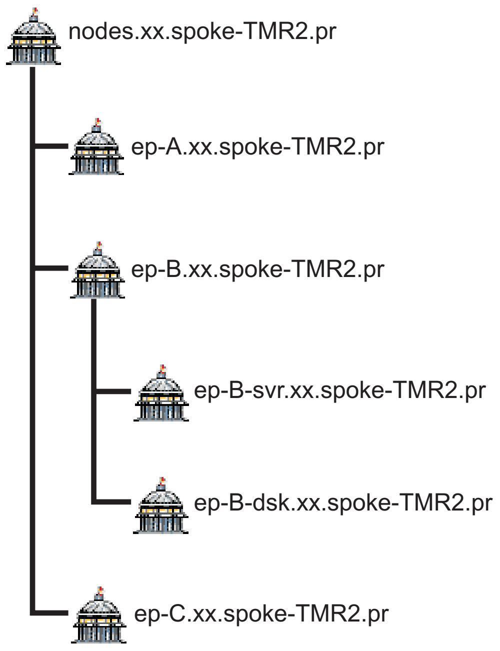


Figure 48. Node policy region for spoke 2 Tivoli region

For application policy region schema, it is recommended that this policy region should exist only in the hub Tivoli region. This way all the operations can be performed from the hub Tivoli region, and no operations need to be performed from the spoke Tivoli regions.

Create the subscribers policy region hierarchy individually in each of the Tivoli regions. At the hub Tivoli region level, you can consolidate the subscriber lists of the entire organization. The subscriber view and the complete consolidated view in simplistic form are illustrated in Figure 49 on page 156 and Figure 50 on page 157. In Figure 49 on page 156, the managed resources defined in the nodes policy region are subscribed to the profile manager in the subscriber policy region.

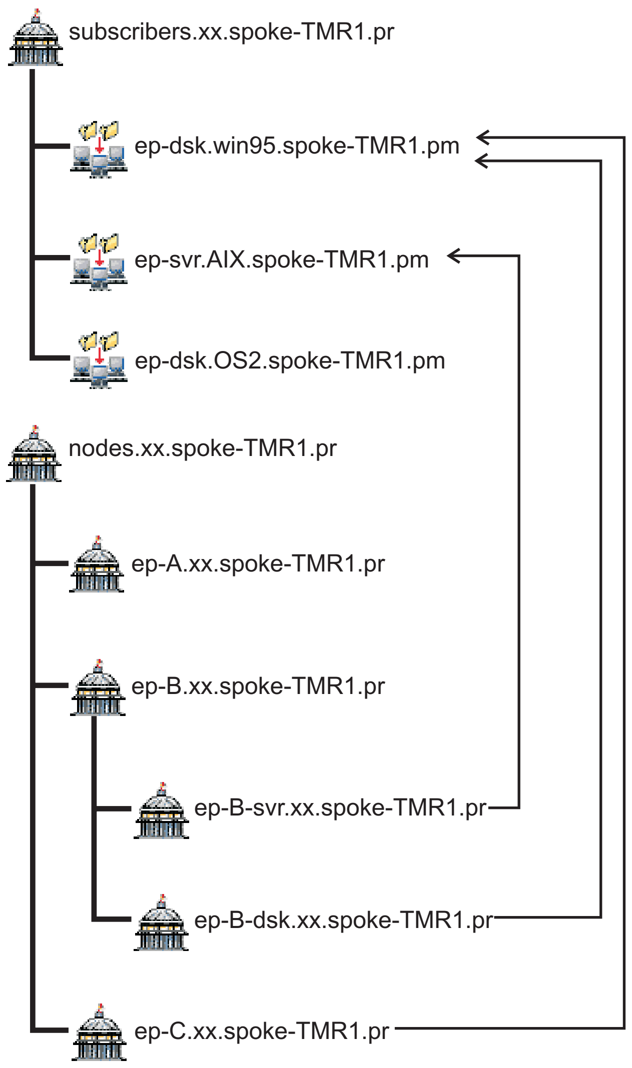


Figure 49. Subscribers policy region in hub and spoke Tivoli region

In Figure 50 on page 157, the subscriber profile managers in the two spoke Tivoli regions are subscribed to the profile manager contained in the software distribution policy region hierarchy.

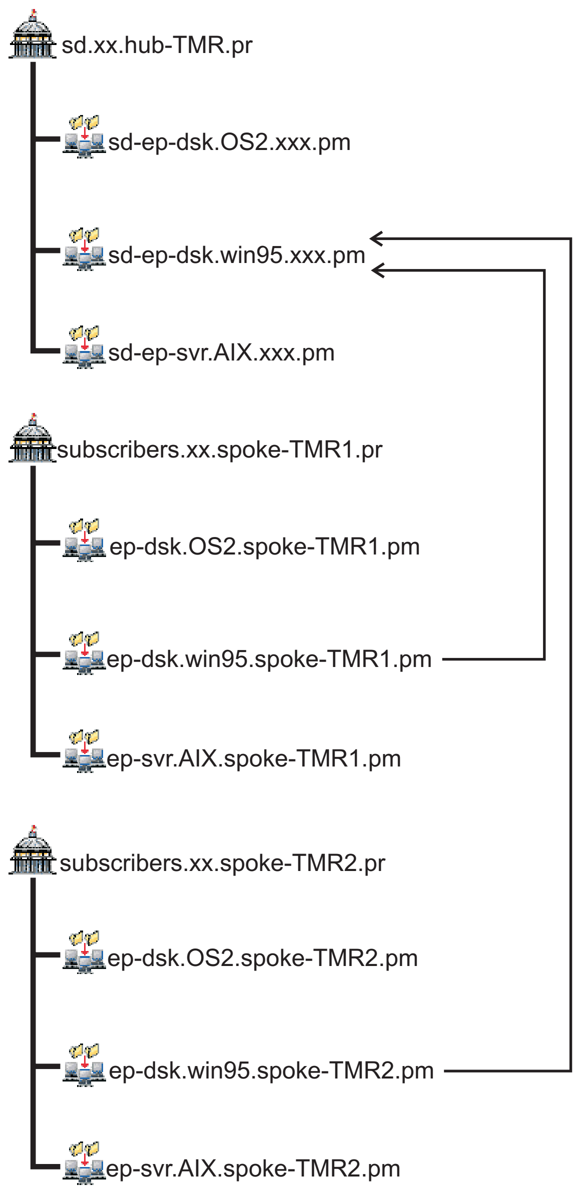


Figure 50. Subscribers profile managers in hub and spoke Tivoli region

Managed nodes and endpoints should not be subscribed across Tivoli region boundaries to profile managers in the hub region. Subscriptions should be made indirectly through profile managers in the spoke regions being subscribed across the region boundaries into the hub regions. Any logical design should attempt to minimize cross-profile manager subscriptions as well. Subscribing endpoints through profile managers forces the management paradigm to focus on profile managers as the managed objects versus endpoints as the managed objects. Cross-profile manager subscriptions impact scalability and performance.

When the profile is pushed, it is important that it is pushed through the profile manager hierarchy. If the push is accomplished by selecting the profile and endpoint and pushing directly, the results are as if the endpoint was subscribed across the region boundaries. The following reasons illustrate why pushing to an endpoint directly is not recommended:

- With applications that perform distributed monitoring and user administration, when endpoints are subscribed across region boundaries, a profile push goes directly from the hub region to the endpoint without passing any of the processing loads to the spoke region. This increases the processing loads on the hub server and increases distribution times.
- When the environment is cleaned up with a **wchkdb** command, these types of cross-Tivoli region subscriptions require the Tivoli server to check a much larger number of objects for consistency than would otherwise be required.
- Database growth on the hub region can grow excessively when these cross subscriptions are made because of the number of profile copies that are kept in the hub region.

Neither push—the direct or the indirect push—would be successful if the profile manager and endpoint resources were not exchanged across the region boundary from the spoke region to the hub. The exchange of the resources is required for the lookup function in Tivoli Management Framework to find the objects involved in the distribution. The subscription of the resources is required to ensure that the operation is to be allowed according to the security model and to distribute the processing load of the distribution across the spoke regions.

Implementing these recommendations can cause some operational difficulties. It requires that operators use special procedures when they target an individual computer system for a distribution. This impact to operations to perform these special procedures is mitigated by the following:

- Using tasks to push profiles instead of pushing them directly
- Using built-in desktop features to perform the pushes
- Pushing to all endpoints when a particular monitor has changed

Operations do not need to be performed from the spoke regions. All operations can be performed from the hub region, but this does require that a proper logical design be put in place.

Resource exchanges and updates

Remember that resources and resource types are created by both Tivoli Management Framework and other Tivoli applications and third-party vendors. Therefore, Tivoli Management Framework and the applications have sets of resources that must be exchanged if they are to work across regions. Also, every application might have specific resource names that are not exchanged and thus might exhibit different behavior when crossing a region boundary. It is essential, therefore, to consider the interregional limitations of specific applications when deciding the logical design.

Chapter 11. Disaster recovery

As organizations become more dependent on information and communication technology to conduct their business and to stay competitive, the availability of the computer systems has become crucial. If computer systems become unavailable, the business processes can come to a halt.

A lengthy outage in a computing environment can result in significant financial losses. More importantly, you can lose customer credibility and subsequent market share. In some cases, such losses could lead to the total failure of the business.

Systems management products, such as Tivoli Management Framework, provide the facilities to manage the deployment and availability. What happens when hardware or software errors occur that affect the availability of Tivoli applications? If one or more components of the Tivoli environment become unavailable, you lose the automated capability to manage the critical systems and applications in your network. For this reason, Tivoli Management Framework should be considered an application whose availability is critical.

With key business functions depending on Tivoli Management Framework to manage and to monitor their operations, a Tivoli recovery system is essential. A *recovery system* is a process or a plan to be used in the event of any outage of a Tivoli application or of all Tivoli products. The outage could be due to numerous reasons such as hardware failure, software failure or deletion, a security breach, a disaster at the site, and so on.

The more an organization relies on Tivoli software products to manage the enterprise, the more important it is to have a recovery system in place. Because of the wide variety of parameters and requirements that could be included in such a system, this chapter provides a limited amount of technical guidance for preparing your recovery system. The following sections provide a means to group and to address organizational requirements.

This subject can be divided into the following topics:
- “Disaster recovery systems”
- “Using Tivoli backups for failure recovery” on page 163

Disaster recovery systems

Unless you have a disaster recovery system in place, you will have difficulty in recovering within a reasonable period of time. The recovery planning is a large subject in itself; however, use this information as a guideline while planning your Tivoli solution.

The more preventative measures you plan, the lower the chances are of a situation resulting in a disaster to the organization. No matter how sophisticated your preventative measures, there is always some risk of an outage. You can reduce the risk of outages as you increase the cost of preventing them, but you cannot totally eliminate the possibility. Hence, the recovery system should be a crucial element of an organization’s strategic business plan.

In addition, a recovery system should also include contingencies such as ongoing changes to the Tivoli environment. Most Tivoli environments continue to be dynamic after the initial installation. For example, an organization could expect to add new buildings with additional computer systems requiring management on a monthly or quarterly basis. This change should be reflected in the Tivoli architecture and environment and ultimately in how backups are performed.

Recovery systems need to consider the magnitude of the failure (the number of elements affected) and the estimated time to repair. The following are sample criteria for what is acceptable in terms of outages:

- A single endpoint can be out for up to a day.
- A gateway can be out for about 4 hours.

Depending on the Tivoli architecture and mission-critical systems in your environment, the criteria for acceptable outage times differ. Long outages affecting a large part of the network justify more elaborate recovery systems than a localized outage that can be fixed by restarting the affected system. To sum up, these items should be considered in a recovery system:

- Availability requirements
- Schedule for backups
- Tivoli application requirements
- Distribution across multiple computer systems and geographic locations
- Contingencies for growth of managed environment
- The number of different backup media

Disaster avoidance

Measures should be taken to prevent disasters if possible or to minimize their impact when they do occur. The best way to determine which measures your organization should include is to conduct a risk analysis to determine where your major vulnerabilities are.

The minimum preventative measures usually consist of on-site recovery procedures that you can use to keep routine problems from escalating into a disaster. Such measures include keeping spare hardware available, performing regular backups, and maintaining a skilled operations staff. In addition, you might want to consider fortifying your building, improving fire protection, implementing rigorous access control procedures, and strengthening corporate policies. The time and cost involved in implementing and maintaining such measures can be justified by comparing the time and cost involved in dealing with a disaster.

If, in spite of the preventative measures, a disaster occurs, you have to determine if the recovery requires on-site recovery or disaster-recovery procedures. The situation is not necessarily obvious, so the decision criteria have to be carefully prepared as part of planning and implementing a recovery solution.

Design considerations

Designing and implementing a recovery system is not a simple task. It can involve considerable effort and expense. As you develop your plan, you will make decisions such as the following:

- What systems and resources have to be managed in a recovery scenario? Will all product resources such as servers, applications and databases be duplicated, or will you duplicate a subset of those resources?

- Will the recovery Tivoli environment be a full copy of the production environment or a subset of it? For example, if the number of resources to be managed after a disaster is smaller, can you manage them all with a single Tivoli region or will you need multiple Tivoli regions?
- Does the entire Tivoli environment need to be recovered immediately, or can management functions be phased into production?
- Will the recovery staff be the same as the predisaster production staff? Are you providing recovery services from another site using staff who might not be familiar with the Tivoli solution? If you are, predisaster education and documentation for the Tivoli products are essential.
- What type of hardware will be available during the recovery? Will it be the same as the production hardware?
- Will it be necessary to change network addressing schemes for managed and managing resources?
- How often will Tivoli backups be done, where will they be stored, and how will they be used during the recovery?
- Will new Tivoli administrator definitions be required to restore the Tivoli environment?
- How will you test the recovery system?

These topics need to be considered in reference to your overall recovery system. If your organization does not currently have a recovery system, you might have to make many assumptions about resources that will be available and procedures that will be followed before and after a disaster occurs.

Defining the process

Recovery planning involves the following high-level activities. As with any design project, a structured approach helps to ensure that all the factors are considered and suitably addressed.

1. **Determine your business requirements**

   Conduct a risk and business impact analysis to determine the business needs. Identify and quantify business processes that cannot be performed without specific information technology (IT) processes, and try to place values on the IT processes involved. Factor in time scales for outages and cost of recovery to help prioritize steps of the recovery process.

   Consider also the risk the organization bears if some business transactions are lost during the disaster. Determine to what degree such a loss can be tolerated for each business process. The results can be used to determine the IT requirements.
2. **Determine your IT recovery requirements**

   Convert your business requirements into terms that your IT system designer can use. The result is usually prepared as a matrix showing, for each business and management application, the required recovery time, maximum allowable data loss, computing power required to run, disk storage required, and dependencies on other applications or data. A good deal of cooperation across organizational and departmental boundaries might be necessary to reach consensus on these matters.
3. **Design the backup and recovery Tivoli environment**

   Define the overall characteristics and major elements of the recovery Tivoli environment. These elements describe, in a generic way, any special hardware and software functions required for the backup and recovery processes, location for recovery items, the recovery configuration and the network and interconnection structures. Often, a high-level design is sufficient to estimate the cost of the solution. After the solution decision is definite, a more detailed design is required.

4. **Select products to match the design**

   Select the products to implement the plan. Your choices influence the cost of replacing the Tivoli environment, so some reiteration might be required. *Products* means the hardware, the software, and possibly the selection of an alternate site for the implementation.

5. **Create the backup and recovery Tivoli environment**

   Create the recovery Tivoli environment according to the design that was developed. To do this, you must make arrangements regarding alternate sites (if any), install any required hardware and software components, and develop the recovery system.

   The development of the recovery system is a project on its own. You must perform the following steps:

   a. Set up the recovery teams
   b. Develop and implement backup and recovery procedures
   c. Document the recovery steps

6. **Keep the solution current**

   As the computing environment changes, the recovery system must be modified to reflect the new realities. You also need to define procedures to ensure that the recovery Tivoli environment remains viable regardless of changes at the primary or alternate site. Do this by developing and implementing procedures for maintaining, testing, and auditing the recovery system.

Often steps of this process are not in sequence. Sometimes, for example, implementation work can begin while portions of the Tivoli environment design are still being developed. In other instances, the activities associated with a certain step are achieved in an earlier step. For example, an organization might decide to use a specific second site before the full business requirements are established.

More importantly, you will often reach a point in the design process where some aspect of the Tivoli environment design requires that you rework an earlier stage. In such cases, the process flow might have to loop back and modify the requirements or the design.

Recovery planning trade-offs

Although you want to develop a recovery solution that matches your business needs, be aware that some requirements involve trade-offs or might even be mutually exclusive. Also, the more stringent the requirements, the more costly the Tivoli recovery system is.

Four common trade-offs are as follows:

- How fast must the recovery be accomplished?
- How much data can be lost?
- What type of disasters does your plan cover?
- What will it cost?

Using Tivoli backups for failure recovery

When hardware or software failures occur, backups provide the necessary means for quick and simple recovery. It is essential to create a recovery system that defines what backups are performed and at what time interval they run in the Tivoli environment.

Moreover, it is important to plan regularly scheduled backups to best optimize a Tivoli environment recovery. The backup plan should take into account the relative importance of the Tivoli applications installed as well as the relative importance of the various managed nodes in the installation. For example, your managed environment might have its most important data and Tivoli applications residing on one or two key managed nodes. Thus, these managed nodes would require more frequent backups than other managed nodes in the Tivoli environment, so that there would be a recent copy of the data to facilitate a quicker recovery. For many organizations, mission-critical business systems are managed by Tivoli products. A good and recent backup of the managed nodes responsible for that business system allows a quick and easy recovery back to normal operations.

The following is an example schedule for maintaining and backing up the Tivoli environment with connected Tivoli regions:

- On a *daily* basis, run the following commands:
  - **wchkdb –u**

    Validates the integrity of the database and fixes any discrepancies found
  - **wupdate –r All All**

    Synchronizes the Tivoli name registries between connected Tivoli regions
  - **wbkupdb** *Tivoli_server*

    Backs up the Tivoli server only
- On a *weekly* basis, run the following commands:
  - **wchkdb –ux**

    Validates the integrity of the database, fixes any discrepancies found, and checks object references across region boundaries
  - **wupdate –r All All**

    Synchronizes the Tivoli name registries between connected Tivoli regions
  - **wbkupdb** *entire_region*

    Backs up all managed nodes and Tivoli server

The **wchkdb –u** command should be performed more frequently after changes to the configuration of the system, such as adding new endpoints or moving endpoints between Tivoli regions. Also, managed nodes that support key applications might need to be backed up more often than weekly.

The high-availability design as described in "High-availability design" on page 164 provides protection against hardware failures primarily through the use of redundant hardware. Sometimes, however, a high availability configuration is not available or it does not work in situations where software failed, databases became corrupt, or files were accidentally deleted.

The way to recover from these software failures is to prepare regular maintenance procedures that include checking the Tivoli database for consistency and backing it up, as discussed in "Using Tivoli backups for failure recovery." In the event of failures, the backups could be used to restore clients, thereby avoiding reinstalling the Tivoli applications. In cases where you must rebuild the environment from the beginning, you should have the backup of Tivoli databases and files to enable you to move into a production environment quickly.

For each failure scenario, develop and design a corresponding plan of action to recover from these failures. For specific maintenance procedures, including suggestions for backup and restore, see the *Tivoli Management Framework Maintenance and Troubleshooting Guide*

High-availability design

To help ensure the availability of specific systems and applications, many vendors offer product solutions generally known as high-availability (HA) products. HA products are designed to monitor critical systems and applications and to restart those applications (possibly on a different physical system) in case of failure.

By implementing Tivoli Management Framework in an HA environment, you can more consistently ensure the capability to manage the resources in your network.

Several HA products are available. Though each has its own unique benefits, they all work on the same basic concepts. *Availability* is simply the proportion of time that a system can be used for its intended purpose. An acceptable level of availability is different for every system in every environment and in general is dependent on the cost associated with the system or application being unavailable for a period of time.

Cluster concepts

In an HA environment, one of the key techniques is to eliminate single points of failure. This means if one part of your system fails, the system can continue to run or to recover in an acceptable period of time with minimal impact to the managed systems. Only in a case where a second error occurs before the first is fixed should a more prolonged outage occur.

Those components that are typically considered single points of failure include the following:

- Individual processors or nodes in a cluster
- Disks
- Adapters, controllers, and cables used to connect the nodes to the disk
- Network adapters attached to each node
- Network backbones over which the users are connected to the cluster
- Asynchronous adapters
- Application programs

Most HA products provide for monitoring of these resources and dynamically utilizing redundant hardware in the case of a failure.

Nodes in HA clusters

When looking at the implementation of Tivoli products in HA environments, the first consideration is Tivoli Management Framework. A specific system can be installed as a Tivoli server, managed node, or endpoint. In addition, the managed node might take on additional roles such as a gateway or the event server for the IBM Tivoli Enterprise Console product.

The Tivoli server is a critical element of the Tivoli region, and therefore is an obvious candidate to be implemented in an HA environment. Assuming that the network itself is available, the availability of the Tivoli server ensures that management operations can be carried out on active managed nodes.

Obviously, the primary motivation for increasing the availability of Tivoli servers is to help ensure that you can manage critical systems in your network. Managed systems not only include end-user systems, but also include application servers and other systems critical to the day-to-day operations of your business.

These application servers are also potential candidates to be implemented as HA nodes. However, if these nodes are also managed by Tivoli applications (for monitoring, software distribution, and so on), they must also be managed nodes or endpoints. If the application server fails over to another node in the HA cluster, the managed node or endpoint function should also fail over. Though system administrators might need to be alerted to the fact that a failover has occurred, they should not have to be aware of which physical node the application server is currently running on to perform other management functions. Therefore, the failover of a managed node or endpoint should be mostly transparent to Tivoli Management Framework.

Gateways do not typically need to be addressed from an HA perspective. The Tivoli architecture allows for endpoints to connect through alternate gateways if their current gateway is unavailable. Refer to Chapter 5, “Endpoints and gateways,” on page 47 for more information.

Most HA products provide for two types of cluster configurations, hot standby or mutual takeover. Basically, a *hot standby* configuration assumes a second physical system is capable of taking over for the first. This second system is not utilized except in the case of a failover. The *mutual takeover* configuration consists of two systems, each with their own set of applications, that can take on the function of the other in the case of a failure. In this configuration, typically the applications on both systems run in a degraded mode after the takeover, because one system is doing the job previously done by two. Mutual takeover is typically a less expensive choice in terms of hardware costs because it avoids having a system installed that is used only when a failure occurs.

Tivoli Management Framework supports multiple object dispatcher processes running on a single system. Therefore, both hot standby and mutual takeover configurations can be used with Tivoli software in UNIX environments. One solution that is often desirable is to have the Tivoli server and the event server configured into a cluster configured for mutual takeover. Though performance might be somewhat degraded while both the Tivoli server and the managed node with the event server are running on the same physical system, it allows for management to continue.

HA on UNIX Tivoli servers

The following sections describe some of the considerations for implementing Tivoli Management Framework in UNIX-based HA environments.

In the HA implementation of any application, you must understand what resources and facilities are used by the application. For example, the typical information that must be gathered for any application before configuring the HA product to support it includes the following:

- Resources used by the application, such as files and network facilities

- System configuration requirements
- Information regarding how to start and stop the application as well as how to test whether it is currently running

Files and file systems

When you install a Tivoli server or managed node using the default settings, files are created or modified in the following locations:

- /usr/local/Tivoli

  This directory tree contains the binary files required to run the Tivoli environment. In general, system- or Tivoli region-specific information is not stored in this directory tree. Therefore, it is relatively static with one exception described in “Placing Files on a Shared Disk” on page 167.
- /var/spool/Tivoli

  This directory tree contains the Tivoli region database. It is dynamically updated during the operation of Tivoli Management Framework and its availability and consistency is critical to the ongoing operation of Tivoli Management Framework. In an HA environment, the file system containing this tree should be located on a shared disk whose ownership can be taken over by another node in the cluster when a failure occurs. In this way, the node that has taken over for a failing node has full access to the Tivoli database. Because files located in the database are not executed, locked executable files interfering with the unmounting of the shared disk are not a concern as with the /usr/local/Tivoli directory.
- /etc/Tivoli

  When Tivoli Management Framework is installed, several files customized for the specific environment are placed in this directory. For example, the setup_env.sh file that sets all the appropriate environment variables for proper Tivoli Management Framework operation is included in this directory. The IBM Tivoli Enterprise Console product also places customized files within this directory tree by default. The number and size of the files in this directory is quite small, and they are typically not modified after they are created. Therefore, you can easily duplicate this directory across nodes in your cluster.
- /usr/lib/X11/app-defaults

  Any required resource files used by the X-based user interface for the desktop are stored in this directory. After Tivoli Management Framework is installed on the first node in a cluster, the files added by the installation can be copied to the same directory on the other nodes in the cluster.
- /tmp

  Tivoli Management Framework generates some log files in the/tmp directory. These are typically created during installation and, in general, you do not need to share or duplicate the files in this directory for proper operation of Tivoli Management Framework.

In addition to the directories in the preceding list, some standard system files are modified by the installation of Tivoli Management Framework. When setting up your HA environment, it is important to understand these modifications so that they can be replicated while setting up alternate nodes in the cluster.

- /etc/services

  A port for Universal Datagram Protocol (UDP) and Transmission Control Protocol (TCP) communication for Tivoli Management Framework is added to this file. The following are the lines that are appended to the file:

```
#
# Tivoli framework daemon
#
objcall         94/tcp          # Tivoli daemon
objcall         94/udp          # Tivoli daemon
```

- /etc/rc.nfs

  On AIX operating systems, the following commands for starting the object dispatcher are added to the /etc/inittab file. During installation, an option is presented related to whether the object dispatcher should be automatically started when the system is rebooted. For an HA environment, you do not want the object dispatcher started through this process. Rather, it is imperative that the HA product be responsible for starting the object dispatcher. If you chose the option to have the object dispatcher automatically started when the system is restarted, edit the /etc/inittab file and comment out the line that starts the object dispatcher (oserv). The line to look for in the file is:

  ```
  /usr/sbin/mkitab "$IDENT:2:once:$ETRC start > $DEVNULL 2>&1"
  ```

  Add a colon (:) character at the beginning of this line to comment it out. The change will take effect when the AIX machine is restarted. The edited line should look like this:

  ```
  :/usr/sbin/mkitab "$IDENT:2:once:$ETRC start > $DEVNULL 2>&1"
  ```

  Note: For other UNIX operating systems, it is important that the HA product starts the object dispatcher. Refer to your operating system documentation to determine how to properly implement the HA start of the object dispatcher.

- /etc/inetd.conf

  An entry for starting the environment is added. This entry is activated through a call to the inetd super daemon and is used by the Tivoli Management Framework remote start process.

  ```
  # Tivoli Framework daemon
  objcall dgram udp wait root /etc/Tivoli/oserv.rc \
  /etc/Tivoli/oserv.rc inetd
  ```

Placing Files on a Shared Disk

The convention in HA clusters is to place binaries relating to shared applications on a local disk.

With Tivoli Management Framework, the binaries themselves are static, but within this directory tree are the binaries for tasks. These are downloaded from the Tivoli server and need to be made available to the system running the service. To allow the task binaries to be made available on the disk shared by the systems in the cluster, you can do one of two things:

- Place the complete Tivoli binary tree on the shared disk.
- Create a separate file system for the portion of the tree containing the tasks on the shared disk.

Environment setup files in the /etc/Tivoli directory

The /etc/Tivoli directory in UNIX environments is used by Tivoli Management Framework for containing files such as environment setup files. Before starting a server installation, the EtcTivoli environment variable can be set to alter the location of these files. However, a number of Tivoli Management Framework processes require the files to remain in the /etc/Tivoli directory. Therefore, it is not recommended to use this variable to alter the default location.

In a hot-standby environment, the files within this directory should be copied to the standby system. In a mutual takeover environment, the installation of the second object dispatcher on the second system creates the /etc/Tivoli directory. However, because each object dispatcher requires its own version of the setup files (to set the appropriate paths for the binaries and database files), these files need to be renamed and procedures for setting the environment for each individual object dispatcher need to be modified.

On the whole, the files and links present in this directory are not customized for any particular installation of the object dispatcher process. The files and links are only read from the/etc/Tivoli directory. The exceptions to this are the following:
- setup_env.sh
- setup_env.csh
- oserv.rc
- The /tec directory
- The /tecad directory

The setup_env files are used to configure the environment so that the user or calling script can interact with a particular instance of the object dispatcher process. The oserv.rc file is used to start, stop, or restart a particular object dispatcher instance. The /tec and /tecad directories are used by the IBM Tivoli Enterprise Console product.

As different setup_env files and oserv.rc files are needed to interact with different object dispatcher processes, you can copy these to the appropriate directories and call them directly with your HA start and stop scripts.

Host name resolution

When Tivoli Management Framework is installed, the object dispatcher binds itself to the host name of the system on which it is being installed, and this information is stored within the database. If the host name of the system is later changed, the object dispatcher will fail to start. This means that if you move the Tivoli server to another machine, you also have to move the host name to this machine. Within Tivoli Management Framework, it is possible to set an environment variable called WLOCALHOST to define a logical host name to be used in place of the actual host name. When installing the Tivoli server, this variable should be set to the name associated with the service interface (IP label) of the cluster before the installation of Tivoli Management Framework. The service interface must be running during installation.

During installation, you need to specify the name of the server to be the same as specified by the WLOCALHOST environment variable. Otherwise, the object dispatcher binds to the actual host name (the name returned by the **hostname** command.) If this occurs, you will not be able to move the Tivoli node to an alternate system. Rather than setting the WLOCALHOST environment variable, it is also possible to get the same effect by creating a file called /etc/wlocalhost. This file should simply contain a single line identifying the desired host name.

In a mutual takeover environment, however, multiple object dispatcher processes can be running on a single system. Each instance of the object dispatcher might require its own particular value for WLOCALHOST. In this situation, you would not want to use the /etc/wlocalhost file because this variable would then be used system wide. Setting the WLOCALHOST variable only effects the shell in which you are running and processes started from that shell.

Managed node considerations

Only a few considerations are related to managed nodes in addition to those applying to Tivoli servers. When you install a managed node, the service interface should be running, and the shared file system should be mounted. If possible, install and configure the HA product first, and then bring the resource group online. After installing, you have to synchronize the same files and file systems as previously described.

You do not have to worry about tasks on managed nodes, because the tasks are stored only on the Tivoli server. If you execute a task on a managed node, the task is copied from the Tivoli server to the managed node /tmp directory and executed from there. Of course, when a task is in the process of being executed at the time of a failover, it might end prematurely and not restart automatically on the alternate node.

Mutual takeover environments

Mutual takeover environments are made possible with running multiple object dispatcher processes. The command that allows you to run multiple object dispatcher processes is **odadmin set_force_bind**. By default, the **set_force_bind** option is set to false. In this state, an attempt to start a second object dispatcher would fail.

Run this command against all the object dispatchers that might be required to run on the same system in the future. After the object dispatcher has **set_force_bind** set to true, it binds to only the IP address that matches the WLOCALHOST variable. Because each instance of the object dispatcher process is binding to a different IP address, you can start multiple object dispatcher processes on the same system. You need to use separate setup_env files and individual oserv.rc files for each instance of the object dispatcher being started. When started, running the corresponding setup_env file allows you to interact with the object dispatcher process.

Endpoint considerations

Implementing an endpoint as an HA resource is similar to implementing a managed node as an HA resource. Endpoints place their configuration and binary files in the /opt/Tivoli/lcf directory for UNIX endpoints. It is possible to put this directory on the shared disk.

When this directory is on the shared disk, failover can occur. When the endpoint restarts, it logs in to the gateway and restarts Tivoli operations such as monitors that were configured to run on this gateway.

Part 3. Reference

Chapter 12. Planning scenario 1

This guide has described methodologies for understanding your environment and using this information to design a Tivoli deployment. This appendix provides an example that demonstrates these concepts and processes.

You can accomplish a successful Tivoli deployment in many ways. Many of the decisions that need to be made are based on your environment. This scenario is fictitious, and by necessity many simplifying assumptions are made. However, you can use this example as a model of how the points discussed earlier in this guide can be applied to your environment.

The scenario is based on a fictitious company named XYZ software company.

XYZ software company—Overview

The XYZ software company designs and manufactures precision test instruments for the process control industry and for analytical laboratories.

The corporate headquarters for XYZ software company is located in New York and consists of the Administration, Engineering, and Marketing departments. These departments are located on a central campus in New York with reliable, high-speed local area network (LAN) connections among all the departments.

The Manufacturing department is located at a separate site in Vermont, connected to the corporate headquarters site through a high-speed link. In addition, the Manufacturing department also has 15 repair centers located across the U.S. They primarily interface with the Manufacturing site. Their network communication needs include all aspects of the business, including the following:

- Ordering parts and supplies
- Billing and cost accounting
- Personnel and other business administration needs
- E-mail

Figure 51 on page 174 illustrates the XYZ software company physical environment.

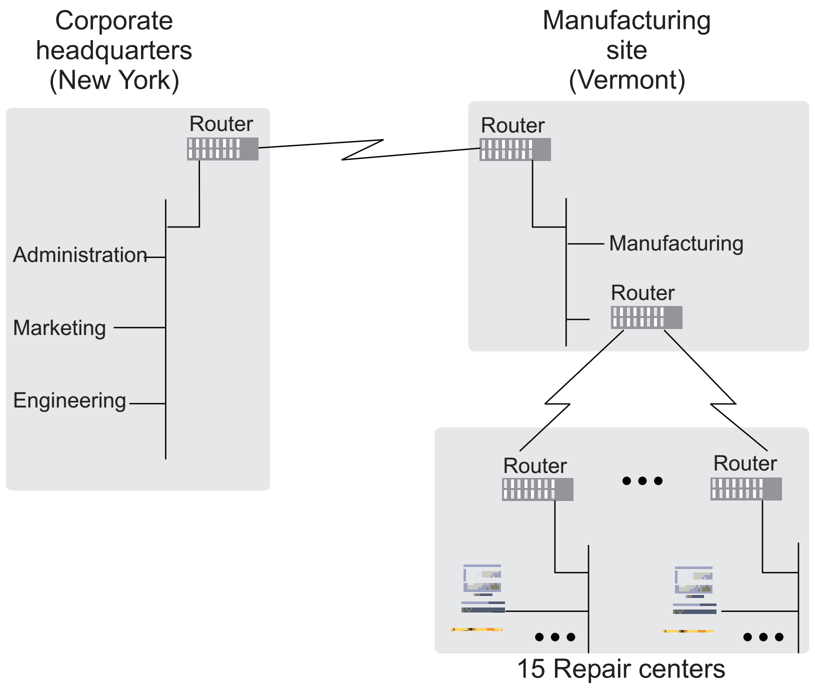


Figure 51. XYZ software company environment

Understanding the XYZ software company environment

As discussed in Chapter 7, “Design guidelines,” on page 85 when planning a Tivoli deployment, you should consider your environment from the physical, application, and organizational perspectives. This section looks at XYZ software company from each of these perspectives and discusses implications related to planning the Tivoli deployment.

Physical perspective

The physical perspective includes network, computer systems, and organizational factors.

Network factors

Both the corporate headquarters and Manufacturing sites have 100 megabit per second (Mbps) Ethernet LANs that were created using high-speed switches. The two LANs are interconnected through routers to give an appearance of one LAN in the organization. The two routers are connected through a T1 (1.544 Mbps) leased line. The repair centers are connected to the organization’s big LAN through 56 kilobit per second (Kbps) dedicated digital links connected through routers. The links are terminated at the Manufacturing site.

The entire networking infrastructure of XYZ software company is highly stable, and XYZ software company is happy with its performance. Based on a recent network analysis, XYZ software company observed that the LAN utilization in its two primary sites is less than 10 percent and there is ample room for more network activity. Because the network is fully owned by XYZ software company and is secure from external threats, the network does not include any firewalls or network address translation (NAT) devices.

XYZ software company uses Transmission Control Protocol/Internet Protocol (TCP/IP) as a standard for all its communication requirements. The user desktops communicate with servers only through TCP/IP. Each system in the organization has a static IP address and does not use Dynamic Host Configuration Protocol (DHCP). Domain name service (DNS) is used, and forward and reverse name resolution is supported.

Computer systems factors

XYZ software company has approximately 2,100 computer systems in the entire organization. There is no standard for a specific operating system. The operating systems include Windows 98, Windows NT, Windows 2000, UNIX, and OS/2. Different divisions use different equipment; however, they all share a common intranet.

The servers' environment is also heterogeneous. There are approximately 40 server class systems, which include Windows NT, Windows 2000, UNIX, AS/400, and OS/390 operating systems. Some of the servers are dedicated to different applications that are deployed throughout the organization, such as Lotus Notes and Systems, Applications, and Products in Data Processing (SAP™).

Each repair center has 5 to 10 user desktops connected to a Windows NT file server.

Organizational factors

A technology design must address the business objectives of the organization.

XYZ software company has several objectives for deploying Tivoli Management Framework, which include the following:

- Increase the availability of computer resources such as servers. This availability is extremely important to XYZ software company business and competitiveness.
- Reduce the costs of maintaining a standard computer system. XYZ software company wants to establish a standard computer system for each department or job function. They need a way to deploy and maintain these standard computer system configurations.
- Obtain a reliable and up-to-date inventory of all computer resources.
- Provide cost-effective remote support for the repair centers.
- Facilitate scalability for future expansion. XYZ software company plans to enlarge its business, including adding new sites in Europe, Asia-Pacific, and Latin America.
- Improve systems audits and communications. XYZ software company wants to monitor and control system changes or problems. Management of system changes and problems is vital to the corporate environment.
- Increase the availability of the managed environment. XYZ software company conducts business from 6:00 a.m. to 10:00 p.m. During this time, the availability of the managed environment is essential. The hours after 10:00 p.m. are generally used by support staff for maintenance activities such as backups, problem recovery, and testing.

Physical perspective summary

XYZ software company network connections are reliable and fast with the exception of the connections between the repair centers and the corporate headquarters site. The speed of the lines to these repair centers is of concern because it limits the ability to transfer large amounts of data to these sites.

The slow connections between the Manufacturing site and the repair centers must be considered in the architectural design. It is important to evaluate the effect of these links on software distribution and how gateways and repeaters can help limit their impact on system performance.

All systems support TCP/IP, so there is no need for multiprotocol support. There is a primary and a backup DNS server for host name resolution. A central administrator maintains the DNS tables.

Application perspective

The application perspective includes Tivoli applications and in-house applications factors.

Tivoli applications factors

To meet its objectives for systems management, XYZ software company plans to implement Tivoli Management Framework with IBM Tivoli Enterprise Console, IBM Tivoli Configuration Manager, and Remote Control. The following sections summarize how each Tivoli application will be used in the XYZ software company environment.

IBM Tivoli Enterprise Console: IBM Tivoli Enterprise Console will monitor the overall health of the computing environment and ensure that any problems are assigned to and handled by the appropriate support person.

IBM Tivoli Enterprise Console receives and processes all events sent by the other Tivoli applications. Through event correlation, actions can be automated and event escalation can be performed. Conservatively, the estimated number of events expected to be handled by the event server is fewer than two messages per second.

Software Distribution: Software Distribution will be used by both the user desktop and server support groups to keep the systems current with the appropriate software. Though standard software to be distributed might be defined and packaged by a central administrator, personnel in each support group might be responsible for initiating the distribution as well as defining additional file packages for distribution.

Except for the repair centers, which are connected through slow links, the organization has high-speed links. XYZ software company understands the limitation of slow links for transferring large amounts of software and will restrict itself. Large file transfers, such as an upgrade of an operating system, will be highly infrequent. These kinds of operations will be restricted to the weekends to avoid congestion on the slow links during the weekdays.

Distributed Monitoring: Distributed Monitoring will be used to monitor various server resources. There is no requirement or plan to monitor individual user desktops.

Inventory: An inventory of all hardware and software installed will be tracked. All inventory scans will be scheduled and initiated centrally. With the computer system standardization effort, the XYZ software company system inventory has to be reliable and constantly updated. Software inventory must be updated at least every three months.

Remote Control: Remote Control will be used by the user desktop support group so that they can take control of the displays and keyboards of users when reported problems cannot otherwise be resolved.

Because users have had the current applications for a long time, it is not expected to use Remote Control often. Its usage for the user desktops on the local LAN will not have an impact on the performance, as there is sufficient network bandwidth available. The only concern is for taking control of repair center desktops, which are connected through slow links. To avoid performance degradation, XYZ software company intends to restrict the usage of Remote Control for repair centers through administrative procedures, allowing only one session to a remote site at a time.

Organizational perspective

XYZ software company has a staff to provide both support and system administration for its systems. The administrative support group is divided into these groups: user desktop, server, and engineering support. Engineering support works exclusively with the Engineering department to support their specialized requirements. The engineering support group works with the user desktop and server support groups as necessary.

Therefore, the support staff consists of these groups:
- User desktop support
- Server support
- Engineering support

Administrators for user desktop support have the following responsibilities:
- Support users with day-to-day problems
- Perform routine maintenance
- Install new applications and upgrade existing ones
- Back up users' desktop data

Administrators for server support maintain the various servers and have the following responsibilities:
- Install, upgrade, and maintain file packages
- Maintain inventories of hardware and software
- Monitor network health, server loads, and so on
- Back up servers

In addition, other administrators implement the overall Tivoli deployment. They will maintain the company-wide inventory of hardware and software. In addition, they will define the standard software packages for user desktops and servers. However, unique requirements in each division might require some modifications or additions to the standard file packages before they are distributed and installed.

Tivoli architecture and design process

Designing the Tivoli architecture has two aspects: the physical design and the logical design. For the physical design, consider the number of Tivoli regions and the placement of key systems such as the Tivoli server, gateways, repeaters, and so on. The logical design consists of the policy region hierarchy, profile manager hierarchy, administrator definitions, and naming conventions.

Physical design

The factors that contributed to design decisions are described in the following sections; however, this design is only one possibility. Other factors or other interpretations of the requirements can lead to alternative design decisions.

Single or multiple Tivoli regions

The following considerations suggest the use of a single Tivoli region:

- **Performance considerations**

  XYZ software company plans to procure new servers to be used for the Tivoli server and managed nodes. The size and speed of these servers are appropriate, so there is no need to consider multiple Tivoli regions for server performance reasons.

  Regarding network performance, the T1 link between the two primary sites is sufficient to support a Tivoli server on one side and managed nodes or gateways on the other side. Also, the current utilization of the network is less than 10 percent, which means that there is sufficient network availability for a single region.
- **Scalability limits**

  There will be approximately 2,100 managed systems in the environment. Most of these will be endpoints, and there will be less than 200 managed nodes (including gateways). Based on the number of managed systems, a single region will suffice.
- **Organizational considerations**

  Though the engineering support staff is physically located within the Engineering department, they work closely with the server and user desktop support staffs. Therefore, the support staff can be considered a single entity. There is no administrative need to split the managed systems into multiple Tivoli regions.
- **Security considerations**

  The XYZ software company network is dedicated, stable, and secure. There are no requirements to use multiple encryption levels, to restrict administrator access, or to implement a firewall. Thus, a single region will suffice.
- **Tivoli applications considerations**

  There are no concerns related to the combination of the Tivoli applications that are being considered that require multiple Tivoli regions. The size of this environment and the intended use of the Tivoli applications can be handled by a single region.
- **Managing servers versus user desktops**

  The number of systems to be managed is not large enough to warrant separating the management of servers and user desktops into multiple Tivoli regions.
- **Slow network links**

  Each of the repair centers has only 5 to 10 user desktops and a single server. There will be one gateway at each of these sites to manage the user desktops. This managed node will also be a repeater for carrying out software distribution to these user desktops and the server. Because each repair center has only one managed node, the maintenance tasks can be carried out appropriately across the slow link during the scheduled time for maintenance.

Number and placement Tivoli servers

In the following sections, the number and network placement of Tivoli servers are discussed.

Tivoli server

It is recommended that the Tivoli server be located near the administrators who manage it. It will be located within the IT administration department at the corporate headquarters.

Gateway

Next, XYZ software company will consider the number and placement of gateways within the region. Based on the placement guidelines in “Gateways” on page 128, the following gateways are included in the environment:

- Two gateways at the corporate headquarters site, gateway A and gateway B. Gateway A will support endpoints at that site. Gateway B is an alternate gateway that will provide a level of redundancy for failure of any other gateway in the organization.
- One gateway at the Manufacturing site, gateway C. This gateway will suffice for the number of systems at that site.
- One gateway at each repair center. Currently, each repair center has a Windows NT server that also provides the connectivity between the systems at the repair center and the corporate network. These servers will become managed nodes and gateways for the endpoints at each site.

Repeater

The gateways will also be used as repeaters (see Figure 52 on page 180). The redundant gateway at the corporate headquarters (gateway B) serves as a source host for Software Distribution. The file package residing on gateway B will be distributed to both gateway A at the corporate headquarters, attached to the local LAN (at 100 Mbps), and to gateway C at the Manufacturing site. Upon receiving the file package, gateway A will distribute the file package to all its endpoints connected on the 100 Mbps LAN. Similarly, gateway C will distribute the file package to all its endpoints connected on the 100 Mbps LAN and a copy of the file package to each of the repair center gateways, for fanout to their endpoints.

When you configure gateway B and gateway C as repeaters, consider the netload factor. If you configure repeater settings on gateway B with a positive netload, you could flood the network link to a slower repeater, for example gateway C. Whereas if you configure gateway B with a negative netload, you likely will have an unnecessarily slow distribution to a fast repeater, for example gateway A. There are similar limitations associated with both netload settings for gateway C.

Therefore, add an additional repeater, repeater D, at the corporate headquarters that will receive data from the source host for Software Distribution at local LAN speeds (100 Mbps) and distribute it to the gateway C at the Manufacturing site. Gateway B can now be configured with a positive netload, and repeater D will be configured with a negative netload. Adding repeater D optimizes the source host repeater (gateway B) for the 100 Mbps speed and provides the best service to the local endpoints at the corporate headquarters.

Likewise, at the Manufacturing site, add an additional repeater, repeater E, that will receive distributions from gateway C at LAN speeds and then redistribute to the repair center gateways. Gateway C can now be configured with a positive netload, and repeater E will be configured with a negative netload.

Note: Keep in mind that if your Tivoli applications are using both MDist and MDist 2 services in a Tivoli region, you must configure the settings for a repeater for both services. For example, this scenario discusses setting the netload of a repeater. For MDist, netload is specified using the **net_load** setting of the **wrpt** command. For MDist 2, netload is specified using the **net_load** and **target_netload** settings of the **wmdist** command. For more information about configuring a repeater, refer to the *Tivoli Management Framework User's Guide*.

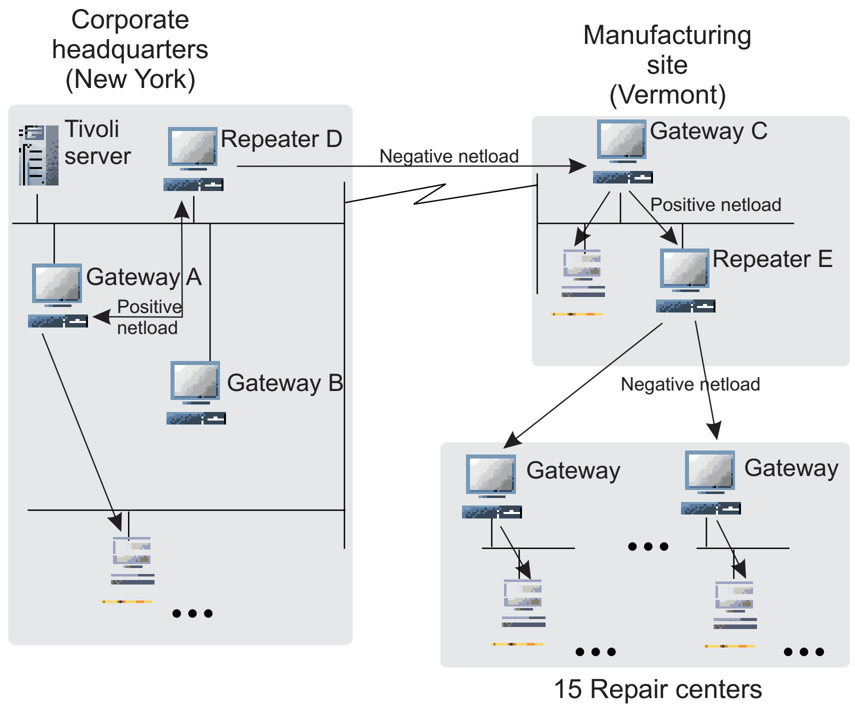


Figure 52. XYZ software company repeater hierarchy

RIM server

XYZ software company must define RDBMS Interface Module (RIM) objects for Inventory and IBM Tivoli Enterprise Console. These RIM objects can be configured differently, and XYZ software company can configure them to run on either two different RIM host servers or from a single RIM host server.

For this example, assume that the RIM objects for each of the two applications must be configured to run on two different RIM host servers. The reasons for this decision could be performance of different applications, automation tasks planned, number of events to be handled, and so on. Refer to the application documentation for further details.

IBM Tivoli Enterprise Console will be configured to run with the IBM Tivoli Enterprise Console RIM object and with the event database on the same system where the event server is configured to run. For Inventory, the RIM object will run on the Tivoli server, and the inventory database will reside on a separate system.

Application servers

For IBM Tivoli Enterprise Console, an event server is required for the XYZ software company environment. For Software Distribution, a source host is required for the XYZ software company environment.

Event server: Tivoli Management Framework supports one event server per region. As mentioned earlier, this event server will also host the IBM Tivoli Enterprise Console database and the RIM object.

Source host: In the environment, there will be one source host for the Software Distribution component of IBM Configuration Manager at the corporate headquarters. Depending on the frequency of the distributions, the size of the file packages, and the usage of the T1 link between the corporate headquarters site and the Manufacturing site, XYZ software company could plan to have another source host at the Manufacturing site at a later time.

Instead of using a separate computer system for the source host, XYZ software company will use the redundant gateway, gateway B, at the corporate headquarters site for this purpose. The computer system hosting gateway B will also act as a Tivoli Software Installation Service repository.

Physical design summary

The overall requirement is for 22 managed nodes to manage approximately 2,100 endpoints. Each of the managed nodes will also be running an endpoint. The endpoint allows XYZ software company to perform system management operations on the computer system hosting the managed node.

Logical design

The logical design includes the policy regions, profile managers, task libraries, administrator definitions, and so on. It is best to define a naming convention for the various resources that will be used in the environment first.

Naming conventions

The guidelines in “Naming conventions for Tivoli objects” on page 135 provide a basis for the naming conventions used for XYZ software company. In general, the names of objects follow the *usage.platform.tmr-name.resource-type* convention. The resource type names are shown in the following table.

Resource Type	Name
Policy region	pr
Task library	tl
Job	jb
Indicator collection	ic
Collection	co
Profile manager	pm
Distributed Monitoring profile	dm
User profile	up
Group profile	gp
Host profile	hp
Print profile	pp
File package	fp
AutoPack	ap
File block	fb

At this time, there is only a single region, but to allow for future expansion, a region name of XYZ software company is included in the format.

For this example, the *platform* variable for resources is not used in the names. The *usage* variable might have subcomponents to help the administrator understand the purpose of the object and its location within the object hierarchy. For example, a policy region containing managed systems (nodes) within the Engineering department would have the name: ENG-Nodes.XYZ.pr. This policy region would be contained within the Nodes.XYZ.pr policy region.

Policy region structure

In this section, the policy region hierarchy for XYZ software company is discussed. The three prongs of the policy region hierarchy (as described in "Overview of a policy region hierarchy" on page 143) will include the following:

- Nodes policy region
- Subscribers policy region
- Policy regions for each Tivoli application

Nodes policy region

This policy region will contain a set of subregions to be used as containers for the various groups of managed systems within the XYZ software company environment. Creating a hierarchy of subregions provides the ability to organize the managed nodes and easily navigate to them when needed. It also enables XYZ software company to limit the administrative authority to add, change, or delete managed systems.

The policy region hierarchy for XYZ software company consists of two layers of subregions that represent the separation of user desktop administration and server administration as well as the separation of the Engineering department from all other business departments. The following represents the policy region hierarchy:

```
Nodes.XYZ.pr
   DSK-Nodes.XYZ.pr
      ENG-DSK-Nodes.XYZ.pr
      BUS-DSK-Nodes.XYZ.pr
   SVR-Nodes.XYZ.pr
      ENG-SVR-Nodes.XYZ.pr
      BUS-SVR-Nodes.XYZ.pr
```

where "DSK" refers to user desktops, "SVR" refers to servers, "ENG" refers to Engineering department systems, and "BUS" refers to all other non-Engineering department systems.

The above hierarchy could be extended to include specific operating system platforms and more departments. For example, if there was a reason to manage the Marketing systems separately from the rest of the business (non-engineering) systems, XYZ software company could add a policy region named MKT-DSK-Nodes.XYZ.pr. For this scenario, this policy region hierarchy provides the navigation and access control required for XYZ software company. For example, administrators responsible for the Engineering department would be given the authority to add managed systems (nodes) to the policy regions with the ENG prefix, but would not have the authority to add managed systems to other policy regions.

Subscribers policy region

For this scenario, the policy region hierarchy is simple, and complexity will be built into a profile manager hierarchy as shown in the following examples. In the policy region hierarchy, a specific policy region can contain both subregions and profile managers appropriate for that region.

The Subscriber.XYZ.pr policy region contains three different subregions, based on the administrative organization (user desktops, servers, and engineering). These subregions have been created to enable the administrators to create their own subscription lists to be used by the appropriate support staff. Inside each region will be sets of profile managers to be used by the administrators who are authorized for that region. As the environment expands and requires more separation of authority, additional policy regions and subregions can be added.

The policy region hierarchy will consist of the following:

```
Subscriber.XYZ.pr
   DSK-Subscriber.XYZ.pr
   SVR-Subscriber.XYZ.pr
   ENG-Subscriber.XYZ.pr
```

Within each of these policy regions, there will be a profile manager hierarchy that will provide the required distribution lists for use by Tivoli administrators. Note that a profile manager hierarchy is created by having a series of subscriber-only profile managers subscribed to other profile managers. In the DSK-Subscriber.XYZ.pr policy region the profile manager hierarchy is as follows:

```
DSK-Subscriber.XYZ.pm
   MKT-DSK-Subscriber.XYZ.pm
      NT-MKT-DSK-Subscriber.XYZ.pm
      AIX-MKT-DSK-Subscriber.XYZ.pm
      .
      .
   MFG-DSK-Subscriber.XYZ.pm
      NT-MFG-DSK-Subscriber.XYZ.pm
      AIX-MFG-DSK-Subscriber.XYZ.pm
      .
      .
SVR-Subscriber.XYZ.pm
   MKT-SVR-Subscriber.XYZ.pm
      NT-MKT-SVR-Subscriber.XYZ.pm
      AIX-MKT-SVR-Subscriber.XYZ.pm
      .
      .
   MFG-SVR-Subscriber.XYZ.pm
      NT-MFG-SVR-Subscriber.XYZ.pm
      AIX-MFG-SVR-Subscriber.XYZ.pm
      .
      .
```

where “MFG” refers to Manufacturing and “MKT” refers to Marketing.

Some of the lower-level profile managers that are subscribers to higher-level profile managers can be used to create distribution lists that are based on other criteria. For example, XYZ software company can define a profile manager for all user-desktop Windows NT systems in XYZ software company. This profile manager would have the following hierarchy:

```
NT-DSK-Subscriber.XYZ.pm
   NT-MFG-DSK-Subscriber.XYZ.pm
   NT-MKT-DSK-Subscriber.XYZ.pm
   .
   .
```

As administrators require distribution lists based on other categories, a senior administrator can create the required profile managers. If the number of subscriber profile managers in a particular policy region becomes too large for administrators to navigate easily, XYZ software company can use generic collections as containers for homogeneous sets of profile managers.

The profile manager hierarchy is independent of the policy region hierarchy. That is, assuming the administrator has authority in all the proper policy regions, profile managers making up the hierarchy can reside in different policy regions. Therefore, if needed, one administrator could be authorized to add new nodes to lower-level profile managers, while a different administrator could be responsible for building the higher-level profile managers that are subscribed directly to the application profiles.

Thus, XYZ software company can build a profile manager hierarchy that will, at the node level, contain very specific sets of systems. XYZ software company can add new systems in their environment to lowest level node profile managers, which automatically include the systems in actions taken on the higher-level profile managers. To control the authority for adding new systems to specific subscriber lists, XYZ software company can create policy regions to contain these profile managers and limit administrator authority to access these policy regions.

Application policy regions

The third aspect of the three-prong policy region approach is a set of top-level policy regions. Because these policy regions contain the profile managers that are used for day-to-day management operations, it is useful to make them as accessible as possible to the administrators. There will be one top-level policy region for each Tivoli application deployed within XYZ software company. Thus, there is a policy region for Software Distribution, Inventory, server management (Distributed Monitoring and IBM Tivoli Enterprise Console), and Remote Control. This approach assumes that administrators will be categorized according to the managing application or management discipline that they use. For example, some administrators use Distributed Monitoring and others use Software Distribution.

Within the top-level policy regions, a hierarchy of subregions containing task libraries and application-specific profile managers will be defined. These profile managers have various subscriber-only profiles (as described in the previous section) subscribed to them and will be used by the appropriate administrators.

Software Distribution: The following describes the XYZ software company guidelines for using Software Distribution:

- Corporate headquarters in New York defines the standard file packages to be distributed to user desktops and servers.
- Any modification and distribution is performed separately by the three groups (user desktop, server, and engineering).

Using these guidelines, the policy region hierarchy will be as follows:

```
SD.XYZ.pr
   DSK-SD.XYZ.pr
      DSK-SD.XYZ.pm
   SVR-SD.XYZ.pr
      SVR-SD.XYZ.pm
   ENG-SD.XYZ.pr
      ENG-SD.XYZ.pm
```

where SD refers to software distribution.

The different application profile managers in this hierarchy will have subscribers that are platform specific. For example, the hierarchy for the DSK-SD.XYZ.pm profile manager is the following:

```
DSK-SD.XYZ.pm
   WIN95-MKT-DSK-Subscriber.XYZ.pm
   NT-MKT-DSK-Subscriber.XYZ.pm
   WIN98-MFG-DSK-Subscriber.XYZ.pm
   .
   .
```

The administrators within the various support staffs will have the **admin** role for the policy regions associated with their customers. For example, members of the server support staff will be given **admin** authority over the SVR regions. This enables those administrators to distribute the defined profiles to the appropriate audience. An administrator with **senior** role would be responsible for creating the profiles and subscribing the appropriate profile managers.

Inventory: XYZ software company wants to use Inventory to capture the configuration setup of all target systems. This includes the hardware and software configuration of the designated endpoints.

XYZ software company has a single administrator placed at the corporate headquarters who is responsible for handling the inventory scanning. For Inventory, the application policy region hierarchy will be as follows:

```
INV.XYZ.pr
   INV.XYZ.pm
      DSK-Subscriber.XYZ.pm
      SVR-Subscriber.XYZ.pm
      ENG-Subscriber.XYZ.pm
```

where INV refers to inventory.

Server management: The function of server management at XYZ software company using IBM Tivoli Monitoring and IBM Tivoli Enterprise Console is to provide the following:

- Continuous monitoring of critical events occurring on different servers
- Notification to appropriate personnel from the appropriate support organization
- Root cause analysis
- Standard automation tasks

The following are the XYZ software company guidelines for server management:

- Define server monitors centrally at the corporate headquarters.
- Engineering server support personnel can either accept a monitor and alter it or reject a monitor. Also, they can distribute monitors solely to the Engineering department.

Using these guidelines, the server management policy region hierarchy will be as follows:

```
MON.XYZ.pr
   BUS-SVR-MON.XYZ.pr
      BUS-SVR-MON.XYZ.pm
   ENG-SVR-MON.XYZ.pr
      ENG-SVR-MON.XYZ.pm
```

where “MON” refers to distributed monitoring.

The application profile managers in this hierarchy will have subscribers, which can be platform specific. For example, the hierarchy for the BUS-SVR-MON.XYZ.pm profile manager is as follows:

```
BUS-SVR-MON.XYZ.pm
   AIX-MKT-SVR-Subscriber.XYZ.pm
   NT-MKT-SVR-Subscriber.XYZ.pm
```

Remote Control: Though Remote Control does not use profiles like other Tivoli applications, there is still a requirement for a separate policy region for Remote Control. This policy region will restrict which administrators can open remote sessions with managed systems.

Remote Control usage is divided into the server, user desktop, and engineering groups. Accordingly, the policy region hierarchy is as follows:

```
RC.XYZ.pr
   DSK-RC.XYZ.pr
   SVR-RC.XYZ.pr
   ENG-RC.XYZ.pr
```

where "RC" refers to remote control.

There is no profile manager hierarchy below these policy regions, because there is no need to send any profiles in this application. The policy regions will contain an administrator icon to access the applications. Administrators for the DSK-RC.XYZ.pr policy region will be able to open remote sessions with user desktops on which they have been given authority through the nodes policy region.

Administrators

As described earlier, the XYZ software company support staff is divided according to the types of systems they support. Administrators support user desktops and servers. In addition, the Engineering department has its own support staff to help with its special requirements.

Based on these responsibilities, roles will be assigned to the various administrators for certain policy regions. For example, a user desktop support administrator requires authority to monitor remote user desktops, so the administrator will require the **admin** role for the remote control policy region.

The only use of a global role (as opposed to policy region-specific roles) will be the **user** role. All administrators will have a global role of **user** that enables them to look at managed resources throughout the Tivoli region. However, to perform management operations, administrators require at least the **admin** role. This role is applied to each administrator on a policy region by policy region basis. This ensures that administrators do not exceed their authority.

The following are general groups of administrators for XYZ software company. There can be one or more administrators in each group. In addition, each administrator can be a member of one or more groups.

Note: Tivoli Management Framework does not have the administrator groups as described here. These groups are simply general definitions of the various responsibilities that different administrators will have. When defining the actual administrators, XYZ software company needs to map roles to the appropriate policy regions for each administrator. Grouping them logically makes it easier to assign the correct roles to each administrator.

- Corporate Senior: This group will have the **senior** role across the entire Tivoli region. Administrators with this level of access will be responsible for creating policy regions, profile managers, profiles, and other resources that are used by the administrators that perform day-to-day management operations. These administrators will also be responsible for performing corporate-wide inventory scans.

  The profiles these administrators create apply across the corporation. However, in some cases, profiles might be modified by senior administrators who are responsible for various types of systems or departments, such as Engineering.
- User Desktops Senior and Servers Senior: These administrators will be responsible for user desktops and servers. They can create profiles unique to their area and modify profiles provided by the corporate senior administrator.
- User Desktops Admin: This group of administrators will be responsible for the daily management of the user desktop systems. They will require the **admin** role in the DSK policy regions as well as the subscriber policy regions.
- Server Admin: Similar to the User Desktops Admin group, these administrators will access the SVR application policy regions and the subscriber policy regions. This will allow them to manage the various server systems within their area of responsibility.
- Engineering Senior: These administrators are responsible for defining any special resources required within the Engineering department.
- Engineering Admin: These administrators perform all day-to-day management that is unique to the Engineering department. They will have the **admin** role over only those policy regions associated with Engineering and will have no authority outside of those regions (other than the global **user** role).

Administrators for individual Tivoli applications can define their own roles beyond those that are defined as part of Tivoli Management Framework. See the product documentation for details on any application-specific roles.

Other considerations

The following sections describe additional design considerations.

XYZ software company operational tasks

Task libraries will be located in the appropriate policy regions as defined in "Application policy regions" on page 184. XYZ software company will implement task library objects for the following functions:

Start The administrator must be able to start a service without knowing the exact command invocation. This function might not be implemented for core platform resources, such as the operating system or the network.

Stop Gracefully halt the service.

Restart
Stop the service unconditionally and then restart it. This function is essential to ensure service availability.

Get-Configuration
The administrator must be able to retrieve the static configuration of any managed resource, for example, software versions and database names served.

Get-Status
The administrator must be able to determine the dynamic operational state of the service, for example, current load and resource usage.

Get-Errorlog
The administrator must be able to locate the service error log for purposes of troubleshooting and further escalation.

XYZ software company Tivoli region security

Passwords and traffic encryption are described in the following sections.

Passwords

Installation passwords will not be used with the installation of Tivoli products. Only the senior headquarters administrators will have the **install_product** role.

Traffic encryption

The network is reliable and secure. Inside a region, encryption will not be used. It is set to **none**.

Tivoli region backup policy for XYZ software company

XYZ software company will make Tivoli backups of the Tivoli region daily. XYZ software company requires additional backups whenever the management environment is significantly altered, both before and after the change. These are called checkpoint backups. For example, after installing a new Tivoli application or service pack, back up the Tivoli region.

The name of the checkpoint backup file indicates date, time, and its additional content related to the backup procedure.

Every 10 days, XYZ software company will archive daily backup files. Checkpoint backup files remain in an easily accessible location.

Systems backup policy

XYZ software company will perform a complete system image backup on all servers (file, applications, management and gateways and repeaters) every week. This backup occurs on Sundays at 12:00 a.m. The backup tapes are stored at an off-site location.

Also, XYZ software company will perform incremental backups every night for strategic servers, SAP R/3 servers, and Lotus Notes servers. Tivoli ADSM is used for these backups, and they are useful in the case of server failure. For example, with the weekly image and incremental backups, a failure in a Lotus Notes server can be reconstructed without losing more than one day of information.

Tivoli region recovery for XYZ software company

For the XYZ software company environment, the following table displays levels of Tivoli region failure and the appropriate recovery procedure. Failures are assumed to be transient unless proven otherwise.

Failure	**Recovery**
Server database inconsistency	Run the **wchkdb –ux** command.
Server database corruption	Restore from the Tivoli backup.

Failure	Recovery
Server transient failure	Restart the Tivoli server object dispatcher.
Client transient failure	Restart the managed node or gateway object dispatcher.
Server full failure	Reboot the server operating system.
Client full failure	Reboot the client operating system. Migrate the endpoints to a contingency gateway.
Client database corruption	Run a selective restore operation from backup. Migrate the endpoints to a contingency gateway.
Client environment corruption	Delete the binaries and database, and re-create the client. Migrate the endpoints to a contingency gateway.
Server CPU failure	Failover to the secondary CPU.
Server shared disk failure	Reboot the server from an external disk.
Tivoli environment corruption	Restore from the checkpoint. On failure, reload from images, and reinstall the Tivoli server, managed nodes, and gateways.
Server database inconsistency	Run the **wchkdb –ux** command.

Databases are checked for inconsistencies every week on Saturdays at 12:00 a.m. This check is important because it ensures the reliability of the Tivoli environment.

Chapter 13. Planning scenario 2

Chapter 12, “Planning scenario 1,” on page 173 uses a sample scenario based on a fictitious company, ABC Instruments, to describe a Tivoli region. This appendix describes a multiple Tivoli region design for the same organization. The design aspects covered are related to the change in topology from a single region to multiple regions. Hence, it is recommended that you understand scenario 1 before reading this scenario.

This scenario is fictitious, and by necessity many simplifying assumptions have been made. However, you can use this example as a model of how to apply information in this guide to your environment to create a Tivoli environment that includes multiple regions.

ABC Instruments—overview for scenario 2

This overview describes additional ABC Instruments systems to be managed in Europe. The business goals are the same as those described in “XYZ software company—Overview” on page 173, with some changes to the location of activities (refer to Figure 53 on page 192).

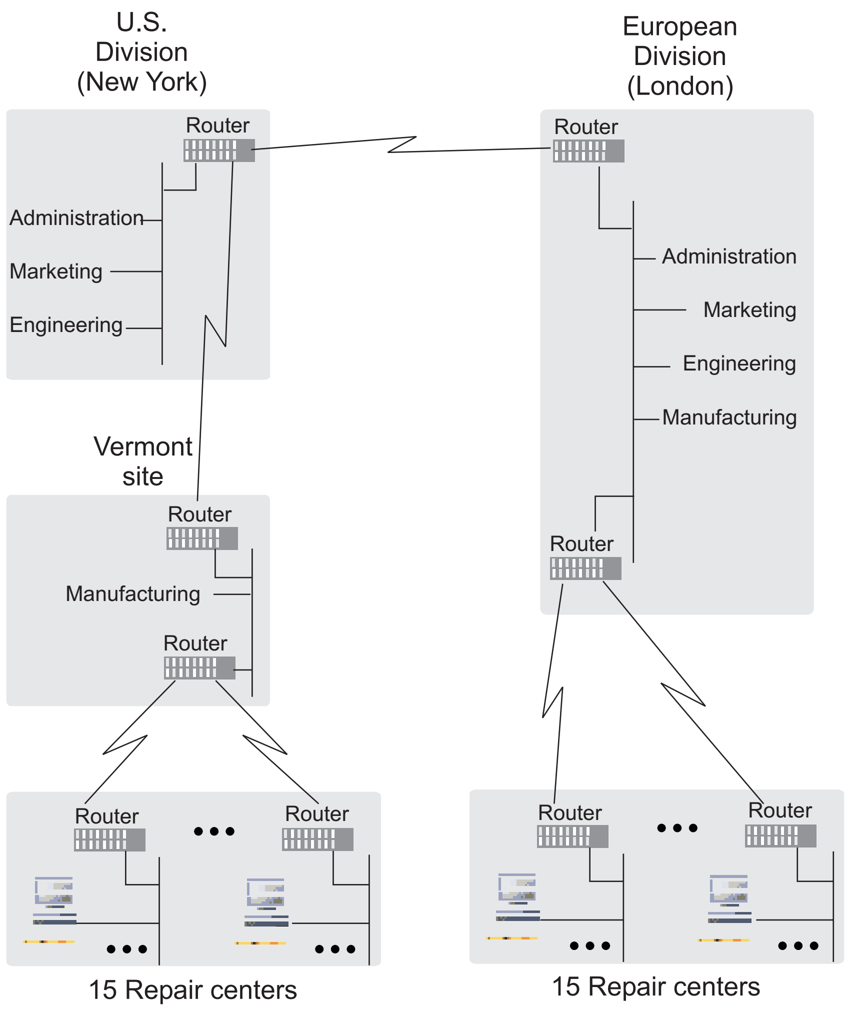


Figure 53. ABC Instruments environment for scenario 2

ABC Instruments corporate headquarters is located in the U.S.; however, ABC Instruments has a second division in Europe. The European division is located in London. Both the U.S. and European divisions consist of the Administration, Engineering, and Marketing departments with reliable, high-speed local area network (LAN) connections among all the departments.

In the U.S., the Manufacturing department is located at a separate site in Vermont and connected to the corporate headquarters site through a high-speed link. Repair functions are performed at 15 remote locations that are geographically dispersed across the U.S. Their communication needs are the same as described on page "XYZ software company—Overview" on page 173.

In Europe, the Manufacturing department is located at the London site. Repair functions are performed at 15 remote locations geographically dispersed across Europe. Repair functions, however, are arranged contractually; that is, each of the centers is an independently-owned business that negotiates contracts directly with the Manufacturing department. In their daily work, they communicate almost exclusively with the European Manufacturing department, and their only communication requirements are as follows:

- Ordering parts
- Billing

- Occasional distribution of parts catalogs

Understanding the ABC Instruments environment

In this section, the European environment of ABC Instruments is examined from the physical, application, and organizational perspectives. The U.S. environment is discussed on page “Understanding the XYZ software company environment” on page 174. Unless otherwise specified, the discussions in scenario 1 apply to this scenario also.

Physical perspective

The physical perspective includes network, computer systems, and organizational factors.

Network factors

The European network is concentrated at the London site, except for the repair centers, which are dispersed across Europe. Similar to the U.S. network, the London site has a 100 megabit per second (Mbps) Ethernet LAN that uses high-speed switches. The network functions as one high-speed LAN, shared by servers and user desktops in all four departments. Except for the repair centers, the managed resources in all the divisions are connected directly to this LAN. The independent repair centers connect to servers on the LAN through 56 kilobit per second (Kbps) dial-up connections, which are set up through routers that are used periodically. Little management is required for these systems, but occasionally an updated version of a client application or a parts catalog must be distributed to these systems.

The U.S. and European divisions are connected to each other through routers. The two routers are connected through a full T1 (1.544 Mb per second) leased line.

Computer systems factors

ABC Instruments has approximately 2,100 user desktops in the U.S. and approximately 1,000 user desktops in Europe. The operating systems include Windows 95, Windows 98, Windows NT, Windows 2000, UNIX, and OS/2. Approximately 40 servers in the U.S. division and 20 servers in the European division include Windows NT, Windows 2000, UNIX, AS/400, and OS/390 platforms. Some of the servers are dedicated to applications such as Lotus Notes and SAP, which have been deployed throughout the organization.

The 15 repair centers in the U.S. division each have between 5 and 10 user desktops connected to a Windows NT file server. The 15 repair centers in Europe each have either 1 or 2 standalone user desktops.

Organizational factors

The considerations are the same as those discussed on page “Organizational factors” on page 175.

Physical summary

ABC Instruments network connections are reliable and fast, with the exception of the connections between the repair centers and the headquarters sites. Those slow lines to repair centers might limit the ability to transfer large amounts of data to these sites. Another consideration is the time difference between the U.S. and European divisions. Specifically, the appropriate time for maintenance of the two divisions is different.

Application perspective

The considerations are the same as those discussed on page “Application perspective” on page 176.

Organizational perspective

In both the U.S. and Europe, ABC Instruments has a staff of individuals to provide support and system administration. Therefore, in the New York and London sites, the staff consists of these groups:

- User desktop support
- Server support
- Engineering department support

The administrators, who implement the overall Tivoli deployment, work at the corporate headquarters in New York. They are responsible for maintaining the company-wide inventory of hardware and software. In addition, they define the standard software packages to be used on user desktops and servers. However, unique requirements in each division might require some modifications to these standard software packages before they are distributed and installed.

The U.S. and European support organizations are autonomous.

Tivoli architecture design process

Designing the Tivoli architecture has two aspects: the physical design and the logical design. For the physical design, consider the number of Tivoli regions and the placement of key systems such as theTivoli server, gateways, repeaters, and so on. The logical design consists of the policy region hierarchy, profile manager hierarchy, administrator definitions, and naming conventions.

Physical design

For scenario 2, ABC Instruments requires a Tivoli environment with multiple Tivoli regions.

Single or multiple Tivoli region

The following are some of the considerations, in order of priority, that suggest a multiple Tivoli region design.

Performance considerations: ABC Instruments plans to enlarge its business soon by creating new locations in Asia-Pacific and Latin America. The long-term goals are to manage systems in major countries across the globe.

The operations in Asia-Pacific and Latin America will be similar to the existing divisions in Europe and the U.S. The two new sites will have infrastructures with approximately the same number of servers and user desktops as the U.S. and European divisions. These systems must also be managed in the Tivoli environment.

For now, a single Tivoli region could handle the load from the U.S. and European divisions. Eventually, the infrastructure load from additional divisions will demand multiple regions rather than a single region. Deploying multiple regions after the initial deployment will require additional design efforts. In addition, down time in the managed environment can occur because ABC Instruments will be required to migrate to the multiple region topology.

Therefore, with expansion to more divisions imminent, it is suggested that the U.S. and European divisions each have its own region for managing the environment within its division. In the future, this means one region each for the U.S., European, Asia-Pacific, and Latin American divisions. This makes expansion feasible within each division.

Scalability limits: Approximately 3,100 managed systems are in the ABC Instruments organization. However, because of expansion plans, the recommended limits for a single region will be exceeded. Based on performance considerations, plan for multiple regions now so that the new divisions can be managed as they are created without any further planning.

Organizational considerations: From an administrative viewpoint, the U.S. and European divisions of the organization are autonomous. The administrators located in the two divisions work with systems in their own geography only. Therefore, ABC Instruments can divide the managed systems into multiple regions according to division.

ABC Instruments will perform the consistency checks and backups of the Tivoli object database regularly. These checks and backups should be run only when no other major Tivoli region processes are running. Because of the time difference between Europe and the U.S., it will be difficult to arrive at a common maintenance schedule to perform these activities.

ABC Instruments can perform these maintenance tasks efficiently by dividing the managed environment into multiple regions according to division. By having a region in each of the two divisions, ABC Instruments can assign individual maintenance schedules.

Security considerations: The ABC Instruments network is dedicated, stable and secure, so security considerations are not part of the Tivoli deployment.

Application considerations: ABC Instruments will use IBM Tivoli Enterprise Console to monitor the overall computing environment (servers) and ensure that any problems are assigned to and handled by the appropriate support engineer. Because of the scalability limits of IBM Tivoli Enterprise Console and the limitation of one event server per region, determine the event rate to be handled by IBM Tivoli Enterprise Console in the organization before planning the number of regions.

ABC Instruments uses Lotus Notes and SAP applications, but will not manage these applications through the Tivoli modules in the first phase of implementation. After deployment of Tivoli Management Framework, ABC Instruments plans to deploy appropriate modules to manage these applications. These modules use IBM Tivoli Enterprise Console extensively, because IBM Tivoli Enterprise Console gathers data on failure conditions found in both the Lotus Notes and SAP environments. IBM Tivoli Enterprise Console will also be used to take corrective action by running tasks in these environments.

A theoretical estimation based on the events and failure logs from the past few years at ABC Instruments predicts that one event server is sufficient to handle the events in the first phase. However, in the second phase, the number of events to be handled by IBM Tivoli Enterprise Console will increase considerably. At that time, ABC Instruments might need multiple event servers and multiple regions in the environment.

One event server per division permits the management operations to grow when new divisions are included in the organization. This structure also provides a possibility for expansion within the division.

Slow network links: When the use of tasks and task libraries by IBM Tivoli Enterprise Console to take corrective actions becomes intensive, performance might dictate that the region be geographically close to the systems being managed. With a single region, the IBM Tivoli Enterprise Console network traffic must travel all the way from the European division to the event server located in New York. Even with the availability of a T1 link between the two sites, the wide area network (WAN) bandwidth might not be large enough if problem sites in the European division start sending large numbers of events to the event server, thus delaying the corrective actions.

The goal is to limit the network traffic across the WAN by isolating problems locally and taking corrective actions through the local event server in each of the divisions. The event servers in each division can automate the event management of the organization, and they can send the events to the corporate event server only for escalation purposes.

Multiple Tivoli region design topology

Each of the Tivoli regions is responsible for managing a different physical segment of ABC Instruments. These managed environments should be connected to create a common environment.

To meet the expected performance, management, and growth requirements of ABC Instruments, the physical topology will be configured in the hub and spoke architecture.

The hub region provides a centralized Tivoli server in its own dedicated Tivoli region. The hub Tivoli server directly manages a limited number of resources, so that it is dedicated primarily to the support of administering the Tivoli environment. The other functions performed by the hub region are as follows:

- Provide the global assets view for the entire organization based on information gathered by Inventory
- Act as the focal point for organization-wide activities, such as creating global standards
- Configure and distribute generic monitoring profiles to the two divisions for their servers
- Configure and distribute file packages to the two divisions according to the standards of the organization
- Receive Distributed Monitoring responses from the spoke regions as a part of an escalation process
- Provide access to the Tivoli object database used by the user desktops at the central support organization

Each of the spoke regions will be connected to the hub region through a two-way connection. The functions of the spoke regions in the U.S. and European divisions are as follows:

- Directly control all endpoints and managed nodes in the division
- Improve performance by grouping managed resources by network location and localizing Tivoli functions to that physical location
- Act as a primary point for distributed monitoring responses
- Provide the inventory view for the division
- Enable the support staff for the division to access the Tivoli object database used by the user desktops in the division
- Modify and distribute the file packages and server monitoring profiles within the division

Number and placement of Tivoli servers

The following section discusses the number and placement of the required Tivoli servers, managed nodes, and gateways.

Tivoli servers: There will be three Tivoli regions:

- The hub region
- The U.S. spoke region
- The European spoke region

The hub Tivoli server and the U.S. Tivoli server will be located at corporate headquarters in New York. The European Tivoli server will be placed in London. London is closer to the managed nodes and endpoints it manages. The placement in London keeps the European Tivoli server on the other side of the WAN.

Gateway: A gateway is not required for the hub region, because it is not supporting any endpoints. For the U.S. region, placement of gateways will be similar to that mentioned in scenario 1 on page "Gateway" on page 179.

For the European region, three gateways will be used. The endpoints could be supported on a single gateway, but three gateways create redundancy in the case of a gateway failure. Because the gateway is also a repeater, ABC Instruments can distribute the load during distributions to large numbers of endpoints. The gateway distribution will be as follows:

- Gateway L will support approximately 450 Administration and Marketing department endpoints.
- Gateway M will support approximately 525 Engineering and Manufacturing department endpoints.
- Gateway N will act as backup to the two other gateways in case one fails and will also be used as the gateway for dial-up endpoints, such as those at the independent repair centers.
- The European repair centers are not part of ABC Instruments network and will not have dedicated gateways. When these systems dial in to the ABC Instruments network, they will connect to gateway N at the London site. These repair centers generate very little traffic, requiring only infrequent distributions of moderately sized database files and updates to a small client program.

Repeater: In addition to the U.S. repeaters described in "Repeater" on page 179, more repeaters are required in Europe to distribute software to all geographies efficiently. For Europe, it is necessary to configure gateways and the Tivoli server as repeaters (see Figure 54 on page 198).

Gateway L, which is supporting Administration and Marketing endpoints at the London site, will also serve as the Software Distribution source host. When you configure gateway L as a repeater, you will need to use a positive netload for distribution to its endpoints and the other two repeaters (gateway M and gateway N) on the same LAN. Because the three repeaters reside on the same 100 Mb per second LAN, a positive netload ensures an efficient distribution. Gateway N for dial-up repair centers will in turn distribute to repair center endpoints.

The hub Tivoli server needs to distribute to the two spoke Tivoli servers at different speeds: the U.S. Tivoli server at 100 Mb per second and the European Tivoli server at T1 speed. To have an efficient repeater configuration, add one more repeater, repeater P, in the hub region. The hub Tivoli server configured with a positive netload will distribute to both the U.S. Tivoli server and repeater P. Repeater P, which is configured with a negative netload, will distribute to only the European Tivoli server.

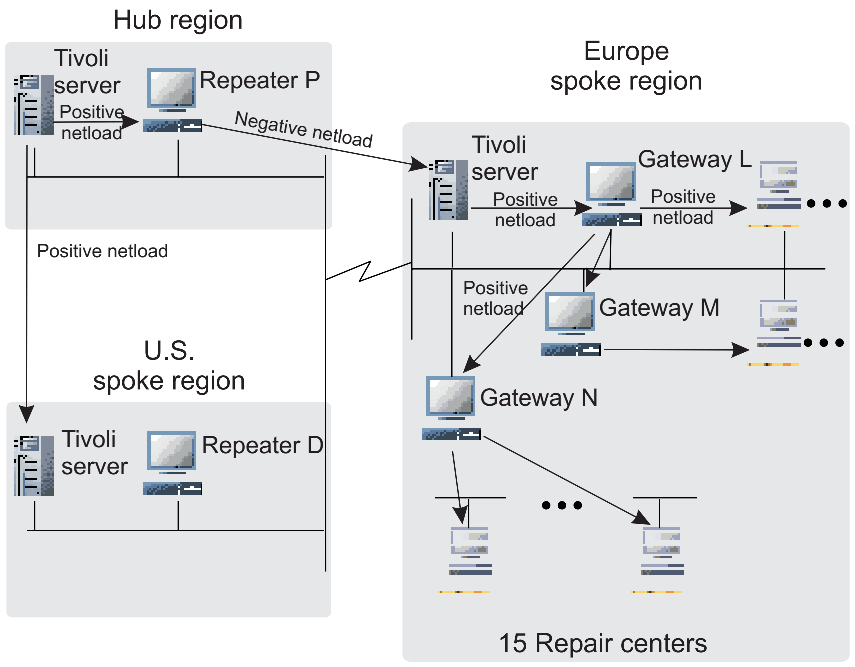


Figure 54. ABC Instruments repeater hierarchy

RIM servers: RDBMS Interface Module (RIM) objects will be placed in a configuration similar to that in scenario 1, as discussed on page “RIM server” on page 180.

For Inventory, a RIM object will run on all three Tivoli servers in the environment. These RIM objects will refer to the inventory database, which will reside on a separate system at corporate headquarters. Scans originate from a inventory profile distribution in each of the spoke regions, and the scan results are saved in the inventory database. Queries against the inventory database are run from the hub region.

In each of the three regions, the IBM Tivoli Enterprise Console RIM object and the event database are located on the same system as the event server.

Application servers: For IBM Tivoli Enterprise Console, the event servers will be placed close to the Tivoli server. The role of the event servers in the two divisions will be as follows:

- Perform automation tasks on events using the defined rules tables

- Through event correlation, perform actions such as escalation to administrators
- Escalate the events to the hub event server for administrative action

There will be two source hosts for Software Distribution for ABC Instruments: a server in the U.S. region (described in scenario 1 on page “Source host” on page 181) and a server in the European region. Instead of using a separate computer system for the source host, ABC Instruments will use the gateway that manages Administration and Marketing department endpoints in the European region, gateway L. The computer system hosting gateway L will also act as a Tivoli Software Installation Service repository.

Physical design summary

Thirty managed nodes will manage approximately 3,100 endpoints. Each of the managed nodes and Tivoli servers will also be running an endpoint. The endpoint allows you to perform system management operations on the computer system hosting the managed node.

Logical design

The logical design includes the policy regions, profile managers, task libraries, administrator definitions, and so on. Before these structures are planned, it is best to define a naming convention for the various resources that will be used in the environment first.

Naming conventions

The naming conventions for resources in the ABC Instruments environment will be similar to scenario 1. The names of the three Tivoli regions will be as follows:

- Hub region: ABC
- U.S. spoke region: USA
- European spoke region: EUR

For example, a policy region containing systems from the user desktop division in the European region would have the name DSK-Nodes.EUR.pr. This policy region will be contained within the Nodes.EUR.pr policy region. Use these conventions for all future Tivoli regions.

Policy region structure

This section describes the policy region hierarchy to be implemented at ABC Instruments according to the recommendations described in “Policy region structure” on page 182.

Nodes policy region: There will be a nodes policy region in each of the three Tivoli regions. The policy regions in the two spoke regions, that is USA and EUR, will contain a set of subregions to be used as containers for the various managed systems within their own divisions. The hierarchy for these two regions consists of three layers of subregions that represent the administrative and operational separation (servers and user desktops) and a departmental separation.

The following represents the nodes policy region hierarchy within the U.S. region. A similar hierarchy will be created for the European region.

```
Nodes.USA.pr
   DSK-Nodes.USA.pr
      ENG-DSK-Nodes.USA.pr
      BUS-DSK-Nodes.USA.pr
   SVR-Nodes.USA.pr
      ENG-SVR-Nodes.USA.pr
      BUS-SVR-Nodes.USA.pr
```

The hub region will directly manage very few systems. These systems will be managed nodes that have been added to support Tivoli user desktops, event servers, Software Distribution source hosts, repeaters, and so on. The hub region should not grow to support endpoints; that is the function of a spoke region. The systems in the hub region should be systems dedicated to the operation of the Tivoli environment. They should not be systems that could otherwise be managed by a spoke region.

In ABC Instruments, there will be two managed nodes, which will be managed directly by the corporate Tivoli server. These nodes are the event server and repeater. The repeater system will also be a gateway for the hub region. It will serve as a management station for Tivoli administrators. As a gateway, it is used for new endpoints as they enter the environment. It is also the gateway from which the hub region manages the hub event server, the Tivoli region, and each of the spoke regions.

Because there are only two managed nodes to be managed within the corporate Tivoli region, there is no need to create subregions within the nodes policy region. A top-level nodes policy region will suffice.

Subscribers policy region: ABC Instruments will create a subscribers policy region in each of the three Tivoli regions. If there is a subscribers policy region in only the hub region, the various resources in the spoke regions must be subscribed across the Tivoli region boundaries multiple times in different profile managers, which is not recommended. These three subscribers policy regions will contain a hierarchy of subregions designed to contain subscriber-only profile managers. The U.S. and European regions will have a policy region hierarchy similar to the one described in scenario 1. For example, the structure for the European region is as follows:

```
Subscriber.EUR.pr
   DSK-Subscriber.ABC.pr
   SVR-Subscriber.ABC.pr
   ENG-Subscriber.ABC.pr
```

There will be a similar policy region hierarchy within the U.S. region. The hub region will not include subregions within the subscriber policy region.

Within the subscribers policy regions in each of the three regions, there will be a profile manager hierarchy that will provide distribution lists to Tivoli administrators. The U.S. and European regions will have a profile manager hierarchy similar to the one created for the U.S. region in scenario 1.

The requirements for the hub region profile managers hierarchy are as follows:

- Resources from the two divisions must be consolidated in a single distribution list. For example, ABC Instruments can have a distribution list for all the Windows NT servers, all the user desktops, all servers, or all Engineering user desktops in ABC Instruments. Application administrators that perform management actions on sets of managed systems will use these hub profile managers.
- The application profile managers in the application policy region in the hub region need a distribution list relevant to the U.S. and European divisions. This distribution list already exists in the subscribers policy region in the spoke regions and can be used for this purpose. However, because these spoke subscribers profile managers will be required by multiple application profile managers, they will need to be subscribed multiple times across the region boundaries, which is not recommended.

ABC Instruments will create a profile manager hierarchy in the hub region similar to those in the spoke regions. The subscribers to these will be similar to those in the spoke regions. The hub profile managers in their subscribers policy region are then subscribed to the appropriate hub application profile managers in the application policy regions in the hub region. The advantage to this approach is that you subscribe the spoke region profile managers across the region boundaries only once.

The subscribers profile manager hierarchy for the hub region is as follows:

```
Subscriber.ABC.pr
   Global-Subscriber.ABC.pm
   USA-Subscriber.ABC.pm
   EUR-Subscriber.ABC.pm
```

The Global-Subscriber.ABC.pm profile manager will have the following structure for all the Windows NT servers in ABC:

```
Global-Subscriber.ABC.pm
   NT-SVR-Subscriber.ABC.pm
      NT-ENG-SVR-Subscriber.EUR.pm
      NT-ENG-SVR-Subscriber.USA.pm
      NT-MKT-SVR-Subscriber.EUR.pm
      NT-MKT-SVR-Subscriber.USA.pm
      NT-MFG-SVR-Subscriber.EUR.pm
      NT-MFG-SVR-Subscriber.USA.pm
      .
      .
```

Applications policy regions: The applications policy region will contain the profile managers that are used for day-to-day management operations. The application administrators will use these policy regions to perform management actions on sets of managed systems within their scope of authority. There will be a top-level policy region for each Tivoli application deployed within ABC Instruments.

The applications policy region hierarchy within a Tivoli region might or might not exist, dependent on the organizational structure and requirements. The next section discusses the different applications and defines the policy region hierarchy within the three regions.

Software Distribution: The ABC Instruments guidelines for the policy region for Software Distribution are as follows:

- Corporate headquarters in New York will define the standard file packages to be used on user desktops and servers.
- The user desktop, server, and engineering groups will separately modify and distribute file packages within each of the divisions.
- The three groups should have permissions to distribute software within only their own managed resources. For example, the server group should not be able to distribute software to user desktops within a division.

Using these guidelines, the policy region hierarchy for Software Distribution will exist in only the hub region and will have the following structure:

```
SD.ABC.pr
   USA-SD.ABC.pr
      DSK-USA-SD.ABC.pr
      SVR-USA-SD.ABC.pr
      ENG-USA-SD.ABC.pr
   EUR-SD.ABC.pr
      DSK-EUR-SD.ABC.pr
      SVR-EUR-SD.ABC.pr
      ENG-EUR-SD.ABC.pr
```

where SD refers to software distribution.

The various application profile managers in this hierarchy will have subscribers as defined in “Software Distribution” on page 184.

Inventory: The ABC Instruments guidelines for the policy region hierarchy for Inventory are as follows:

- Inventory scanning will be performed by the senior administrator for each division
- The senior administrator at corporate headquarters must be able to view the complete ABC Instruments inventory.

Using these guidelines, the policy region for Inventory will be created in each of the three regions. The U.S. and European regions will each distribute inventory profiles and store scan results in their local inventory databases. A database administrator will join the two inventory databases to provide a composite view in the hub region. The policy region for Inventory in the hub region will be used only to view the ABC Instruments assets from this composite database view.

The policy region hierarchy for Inventory will be as follows for the European region:

```
INV.EUR.pr
   INV.EUR.pm
```

The profile manager for Inventory will have the following subscribers defined in the subscribers policy region structure:

```
INV.EUR.pm
   DSK-Subscriber.EUR.pm
   SVR-Subscriber.EUR.pm
   ENG-Subscriber.EUR.pm
```

There will be a similar structure in the U.S. region. The policy region for Inventory in the hub region will not have any profile managers because it does not need to perform the scan operations.

Server Management: The guidelines for server management at ABC Instruments using Distributed Monitoring and IBM Tivoli Enterprise Console are as follows:

- Corporate headquarters in New York defines the standard server monitors to be used on all servers in the organization.
- Monitors are modified and distributed separately by the server (business) and engineering groups within each of the divisions.
- The server and engineering groups should have permissions to distribute monitors within only their own managed resources. For example, the business server group should not be able to distribute monitors to servers in the Engineering department within a division.

Using these guidelines, the hierarchy of the server management policy region will exist only in the hub region and will have a structure like the following:

```
MON.ABC.pr
   USA-MON.ABC.pr
      BUS-SVR-USA-MON.ABC.pr
      ENG-SVR-USA-MON.ABC.pr
   EUR-MON.ABC.pr
      BUS-SVR-EUR-MON.ABC.pr
      ENG-SVR-EUR-MON.ABC.pr
```

The various application profile managers in this hierarchy will have subscribers, which could be platform specific as explained in "Server management" on page 185.

Remote Control: The ABC Instruments guidelines for Remote Control are as follows:

- The remote control operations will be performed by administrators in the two divisions. Corporate headquarters in New York is not involved in this activity.
- The user desktop, server, and engineering groups within each division should be able to take control of resources only within their group.

Using these guidelines, the policy region hierarchy for Remote Control will exist only in the U.S. and European regions and have the following structure for the European region:

```
RC.EUR.pr
   DSK-RC.EUR.pr
   SVR-RC.EUR.pr
   ENG-RC.EUR.pr
```

There will be a similar structure in the U.S. region. This policy region structure will restrict which administrators can open remote sessions with managed systems. Below these policy regions there will not be a profile manager hierarchy because there is no need to send any profiles with Remote Control. The policy regions will contain an administrator icon to access the applications.

Chapter 14. Authorization roles

This section contains tables that describe the authorization roles that you need to perform activities in Tivoli Management Framework. It contains one table for each defined resource or service. Using these tables, you can determine which activities you can perform with your assigned role or which role is required to perform a specific task.

Roles

To perform system administration activities in Tivoli Management Framework, you must be a Tivoli administrator. Depending on what activities you are required to perform, you must have one or more of the following roles:

- **admin**
- **backup**
- **Dist_control**
- **install_client**
- **install_product**
- **policy**
- **restore**
- **RIM_view**
- **RIM_update**
- **senior**
- **super**
- **user**

Other roles might be available in your Tivoli environment depending on which Tivoli software products are installed. When you create a new administrator or change the properties of an existing administrator, the list of defined and available resource roles and region roles is displayed in scrolling lists. Consult the document for each installed Tivoli software product for information about the capabilities associated with each authorization role.

Installation

Table 4 presents the authorization roles required to perform installation activities in a Tivoli region. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 4. Required authorization roles associated with installation

Required role	Context	Activity
Not applicable (root access or Administrator privileges)	Tivoli server	Install a Tivoli server on the designated server machine.
install_client	Policy region	Install a managed node in the Tivoli region. Remove a managed node in the Tivoli region.

Table 4. Required authorization roles associated with installation (continued)

Required role	Context	Activity
install_product	Tivoli region	Install a product or an application in the Tivoli region. Install a patch or upgrade in the Tivoli region.

For information about these tasks, refer to the *Tivoli Enterprise Installation Guide*.

Repeater configuration

Table 5 presents the authorization roles required to perform configuration operations on multiplexed distribution repeaters. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 5. Required authorization roles associated with repeater configuration

Required role	Context	Activity
senior	Tivoli region	Create a repeater. Delete a repeater. Tune a repeater for network load, maximum number of connections, maximum amount of memory, and maximum amount of disk paging space. Query an MDist repeater for its current configuration and active distributions.
senior or both**Dist_control** and **RIM_view**	Tivoli region	Cancel, pause, and resume a distribution. Delete distributions from the database.

Administrators

Table 6 presents the authorization roles required to perform activities on Tivoli administrators. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 6. Required authorization roles associated with Tivoli administrators

Required role	Context	Activity
senior	Administrators collection	Create an administrator. Delete an administrator. Add or remove resource roles. Add or remove login names. Add or remove notice group subscriptions. Edit the properties of a Tivoli administrator
	Tivoli region	Add or remove Tivoli region roles.
user	Tivoli desktop	Create a collection and drag and drop resources onto it. Drag and drop resources onto an administrator desktop.

Managed nodes

Table 7 presents the authorization roles required to perform activities on managed nodes. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 7. Required authorization roles associated with managed nodes

Required role	Context	Activity
install_client	Policy region	Install a managed node. Remove a managed node.
admin	Policy region containing a managed node	Add an entry to the IP interface list of a managed node. Remove an entry from the IP interface list of a managed node.
user	Policy region containing a managed node	Display the properties window of a managed node.
user and a system account on the managed node	Policy region containing a managed node	Open a managed node window. Open an X terminal on a UNIX managed node.

Tivoli region connections

Table 8 presents the authorization roles required to connect and disconnect Tivoli regions and to perform activities on connected regions. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 8. Required authorization roles associated with connected Tivoli regions

Required role	Context	Activity
super	Tivoli region	Disconnect Tivoli regions. Make a remote connection between two Tivoli regions. Make a secure connection between two Tivoli regions.
senior	Tivoli region	Exchange resource information between Tivoli regions. Schedule an exchange of resource information between Tivoli regions.

Policy regions

Table 9 presents the authorization roles required to perform activities on a policy region. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 9. Required authorization roles associated with policy regions

Required role	Context	Activity
senior and **policy**	Tivoli region	Create a top-level policy region.†
	Parent policy region	Create a subregion.†
	Policy region	Add or remove a managed resource type to or from a policy region. Assign or change the policy for a managed resource type.
senior	Parent policy region	Delete a subregion.
	Policy region	Change the name of a policy region. Delete a top-level policy region.
	Source and destination policy regions	Move resources from a policy region to another policy region.
admin	Policy region	Check policy in a policy region.
user	Policy region	Open a policy region window.

† Whether both the **senior** and **policy** roles are required when performing these operations from the command line depends on the command line options used. Refer to the documentation for the **wcrtpr** command in *Tivoli Management Framework Reference Manual* for information about roles and policy regions.

Profiles and profile managers

Table 10 presents the authorization roles required to perform activities on profile managers and profiles. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 10. Required authorization roles associated with profiles and profile managers

Required role	Context	Activity
senior	Policy region	Change policy on the records of a profile in a profile manager. Create a profile manager in a policy region. Delete a profile manager from a policy region. Edit a profile manager. Clone a profile in a profile manager. Copy a profile in a profile manager. Create a profile in a profile manager. Delete a profile from a profile manager.
	Source and destination policy region	Move a profile manager from one policy region window to another policy region window. Move a profile from one profile manager window to another profile manager window.
	Policy region	Set policy on the records of a profile in a profile manager. Synchronize a profile in a profile endpoint.
admin	Policy region of the profile manager and policy region of the subscriber	Distribute one or more profiles from a profile manager. Subscribe a managed node or another profile manager. Remove a subscription for a managed node or another profile manager.

Task libraries

Table 11 on page 210 presents the authorization roles required to perform activities on tasks and jobs contained in task libraries. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 11. Required authorization roles associated with tasks and jobs in task libraries

Required role	Context	Activity
senior	Policy region	Change the policy for a task library in a policy region. Create a task library in a policy region. Delete a task library from a policy region. Set the policy for a task library in a policy region.
	Source and destination policy region	Move a task library from a policy region window.
admin	Policy region	Create a job or task in a task library. Delete a job or task from a task library. Edit a job or task in a task library.
	Job	Schedule a job for future execution.
The role defined in the task or job specification	Job	Execute a job.
	Task	Execute a task through either drag-and-drop or double-click. Save the output from task execution. Save the output from job execution.

Notice groups

Table 12 presents the authorization roles required to perform activities on notice groups. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 12. Required authorization roles associated with notice groups

Required role	Context	Activity
senior	Tivoli region	Expire notices in a notice group. Set notice expiration length for a notice group.

Table 12. Required authorization roles associated with notice groups (continued)

Required role	Context	Activity
user	Tivoli desktop	Combine multiple related notices in a notice group into a single listing. Display old notices in a notice group. Filter notices in a notice group. Forward one or more notices through e-mail. Mark notices in a notice group as read or unread. Read notices in a notice group. Save notices in a notice group to a file. Sort notices in a notice group.

Scheduler

Table 13 presents the authorization roles required to perform activities on the scheduler. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 13. Required authorization roles associated with the scheduler

Required role	Context	Activity
super	Tivoli region	Start the scheduler.
admin	Tivoli region	Disable or enable a scheduled job. Edit a previously scheduled job. Remove a previously scheduled job.
	Scheduler	Schedule a job.
user	Scheduler	Browse the list of scheduled jobs. Change the information displayed in the list of scheduled jobs. Find one or more jobs displayed in the list of scheduled jobs. Sort the jobs displayed in the list of scheduled jobs.

Maintenance

Table 14 presents the authorization roles required to perform maintenance activities. It lists the required authorization role, the context in which you need the role, and the activities you can perform with each role.

Table 14. Required authorization roles associated with maintenance

Required role	Context	Activity
super	Tivoli region	Check the integrity of the Tivoli management region database. Enable or disable diagnostic tracing. Run the **odadmin** and **odstat** commands. Place the Tivoli region in single-user mode.

For information about maintenance tasks, see *Tivoli Management Framework Maintenance and Troubleshooting Guide*.

Chapter 15. Directory structure and system variables

This section describes the directory structures and system variables created during the installation of Tivoli Management Framework.

UNIX Tivoli servers and managed nodes

When you install Tivoli Management Framework on UNIX operating systems, many new files and directories are created, some existing files are modified, and several system variables are defined.

Note: When you install Tivoli Management Framework on a UNIX operating system, the /tmp/.tivoli directory is created. This directory contains files that are required by the object dispatcher process. You should not delete this directory or any of its contents unless explicitly directed to by your Tivoli support provider. You should also ensure that regularly scheduled disk clean-up jobs (cron or Tivoli jobs) do not remove this directory or its contents.

To use a different directory, you must set an environment variable in both the object dispatcher and the shell. After installing Tivoli Management Framework, perform the following steps to set the necessary environment variables:

1. Create a directory. This directory must have at least public read and write permissions. However, define full permissions and set the sticky bit to ensure that users cannot modify files that they do not own.
2. Set the environment variable in the object dispatcher:
   a. Enter the following command:
      ```
      odadmin environ get > envfile
      ```
   b. Add the following line to the `envfile` file and save it:
      ```
      TIVOLI_COMM_DIR=new_directory_name
      ```
   c. Enter the following command:
      ```
      odadmin environ set < envfile
      ```
3. Edit the Tivoli-provided setup_env.csh, setup_env.sh, and oserv.rc files in the /etc/Tivoli directory to set the TIVOLI_COMM_DIR variable.
4. For HP-UX and Solaris systems, add the following line to the file that starts the object dispatcher:
   ```
   TIVOLI_COMM_DIR=new_directory_name
   ```
   Insert the line near where the other environment variables are set, in a location that runs before the object dispatcher is started. The following list contains the file that needs to be changed on each operating system:
   - For HP-UX operating systems: /sbin/init.d/Tivoli
   - For Solaris operating environments: /etc/rc3.d/S99Tivoli
5. Shut down the object dispatcher by entering the following command:
   ```
   odadmin shutdown all
   ```
6. Restart the object dispatcher on the Tivoli server by entering the following command:
   ```
   odadmin reexec 1
   ```

7. Restart the object dispatcher on the managed nodes by entering the following command:

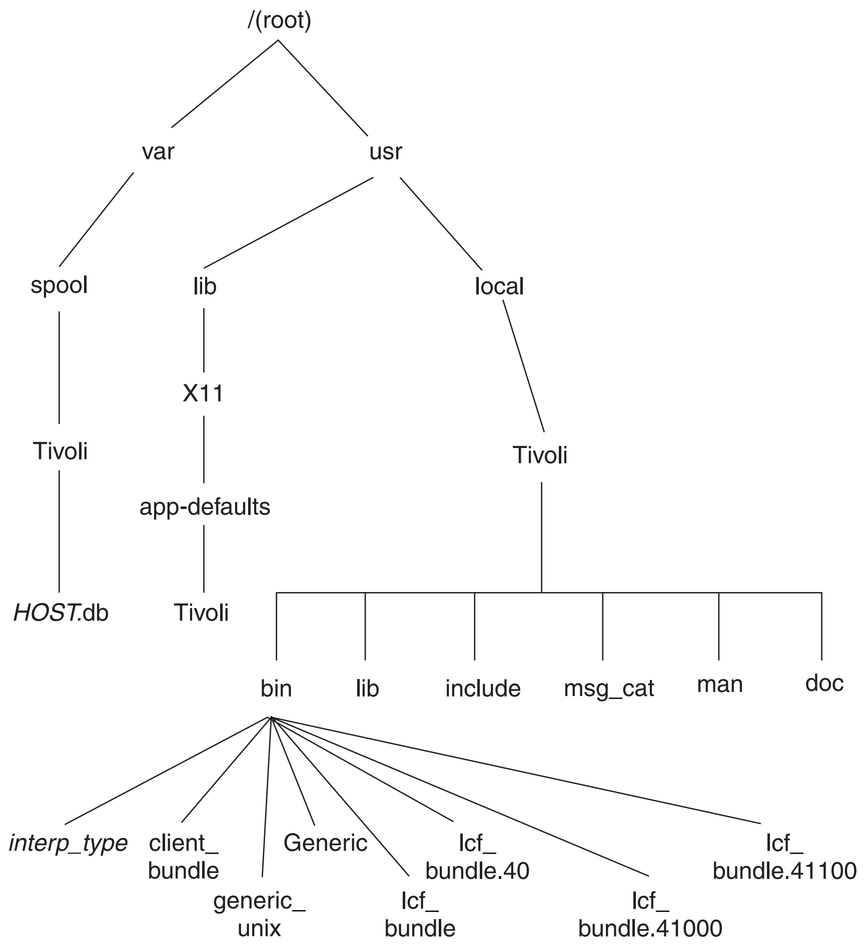
```
odadmin reexec clients
```

Directory Structure

The Tivoli Management Framework installation process installs the following files on UNIX Tivoli servers or managed nodes:

- Libraries
- Binaries
- Command reference pages (manual pages)
- X11 resource files
- Message catalogs
- Databases (server and client)
- Environment setup files

You can install these directories in the root directory (/) or in an installation directory that you create. These files are installed in the directories shown in the following figure.



/var/spool/Tivoli/*host*.db
: Contains the database files for the Tivoli servers or managed nodes. These files should always be installed locally on each host.

/usr/lib/X11/app-defaults
: Contains the X11 resource files.

/usr/local/Tivoli/bin
: Contains the following Tivoli Enterprise binary directories:

  client_bundle
  : Contains files necessary for bootstrap client installation. This directory is installed on the Tivoli server only.

  generic
  : Contains Web pages and language code sets.

  generic_unix
  : Contains generic binaries for all supported platforms.

  lcf_bundle
  : Contains the endpoint binaries required for each supported platform running Tivoli Enterprise software released prior to Tivoli Management Framework, Version 3.6.

  lcf_bundle.40
  : Contains the endpoint binaries required for each supported platform running Tivoli Management Framework, Version 3.7.*x*.

  lcf_bundle.41000
  : Contains the endpoint binaries required for each supported platform running Tivoli Management Framework, Version 4.1.

  lcf_bundle.41100
  : Contains the endpoint binaries required for each supported platform running Tivoli Management Framework, Version 4.3.1.

  interp_type
  : Indicates the interpreter type. A separate subdirectory is created for each interpreter type that you have in your Tivoli region. The contents of these directories depend on the Tivoli Enterprise products that you have installed.

/usr/local/Tivoli/lib
: Contains a subdirectory for each interpreter type that you have installed.

  interp_type
  : These subdirectories contain the operating system-specific Tivoli libraries.

/usr/local/Tivoli/include
: Contains the header files used by Tivoli Application Development Environment (ADE).

/usr/local/Tivoli/msg_cat
: Contains the Tivoli message catalogs.

/usr/local/Tivoli/man
: Contains the UNIX manual pages. These files can be installed on a file server accessible by the Tivoli server and its clients.

/usr/local/Tivoli/doc
: Contains the most recent Tivoli documentation.

The Tivoli environment setup files are created in the /etc/Tivoli directory. This directory contains the following subdirectories:

bin
: Contains the Perl binaries required for many Tivoli operations.

lib
: Contains the Perl language library required for many Tivoli operations.

tll.conf
: Contains configuration information for the Task Library Language.

The following files are created during installation in the /etc/Tivoli directory:

oserv.rc
: Contains scripts to start or stop the object dispatcher.

setup_env.sh
: Sets up the Tivoli system variables if you are using a Bourne-type shell. Run this script before you start the Tivoli desktop.

setup_env.csh
: Sets up the Tivoli system variables if you are using C shell. Run this script before you start the Tivoli desktop.

Files modified

The following system startup files are added or modified during the installation of a UNIX managed node or Tivoli server.

Operating system	Files modified
AIX	/etc/inittab /etc/services /etc/inetd.conf
HP-UX	/sbin/rc0.d/K100Tivoli /sbin/rc1.d/K100Tivoli /sbin/rc2.d/K100Tivoli /sbin/rc3.d/S500Tivoli
Solaris	/etc/init.d/Tivoli /etc/rc3.d/S99Tivoli /etc/rc2.d/K50Tivoli /etc/rc0.d/K50Tivoli /etc/rc1.d/K50Tivoli /etc/services /etc/inetd.conf
SuSE Linux	/etc/rc.d/rc2.d/S99Tivoli /etc/rc.d/rc3.d/S99Tivoli /etc/rc.d/rc4.d/S99Tivoli /etc/rc.d/rc5.d/S99Tivoli /etc/rc.d/rc2.d/K10Tivoli /etc/rc.d/rc3.d/K10Tivoli /etc/rc.d/rc4.d/K10Tivoli /etc/rc.d/rc5.d/K10Tivoli
Other types of Linux	/etc/rc.d/rc2.d/S99Tivoli /etc/rc.d/rc3.d/S99Tivoli /etc/rc.d/rc4.d/S99Tivoli /etc/rc.d/rc5.d/S99Tivoli /etc/rc.d/rc0.d/K10Tivoli /etc/rc.d/rc1.d/K10Tivoli /etc/rc.d/rc6.d/K10Tivoli

System variables

The following system variables are set when you run either of the setup scripts contained in the /etc/Tivoli directory. If you have previously set system variables, the Tivoli setup program overrides these variables: BINDIR, DBDIR, INTERP, LIBDIR, o_dispatch, EtcTivoli, TMR, and WLOCALHOST. For LD_LIBRARY_PATH, LIBPATH, SHLIB_PATH, MANPATH, NLSPATH, and PATH, the Tivoli settings are added to your existing variable.

System variable	Default setting
BINDIR	`/usr/local/Tivoli/bin/`*`interp_type`*
DBDIR	`/var/spool/Tivoli/`*`host`*`.db`
INTERP	*interp_type* Refer to the *Tivoli Management Framework Release Notes* for a complete list of interpreter types.
`LD_LIBRARY_PATH` `LIBPATH` (AIX only) `SHLIB_PATH` (HP-UX only)	`"$LIBDIR":/usr/lib:/usr/ucblib`
LIBDIR	`/usr/local/Tivoli/lib/`*`interp_type`*
MANPATH	`/usr/local/Tivoli/man/`*`interp_type`*`:$MANPATH`
NLSPATH	`/usr/local/Tivoli/msg_cat/%L/%N.cat`
PATH	`"$BINDIR/bin:$BINDIR/ADE:$BINDIR/AEF` `:$BINDIR/kerberos:$PATH"`
XKEYSYMDB	`/usr/lib/X11/XKeysymDB`
XUSERFILESEARCHPATH	/usr/lib/X11/app-defaults/%L/%N.cat
o_dispatch	`94`
TISDIR	`$BINDIR/../generic`
TMR	*region_number*
WLOCALHOST	Host name to listen on

Windows Tivoli servers and managed nodes

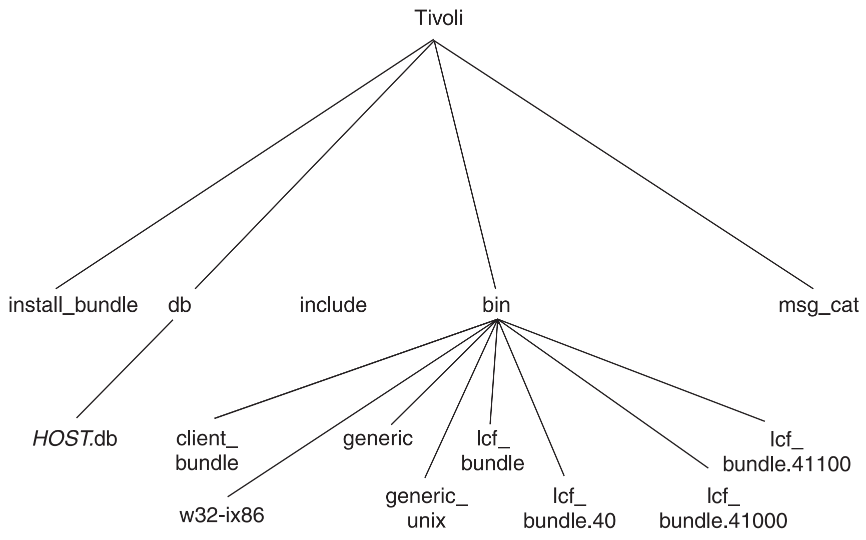
The following sections detail the directory structure created and the system variables defined when you install Tivoli Management Framework on a Microsoft Windows Tivoli server or managed node.

Directory structure

When you install Tivoli servers and managed nodes on Windows operating systems, the installation process creates several directories that contain the following files:

- Libraries
- Binaries
- Header files
- Message catalogs
- Databases

These files are installed in the directories shown in the following figure.



\Program Files\Tivoli\db
: Contains the database files for the Tivoli server and managed nodes. These files must be installed on a local Windows NT File System (NTFS).

\Program Files\Tivoli\include
: Contains the header files used by Tivoli Application Development Environment (ADE).

\Program Files\Tivoli\bin
: Contains the following Tivoli binary directories:

  client_bundle
  : Contains files necessary for bootstrap managed node installation. This directory is installed on the Tivoli server only.

  w32-ix86
  : Contains the Tivoli binaries used by Windows Tivoli servers and managed nodes. The content of this directory depends on the Tivoli Enterprise products that you have installed.

  generic
  : Contains Web pages and language code sets.

  generic_unix
  : Contains generic binaries for all supported platforms.

  lcf_bundle
  : Contains the endpoint binaries required for each supported platform running Tivoli Enterprise software released prior to Tivoli Management Framework, Version 3.6.

  lcf_bundle.40
  : Contains the endpoint binaries required for each supported platform running Tivoli Management Framework, Version 3.7.*x*.

  lcf_bundle.41000
  : Contains the endpoint binaries required for each supported platform running Tivoli Management Framework, Version 4.1.

lcf_bundle.41100
: Contains the endpoint binaries required for each supported platform running Tivoli Management Framework, Version 4.3.1.

\Program Files\Tivoli\lib
: Contains the static Tivoli libraries required for Tivoli Application Development Environment (ADE). Shared libraries for Windows operating systems are located in the \Tivoli\bin directory.

\Program Files\Tivoli\msg_cat
: Contains the Tivoli message catalogs.

The Tivoli environment setup files are created in the %SystemRoot%\system32\drivers\etc\Tivoli directory. This directory contains the following subdirectories:

tll.conf
: Contains configuration information for the Task Library Language.

The following files are created in the %SystemRoot%\system32\drivers\etc\Tivoli directory during installation:

setup_env.sh
: Sets up the Tivoli system variables if you are using a Bourne-type shell. Run this script before you start the Tivoli desktop.

setup_env.bat
: Sets up the Tivoli system variables if you are using the DOS shell. Run this script before you start the Tivoli desktop.

Registry variables

When you install Tivoli Management Framework on Windows operating systems, the appropriate system variables and directory paths are added to the Windows registry.

The HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Platform key contains the value of localhost. This value identifies the preferred alias for the local host and is set using the **wlocalhost** command. The localhost key will not exist until the **wlocalhost** command is run.

System variables

The following system variables are set when you run either of the setup scripts contained in the %SystemRoot%\system32\drivers\etc\Tivoli directory. If you have previously set these system variables, Tivoli Management Framework overrides the variables except for PATH. For PATH, the settings are added to the existing variable.

For systems running Windows 2000 and beyond, the default object dispatcher setting for the TMP and TEMP variables, %DBDIR%\tmp, is overwritten with the system values for these variables. This is because, by default, these variables are included in the system environment set, which is not the case for Windows NT systems.

The system settings for TMP and TEMP are unrelated to the values the object dispatcher uses for these variables. The object dispatcher uses %DBDIR%\tmp by default, unless it is overridden by a value for the TMP variable placed in the object dispatcher environment and specified using the **odadmin environ** command, as in the following example:

1. Enter the following:
   ```
   odadmin environ get > env
   ```
2. Add the following statement to env:
   ```
   TMP=temporary_directory
   ```
   where *temporary_directory* is the fully qualified path to the desired temporary directory.
3. Enter the following:
   ```
   odadmin environ set < env
   ```
4. Recycle the object dispatcher.

System variable	Default setting
BINDIR	`%SystemDrive%\Program Files\Tivoli\bin\w32-ix86`
DBDIR	`%SystemDrive%\Program Files\Tivoli\db\`*`host`*`.db`
INTERP	`w32-ix86`
NLSPATH	`%SystemDrive%\Program Files\Tivoli\msg_cat\%L\%N.cat`
o_dispatch	`94`
PATH	`"%BINDIR%\bin;%BINDIR%\tools; %BINDIR%\ADE;%BINDIR%\AEF; %PATH%"`
PERLLIB	`%BINDIR%\tools\lib\perl`
TEMP	`%DBDIR%\tmp`
TISDIR	`%BINDIR%\..\generic`
TMP	`%DBDIR%\tmp`
TMR	*region_number*
WLOCALHOST	Host name to listen on

RIM host

You can set a timer on the RIM agent to throw an exception and exit if the RIM agent does not receive communication from the client application for a specified timeout period. By default, the timer is not active and the RIM agent waits indefinitely for the next command from the client application. This timer can be activated and configured by using the **odadmin** environment variable **RIM_IOM_TIMEOUT**. The value of this variable is the maximum amount of time in hours that the agent waits to receive a command from the client.

The variable **RIM_IOM_TIMEOUT** can be set when you install a RIM host. If you have previously set this system variable, Tivoli Management Framework does not override the variable.

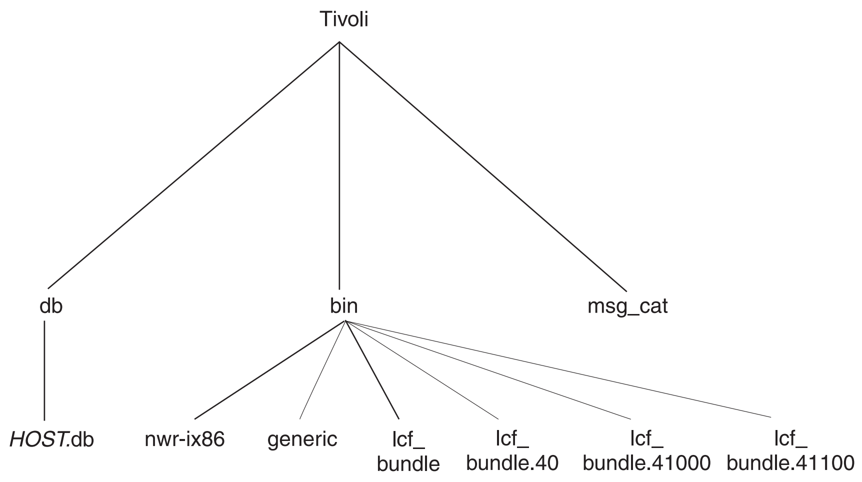
System variable	Value	Is the timer active?	Agent behavior
RIM_IOM_TIMEOUT	Not set	No	The agent waits indefinitely for the next command.
	≤ 0	No	The agent waits indefinitely for the next command.
	≥ 1	Yes	The agent waits up to the number of hours specified for the next command. The maximum number of hours that the agent waits is 48. If the number of hours specified is greater than 48, the agent waits up to 48 hours.

NetWare gateways directory structure

When you install a gateway on a Novell NetWare operating system, the installation process creates several directories that contain the following files:

- Binaries
- Message catalogs
- Databases

These files are installed in the directories shown in the following figure.



\Tivoli\db
: Contains the database files for the managed nodes.

\Tivoli\bin
: Contains the following Tivoli binary directories:

  nwr-ix86
  : Contains the Tivoli binaries used by NetWare managed nodes. The content of this directory depends on the Tivoli Enterprise products that you have installed.

  generic
  : Contains Web pages and language code sets.

lcf_bundle
: Contains the endpoint binaries required for each supported platform running Tivoli Enterprise software released prior to Tivoli Management Framework, Version 3.6. These binaries support product compatibility with Tivoli Management Framework, Version 4.3.1.

lcf_bundle.40
: Contains the endpoint binaries required for each supported platform running Tivoli Management Framework, Version 3.7.*x*.

lcf_bundle.41000
: Contains the endpoint binaries required for each supported platform running Tivoli Management Framework, Version 4.1.

lcf_bundle.41100
: Contains the endpoint binaries required for each supported platform running Tivoli Management Framework, Version 4.3.1.

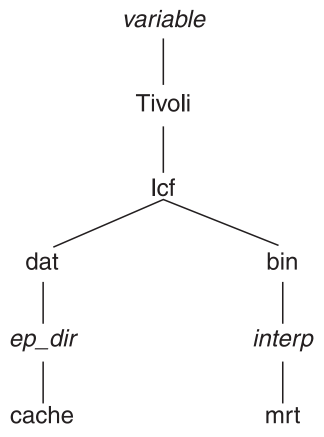
\Tivoli\msg_cat
: Contains the Tivoli message catalogs.

Endpoints

The following sections detail the directory structure created and the system variables defined when you install a Tivoli endpoint on supported UNIX or Windows operating systems.

Default directory structure

The default directory structure created when you install a Tivoli endpoint is shown in the following figure.



variable
: Drive, directory, or volume, which varies depending on the platform where the endpoint is installed:

UNIX operating systems
: /opt

Windows operating systems
: %SystemDrive%\Program Files

OS/2 operating systems
%BootDrive%

Tivoli\lcf\dat\ep_dir
Endpoint working directory, which contains the following directories and files, among others:

cache	Contains the binaries and files used by the endpoint to execute Tivoli Management Framework operations. These binaries and files are downloaded from the assigned gateway as needed by the endpoint.
last.cfg	Contains the latest configuration information used to start the endpoint.
lcf.dat	Contains login information.
lcfd.log	Contains log messages generated by the endpoint.
lcfd.sh	Contains the script file for starting the endpoint. (UNIX only)
lcfd.st	Contains the status of the endpoint.

Tivoli\lcf\bin\interp\mrt
Contains the platform-specific binaries and files needed to start, stop, and manage the endpoint.

Files modified

The following system startup files are added or modified during the installation of a UNIX endpoint.

Operating system	Files modified
AIX	/etc/rc.tma1 /etc/inittab.before.tma1
HP-UX	/sbin/init.d/lcf1.sh /sbin/rc0.d/K100Tivoli_lcf1 /sbin/rc1.d/K100Tivoli_lcf1 /sbin/rc2.d/K100Tivoli_lcf1 /sbin/rc3.d/S500Tivoli_lcf1
Solaris	/etc/init.d/lcf1.rc /etc/rc0.d/K50Tivoli_lcf1 /etc/rc1.d/K50Tivoli_lcf1 /etc/rc2.d/K50Tivoli_lcf1 /etc/rc3.d/S99Tivoli_lcf1
SuSE Linux	/etc/rc.d/rc2.d/S99Tivoli_lcf1 /etc/rc.d/rc3.d/S99Tivoli_lcf1 /etc/rc.d/rc4.d/S99Tivoli_lcf1 /etc/rc.d/rc5.d/S99Tivoli_lcf1 /etc/rc.d/rc2.d/K10Tivoli_lcf1 /etc/rc.d/rc3.d/K10Tivoli_lcf1 /etc/rc.d/rc4.d/K10Tivoli_lcf1 /etc/rc.d/rc5.d/K10Tivoli_lcf1

Operating system	Files modified
Other types of Linux	/etc/rc.d/rc2.d/S99Tivoli_lcf1 /etc/rc.d/rc3.d/S99Tivoli_lcf1 /etc/rc.d/rc4.d/S99Tivoli_lcf1 /etc/rc.d/rc5.d/S99Tivoli_lcf1 /etc/rc.d/rc0.d/K10Tivoli_lcf1 /etc/rc.d/rc1.d/K10Tivoli_lcf1 /etc/rc.d/rc6.d/K10Tivoli_lcf1

System variables

The following system variables can be set after you install an endpoint.

System variable	Default setting
LCFROOT	Endpoint working directory
LCF_DATDIR	`$LCFROOT/dat/`*`ep_dir`*
TISDIR	`$LCFROOT/dat/`*`ep_dir`*
INTERP	*interp_type*

To use these system variables, you must run the environment setup scripts for the endpoint. For details on running these scripts, refer to "Setting system variables for endpoints."

Setting system variables for endpoints

For UNIX operating systems, the installation process creates the following setup scripts:

- /etc/Tivoli/lcf/*ep_dir*/lcf_env.csh
- /etc/Tivoli/lcf/*ep_dir*/lcf_env.sh

You can change your login initialization procedure to use the appropriate setup file so that the necessary environment variables and search paths are set to allow you to start the endpoint.

For example, you might add the following to your initialization procedure:

For C shell (csh):

```
if ( -f /etc/Tivoli/lcf/1/lcf_env.csh ) then
   source /etc/Tivoli/lcf/1/lcf_env.csh
endif
```

For Bourne or Korn shell (sh or ksh):

```
if [ -f /etc/Tivoli/lcf/1/lcf_env.sh ]; then
   . /etc/Tivoli/lcf/1/lcf_env.sh
fi
```

For Windows operating systems, except Windows 98, the installation process creates the following setup scripts:

- %SystemRoot%\Tivoli\lcf\ep_dir\lcf_env.cmd
- %SystemRoot%\Tivoli\lcf\ep_dir\lcf_env.sh

For Windows 98 operating systems, the installation process creates the following setup script:

- %SystemRoot%\Tivoli\lcf\ep_dir\lcf_env.bat

Run one of these scripts to initialize the Tivoli environment variables for the endpoint.

Tivoli Desktop for Windows installations

The following sections detail the directory structure created and the system variables defined when you install Tivoli Desktop for Windows. This product can be installed on supported Windows and OS/2 operating systems.

Note: You do not need to install Tivoli Desktop for Windows on UNIX operating systems.

Directory Structure

The default directory structure for Tivoli Desktop for Windows on supported Windows operating systems is the \Program Files\Tivoli\Desktop directory. The default directory structure for Tivoli Desktop for Windows on supported OS/2 operating systems is the \tivoli\desktop directory.

Registry contents

Tivoli Desktop for Windows stores three values in the Windows registry. The following table contains the values that are stored under the HKEY_CURRENT_USER\Software\Tivoli\Desktop\4.1.1 registry key.

Value name	**Data type**	**Description**
host	REG_SZ	Specifies the name of the host system.
port	REG_SZ	Specifies the port number used to connect to the host.
user	REG_SZ	Specifies the name of the user to connect to the host.

These registry values are documented for your information. You can modify all of these settings from the Tivoli command line, although you should exercise caution when doing so. You should not need to directly modify these registry values.

Note: You cannot view these values while you are running an OS/2 operating system.

Appendix. 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.

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.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

AIX, AS/400, DB2, IBM, MVS, OS/2, OS/400, S/390, Tivoli, Tivoli Enterprise, Tivoli Enterprise Console, TME, TME 10, WebSphere, WIN-OS/2, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

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

ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see http://www.intel.com/sites/corporate/tradmarx.htm.

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

Glossary

A

authorization roles. A role assigned to Tivoli administrators to enable them to perform their assigned systems management tasks. A role can be granted over the entire Tivoli region or over a specific set of resources, such as those contained in a policy region. Examples of authorization roles include **super**, **senior**, **admin**, and **user**.

B

bulletin board. The mechanism by which the Tivoli Management Framework and Tivoli applications communicate with Tivoli administrators. The bulletin board collects notices in notice groups. Administrators can access the bulletin board from the Tivoli desktop. The bulletin board is an audit trail for important operations that the administrators perform.

C

Configuration Change Management System (CCMS). In a Tivoli environment, a data store of profiles that contain confugration data that is used by system management applications to make configuration changes on groups of systems. See also profile manager.

collection. A container that provides a single view of related resources.

configuration repository. The relational database that contains information collected or generated by inventory or software distribution operations.

D

default policy. A set of resource property values assigned to a resource when the resource is created. Contrast with validation policy.

Distribution Status console. An MDist 2 interface provided by Tivoli Management Framework that enables administrators to monitor and control distributions across a network. See also MDist 2.

downcall. A method call from the gateway to an endpoint. Contrast with upcall.

E

endpoint. In a Tivoli environment, the computer system that is the ultimate target for most Tivoli operations. Contrast with managed node.

endpoint list. In a Tivoli environment, A list of all endpoints in a Tivoli region with their assigned gateways. See endpoint manager.

endpoint manager. In a Tivoli environment, a service that runs on the Tivoli server, assigns endpoints to gateways, and maintains the endpoint list.

endpoint method. In a Tivoli environment, a method that runs on an endpoint as the result of a request from another managed resource. Results of the method are forwarded to the gateway and then to the calling managed resource.

F

fanout. In communications, the process of creating copies of a distribution to be delivered locally or sent over the network.

G

gateway. Software that provides services between endpoints and the rest of the Tivoli environment.

I

instance. A single occurrence of a resource.

J

job. In a Tivoli environment, a resource consisting of a task and its preconfigured execution parameters. Among other things, the execution parameters specify the set of hosts on which the job is to run.

L

lcfd. The Tivoli service that is used by an endpoint to communicate with a gateway. Contrast with oserv.

M

managed node. In a Tivoli environment, a computer system where Tivoli Management Framework is installed. Constrast with endpoint.

MDist. A multiplexed distribution service provided by Tivoli Management Framework that enables efficient transfer of data to multiple targets. Another distribution service, MDist 2, provides additional management features. See also MDist 2.

MDist 2. A multiplexed distribution service provided by Tivoli Management Framework that enables efficient transfer of data to multiple targets. Administrators can monitor and control a distribution throughout its life cycle. Another multiplexed distribution service, MDist, lacks these management features. See also Distribution Status console.

member. The contents of a collection. See also collection.

multiplexed distribution. The mechanism used by Tivoli Enterprise applications to transfer data to multiple targets. Tivoli Management Framework provides two multiplexed distribution services, MDist and MDist 2. See also MDist and MDist 2.

N

name registry. See Tivoli name registry.

notice. In a Tivoli environment, a message generated by a systems management operation that contains information about an event or the status of an application. Notices are stored in notice groups. See also bulletin board.

notice groups. In a Tivoli environment, an application- or operation-specific container that stores and displays notices pertaining to specific Tivoli functions. The Tivoli bulletin board is comprised of notice groups. A Tivoli administrator can subscribe to one or more notice groups; the a bulletin board contains only the notices that reside in a notice group to which the administrator is subscribed. See also bulletin board and notice.

NT repeater. The first Windows managed node where Tivoli Remote Execution Service is installed. Using fanout, the NT repeater distributes Tivoli Remote Execution Service to all other Windows managed nodes during the installation process.

O

object path. In a Tivoli environment, an absolute or relative path to a Tivoli object, similar to a path in a file system.

object reference. In a Tivoli environment, the object identifier (OID) that is given to an object during its creation.

object request broker (ORB). In object-oriented programming, software that serves as an intermediary by transparently enabling objects to exchange requests and responses.

oserv. The Tivoli service that is used as the object request broker (ORB). This service runs on each Tivoli server and managed node. Contrast with lcfd.

P

policy. A set of rules that are applied to managed resources. A specific rule in a policy is referred to as a policy method.

policy region. A group of managed resources that share one or more common policies and which model the management and organizational structure of a network computing environment. Administrators use policy regions to group similar resources, to define access to the resources, and to associate rules for governing the resources.

policy subregion. In a Tivoli environment, a policy region created or residing in another policy region. When a policy subregion is created, it initially used the resource and policy properties of the parent policy region. A Tivoli administrator can later change or customize these properties to reflect the specific needs and differences of the subregion.

profile. In a Tivoli environment, a container for application-specific information about a particular type of resource. A Tivoli application specifies the template for its profiles, which includes information about the resources that can be managed by that Tivoli application.

profile manager. In a Tivoli environment, a container for profiles that links the profiles to a set of resources, called subscribers. Tivoli administrators use profile managers to organize and distribute profiles. A profile manager can operate in dataless mode or database mode. created in the context of a policy region and is a managed resource in that policy region. See also Configuration Change Management System (CCMS).

proxy endpoint. In a Tivoli environment, a representation of an entity (such as a network device or a host) that functions as a subscriber for profile distribution. The proxy endpoint is created on a managed node, which performs the proxy role during profile distribution. Multiple proxy endpoints can be created on the same managed node.

pull. An operation that initiates an action by requesting it of a resource.

push. An operation that sends information to other resources.

Q

query. In a Tivoli environment, a combination of statements that are used to search the configuration repository for systems that meet certain criteria. The query object is created within a query library. See also query library.

query facility. In a Tivoli environment, a facility that enables the use of SQL functions to access information in an RDBMS Interface Module (RIM) repository.

query library. In a Tivoli environment, a facility that provides a way to create and manage Tivoli queries. See also query.

R

RDBMS. See relational database management system (RDBMS).

RDBMS Interface Module (RIM). In Tivoli Management Framework, the module in the distributed object database that contains information about the installation of the relational database management system (RDBMS).

region. See Tivoli region.

registered name. In a Tivoli environment, the name by which a particular resource is registered with the name registry when the resource is created.

relational database management system (RDBMS). A collection of hardware and software that organizes and provides access to a relational database.

repeater. In a Tivoli environment, a managed node that is configured for multiplexed distribution. A repeater receives a single copy of data and distributes it to the next tier of clients.

resource. A hardware, software, or data entity that is managed by Tivoli software products.

resource type. In a Tivoli environment, one of the properties of a managed resource. Resource types are defined in the default policy for a policy region.

RIM. See RDBMS Interface Module (RIM).

RIM repository. In a Tivoli environment, the relational database that contains information that is collected or generated by Tivoli software products.Examples of a RIM repository include the configuration repository and the event database.

root administrator. In a Tivoli environment, the account for the initial Tivoli administrator that is created during the installation of Tivoli Management Framework. This account is the equivalent of the root user on UNIX operating systems and a member of the administrator group on Microsoft Windows systems.

S

subscriber. In a Tivoli environment, a resource that is subscribed to a profile manager.

subscription. In a Tivoli environment, the process of identifying the subscribers to which profiles are distributed.

subscription list. In a Tivoli environment, a list that identifies the subscribers to a profile manager. A profile manager can be included in a subscription list to subscribe several resources simultaneously rather than adding each resource individually.

T

TAP. See Tivoli Authentication Package.

task. In a Tivoli environment, the definition of an action that must be routinely performed on various targets throughout the network. A task defines the executable files to be run when the task is initiated, the authorization role required to execute the task, and the user or group name under which the task runs.

task library. In a Tivoli environment, a container in which a Tivoli administrator can create and store tasks and jobs.

Tivoli administrator. In a Tivoli environment, a system administrator that is identified by system account maps who is authorized to perform systems management tasks and manage policy regions in one or more networks.

Tivoli Application Development Environment (ADE). A toolkit that contains the complete application programming interface (API) for Tivoli Management Framework. This toolkit enables customers and Tivoli Partners to develop their own applications for a Tivoli environment.

Tivoli Application Extension Facility (AEF). A toolkit that enables customers to extend the capabilities of Tivoli applications. For example, they can add fields to a dialog, create custom attributes and methods for application resources, or create custom icons and bitmaps.

Tivoli Authentication Package. A dynamically linked library (DLL) installed by Tivoli Management Framework, that is capable of creating Windows security tokens for a different user context. These tokens can be used for accessing network resources or creating processes in a different user context.

Tivoli client. A client of a Tivoli server. See Tivoli management region client (managed node) and Tivoli server.

Tivoli desktop. In a Tivoli environment, the desktop that system administrators use to manage their network computing environments.

Tivoli Event Integration Facility (EIF). A toolkit that provides a simple application programming interface (API) to enable customers and Tivoli Partners to develop new event adapters that can forward events. A customer can also translate events from third-party or in-house applications.

Tivoli environment. The Tivoli applications, based upon Tivoli Management Framework, that are installed at a specific customer location and that address network computing management issues across many platforms.

Tivoli Management Framework. The base software required to run many Tivoli software applications. This software infrastructure enables the integration of systems management applications from Tivoli and the Tivoli Partners. Tivoli Management Framework includes the following components:

- Object request broker (oserv service)
- Distributed object database
- Basic administration functions
- Basic application services
- Basic desktop services, such as the graphical user interface (GUI)

Tivoli client (managed node). In a Tivoli environment, any computer system—except the Tivoli server—on which Tivoli Management Framework is installed. The object dispatcher (or oserv service) runs on each client, and each client maintains a local object database. See Tivoli server.

Tivoli name registry. In a Tivoli environment, the table that map names of managed resources to resource identifiers (and the corresponding information) with a Tivoli region.

Tivoli region. In a Tivoli environment, a Tivoli server and the set of clients it serves. An organization can have more than one region. A Tivoli region addresses the physical connectivity of resources whereas a policy region addresses the local organization of resources.

Tivoli Remote Execution Service. The service that enables a Tivoli environment to perform remote operations on machines. These operations include remotely installing clients, connecting Tivoli management regions, and starting the object request broker from a remote machine.

Tivoli server. In a Tivoli environment, the server for a specific Tivoli region that holds or references the complete set of Tivoli software, including the full object database.

transaction. A specific set of input data that triggers a specific process or job.

U

upcall. A method invocation from an endpoint to the gateway. Contrast with downcall.

user login map. A mapping that associates a single user login name with user accounts on various operating systems. User login maps enable Tivoli administrators to log in to the Tivoli environment and perform operations within the Tivoli environment with a single user login name, independent of system accounts used on the various operating systems.

V

validation policy. In a Tivoli environment, the policy that ensures that all resources in a policy region comply with the established policy for the policy region. A validation policy prevents Tivoli administrators from creating or modifying resources that do not conform to the policy of the policy region in which the resouces were created. Contrast with default policy.

virtual user. A user ID (UID) mapping set up in Tivoli Management Framework. A single UID can be mapped to different actual users on different types of architectures. For example, the virtual user $root_user can be mapped to root on UNIX operating systems and **Administrator** on Windows operating systems. See user login map.

Index

Part Number: CT2JSIE

Printed in USA

(1P) P/N: CT2JSIE

GC32-0803-02