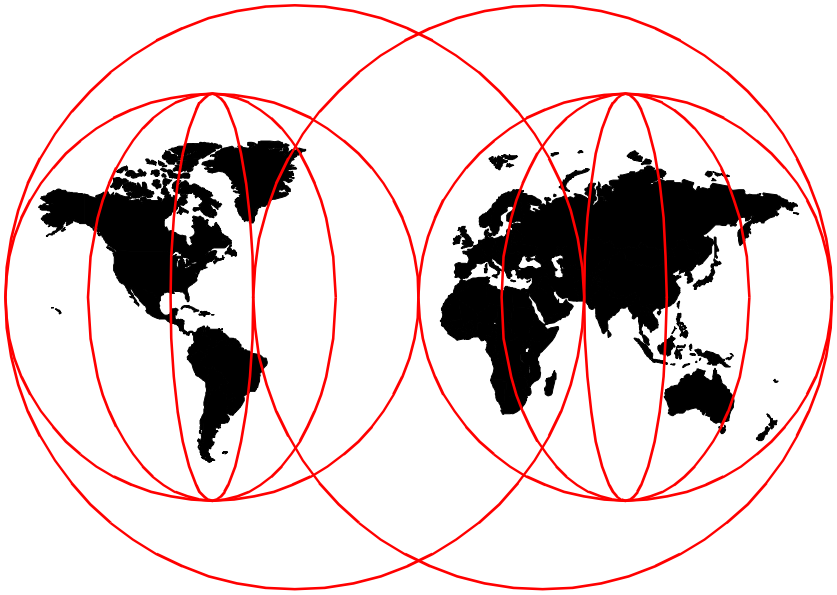
Tivoli Enterprise Performance Tuning Guide

Yoichiro Ishii, Chiara Castoldi, Jesus Del-Cura, Jesus Toledo Perez



International Technical Support Organization

www.redbooks.ibm.com

SG24-5392-00

SG24-5392-00

International Technical Support Organization

Tivoli Enterprise Performance Tuning Guide

December 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix A, “Special notices” on page 305.

First Edition (December 1999)

This edition was written between February and June 1999. The contents will mostly apply to the levels of the respective products current at that time, in particular 3.6 and 3.6.1. However, most of the information will be of use at both prior and later levels of the Tivoli Management Framework and applications. Where information is very specific to release level, a mention of that fact is included in the text.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. OSJB Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Figures

Tables

Preface

The Tivoli Enterprise Environment can be complex and can include a variety of Tivoli applications. Tuning this environment to provide the best possible performance for a particular application mix can be challenging. This project produces guidelines showing how to optimize the performance of typical Tivoli Enterprise installations. These guidelines are based on a Tivoli 3.6 Enterprise environment and include all the primary Tivoli management applications. In this redbook we develop a methodology that helps the customer or Tivoli professional to implement a tuning strategy to address the Tivoli Enterprise Environment.

The team that wrote this redbook

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

This project was designed and managed by:

Yoichiro Ishii is an Advisory I/T Specialist working as a project leader in the International Technical Support Organization Tivoli Group, Austin Center. He manages Tivoli projects and develops technical documents, including Redbooks on Tivoli Enterprise topics. He also teaches IBM classes worldwide on all areas of systems management, application management and network management. Before joining the ITSO in mid-1998, Yoichiro worked in IBM Japan as an I/T Specialist for systems and network management, consulting major IBM customers. In this role, he architected systems management design and solution for large-scale government projects. He can be reached at ishii@us.ibm.com.

The other authors of this redbook are:

Chiara Castoldi is an IT Specialist in IBM Vimercate, Italy. She is working in the Network Station Management department developing Tivoli Solutions for large environments to support Command Center and Help Desk sites. She holds a degree in Physics from the University of Study of Milan. Her areas of expertise include Tivoli core applications, AIX, Windows NT, and TCP/IP Protocol.

Jesus Del-Cura is an IT Specialist in Grupo BBV, Spain. He is working in the Operative Systems Technical Support department of BBV in Madrid. He is currently working with Tivoli core applications and OS/390 products. He holds a degree in Industrial Engineering from the Instituto Catolico de Artes e

Industrias (ICAI) de la Universidad Pontificia Comillas de Madrid. His areas of expertise include OS/390, Unix, Windows NT, and several products on these platforms.

Jesus Toledo Perez is an IT Specialist in Laboratorios Magneticos S.A. de C.V. in Mexico City. He is currently engaged in projects providing Tivoli Solutions for large customer environments. He holds a degree in Electronics and Communications Engineering from Universidad Tecnologica de Mexico, Campus Cuitlahuac. His areas of expertise include Tivoli, UNIX, WindowsNT, and Telecommunications.

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

Victoria Stevens
Tivoli Systems

Kathy Hebblethwaite
Tivoli Systems

Luca Loiodice
Tivoli Systems

Brian Vassberg
Tivoli Systems

Jeff Mills
Tivoli Systems

Rowland Reed
Tivoli Systems

Rafael Sanchez
Tivoli Systems

Sean Starke
Tivoli Systems

Linda Allen
Tivoli Systems

Christopher Doan
Tivoli Systems

Jerry Moffitt
Tivoli Systems

Jan Braddock
Tivoli Systems

Riccardo Colella
Tivoli Systems

Caroline Cooper
International Technical Support Organization, Austin Center

Stefan Uelpenich
International Technical Support Organization, Austin Center

John Owczarzak
International Technical Support Organization, Austin Center

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in “IBM Redbooks evaluation” on page 323 to the fax number shown on the form.
- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Introduction

Tivoli Management Framework and Tivoli Management Applications provide a set of common services and facilities to enable powerful systems applications. Version 3.2 of Tivoli introduced major new extensions to the framework's architecture. These new extensions included the Lightweight Client Framework (LCF) architecture.

Now, Version 3.6 of Tivoli has been released, and the applications take full advantage of the new architecture extensions. We now have a more powerful set of functions and services for distributed systems management. Version 3.6 of Tivoli is a testament to both stability and extensibility of the Tivoli product's architecture.

With Tivoli 3.6, most customer management environments have been getting larger and more complex. To manage these large-scale management environments efficiently, tuning Tivoli Management environments has become very important and almost mandatory. This redbook will provide guidelines showing how to optimize the performance and throughput of typical large-scale Tivoli Management environments.

In this chapter, we introduce the overview of Tivoli 3.6, the approaches of Tivoli performance tuning, and its strategies.

1.1 About this redbook

In general, performance tuning documents can be categorized as follows:

- Documents that explain how to improve performance in the specific environment, such as project documents for the customer.
- Documents that explain how to improve performance in most environments, such as manuals.

Some documents provide very detailed information including each parameter setting for the specific environment or configuration. On the other hand, some documents provide general information that is useful for most environments and configurations.

In this redbook, we will provide general information and techniques that will be useful to improve performance and throughput in your Tivoli Management environment. To classify the information about the performance tuning, this redbook should be useful for most customer environments. The following

figure (Figure 1) shows what kind of information we will introduce in this redbook.

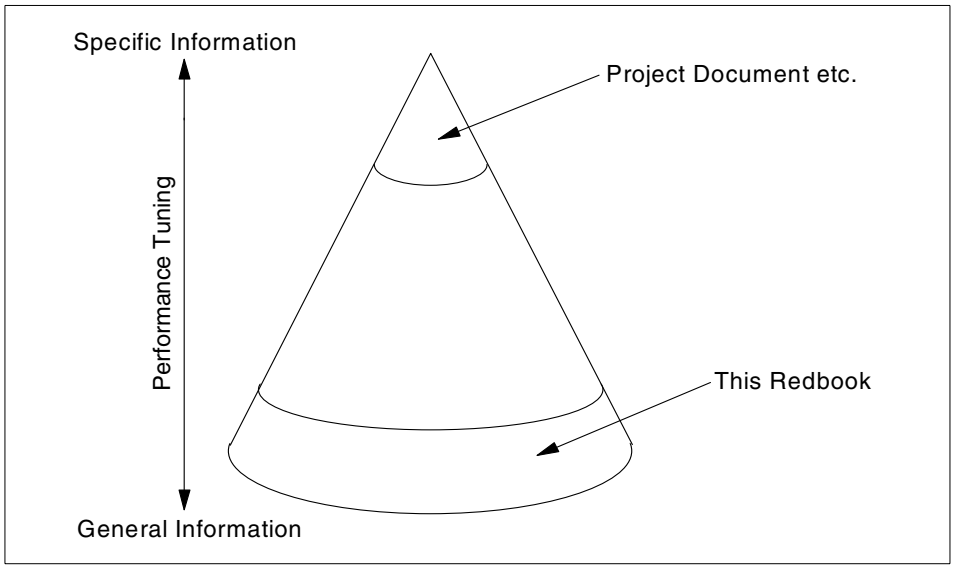


Figure 1. Information structure for Performance tuning

As you can see in Figure 1, the specific performance information is based on the general and basic performance information, and the specific performance information depends on each management (customer) environment. For example, the following factors can be dependencies:

- How you manage your environment.
- Which management applications your environment uses.
- How many managed resources your environment has.

Normally, the project documents that are created for the specific customer environment may include very detailed information about tuning performance. For example, each parameter configuration, very detailed design considerations, and so on. However, project documents are applicable only to specific customer environment.

To provide information that will be helpful for most customer's environments and their configurations, we will focus on environments with the following characteristics:

- Three-tiered management structure.
- A large-scale management environment.

- Each management site is connected via Wide Area Network (WAN).
- Version 3.6 of Tivoli products environment.

As shown in the following figure (Figure 2).

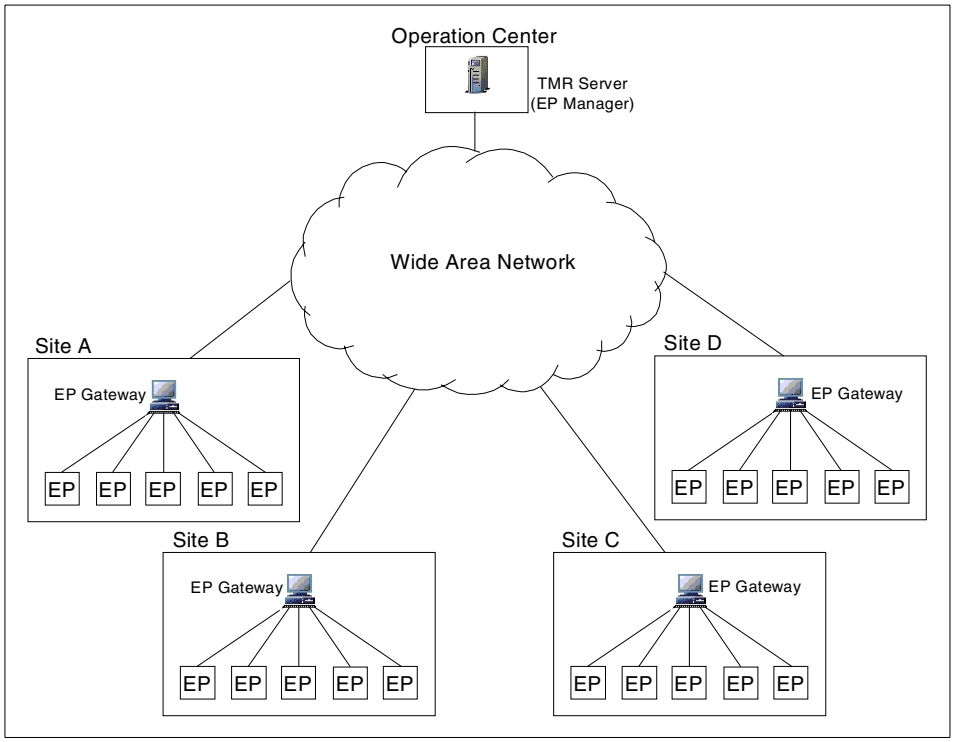


Figure 2. Sample Tivoli Management Environment

The Tivoli Management Application tuning information provided in this redbook is based on this environment.

Note

As mentioned before, we will focus on Tivoli 3.6 products and their three-tiered management structure. Therefore, we will not provide information about the following managed resources or components:

- Managed Nodes (except TMR server and Endpoint Gateway)
- PC Managed Nodes
- NetWare Managed Site
- Prior TMR structure (Version 3.2 of Tivoli or lower)

This redbook will provide the information about how to improve performance and throughput in your Tivoli management environment. Therefore, this redbook will not contain capacity planning information, for example, the appropriate sizing information for each hardware (TMR server, Endpoint Gateway), and so on.

1.2 Overview of Tivoli 3.6

Version 3.6 of the Tivoli Enterprise applications provides an extremely strategic set of applications. Tivoli 3.6 provides almost all features and services that were provided by the previous version of the Tivoli products and also extends these features. Tivoli 3.6 provides real extensibility for customers.

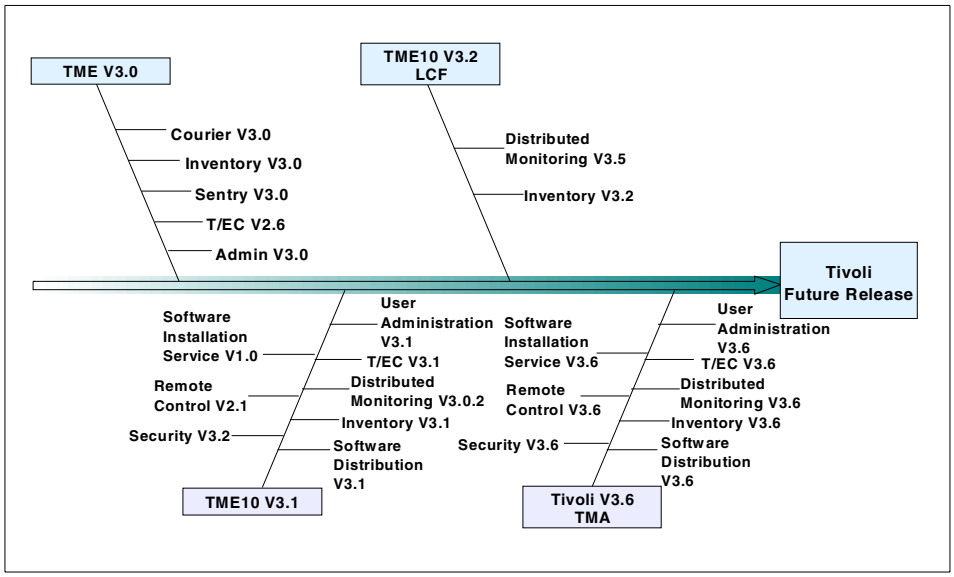


Figure 3. Tivoli Products History

1.2.1 Overview of Tivoli Management Agent

The most visible new feature in Tivoli 3.6 is the Tivoli Management Agent (TMA), previously called the Lightweight Client Framework (LCF) Endpoint. The TMA is an extension of the classic TME 10 Framework that increases scalability of TMRs while reducing the hardware and software requirements on the managed systems. The following sections introduce this new architecture and its main components.

1.2.1.1 Tivoli Management Agent basics

The Tivoli Management Agent (TMA) related extensions to the framework introduce three object types that represent system roles in a TMR.

- Endpoint (TMA)
- Endpoint Gateway
- Endpoint Manager

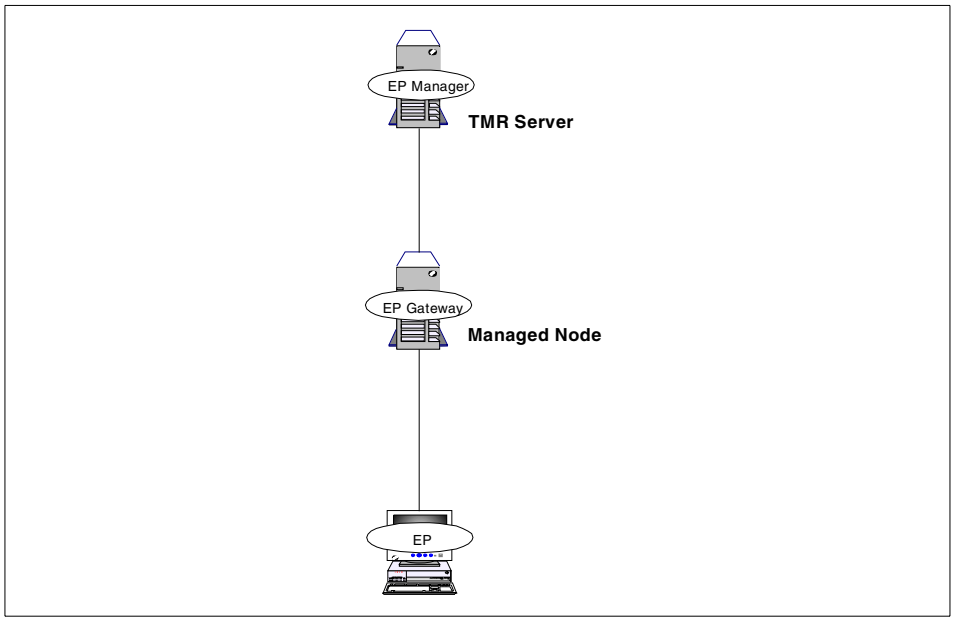


Figure 4. New object types in Tivoli Management Agent Environment

Although each of the above items logically represents a different system's role in the Tivoli environment, it should be noted that a single physical system can obtain more than one of the above object types. That is, one system could contain an Endpoint Manager, an Endpoint Gateway and an Endpoint. However, in most environments, Endpoint Gateways will reside on different systems than Endpoint Managers for performance reasons. You should not have more than one of each type on a single system from performance point of view. The following sections introduce an overview of each management object.

1.2.1.2 Endpoint (Tivoli Management Agent)

The Endpoints are managed-only systems taking advantage of the Lightweight Client Framework. You can gather required management information from thousands of Endpoint machines and remotely manage those machines with very little overhead. The Endpoints make relatively small demand on computer resources.

1.2.1.3 Endpoint Gateway

The Endpoint Gateway provides the primary interface between a set of Endpoints and the rest of the TMR. As part of this role, it also assumes some of the functions previously performed by the TMR server. By shifting a share

of the management process to the Endpoint Gateway, the TMR server is free to service more managed systems than with the previous versions of Tivoli. A single Endpoint Gateway can support communications with thousands of Endpoints. In Version 3.6 of the Tivoli Management Framework, an Endpoint Gateway must be installed on a Managed Node.

1.2.1.4 Endpoint Manager

The Endpoint Manager runs on the TMR server. The Endpoint Manager maintains the information related to known Endpoints and Endpoint Gateways that are running in a given TMR. The Endpoint Manager's primary role is to assign the Endpoint to the Endpoint Gateway when the Endpoint performs the initial login. The Endpoint Manager assigns an alternate Gateway in case the Endpoint needs to migrate. This might occur through explicit administrative action to migrate an Endpoint from one Endpoint Gateway to another or if an Endpoint attempts another login due to its assigned Gateway being unavailable for any reason.

1.2.2 Three-Tiered management structure

In Version 3.6 of Tivoli, the management topology is changed from a two-tiered structure to a three-tiered structure. The three-tiered management structure provides the following advantages:

- Off-loading the TMR server
- Increasing the number of systems a single TMR can manage
- Flexible configuration for Endpoints
- High availability for Endpoint operations
- Configuring the MDist repeater automatically

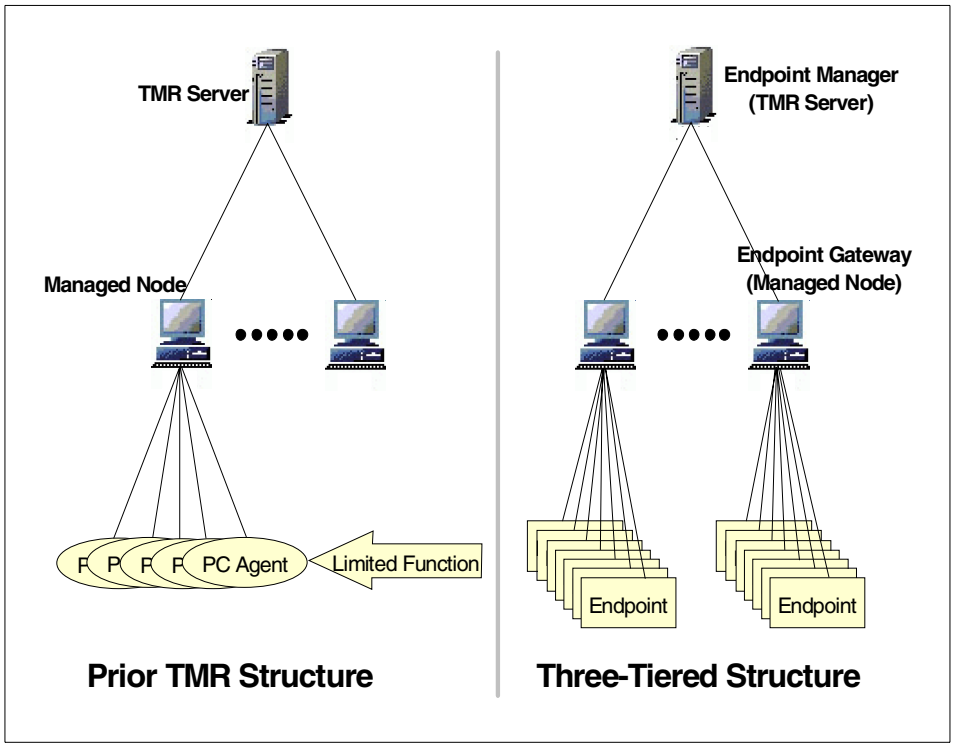


Figure 5. Three-tiered management structure

The three-tiered management structure is a natural concept for managing a large-scale environment. According to this concept, the Endpoint Gateway plays the role of mid-level manager. In other words, the Endpoint Gateway takes responsibility for managing the Endpoints which have logged into the Endpoint Gateway. Therefore, all requests from the Endpoints must be received by the Endpoint Gateway and most of them will be processed by the Endpoint Gateway instead of the TMR server. Only requests which the Endpoint Gateway cannot handle would be forwarded to the TMR Server where the TMR Server processes them.

Tivoli 3.6 provides improved functions, however, we have to understand them thoroughly if we want to use them efficiently and improve its performance. From this point of view, in the three-tiered environment, the following concepts are very important:

- TMR Design
- Management Resource Allocation (EP Manager, EP Gateway, Endpoint)
- Endpoint Configurations (Login interfaces information)

To improve the performance and throughput in your management environments, we will talk about the above subject in the following chapters.

1.2.3 Tivoli Management applications

Tivoli 3.6 provides many management applications that run on the Tivoli Management Framework. The following figure (Figure 6) shows the relationship between Tivoli Management Applications and Tivoli Management Framework.

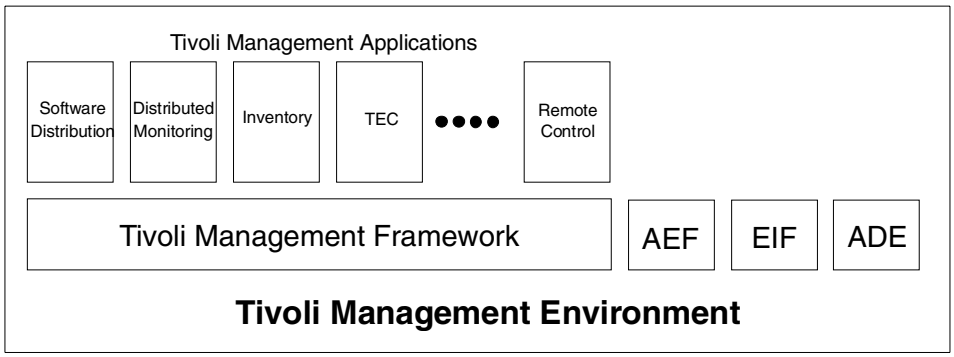


Figure 6. Tivoli Management Applications on Framework Service

In this book, we introduce how to improve performance and throughput for the following Tivoli products:

- Tivoli Management Framework
- Tivoli Software Distribution
- Tivoli Distributed Monitoring
- Tivoli Inventory
- Tivoli Enterprise Console
- Tivoli Remote Control

The Tivoli Management Framework provides many important basic services, such as the MDist feature for each application that runs on the framework. Therefore, improving the Tivoli Management Framework performance enhances the performance of each application that runs on the framework services.

1.2.4 Categorizing Tivoli Management Applications

When tuning Tivoli performance, understanding Tivoli Management Application behavior is important. The following table (Table 1) categorizes the Tivoli Management Applications that will be introduced in this book.

Table 1. Tivoli Management Application Categories

Downcall-oriented	Upcall-oriented	Other
Software Distribution Inventory Remote Control User Administration	Distributed Monitoring Event Adapter	Tivoli Enterprise Console

1.2.4.1 Downcall-oriented applications

In the Tivoli 3.6 Enterprise applications, most applications are downcall-oriented. The downcall-oriented application issues a downcall from the TMR server (Endpoint Manager) through the Endpoint Gateway to the Endpoint when it runs on a Endpoint. Basically, the behavior of the downcall-oriented application is less complicated than the behavior of the upcall-oriented application.

1.2.4.2 Upcall-oriented applications

In Tivoli 3.6 products, Distributed Monitoring (DM) is the most common upcall-oriented application. The upcall-oriented application issues an upcall from the Endpoint into the Endpoint Gateway when it runs on a Endpoint. Normally, the upcall-oriented application provides a daemon-like process on TMA, such as Sentry engine (dm_ep_engine.exe). This process runs on a TMA and issues an upcall if needed. Another example of an upcall-oriented application is Event Adapter.

1.2.4.3 Other applications

Tivoli Enterprise Console (TEC) is different from the applications above because TEC does not issue any downcall or upcall. TEC just waits for and receives the TEC events from its managed resource.

Note

TEC consists of some components. TEC Server and TEC Console do not issue any downcall and upcall, but TEC Adapter Configuration Facility issues a downcall, and TEC Event Adapter issues an upcall.

1.3 Overview of Tivoli Performance tuning

In this section, we introduce Tivoli performance tuning and throughput improvement. In all complex systems many factors effect performance interactively. It is the same in the Tivoli Management environments. The following figure (Figure 7) shows this situation.

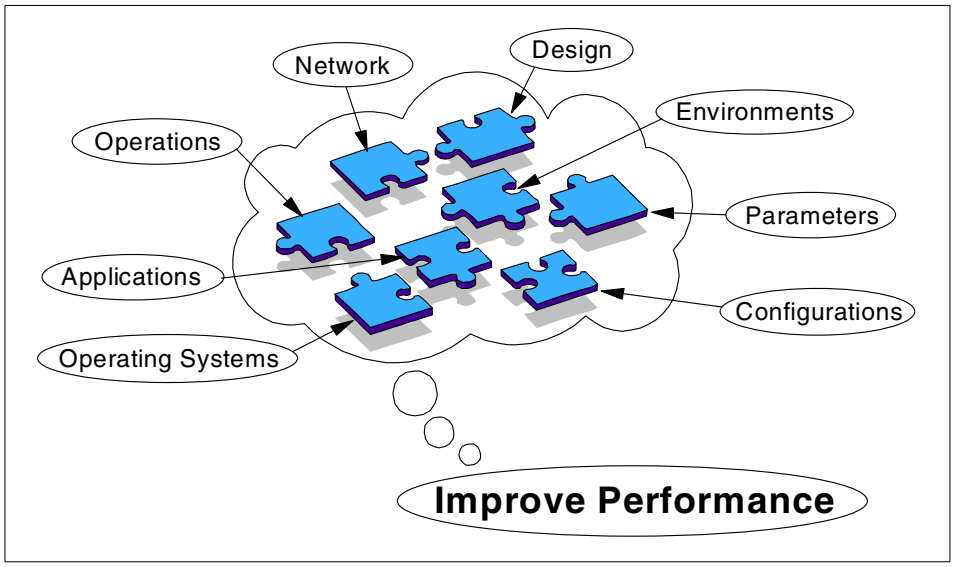


Figure 7. Performance improvement factors

To improve the performance in your Tivoli Management environments, you have to consider each factor carefully. The relationship between each factor is complicated. However, understanding these relationships is mandatory if you are going to attempt to detect some performance bottlenecks, resolve them, and improve performance.

Performance tuning involves many processes and these are divided into the two different phases as follows.

- Before Deployment
- After Deployment

The following table (Table 2) shows the performance tuning processes for each phase.

Table 2. Performance tuning processes

Phase	Performance Tuning Processes
Before Deployment	Designs Configurations
After Deployment	Detecting Bottlenecks Problem Determinations Configurations

If you understand how to improve performance before your deployment, you can plan and design your management environment more efficiently. Usually, tuning your management environment after deployment is more difficult than tuning the environment during its design phase. This is very important when you plan or design your management environment.

1.3.1 Performance and throughput

In this section, we introduce performance tuning and throughput improvement. We frequently use the terms, performance and throughput, for talking about the Tivoli performance issues.

1.3.1.1 Performance tuning

Performance tuning involves many things. Basically, performance tuning means reduction of the elapsed time that the process takes. The following figure (Figure 8) shows how performance tuning affects the process and its elapsed time.

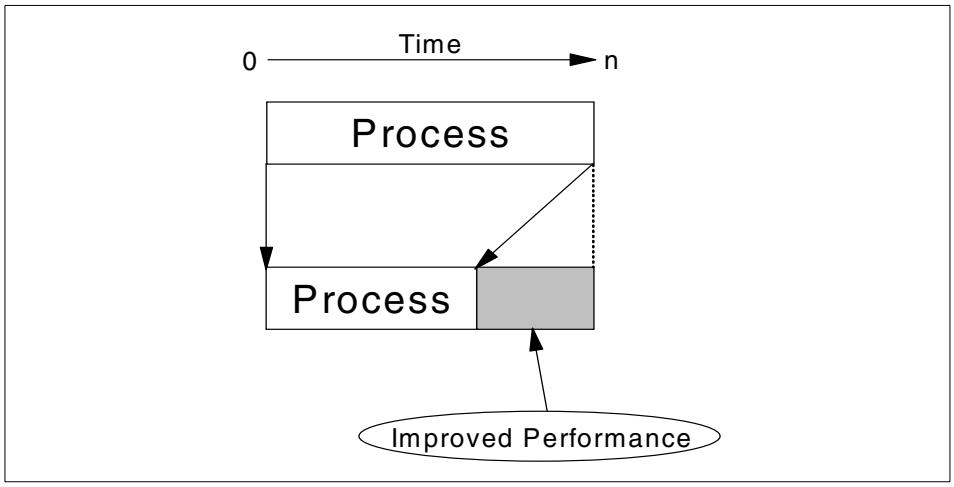


Figure 8. Tuning Performance

As you can see, after the tuning, the elapsed time of this process is improved. In a real Tivoli Management environment, many things may affect the elapsed time, for example, design, configurations, and so on.

1.3.1.2 Improving throughput

In general, throughput means how many processes can be completed within a specific period. To improve throughput, there are usually two different approaches:

- Improve the elapsed time of each process.
- Reduce the intervals between each process.

The following sections introduce each approach for throughput improvement.

Improving the elapsed time of each process

Improving the elapsed time of each process is the basic approach. To complete many processes within a specific period, we will attempt to improve the elapsed time of each process. As a result, we can complete many more processes within the specific period. The following figure (Figure 9) shows how this approach affects your environment.

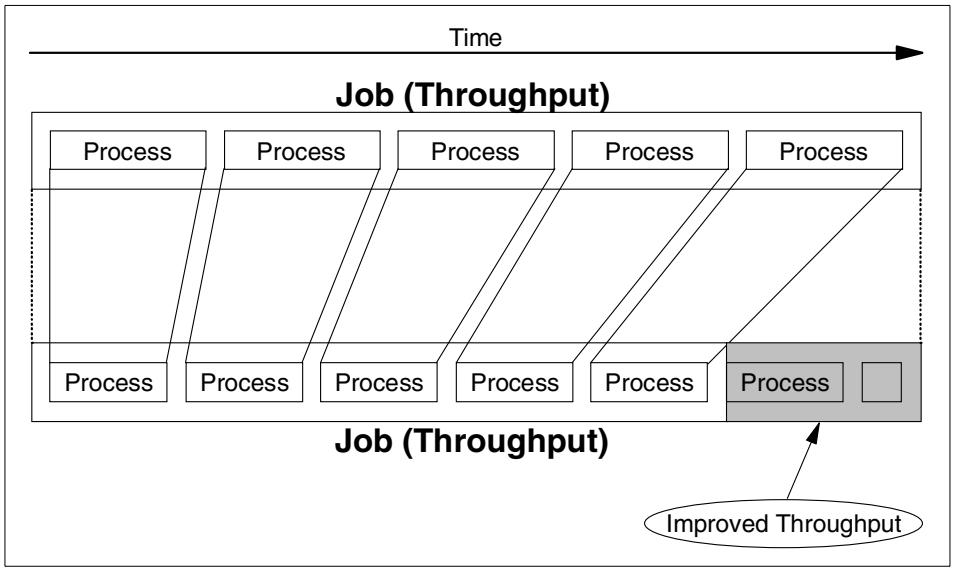


Figure 9. Example 1: Improving throughput

As you can see, this approach improves the throughput of the job by reducing the elapsed time of each process. In the real management environment, this approach (refer to the Figure 9) is performed by tuning each application's parameter or modifying the design or configurations.

Reducing the intervals between each process

In a real management environment, this approach could be easier than the approach that is introduced in the Figure 9. To complete many processes within the specific period, we attempt to reduce the interval time between each process. As a result, we can complete many more processes within the specific period. The following figure (Figure 10) shows how this approach improves the throughput in your environment.

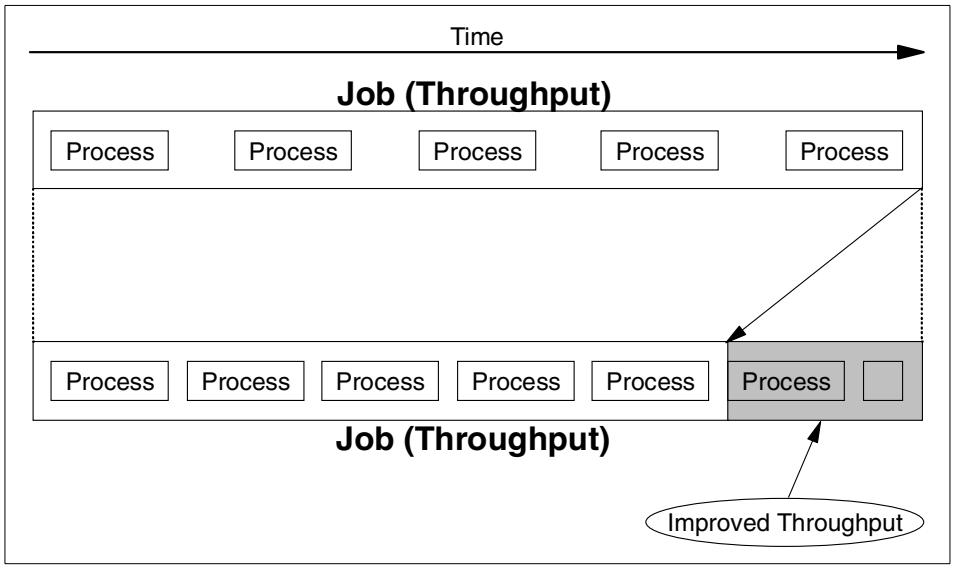


Figure 10. Example 2: Improving throughput

As you can see, this approach improves the throughput of the job by reducing the interval time between each process. In a real management environment, this approach is performed by tuning the operations or by customizing the configurations.

In the Tivoli Management environment, the situations are complicated and we need to improve the throughput of management operations by using both approaches. Some of Tivoli Management Applications may not provide many tuning parameters; so, the operations in your environment and design of your management environment are important in the Tivoli performance tuning. From this point of view, the second approach that is introduced in Figure 10 could be the practical approach in the real Tivoli Management environment.

1.4 Tivoli Performance tuning strategies

Performance is one of the key issues for efficient systems management operations. The three-tiered management structure is designed to improve the performance and throughput in a Tivoli Management environment. However, to optimize the advantages of the three-tiered management structure, you need to understand what Tivoli performance tuning is. In this section, we introduce an overview of Tivoli performance tuning in your Tivoli Management environment.

1.4.1 Understanding Tivoli Performance tuning strategies

To improve performance and throughput in the Tivoli Management environment, what should you do first? This is a very difficult question because it depends on many factors. The two most common ways for improving performance in the Tivoli Management environment include:

- Parameter Tuning (for each machine)
- Design Approach (for the whole management environment)

To understand parameter tuning, we can look at the following model (Figure 11), which consists of six layers.

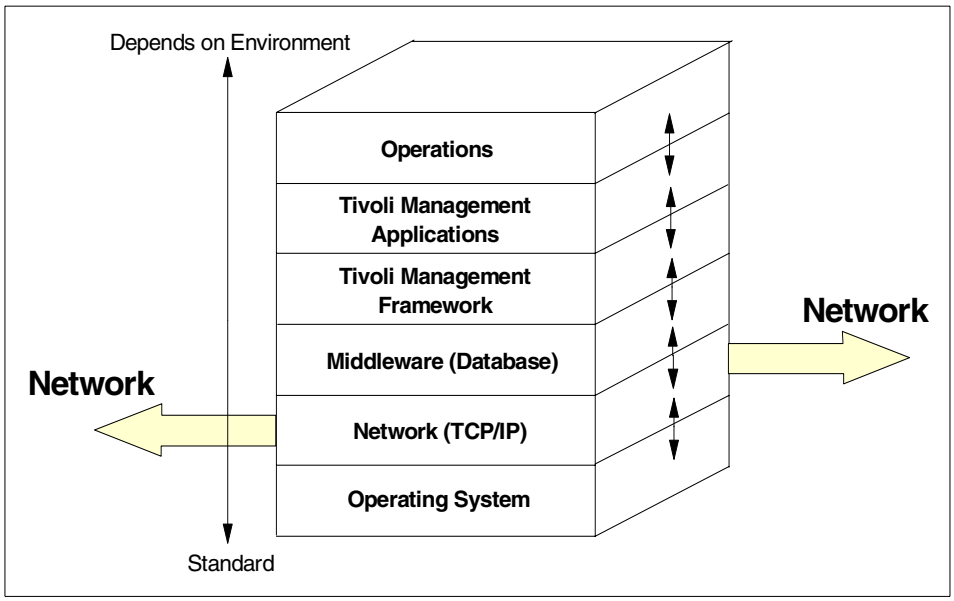


Figure 11. Performance Tuning modeling

As you can see, each layer depends on an upper or lower layer. In each layer, there are many ways to improve performance.

Note

In the example of the performance tuning model (Figure 11 on page 16), the lower layer component must affect the performance of its upper components. For example, when tuning the parameters of the Tivoli Management Application, if your network is not optimized and its performance is very low, the Tivoli Management Application performance will be very low as well.

Therefore, when you tune the Tivoli Management environment, we strongly recommend you to tune your environment from the lower layer to the upper layer. This order is very important in Tivoli performance tuning. Do not forget that the lower layer always affects the performance of its upper layer. In other words, you cannot expect the Tivoli performance to improve unless you tune the lower layer components first.

1.4.1.1 Operating system and network tuning

In the real management environment, of course, the relationship between each layer is complicated; so, it is not, in practice, as simple as the above model shown in the Figure 11. However, we can say that the Tivoli Management Framework runs on each operating system, for example AIX, Windows NT, or Windows 98, and uses the network protocol (TCP/IP) provided by each operating system. Tuning the operating system and its network is important for all distributed applications including Tivoli. For instance, for the RIM (RDBMS Interface Module) host performance tuning, the tuning of operating system parameters, network parameters, and RDBMS parameters is as important as tuning the RIM host parameters themselves.

The system and network performance affect the Tivoli Management Application performance greatly. Network performance is important in the Tivoli Management environment. Network tuning is more standardized than the upper layer tuning because so many applications depend on it, and the concepts are well understood by most network administrators.

1.4.1.2 Middleware (database) tuning

Tuning middleware (typically RDBMS) is also important for Tivoli Management Applications performance as well as operating system tuning or network tuning. Especially, in RDBMS implementation, the considerations are not only database tuning itself but also database server allocation. To improve Tivoli Management Application performance, pay particular attention to the tuning information provided by the database vendor.

1.4.1.3 Tivoli Management Framework tuning

After tuning the operating system, network and database to improve Tivoli Management Framework performance, we need to consider tuning the Endpoint Manager (TMR server), Endpoint Gateways, and Endpoints. This normally depends on each management environment. The following list summarizes some of the more common considerations:

- How many Endpoints a single Endpoint Gateway is managing.
- The hardware configurations of the Endpoint Manager and Endpoint Gateways (such as CPU or memory).
- The network design and speed.
- The TMR design.
- The Tivoli Management Applications running on the Tivoli Management environment (such as Software Distribution).
- Endpoint policy definitions.
- MDist repeater configurations.

If you use Software Distribution, the MDist repeater tuning is very important for its performance and throughput. You have to understand the above considerations and consider the most efficient way to improve Tivoli Management Framework performance.

1.4.1.4 Tivoli Management Application tuning

This depends on several environmental factors as well. In this redbook we will introduce how to configure or customize the following Tivoli Management core applications to improve their performance.

- Software Distribution
- Distributed Monitoring
- Inventory
- Tivoli Enterprise Console
- Remote Control

We will also focus on the three-tiered management environment. In most cases, the same Tivoli Management Applications tuning methods can be used for Managed Nodes (prior TMR structure) and Endpoints (three-tiered structure).

1.4.1.5 Operational considerations

It is important to understand how Tivoli and the Tivoli Management applications will be used in your environment and what other operations are running on your systems. Understanding this can help you to optimize the performance for your given management environment.

For example, in a classroom system, all Endpoints might attempt to perform a Endpoint login at the same time (when the class begins). This is very important information for the administrator. If you understand this situation before the implementation of a Tivoli Management environment, you can anticipate and possibly avoid problems.

As another example, if you use Software Distribution, it is important to know how many files are distributed in a day, how many targets you have, and how large the files are. For example, if you distribute many large files, you should not necessarily send them all to the systems at the same time.

In addition, if you know when the peak of the network utilization is, this information also will be useful. For example, if you plan to perform a bulk distribution, you can schedule this distribution at the lowest network utilization. As a result, this distribution will not adversely affect other applications that use the same network.

Operational considerations may not improve throughput dramatically, however, defining operational procedures based on the environment may help keep the throughput and performance within expectations even if the number of Endpoints (managed resources) increase.

1.5 TMR and network design approach

In the previous section, we talked about an approach to performance tuning strategy by tuning parameters for each machine. However, in most management environments, each system is connected via the network and working together. Therefore, the design of the whole management system is important for performance tuning rather than tuning individual managed resources. The following figure (Figure 12) represents the interaction between each managed resource.

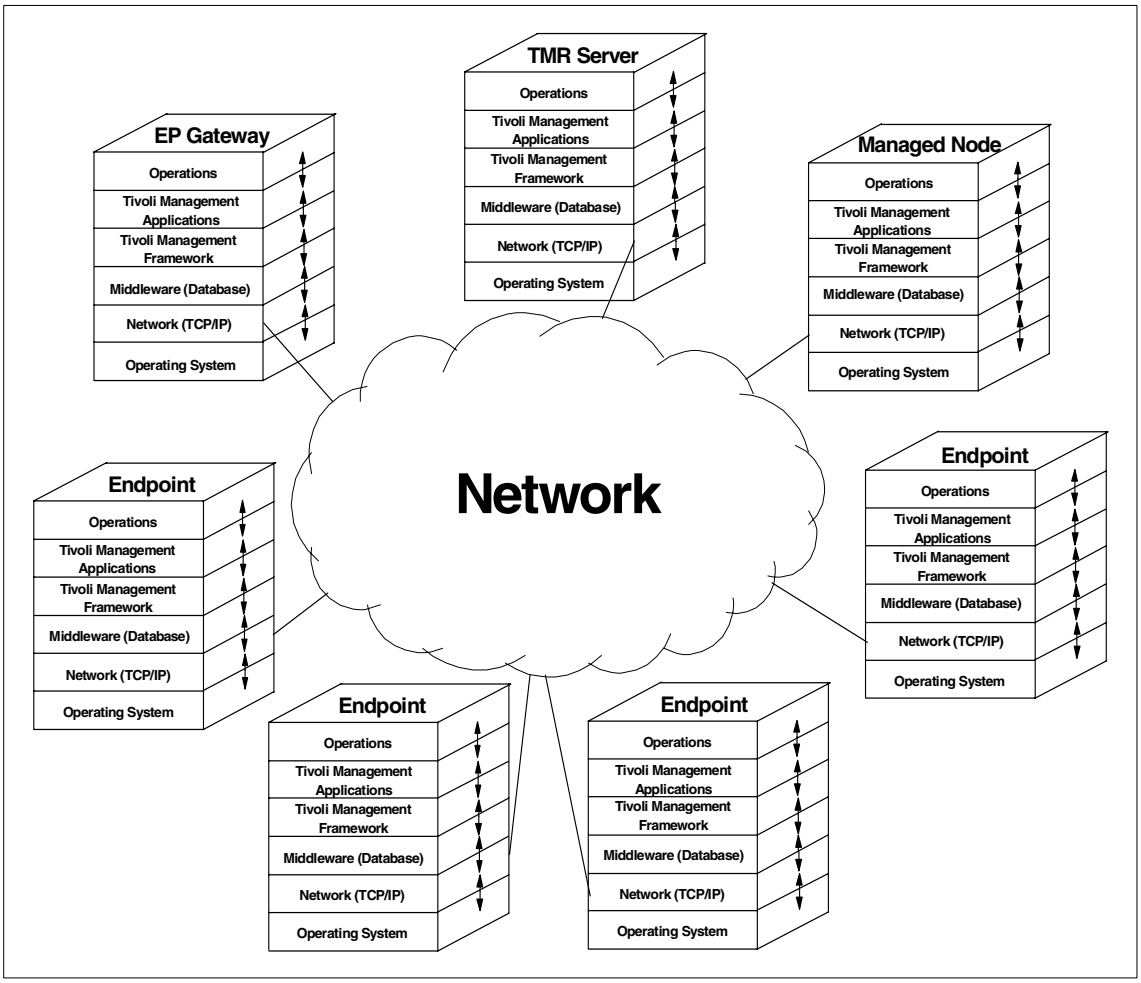


Figure 12. Interactions between each managed resource

What you should consider next is how each box (managed resource) can work together efficiently.

1.5.1 Design and tuning

The design of the whole management environment is as important as tuning each of the systems within their environment. Normally, the design of the management system should be considered in the early planning phase. This means that once you decide on the design, it is very difficult to change it after its deployment or implementation. What about parameter tuning? Normally, it can be done as an on-going exercise after the implementation. The most

important thing to remember is that it is much more difficult to change the design once you have begun the implement of the management systems in a large-scale management environment.

1.5.2 Design considerations

In the next chapter, Chapter 2, “Design considerations” on page 23, we will introduce the TMR design and management examples to improve performance. This information should be useful for tuning performance as well as throughput.

The following should be considered in the design phase of your deployment for improving the performance in a large-scale management environment.

- TMR Design.
- Network Design
- How many Endpoints are managed in the TMR.
- MDist Repeater Allocation.
- RIM Host Allocation.
- RDBMS Server Allocation.
- TEC Server Allocation.

Chapter 2. Design considerations

To improve Tivoli enterprise performance, the design of Tivoli Management environment is very important. The design of your Tivoli enterprise implementation is key to how well it will perform. In this chapter, we introduce some design considerations when you implement Tivoli Management applications and show where you should install each component in a large-scale three-tiered management structure.

2.1 Understanding three-tiered management structure

Version 3.6 of Tivoli provides a new management structure, the three-tiered management structure. It gives us stability, extensibility, and flexibility. This new management structure also introduced some new types of managed resources. These are Endpoint Manager, Endpoint Gateway, and Endpoint (TMA). These new managed resources also contribute to its performance improvement. They allow us to manage each system with less resources. The Endpoint is the typical example. New management structure lightens the load of the TMR server as well. To spread out the manager system resource in some Tivoli operations, the Endpoint Gateway plays the role of manager instead of the TMR Server.

In this three-tiered management structure, what is considered performance improvement? Before this can be answered, the following must be considered if you want to improve performance.

- Network.
- How many Endpoints a single Endpoint Gateway manages.
- How many Endpoints a single TMR manages.
- TMR server and Endpoint Gateway hardware configurations.

This chapter focuses only on the Tivoli infrastructure. Application considerations are covered in later chapters.

2.1.1 Network performance

All Tivoli Management applications use network protocol (mostly TCP/IP protocol). Therefore, the network performance directly affects Tivoli Management applications performance. The network tuning will involve the following factors:

- Link Speed and Quality
- Router configurations

- Other Network device configurations
- Other applications that use the same network

Although you should appreciate how important network tuning is for Tivoli management environments, we do not cover information about network tuning in this redbook. We recommend that you refer to the appropriate manuals or books which provide network tuning information.

2.1.2 Endpoint Gateway configurations

The Endpoint Gateway is a software component that runs on a full Managed Node enabling the Managed Node to operate as a Endpoint Gateway between a cluster of Endpoints and the rest of the TMR. Each TMR can have multiple Endpoint Gateways. The number of Endpoint Gateways will depend on factors, such as available system resources, the number of Endpoints, and network topology. Currently, one TMR server can handle up to approximately 200 Endpoint Gateways. This limit is actually based on the number of Managed Nodes that one TMR Server can manage.

There is no precise limit to how many Endpoints one Endpoint Gateway can handle. This will depend on system resources, performance requirements, and the type of management being performed. However, testing has been done that indicates that in many environments up to 2,000 Endpoints or more may be supported by a single Endpoint Gateway.

However, we do not recommend you to manage 2,000 Endpoints with a single Endpoint Gateway. The Endpoint Gateway is one of the key components to improve Tivoli performance. In the three-tiered management structure, the Endpoint Gateway plays the following roles:

- Handling Endpoint login requests
- Handling downcalls
- Handling upcalls
- MDist Repeater

Especially, when the Endpoint Gateway plays the role of the MDist repeater, a lot of resources are required to improve its performance, such as memory or disk space. And the MDist repeater tuning is the essence of Tivoli performance tuning. This is the reason why we do not recommend you to manage 2,000 Endpoints by a single Endpoint Gateway.

The number of Endpoints that are managed by a single Endpoint Gateway also depends on how each Endpoint works in your management environment.

For example, if all Endpoints attempt to perform the Endpoint login at once (for example, at 8:00am every Monday) in your management environment, you should try to decrease the number of Endpoints. On the other hand, if each Endpoint will attempt to perform the Endpoint login separately, your Endpoint Gateway may be able to manage many Endpoints. A typical example of this case is a classroom system.

As we mentioned, we cannot indicate the exact number of Endpoints that are managed by a single Endpoint Gateway, but a lower number of Endpoints is better for the performance. The following figure (Figure 13) shows the Endpoint Gateway allocation that is one of our recommended designs.

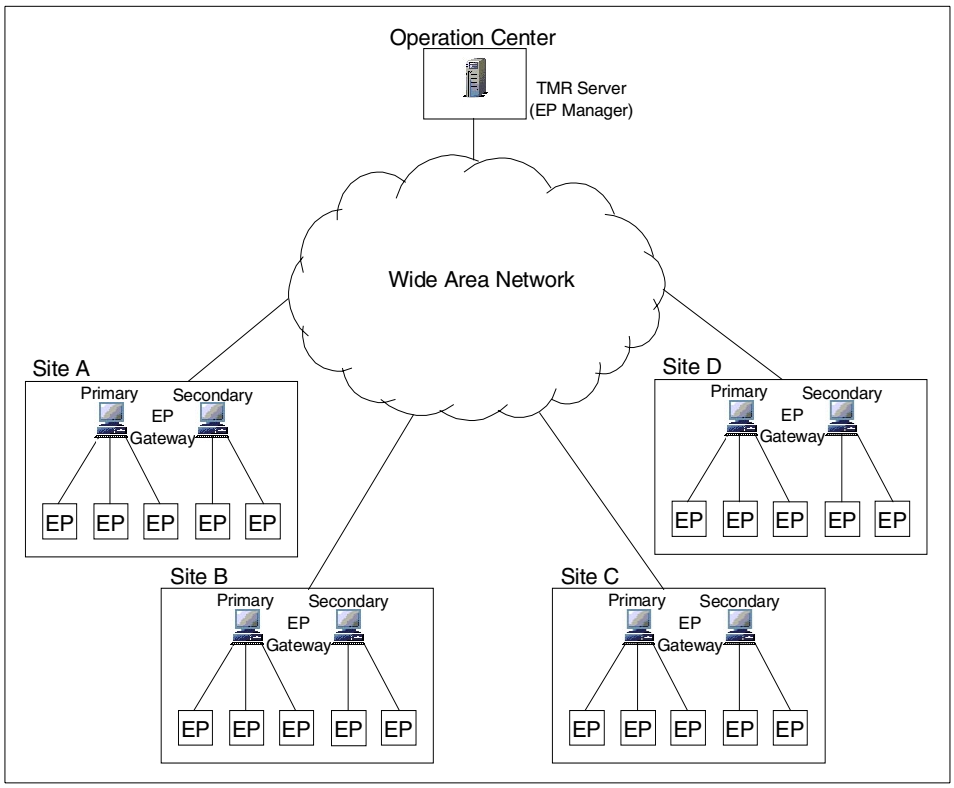


Figure 13. The recommendable Endpoint Gateway allocation

This design configuration is somewhat expensive configuration, but includes some good points in terms of performance.

- Each Endpoint Gateway manages a reasonable number of Endpoints.

In this case, more than one Endpoint Gateway is allocated in each location. Therefore, the load of each Endpoint Gateway is spread out.

- When the primary Endpoint Gateway becomes unavailable, the secondary Endpoint Gateway can take over the operations.

  Normally, when the Endpoint Gateway becomes unavailable, Endpoints will be isolated and attempt to look for another available Endpoint Gateway. If you do not configure two Endpoint Gateways in the same location, these isolated Endpoints will login to an Endpoint Gateway that is located in an other site. This means that these Endpoints are connected to the Endpoint Gateway through Wide Area Network (WAN). Usually, WAN is configured by using slow links so that this adversely affects Tivoli performance. To improve performance and availability, we therefore recommend allocating a primary and secondary Endpoint Gateway in each location.

  It is a good solution not only for the performance but also for the reliability of your management system. If you implement this configuration, then when the primary Endpoint Gateway goes down, the secondary Endpoint Gateway can take over all Endpoints that have logged into the primary Endpoint Gateway. The following figure (Figure 14) shows how the primary Endpoint Gateway takes over the operations from the secondary Endpoint Gateway.

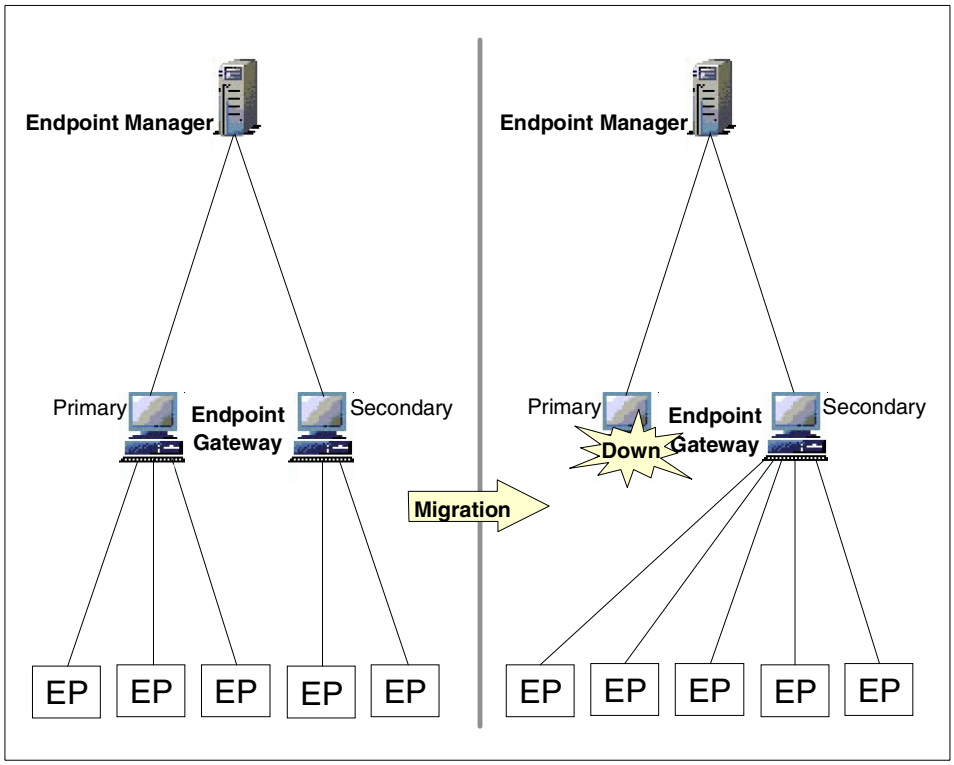


Figure 14. Taking over the operations from primary EPGW to secondary EPGW

This configuration also allows for Endpoint Gateway machine to be periodically shutdown for some reason, such as maintenance, without significantly degrading performance.

2.1.3 TMR configuration

Due to the architecture of Managed Nodes and the distributed database component that they control, a single TMR is typically limited to around 200 Managed Nodes. For environments requiring more than 200 Managed Nodes, the capability exists to interconnected TMRs and manage the resources in one TMR from the other. This solution provides many benefits, but also comes with the cost of increased complexity.

2.1.3.1 Interconnected TMRs

With the new architecture, a single TMR can now support many thousands of Endpoints. However, a maximum of 10,000 Endpoints per single TMR is generally recommended for performance reasons.

Therefore, there will be less of a requirement to interconnected TMRs, which will simplify your Tivoli deployment. If, however, you have interconnected TMRs for other reasons, such as geography or organizational requirements, this is fine. The Endpoints can be managed across interconnected TMRs as well. The following figure (Figure 15) shows the typical example of the interconnected TMRs that resolves the geography requirement.

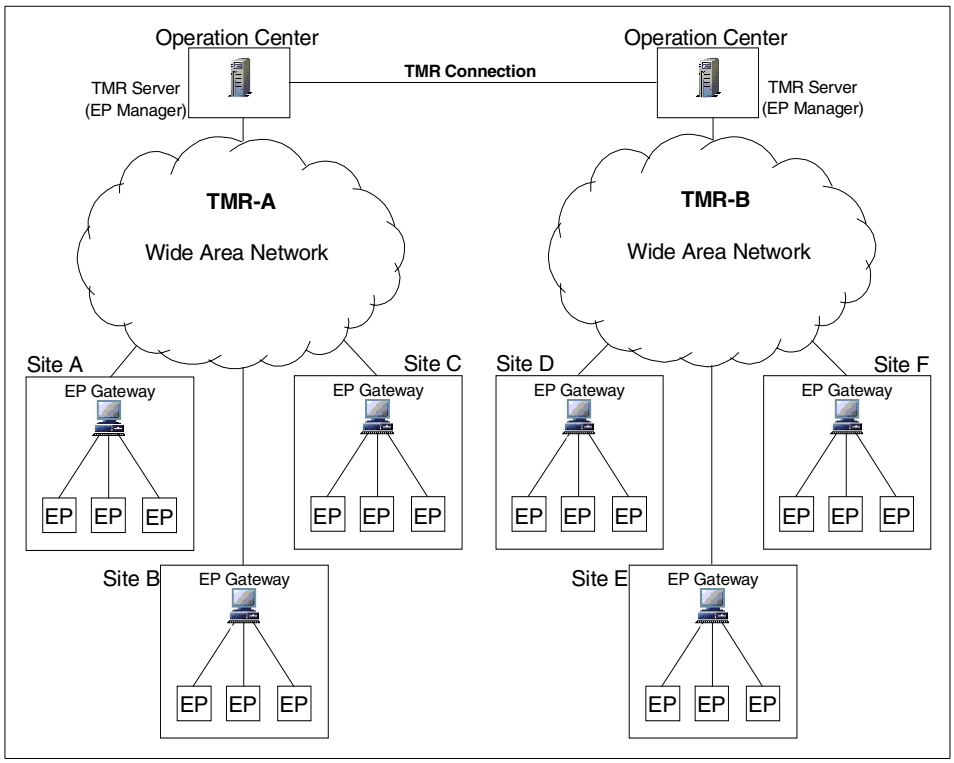


Figure 15. Interconnected TMRs to resolve geography requirements

Interconnected TMRs are particularly appropriate when:

- There are two different operation centers, and these are located in different places and far from each other.
- There are more than 10,000 managed resources (Endpoints) in your management environment.
- There are some organizational requirements that are needed to divide your management environment into two different TMRs.

2.1.3.2 Hub-Spoke management structure

The Hub-Spoke management structure is used for managing a large distributed systems environment using Tivoli. The following factors should be considered when the Hub-Spoke management structure is implemented.

- The number of managed resources (Endpoints).
- Hardware configurations of TMR servers or other infrastructure servers.
- Network configurations or topology.
- Geographical reasons.
- Organizational requirements or reasons.

Note

The number of managed resources can be the biggest consideration for designing your management structure. The following guidelines or limitations should be considered in the physical architecture design phase.

- 10,000 Endpoints per single TMR is generally recommended.
- No more than 2,000 Endpoints per single Endpoint Gateway is recommended.
- The number of Managed Nodes (Endpoint Gateways) supported in a single TMR is 200.

If your management environment does not clear these conditions, you have to consider dividing the TMR. Version 3.6 of Tivoli improved its scalability dramatically, but a large management environment still needs the multiple TMR solutions, such as the Hub-Spoke management structure.

In a large-scale management environment, the Hub-Spoke management structure can be a powerful solution. The following figure (Figure 16) shows an example of the Hub-Spoke management structure.

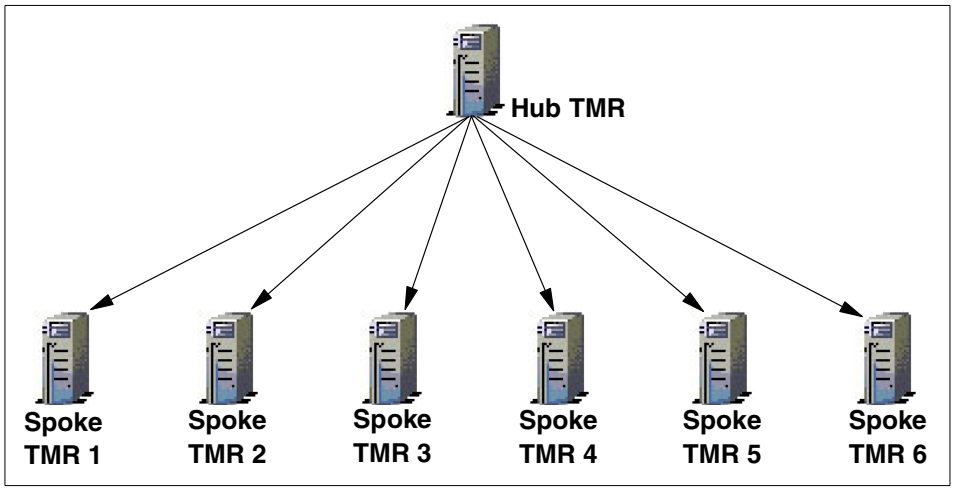


Figure 16. Hub-Spoke TMRs management structure

As you can see, the Hub TMR server provides a centralized management server function and the Hub TMR server works as the focal point in the entire management environment. Each Spoke TMR provides the direct control function to all Endpoints in its TMR. Therefore, most lookup requests from each managed resource in the Spoke TMR can be resolved within the Spoke TMR.

In this redbook, we will not explain Hub-Spoke architecture in detail. However, the Hub-Spoke management structure is one of the best solutions for building an international scale management system.

2.1.3.3 Interconnected TMRs for problem management

To improve the performance of your Tivoli Management environment, we introduce another solution using the interconnected TMRs. The following figure (Figure 17) shows the interconnected TMRs configuration in this case.

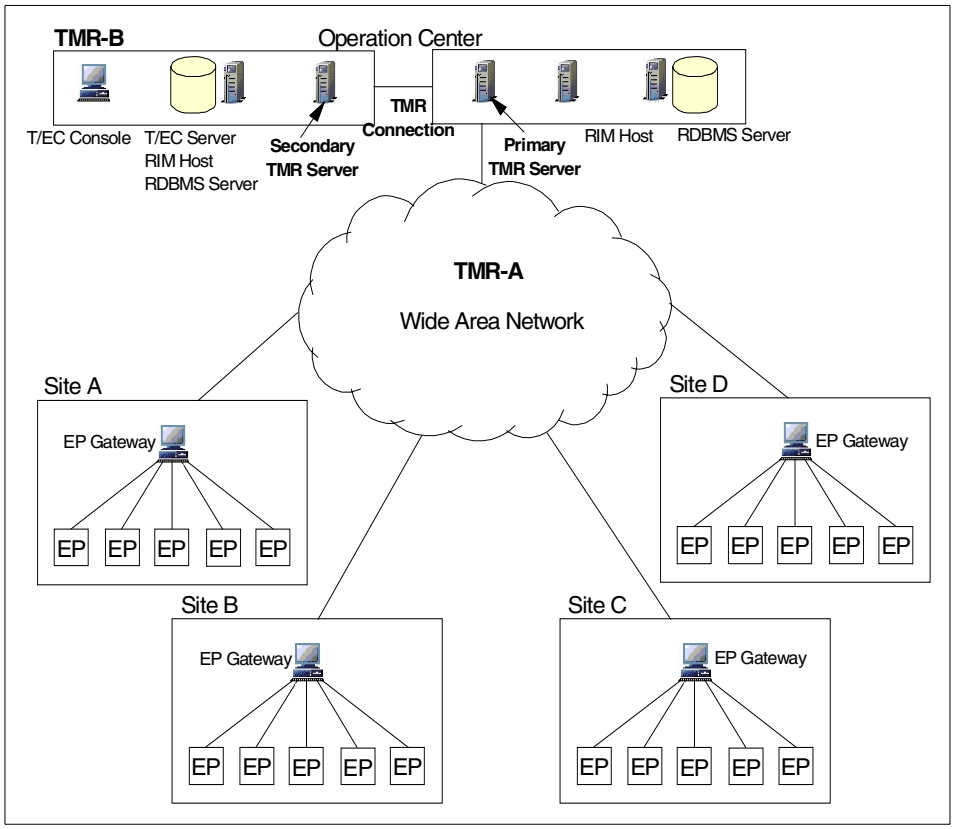


Figure 17. Interconnected TMRs to improve Tivoli Environment performance

In this example, most of managed resources are managed by the primary TMR server and belong to TMR-A. However, Tivoli Enterprise Console (TEC) has its own TMR, which is TMR-B.

TEC application is a heavy user of CPU or memory resources. TEC also consists of many components, for example, TEC server, RIM host, and RDBMS server. Communication among these components frequently occurs, and it can be relatively heavy traffic. In most environments, TEC gives us the best performance when it has its own TMR separate from the TMR that manages most of managed resources. Please refer to the “Improving Tivoli Enterprise Console performance” on page 243 for more detailed information about the Tivoli Enterprise Console performance.

2.1.4 TMR Server and Endpoint Gateway hardware configurations

In Version 3.6 of Tivoli, a single TMR server can manage thousands of Endpoints. As a result, one might be concerned about the load on the TMR Server. The new architecture is designed to allow Endpoint Gateway to off-load many of the functions formerly performed by the TMR Server. Although we have a large increase in the number of managed systems, the load on the TMR Server may actually decline.

On the other hand, based on the type of management you are performing, you will want to ensure that your workload is balanced across the Endpoint Gateways in your environment.

The load of the TMR Server and Endpoint Gateways are still heavy if your TMR manages thousands of managed systems. As we mentioned, the Endpoint Gateway plays the vital role in the three-tiered management structure. The TMR Server still maintains its master object databases and Tivoli Name Registry (TNR), and many of the functions that are performed must be authenticated at the TMR Server.

You must configure these manager resources, TMR Server and Endpoint Gateways, using high performance hardware. It should help you with performance improvement in Tivoli Management environment.

2.2 Management resource allocation

In Tivoli Management environments, there are two different types of resources. The first is manager resources. The following are typical manager resources:

- TMR server
- Endpoint Gateway
- TEC server
- RIM host
- RDBMS server

The second is a managed resource or a target that is managed by your manager resources. The following are typical managed resources:

- Managed Node
- PC Managed Node
- Endpoint (TMA)

To optimize your management environment, you need to balance each resource. In the previous section, we introduced the relationship between the Endpoint Gateway and Endpoints. There are some considerations when you implement your manager resources as well. In this section, we introduce the considerations for allocating manager resources.

2.2.1 Improving database operations performance

In the Tivoli Management operations, database (RDBMS) access is a potential performance bottleneck. In Tivoli environment, access to RDBMS is made through the RIM host. The RIM host provides a common database interface to the Tivoli applications irrespective of which RDBMS is being used. Some Tivoli core applications, such as Tivoli Enterprise Console (TEC) and Inventory, access RDBMS databases through a RIM host. To improve these Tivoli Application's performance, we compare various implementations and examine this impact on performance.

2.2.1.1 Sharing RDBMS is not recommended

In Tivoli Management applications, Tivoli Enterprise Console (TEC) and Inventory are the two biggest database users. Therefore, when you configure TEC and Inventory in your TMR, you should take care of the resource allocation, which will affect the performance of these products. The following figure (Figure 18) shows the configurations that are not recommended.

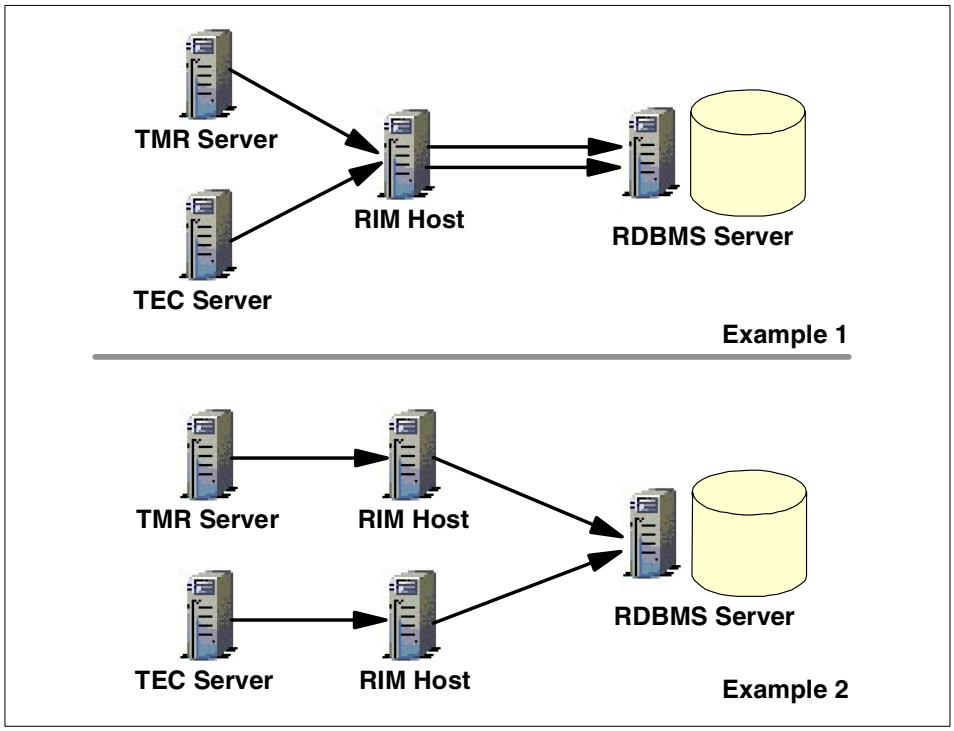


Figure 18. Non-recommended server configuration

As you can see in the both examples, TEC shares some resources with Inventory. For example, in Example 1, TEC shares the both the RIM host and RDBMS server with Inventory, and in Example 2, TEC shares the RDBMS server with Inventory.

In a large-scale environment, the amount of data that is managed by the RDBMS server is very large. Therefore, to improve performance, we strongly recommend you allocate the RIM host and the RDBMS server to each application.

2.2.1.2 Recommended configurations

The following figure (Figure 19) shows the configuration that is recommended to improve performance of Tivoli database operations.

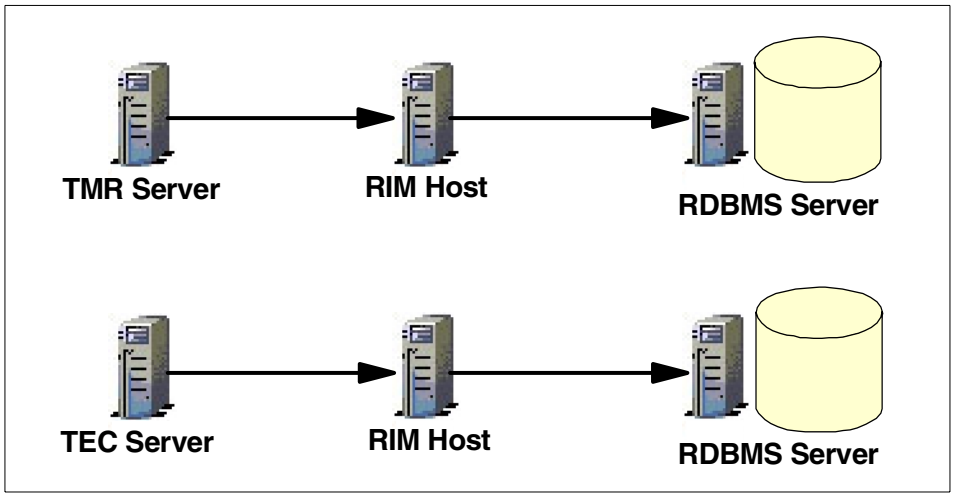


Figure 19. Recommended server configuration

As we mentioned, each application has its own RIM host and RDBMS server. It will be an expensive configuration, but will greatly improve its performance.

2.3 Logical design

In this chapter, we have discussed the physical architecture designs in your Tivoli Management environment. However, logical design is also important in your management operations. Normally, logical design is closely related to your management environment as follows:

- Tivoli Management applications
- Management Operations (how you manage your environment)
- Management Scale (large-scale or small-scale)
- Business Structure
- Geographical Structure

In this section, we talk about some considerations in the logical designs of your Tivoli Management environment.

2.3.1 Policy region design

Policy region design depends on your management environment, for example, management operations, Tivoli Management applications or

business structure. Proper design for policy regions simplifies your management operations and also improves performance and throughput.

If you are planning to manage more than a few thousand Endpoints within a single TMR, you should consider modifying the default policy methods that are used in building desktop panels to restrict the number of objects that are displayed. Processing time for certain desktop operations (for example, looking for a specific Endpoint in your desktop) increases in direct proportion to the number of managed Endpoints.

Policy region structure design is also important for managing your environment efficiently. The following figure (Figure 20) shows an example of policy region structure design.

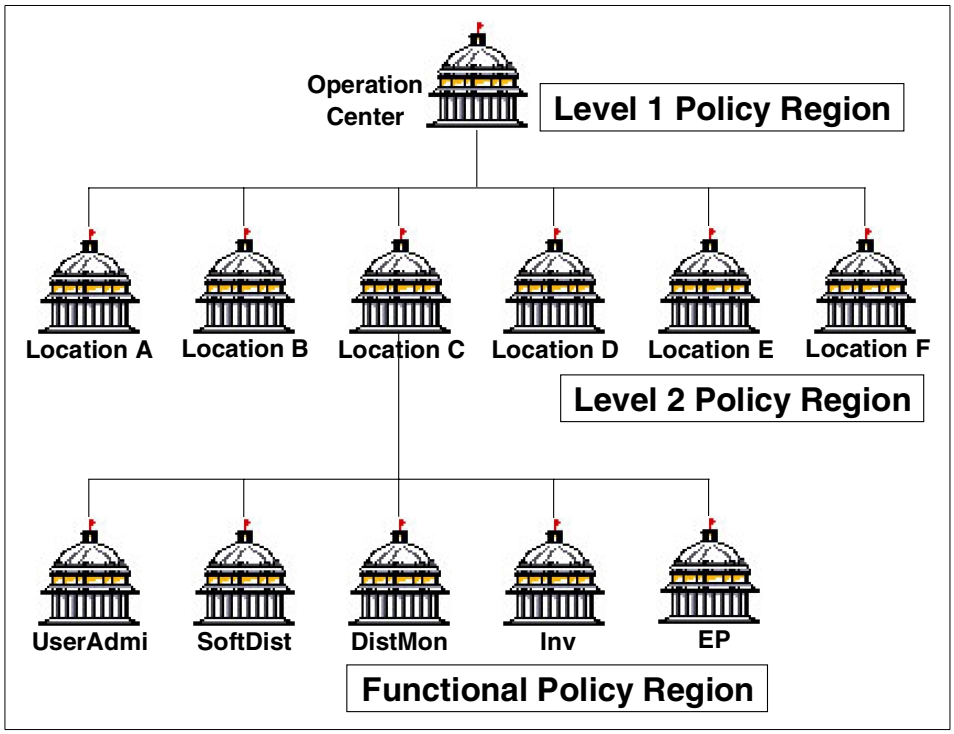


Figure 20. Example of policy region structure design

As you can see, the upper layer of the policy region structure is divided into geographical or organizational sub-policy regions. On the other hand, the lower layer of the policy region structure is divided into functional sub-policy regions. This is a basic approach to policy region design and also recommended way. In policy region structure design, naming conventions are

also important. Proper definitions avoid displaying unnecessary objects on a Tivoli desktop and improve your management efficiency.

2.3.2 Administrator desktops

Tivoli Administrator definition is very important to manage a large-scale environment. The most important attributes to specify for Tivoli administrators are TMR roles, resource roles, user ID, and login IDs. Any TMR role granted automatically applies to all TMR resources, both existing and future.

Tivoli administrator definition is closely related to your management operation. Therefore, in this redbook, we will not explain Tivoli administrator definition in detail. However, to restrict the number of objects that are displayed in each operator's Tivoli desktop, each operator should display only objects that the operator manages, and you should define each Tivoli administrator's role properly. This will improve the performance and throughput of your management operations.

2.3.3 Profile managers

Profile manager configurations may affect performance and throughput for your management operations in a large-scale environment. We will introduce profile manager configurations in the Chapter 5, "Improving software distribution performance" on page 133.

2.4 Conclusion of design considerations

To improve Tivoli Enterprise performance, its design should be considered carefully. In this chapter, we discussed the following:

- Three-tiered management structure
- TMR design
- Resource allocation
- Logical design

These considerations should be important and useful for most customers, especially in the pre-sales phase. As we mentioned, physical design modification is difficult after you implement your Tivoli Management environment; therefore, we strongly recommend you follow the information that this chapter provides.

Chapter 3. Improving operating systems and network performance

To improve performance in a Tivoli Management environment, tuning the operating system and network on each system is important because all Tivoli Management applications are running on operating systems and use the network. The following figure (Figure 21) shows the relationship between the operating system, network, and Tivoli Management applications.

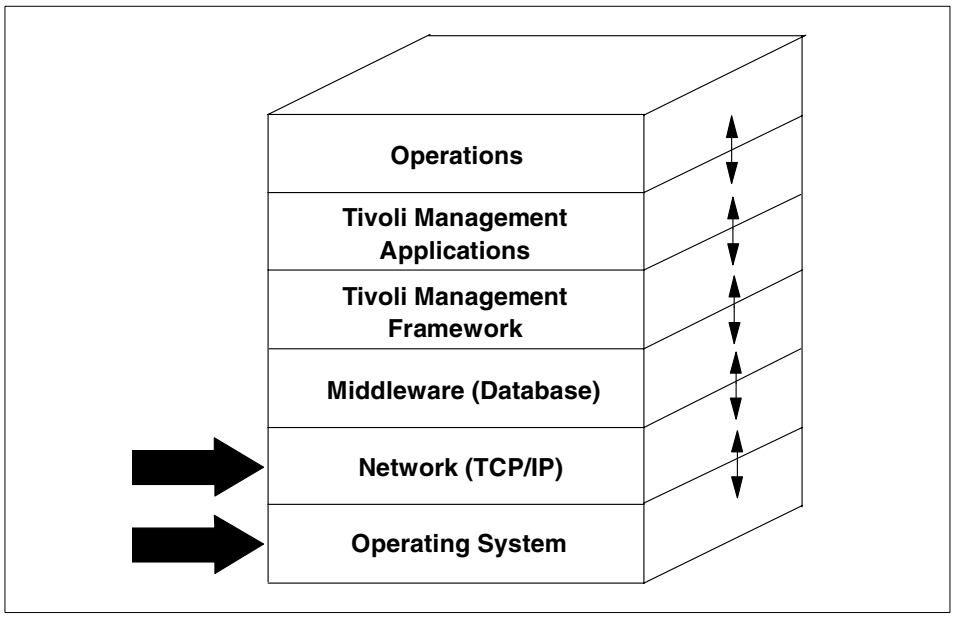


Figure 21. Tuning operating system and network

In this chapter, we introduce hints and tips about how to tune the specific operating systems (AIX and Windows NT) and their network interfaces. This may affect the performance and throughput for all upper layer components that run on the operating systems and their networks.

Note

This chapter introduces general information about performance tuning for specific operating systems and their network interfaces. Please refer to the appropriate manuals or documents for more detailed information about operating systems and network tuning.

3.1 Improving AIX system performance

In this section, we will introduce hints and tips to improve the system performance of an AIX machine.

3.1.1 CPU and Virtual Memory Utilization

The most frequently used tool is the `vmstat` command. The `vmstat` command monitors CPU and virtual memory usage for the system as a whole. It calculates statistic averages by reading an AIX kernel structure from kernel memory at each sample. This structure contains statistics counters that continually increase with system age. The `vmstat` command calculates the difference between two computer samples and comes up with an average over the sampling interval. It is typically run in a separate xterm window in the foreground with the output going to the terminal. It can also be run in the background while redirecting output to a file. The primary argument is the sample interval expressed in seconds. The second optional argument is the number of samples to capture. The most common way to use `vmstat` is to omit the second argument, in which case, `vmstat` will run a sample until interrupted.

Note

The first capture of `vmstat` should be ignored. It is incorrectly computed due to the lack of a proceeding capture with which to calculate statistics counter differences.

```
# vmstat 5 5
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 15643  1505   0   0   0   0    0   0 148  658 137  1  1 97  1
 0  0 15643  1505   0   0   0   0    0   0 178  295  76  1  3 88  9
 0  0 15643  1505   0   0   0   0    0   0 133  240  46  0  2 97  1
 0  0 15643  1505   0   0   0   0    0   0 119  256  43  1  2 97  1
 0  0 15643  1505   0   0   0   0    0   0 134  203  48  2  1 96  1
#
```

In this example, the important fields are as follows:

avm Active virtual memory in units of 4 KB pages. This value is the total number of pages allocated in page space. High values simply mean that the system has a high amount of virtual memory currently allocated. It is not an indicator of poor performance.

fre Size of the list of free RAM pages. The AIX system will try to maintain a minimum size free list so that pages will always be available when needed. The page stealer is responsible for keeping the free list at this minimum.

pi Average page-in rate from page space in units of pages per second. A nonzero value over an extended time (consider one minute an extended time) means that real memory is constrained to some degree. To be more precise, five pages per second would have us worried, while 50 pages per second would have us pounding the desk in frustration. Also note that this statistic does not include paging in from a file system due to I/O.

po Average page-out rate to page space. Everything that applies to pi also applies to po.

us User-mode CPU average. What a good value is for user CPU time depends on whether you are trying to maximize throughput or response time. Ideally, to maximize throughput, this value should be 100percent. To maximize response time, this value would ideally 0 percent.

sy System-mode CPU average. Ideally, system CPU time should be zero, but it never is. We would say that a value greater than 50 percent is usually cause for concern and further investigation.

id Idle-time average. These ideal values are opposite those for user time.

We strongly recommend you to monitor the following systems in your TMR by using the `vmstat` command.

- TMR server
- TEC server
- RIM host
- RDBMS server
- Endpoint Gateway
- MDist Repeater

Then, if the following situations are occurring, you should consider tuning your system.

- The fre value is always low.
- The pi and po value is always nonzero.
- The us value plus sy value is always close to 100.

- The id value is always close to 0.

In this case, first of all, you should consider extending the paging space. The paging space needs to be at least twice that of the your memory size. This is the most easy and steady way to improve performance. If the situation is not changed, even if you extend the paging space, it may be needed to extend the real memory.

The TEC server pins some cache on the real memory so that sometimes you may need to reduce the size of these caches.

3.1.1.1 Other useful tools

The following performance tools will be useful to monitor your system status.

iostat A facility to monitor terminal and disk I/O activity.

sar A tool to collect, report, or save a variety of system activity information.

ps Displays information about active processes.

monitor Displays a collection of the best statistics of `sar`, `vmstat`, `iostat`, and `ps` on one screen report.

tprof Provides a report that includes sorted totals of process names and how many ticks are spent in the three address spaces.

filemon Displays file I/O activity.

svmon Displays how much storage each process is using at any one time.

rmss Simulates reducing the amount of RAM by hiding real memory frame.

vmtune Sets tuning parameters for AIX kernel.

Please refer to the appropriate manuals for more detailed information about the above performance tools.

3.1.2 Network monitoring

The command that most mimics `vmstat` for the network domain is `netstat`. The `netstat` command provides many options and shows various information about the network interfaces on your system. The following shows how to get a repetitive sampling of the network interface every five seconds in the manner of `vmstat` sampling. In this case, our only network interface was tr0, a Token-Ring interface. If we had been using the Ethernet, we would have specified en0. As in `vmstat`, the very first line is a dump of the counters since

boot and should be ignored. The five columns, which are the most useful, are statistics for the tr0 interface, while the last five columns are statistics for all interfaces. The first two columns are input packet rates in packets per second and error rates in errors per second. The next two columns are output packet rates and error rates. Ignore the colls column, which is not used on AIX and will always be zero. On other UNIX systems, this statistic counts the number of Ethernet collisions. The packet counts indicate a rough measure of network throughput through that particular interface. Typical Ethernet or Token-Ring interfaces will be able to sustain about 2,000 to 3,000 packets per second. This figure will depend on how large the packets are and how fast the CPU is.

```
# netstat -I tr0 -i 5
    input   (tr0)      output           input   (Total)    output
 packets  errs  packets  errs colls  packets  errs  packets  errs colls
10088687     0  9689526     0     0 11356132     0 10956971     0     0
      28     0       25     0     0       31     0       28     0     0
      29     0       29     0     0       34     0       34     0     0
      21     0       17     0     0       24     0       20     0     0
      50     0       44     0     0       53     0       47     0     0
     134     0      170     0     0      137     0      173     0     0
      92     0       79     0     0       97     0       84     0     0
```

In this case, our recommendation is the same as with the `vmstat` command. We strongly recommend you to monitor the following systems in your TMR by using the `netstat` command.

- TMR server
- TEC server
- RIM host
- RDBMS server
- Endpoint Gateway
- MDist Repeater

Then, if the following situations always occur, you should consider tuning your system.

- The errs value is always nonzero.

In this case, you should check your network and network interfaces. For example, heavy network traffic or bulk data transfer may cause this situation.

The following shows network memory buffer (mbuf) statistics. The last three lines are important and indicate, at some point, a shortage since boot. A nonzero value here would indicate you should increase the limit of mbuf.

```
# netstat -m

Kernel malloc statistics:

******* CPU 0 *******
By size       inuse     calls failed    free   hiwat   freed
32              244    977561      0     140     640       0
64              116    895891      0     204     320       0
128              82      9293      0      78     160       0
256             251  32139426      0     245     384       1
512             117   1687674      0      27      40      29
1024             88    127008      0      40     100       0
2048              0   3942310      0     100     100     142
4096             67    577836      0      83     120      23
8192              0    245186      0       3      10       0
16384             1    191596      0      21      24       7
32768             1         1      0       0    1023       0

By type       inuse     calls failed  memuse  memmax  mapb

Streams mblk statistic failures:
0 high priority mblk failures
0 medium priority mblk failures
0 low priority mblk failures
#
```

The mbuf specifies the maximum memory to be used for the network. It will be pinned on the real memory; so, be careful when you increase it.

The following shows the detailed network adapter statistics for Ethernet and Token-Ring respectively.

```
netstat -v
-------------------------------------------------------------
ETHERNET STATISTICS (ent0) :
Device Type: IBM 10/100 Mbps Ethernet PCI Adapter (23100020)
Hardware Address: 00:60:94:e9:27:8a
Elapsed Time: 0 days 0 hours 0 minutes 0 seconds

Transmit Statistics:                          Receive Statistics:
--------------------                          -------------------
Packets: 0                                    Packets: 0
Bytes: 0                                      Bytes: 0
Interrupts: 0                                 Interrupts: 0
Transmit Errors: 0                            Receive Errors: 0
Packets Dropped: 0                            Packets Dropped: 0
                                              Bad Packets: 0
Max Packets on S/W Transmit Queue: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 0                          Broadcast Packets: 0
```

```
Multicast Packets: 0                        Multicast Packets: 0
No Carrier Sense: 0                         CRC Errors: 0
DMA Underrun: 0                             DMA Overrun: 0
Lost CTS Errors: 0                          Alignment Errors: 0
Max Collision Errors: 0                     No Resource Errors: 0
Late Collision Errors: 0                    Receive Collision Errors: 0
Deferred: 0                                 Packet Too Short Errors: 0
SQE Test: 0                                 Packet Too Long Errors: 0
Timeout Errors: 0                           Packets Discarded by Adapter: 0
Single Collision Count: 0                   Receiver Start Count: 0
Multiple Collision Count: 0
Current HW Transmit Queue Length: 0

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Reset Count: 0
Driver Flags: Broadcast Simplex Limbo
64BitSupport

IBM 10/100 Mbps Ethernet PCI Adapter Specific Statistics:
---------------------------------------------------
Chip Version: 25
RJ45 Port Link Status : down
Media Speed Selected: Auto negotiation
Media Speed Running: Unknown
Receive Pool Buffer Size: 180
Free Receive Buffers: 180
No Receive Buffers: 0
Inter Packet Gap: 60
Adapter Restarts due to IOCTL commands: 0
Packets with Transmit collisions:
 1 collisions: 0           6 collisions: 0          11 collisions: 0
 2 collisions: 0           7 collisions: 0          12 collisions: 0
 3 collisions: 0           8 collisions: 0          13 collisions: 0
 4 collisions: 0           9 collisions: 0          14 collisions: 0
 5 collisions: 0          10 collisions: 0          15 collisions: 0
Excessive deferral errors: 0x0
------------------------------------------------------------
TOKEN-RING STATISTICS (tok0) :
Device Type: IBM PCI Tokenring Adapter (14103e00)
Hardware Address: 00:06:29:b9:fb:08
Elapsed Time: 14 days 5 hours 11 minutes 23 seconds

Transmit Statistics:                        Receive Statistics:
--------------------                        -------------------
Packets: 9700977                            Packets: 10327816
Bytes: 8014542907                           Bytes: 5682485772
Interrupts: 9700193                         Interrupts: 10327452
Transmit Errors: 1                           Receive Errors: 0
Packets Dropped: 0                           Packets Dropped: 0
                                            Bad Packets: 0
Max Packets on S/W Transmit Queue: 18
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Elapsed Time: 14 days 5 hours 11 minutes 17 seconds
Broadcast Packets: 35                       Broadcast Packets: 1125658
Multicast Packets: 2                        Multicast Packets: 91
Timeout Errors: 0                           Receive Congestion Errors: 0
Current SW Transmit Queue Length: 0
Current HW Transmit Queue Length: 0
```

```
General Statistics:
-------------------
No mbuf Errors: 0                          Lobe Wire Faults: 0
Abort Errors: 0                            AC Errors: 0
Burst Errors: 0                            Frame Copy Errors: 1
Frequency Errors: 0                        Hard Errors: 3
Internal Errors: 0                         Line Errors: 0
Lost Frame Errors: 0                       Only Station: 0
Token Errors: 0                            Remove Received: 0
Ring Recovered: 9                          Signal Loss Errors: 0
Soft Errors: 1                             Transmit Beacon Errors: 0
Driver Flags: Up Broadcast Running
AlternateAddress 64BitSupport ReceiveFunctionalAddr
16 Mbps

IBM PCI Tokenring Adapter (14103e00) Specific Statistics:
---------------------------------------------------------
Media Speed Running: 16 Mbps Half Duplex
Media Speed Selected: Auto negotiation / Half Duplex
Receive Overruns : 0
Transmit Underruns : 0
ARI/FCI errors : 0
Microcode level on the adapter :001PX11B2
Num pkts in priority sw tx queue  : 0
Num pkts in priority hw tx queue  : 0
Open Firmware Level : 001PXRS02
```

In this example, the important fields that you should check are as follows:

Transmit errors Shows the number of errors during transmitting data.

Receive errors Shows the number errors during receiving data.

Packets dropped Shows how many packets were dropped during transmitting or receiving data.

Max Packets on S/W Transmit Queue Indicates the maximum number of packets that the transmit queue has queued.

S/W Transmit Queue overflow Indicates how many times the transmit queue overflowed.

Where, if you meet the following situations, you should check your network or tune the network adapter card configurations.

- The Transmit Errors and Receive Errors values are always high.
- The Packets Dropped value is always high.
- The Max Packets on S/W Transmit Queue value is almost close to the adapter attribute.
- The S/W Transmit Queue Overflow value is non-zero.

You can check the definitions of your network adapter card by using the `lsattr` command as follows:

```
# lsattr -E -l ent0
busio           0xbff400         Bus I/O address                     False
busintr         6                Bus interrupt level                 False
intr_priority   3                Interrupt priority                  False
tx_que_size     256                  TRANSMIT queue size                 True
rx_que_size     256                  RECEIVE queue size                  True
rxbuf_pool_size 384              RECEIVE buffer poof size            True
media_speed     Auto_Negotiation Media Speed                         True
use_alt_addr    no               Enable ALTERNATE ETHERNET address True
alt_addr        0x000000000000   ALTERNATE ETHERNET address          True
ip_gap          96               Inter-Packet Gap                    True
# lsattr -E -l tok0
busio        0xbffc00  Bus I/O address                     False
busintr      1         Bus interrupt level                 False
xmt_que_size 512           TRANSMIT queue size                 True
rx_que_size  64            RECEIVE queue size                  True
ring_speed   autosense RING speed                            True
attn_mac     no        Receive ATTENTION MAC frame         True
beacon_mac   no        Receive BEACON MAC frame            True
use_alt_addr no        Enable ALTERNATE TOKEN RING address True
alt_addr     0x        ALTERNATE TOKEN RING address        True
full_duplex  no        Enable FULL DUPLEX mode             True
#
```

In this example, the `xmt_que_size` and `rx_que_size` show the queue size of the adapter card. If this Max Packets on S/W Transmit Queue value is close to the above `xmt_que_size`, sooner or later, the S/W transmit queue will overflow. Therefore, we recommend you to increase the `xmt_que_size` value. You can change this value by using SMIT.

3.1.2.1 Other useful network tuning tools

The following performance tools will be useful to monitor or tune your system's network interfaces.

no Sets or displays current network options in the kernel.

netpmon The tool to use to analyze network I/O.

iptrace Provides IP layer trace facility.

tcpdump Gives detailed information about the network traffic.

Especially, the `no` command provides many parameters that affect the performance. For example, thewall, sb_max, tcp_sendspace, tcp_recvspace, udp_sendspace, udp_recvspace, tcp_mssdflt, ipqmaxlen or rfc1323 parameters are very important to improve network performance; so, you should tune these parameters very carefully.

3.1.3 Other considerations

We introduced basic and typical performance tuning techniques. Basically, we recommend that you use these methods when you tune your system performance. However, if these are not enough, changing the nice value may help your situation.

The nice value specifies the priority for each process. So, you can modify the Tivoli process priority, such as oserv process, on the system. It is available on the AIX system. Modify this value with caution because the AIX kernel also has the priority that is defined by the nice value. Therefore, you must refer to the AIX manuals that explain the `nice` command in detail if you plan to modify it.

3.2 Improving Windows NT system performance

The main objective of this section is to show which factors affect the system performance of an Windows NT machine and to provide guidelines about how to improve this performance.

3.2.1 Overview

Unlike other operating systems that were designed with the idea that the users should handle the details of the system configuration for optimal performance, Windows NT was designed to be a self-optimizing operating system. This means that Windows NT tries to adjust how it operates to accommodate a particular system for optimal performance. There are very few software settings to change. It is considered optimized for basic workload conditions.

This does not mean that Windows NT users cannot improve their performance. Modifying certain hardware and software components may provide performance improvements.

Servers are made up of a number of subsystems, each of which plays an important role in how the server performs. Depending on the use of the server, some of these subsystems are more important and critical to performance than others.

These can be summarized as:

- Processor and bus architecture
- System memory
- Disk storage

- Network communications

Other factors include:

- Network topology and bandwidth
- Number of servers and users
- Types of applications

Tuning the server by identifying the factors that are restricting the server's optimum performance may achieve your goal.

These factors are interrelated. You might observe that your disk subsystem is performing poorly; whereas, the real bottleneck lies on the insufficient amount of memory causing excessive paging. Solving a problem on one could cause a problem to the other.

The first and general recommendation to improve the performance is to bear in mind that the system performance depends on the proper working and balance of these subsystems.

3.2.2 Processor performance

The processor is one of the starting points to overall system performance. The processor is the heart of the server and is involved in most transactions that occur in the server. The processing horsepower of a computer determines how well a processor-intensive operating system, such as Windows NT, will run.

While the CPU is an important subsystem, there is the mistaken belief that the CPU is often the source of a performance bottleneck; so, buying a server with the fastest CPU is the way to go. In the majority of server installations, the CPU is, in fact, over-powered and the other subsystems are under-powered.

Recommendations for processor performance are:

- RISC processors usually provide faster performance than the CISC processors running 32-bit code applications, such as Tivoli.
- The PowerPC processor offers better performance than most Intel processors.
- Replace 8-bit video cards with 16 or 32 bit cards. 8 bit cards use more processor time under Windows NT.

- Increase the size of your external memory cache. Increasing the amount of external memory cache can often increase performance on a Windows NT computer.
- Add additional processors. You may benefit if you are using multithread applications by adding additional processors. One of the advantages of Windows NT over other operating systems is the support for multiple processors, known as multiprocessing. Multiple processors provide improved performance with applications that are designed with multiprocessing in mind. Most major applications designed for Windows NT will have performance improvements on a computer with multiple processors. However, many older 16-bit applications, or other applications that do no support multiprocessing may not have any noticeable performance improvements. Adding a processor is especially easy when the machine has more than two processors. It is detected automatically. But if the machine has just one processor, you have to put on the version of the Windows NT Kernel for multiprocessing. You can do that with a program (UPTOMP.EXE, found in the Windows NT Resource Kit). Otherwise, you will need to reinstall Windows NT before it will recognize and use the additional processor.

3.2.3 Memory performance

Memory is perhaps the single most critical issue of Windows NT performance. Memory, even more than processing power, in most circumstances, determines just how much Windows NT can work and produce. This is especially true in client/server environments, such as Tivoli, where the emphasis may not be on processing power but simply on having enough memory on the server machine running Windows NT to address each client's application needs.

How much physical memory do you need for good performance? It depends on your system.

If you are planning to implement a new system, you should consider the following:

- The absolute minimum of any computer running Windows NT should be 16 MB. Add at least another 4 MB for each kind of networking service. If you are installing Windows Networks, NetWare Networks, as a service, then you should estimate another 2 MB per service. If you're using your machine as a network server, take the final number and add 50 percent. Add 16 MB if you are working regularly with graphics. This includes programs, such as AutoCAD, Adobe Photoshop, or any photo-realistic image rendering software. Add 8 MB for fault tolerance as system

protection. Add 16 MB if you are doing application development. Development environments, such as Visual C++ and VisualBasic, need memory for themselves and for the applications being generated with them.

If you have a system, and you are going to modify its configurations:

- To determine the amount of memory needed to support twice the amount of users, just double the working set size and then add 30 percent as a buffer for peak activity.

  Remember more is better. Memory is too cheap to worry about compared to accurately predicting the exact amount required.

If you can't add more memory:

- Windows NT's memory demands can be partially alleviated through the creative use of disk space. By taking chunks of live memory that are not actually being used and temporarily storing what is in them on the hard drive (a technique known as swapping or paging), it is possible to get far more use out of the memory you do have. This scheme, while it does give any Windows NT machine a performance boost, has its limitations.
- The size of the hard disk is important. Windows NT takes up about 100 MB.
- The speed of the hard drive. A slow hard drive (or a slow drive controller) will bring Windows NT to a grinding crawl especially when it starts swapping heavily.
- The speed of the computer. Swapping eats CPU cycles. The less time spent swapping, the more time spent actually running applications.

But remember that servers should never regularly page memory to disk. Paging I/O should occur only occasionally, for example, when applications are initializing, but never on a continuous basis.

Servers should be configured so that average memory utilization does not exceed 70 percent. 30 percent is usually enough extra memory so that the server will not expand storage onto disk or page memory onto disk during periods of peak activity.

In order to minimize memory utilization, you should reduce the amount of memory that is in constant use. Here are some suggestions:

- Reduce the number of display colors.
- Reduce the display resolutions.

- Turn off unnecessary services and drivers. You can do this from the Control Panel.
- With physical memory, use consistent sets of chips. One of the more important rules about purchasing and installing RAM in a computer is that it is a bad idea to mix RAM chips of different speeds or even different manufactures. The results can be unpredictable, ranging from the computer simply not performing well, to the computer not working at all. Use as many identical chips as you can, right down to the lot number and manufacturer.
- Use 32-bit programs instead of 16-bit counterparts. Every time a16-bit program is launched, Windows NT has to spawn a virtual machine to handle it. This eats memory. Use as many 32-bit programs as possible.
- Your system should not page to a disk, but if it does page to disk, use a separate hard drive for swap memory. Paging eats up CPU time and forces a drive to labor quite a bit. With this in mind, if you are configuring a fairly high-end Windows NT machine, investing in a second (or third) hard drive and devoting at least part of it to swap spaces makes good sense. The main disk can be reserved for applications and the operating system itself, and the amount of time wasted for the disk to split time between two or more things (paging and regular system operations) comes way down. A computer running Window NT should be outfitted with at least 1 GB of hard disk storage (not just for the operating system and its supports, but for swap space).

3.2.4 Disk performance

All data must be retrieved from, and stored to, disk. Disk accesses are usually measured in milliseconds; whereas, memory and CPU and bus operations are measured in nanoseconds or microseconds. In other words, disk operations are typically thousands of times slower than CPU and bus transfers, memory accesses, and LAN transfers. This explains why the disk subsystem can easily become the major bottleneck for any server configuration.

Disk subsystems are also important because the physical orientation of data stored on disk has a dramatic influence on overall server performance. A detailed understanding of disk subsystem operation is critical for effectively solving many server performance bottlenecks.

A disk subsystem consists of the physical hard disk and the controller. A disk is made up of multiple platters coated with magnetic material to store data. The entire platter assembly mounted on a spindle revolves around the central

axis. A head assembly mounted on an arm moves to and from the platter (linear motion) to read the data stored on the magnetic coating of the platter.

The linear movement of the head is referred to as the seek, and the time it takes to move to the exact track where the data is stored is called seek time. The rotational movement of the platter to the correct sector to present the data under the head is called latency. The ability of the disk to transfer the requested data is called the data transfer rate.

Recommendations for improving the disk performance are:

- Configure Windows NT virtual memory to use multiple drivers.
- If your system has multiple hard disks, creating a virtual memory pagefile for each disk may offer performance improvements.This will offer a performance gain if your disk controller allows a simultaneous reading and writing of multiple disks.
- Move the virtual memory pagefile completely off of the drive that contains the Windows NT systems files. This avoids disk attempts for both system file reads and writes and pagefiles reads and writes.
- Avoid 8-bit disk cards. Replace them with 16 or 32 bit cards.
- The most widely used drive technology today in servers is SCSI or EIDE.
- For performance reasons, do not use EIDE (Enhanced Integrated Drive Electronics) disks in your server. The EIDE interface does not handle multiple simultaneous I/O requests very efficiently and so is not suited to a server environment. The EIDE interface uses more server CPU capacity than SCSI (Small Computer System Interface). We recommend you limit EIDE use to CD-ROM and tape devices.
- A hard disk 0 partition that is above 400 MB should be formatted as NTFS.

3.2.5 NT network performance

The network adapter is the pathway into the server from the clients. All requests to the server, and all responses from the server must pass through the network adapter. Its performance is key for many server applications.

Most LAN adapters consist of components that perform the following functions:

- Network interface/control
- Protocol control
- Communication processor
- Bus interface

- Buffers/storage
- LAN adapter operation

3.2.5.1 Adapter performance

The relationship between the network client workstation and the server is called a request/response protocol. Any time a network client wants to access data on the server, it must issue a request. The server then locates the data, creates a data frame, and issues the `transmit` command to the server LAN adapter to acquire that data from memory to be transmitted to the network client workstation (response). This request/response relationship is a fundamental characteristic that affects server performance.

The LAN adapter has two fundamental operations that occur for each frame.

1. The LAN adapter communication processor must execute the firmware code necessary to prepare each frame to be moved to or from system memory. This is called adapter command overhead. Every LAN adapter has a limit on the number of frames per second that can be processed. Since it takes a certain amount of time to execute this code, the adapter frame throughput rate is reached when the on-board communication processor reaches 100 percent utilization.
2. The LAN adapter must also copy all frames to or from memory. This is the adapter DMA throughput capability. When a LAN adapter is moving large amounts of data per frame, the amount of time spent processing firmware command overhead is small compared to the time necessary to copy the large frame to or from server memory. When an adapter is maximizing its data transfer resources, it is the DMA performance of the adapter, not the on-board communication processor, that limits frame throughput of the adapter.

LAN adapters are efficient when network applications requesting data generate requests for large frames. Applications that request small blocks of data require the LAN adapter communication processor to spend a larger percentage of time executing overhead code for every byte of data transmitted. This is why most LAN adapters cannot sustain full wire speed for all frame sizes. In this case, the solutions are new applications (difficult or perhaps impossible) or additional subnetworks using multiple LAN adapters. Faster LAN adapter technology could be used, but the gains would be minimal. Faster LAN technology offers higher data rates, but when the frames are small, a greater percentage of time is spent in adapter overhead not data transmission.

3.2.5.2 Network design performance

In addition to the network adapter, network bandwidth also contributes to the performance of the overall system. A 100 Mbps network bandwidth is certainly better than a 10 Mbps network.

Depending on the network traffic, segmentation can be implemented to achieve better performance. Networks will be improved if switches are used to split the network into a number of small segments.

With such a simple design, bottlenecks can easily occur. The following design points will help you eliminate those problem areas:

- The design should ensure that most traffic stays within a segment.
- Where most traffic is to and from servers, place the servers on segments of their own. This improves server and network availability.
- Speed up the server connections by configuring their network adapters for full duplex. If most traffic is to or from the servers, this can effectively double the network I/O speed.
- Use the fastest network adapter in the server.
- If there is very high traffic on a particular segment due to specific workstations, and if the bandwidth of that segment is the bottleneck, you can isolate those workstations and put them on separate switch points. This will lower the burden of that network segment.
- For large networks, use multiple switches and connect those switches together using a high-speed backbone connection. Use 100 Mbps Ethernet where only limited distance is needed and average backbone traffic won't exceed 30 Mbps. Otherwise, consider connections, such as ATM.

3.2.5.3 Device drivers performance

Device drivers play a major role in performance of the subsystem with which the driver is associated. A device driver is software written to recognize a specific device. Most of the device drivers are vendor specific. These drivers are supplied by the hardware vendors. Most of the device drivers can be downloaded from the Web site.

Choosing a correct device driver for specific hardware is very important. The device drivers are also specific to the operating system. Some of the device drivers are integrated with Windows NT, and some are supplied with the hardware by the hardware vendor. A technically competent person should select a proper driver during installation. Selecting an incorrect device driver for a specific device can cause poor performance.

The version of the device driver is also a very important factor. Some of the critical performance problems can be resolved by using the correct version of the device driver. Most of the latest drivers can be found on the Web site of the hardware vendor.

Windows NT provides support to most of the network protocols. Choose the protocols that are absolutely necessary for your network and install only those protocols. For example, if you have a very small network and do not intend to use Internet or routing functions, you must only use the NetBEUI protocol. NetBEUI is a fast protocol for small LANs but is a non-routable protocol. If you intend to use Internet or routing functions, then you must install TCP/IP. Similarly, if you intend to connect to a NetWare server, then you must add the IPX/SPX protocol.

Note

Tivoli Management applications must use TCP/IP protocol to perform management operations.

The same rules apply for Network Services: Remove all the unwanted services.

Use only the services that are absolutely necessary.

3.2.5.4 Network performance recommendations

The LAN adapter can become a server bottleneck; so, it is important to monitor LAN activity to prevent the LAN adapter from slowing the server down. Monitoring LAN activity usually requires a LAN analyzer. Examining LAN utilization or bytes transferred will usually not be sufficient without looking at other factors that affect network performance.

The potential two types of bottlenecks in the LAN subsystem are:

1. Bytes per second, which are obtained by most operating system monitor programs or with a LAN analyzer.
2. Packets per second, which can be monitored using a LAN analyzer.

Packets per second can indicate a very busy LAN adapter. When the packet size is small, LAN utilization will be low, but LAN adapter utilization can be quite high causing a LAN bottleneck. Since LAN analysis can be complex and time consuming, we recommend that you simplify the analysis by never exceeding an average LAN utilization of about 40 to 50 percent.

Most LAN adapters available today can drive LAN utilization to 40 to 50 percent for even the smallest packet size. Thus, most LAN bottlenecks can be avoided if you use a 40 to 50 percent threshold as the maximum acceptable average LAN utilization. If you spot sustained LAN utilization above 50 percent, then it is time to employ higher speed LAN technology or divide your network into multiple subnetworks with separate LAN adapters in the server for each subnetwork.

Average LAN utilization should never exceed about 40 to 50 percent.

LAN adapter technology is developing rapidly. The type of adapter installed in a server will reflect the performance capacity available. For example, a server with 100 Mbps LAN adapter on a 100 Mbps network should provide more throughput than a 10 Mbps LAN adapter on a 10 Mbps network.

3.2.5.5 Tuning network performance

Tuning Windows NT network performance is one of the most complex tasks.

Your selection of the processor, disk, and memory subsystem configurations will have a significant impact on the Windows NT networking components' ability to perform optimally. The performance tuning goal is to reduce overhead and increase network responsiveness. Here are some actions you can perform to respond to network bottlenecks.

Hardware options

Consider the following items:

- Use 32-bit bus cards and LAN adapters. These adapters are faster, more efficient, and do not consume a significant amount of CPU time. A good LAN adapter will incorporate features to reduce contention on the network as well as provide good performance with minimum interrupt generation.
- You can also move to a high speed LAN adapter, such as 100 Mbps Ethernet, ATM, or FDDI and backbone devices.
- The network adapter's speed impacts the overall performance of the file server in the following ways:
- It improves the maximum peak transaction rate of the server.
- It may only slightly affect performance under heavy loads because the server is disk bound as it waits for seeks. In this case, the total network transfer time is a relatively small component of the overall transaction.
- As network adapter performance improves, increased CPU performance is required to service the increased request rates from the users on the LAN.

- Balance the network load using segmentation or by having more than one LAN adapter in the server. Alternatively, set workgroup switches to isolate traffic to appropriate segments. Switches serve no useful purpose if most traffic has to travel from one port to another on the same segment. The secret of using switches is to keep local traffic within a segment and stop it from interfering with another segment.
- Create multiple networks or subnetworks. This helps in handling unnecessary broadcasts over the network.
- Upgrade to better performing routers and bridges.
- Add more servers to the network. In this way, you distribute the processing load to other servers.

Software options

Consider the following items:

- Use recent LAN adapter device drivers. LAN adapter manufacturers usually develop their latest drivers to address bugs, and inefficiency in previous releases. The driver should fully support the latest Windows NT networking revisions.
- You should remove network protocols and services not being used to free up the system's needed resources. From the **Control Panel -> Network -> Services**, select the services or services you want to remove and click on the **Remove** button. You have to restart your server to complete the process.

The following figure (Figure 22) shows you how to remove network services from the Windows NT server.

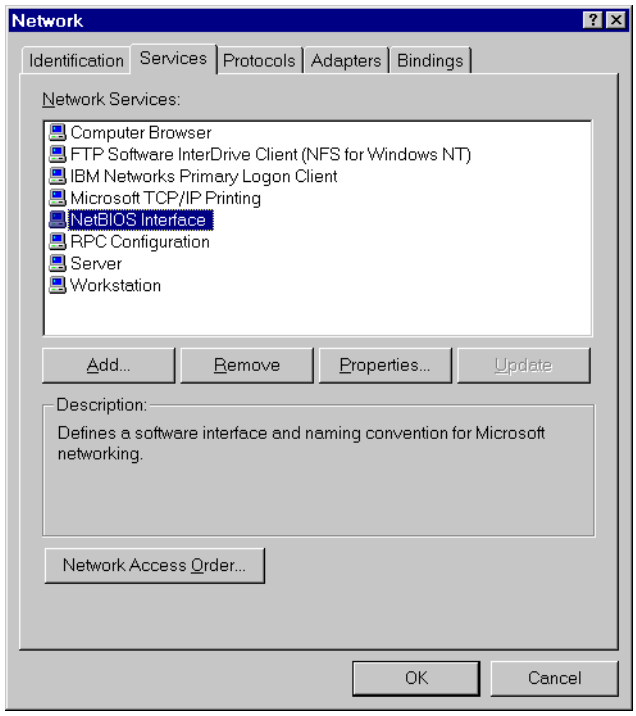


Figure 22. Removing network services from Windows NT server

Network Design Issues

Consider the following items:

- Optimize the protocol binding order. The system will look at the first loaded protocol to fulfill any network I/O requests. You should load the most frequently used protocol first to get the best performance.

  Performance benefits can be gained by optimizing the binding order. For example, if you have a relatively small network, NetBEUI is an appropriate protocol that performs faster. If you need to communicate with remote servers via TCP/IP on a less frequent basis, it would be better to configure your protocol binding order to make NetBEUI first.

  Go to the network bindings window (**Control Panel -> Network -> Bindings**) as shown in the following figure (Figure 23).

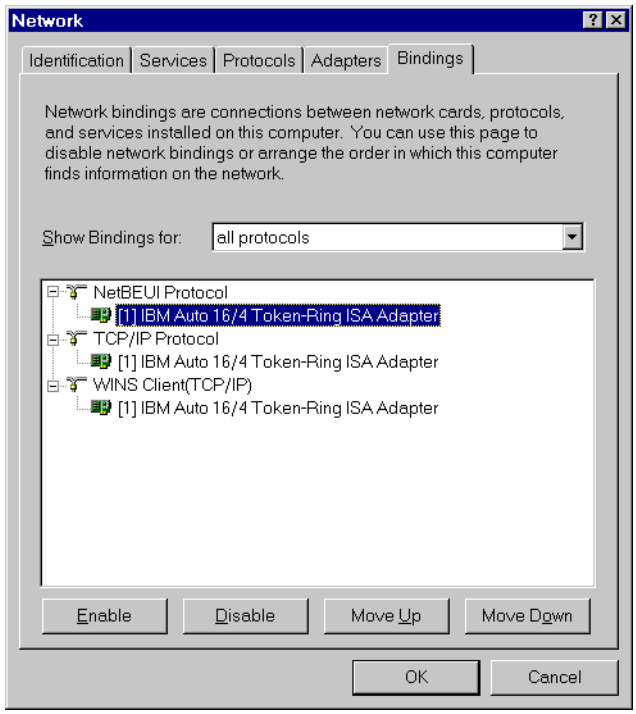


Figure 23. Modifying network protocol binding order

By selecting a protocol and clicking on the **Move Up** and **Move Down** buttons, you will be able to change the binding order of your protocols.

- Use protocols efficient enough to handle the type of your network workload. As much as possible keep your transport protocols to a minimum. Too many protocols makes it hard for troubleshooting and maintenance. Remove protocols not needed for network communications. The following figure (Figure 24) shows how to remove unnecessary protocols.

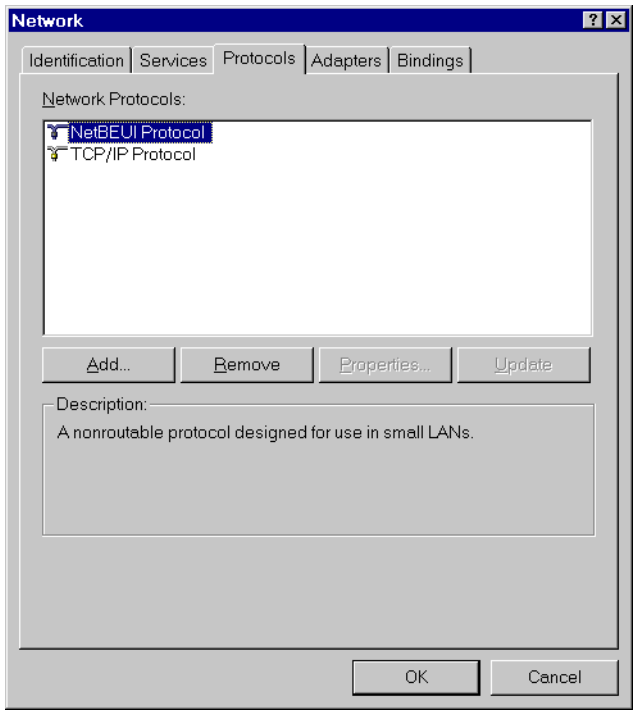


Figure 24. Removing network protocols from Windows NT server

- Move printers and users to other servers to lessen the load on your server.
- Network cabling hints include:
  1. Keeping 10Base-T devices to less than 35 per segment.
  2. Using good-quality Category 5 cable in UTP installations.
  3. Keeping Category 5 cable under the 100 meters total length limit.
  4. Keeping 10Base2 segments to under 185 meters.
  5. Keeping the Token-Ring devices to less than 75 per ring.
  6. Checking for cabling faults, which causes frequent network slowdowns.

Monitoring network protocols

Other than the performance objects and counters, it is also important to monitor how network protocols affect the network. Network protocols installed on your Windows NT affect the number of broadcasts and retransmissions. By monitoring the right counters for the protocols you selected, you can have a better understanding of the use of the network bandwidth.

The first step in optimizing the network component is to gather a baseline measurement of its current performance. Network activity can occur anywhere in the network subsystem especially on the following Windows NT components:

- Redirector
- Server
- Network Interface
- Network Segment

One of the best means to measure and optimize the network is trend analysis on data logged over a period of weeks, months, especially during the peak of network activities. This reflects the actual network usage. This is invaluable in determining when, and if, new hardware must be added.

The network performance object counters that should be investigated are presented at the end of the chapter in the subject Performance Monitor. For Network Performance, you should examine the indications of network bottlenecks based on the object counter readings.

3.2.6 NT performance tuning for Tivoli

There are several actions that can directly improve the performance of a Windows NT machine running Tivoli applications. These are explained in the following section.

3.2.6.1 Configuring the Windows NT System cache

To configure server network services, set on Maximize Throughput for Network Applications.

This option allows the tuning of system-wide caches and optimizes performance for servers providing distributed applications. The way to set this option is: Select **Control Panel -> Network -> Services -> Servers -> Properties** (refer to the Figure 25).

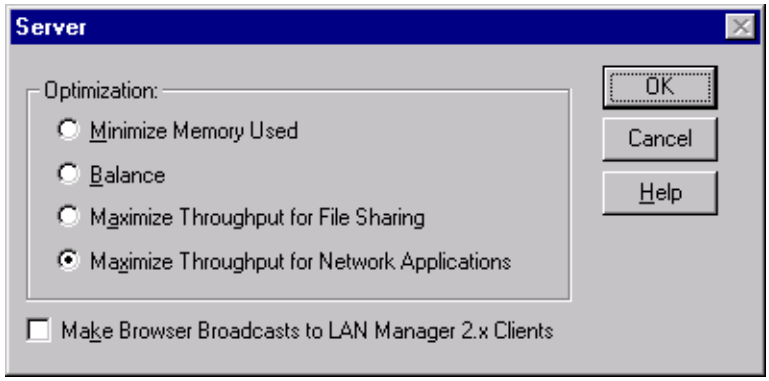


Figure 25. Windows NT system cache setting configuring multitasking option

Set the Performance Boost to the low setting if possible

With this option you can select how Windows NT preemptively prioritizes process threads that the CPU has to attend to. Preemptive multitasking is a methodology that can halt the execution of a process and start the execution of another process at the discretion of the operating system. Be sure to set your application performance accordingly to get the best results. You can do this by clicking on the **Windows NT System Properties** panel. There are three settings for the boost indicator that determine the priority of foreground applications:

- Maximum: For the best response time for the foreground programs.
- Middle: Gives background programs a better response time but still gives more processor time to the foreground programs.
- None: Gives all programs equal amounts of processor time. You should set the Boost indicator to **None** if you are not planning to run applications interactively on the server console. This is the best options for Windows NT machines running Tivoli applications.

You can select this option in Windows NT 4.0 from **Control Panel -> System -> Performance**. Set the Performance Boost to the low setting if possible (refer to the following figure, Figure 26).

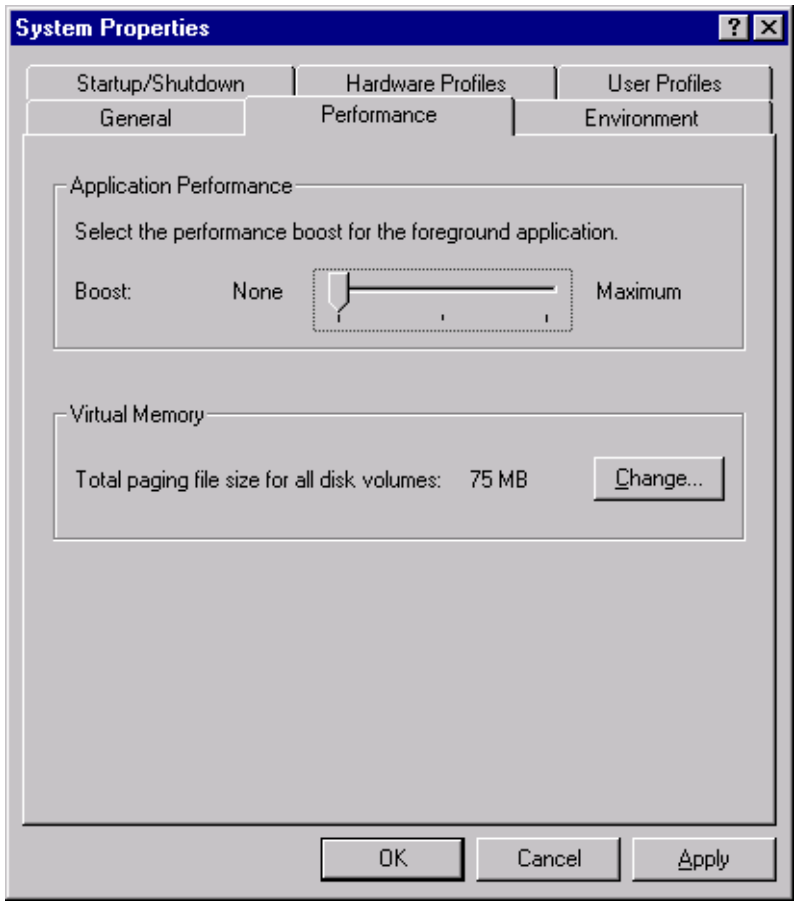


Figure 26. Windows NT application performance boost setting

3.2.6.2 Logging off the server

A simple, but important step, in maximizing your Windows NT Server performance is to keep local users logged off. A locally logged-on user uses a significant amount of CPU time. Windows NT must allocate a certain percentage of CPU time, which could reach 50 percent depending on which applications are being run, to service the logged-on user's session. This will impact the CPU performance servicing other applications and services running on the system.

3.2.6.3 ScreenSavers and wallpaper

Avoid the ScreenSavers. These have dramatic effects on performance. GL Pipes sometimes waste 100 percent of CPU.

Also, use no desktop wallpaper or use smaller wallpaper with a lower bit depth.

3.2.7 Windows NT performance monitor

Windows NT provides a very useful tool for diagnosing performance bottlenecks. The utility is the Performance Monitor (Start -> Programs -> Administrative Tools (Common) -> Performance Monitor). The following figure (Figure 27) shows the Windows NT Performance Monitor.

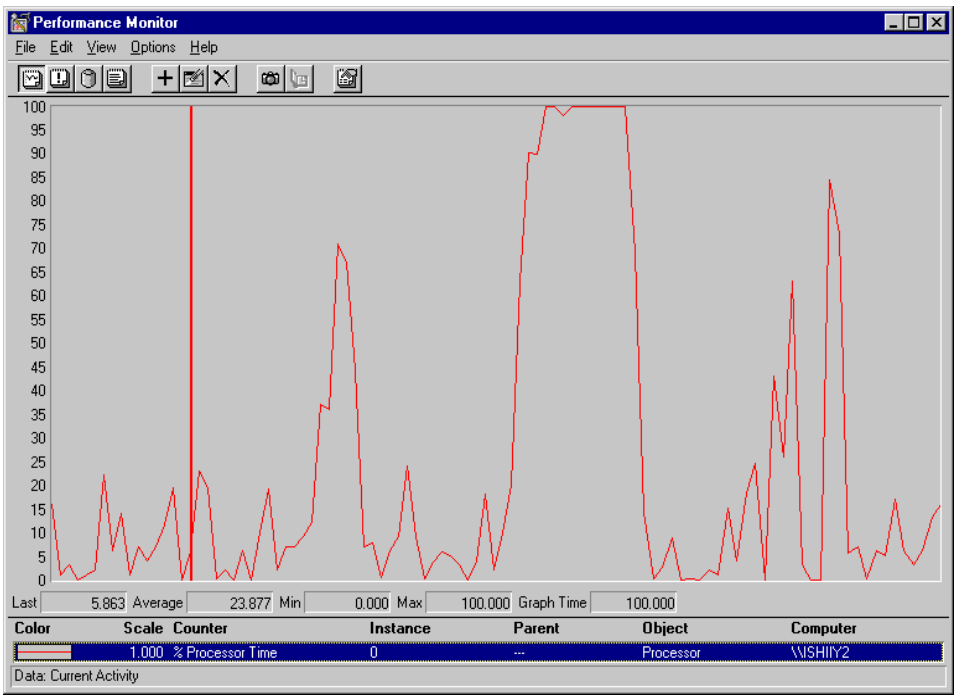


Figure 27. Windows NT performance monitor

With the Performance Monitor you can visually graph data or send it to a log file. The Performance monitor can track a variety of computer functions, such as: processor performance, disk performance, memory performance, process performance, and many more. The most interesting key objects that you should be monitoring are discussed in the following sections.

3.2.7.1 For processor performance

Processor: % time processor. Is one of the most significant item. This object shows how much of the time the processor is actually being used. If it is

usually over 80 percent, either a second processor or a more powerful processor may be needed.

Processor: % interrupts / sec (). Shows how many interrupts from hardware devices are being serviced by the processor.

System: Processor queue length. Shows how many threads are currently in the processor queue waiting to be processed. If it consistently reaches values above 2, then this could also indicate a processor bottleneck.

3.2.7.2 For disk performance

PhysicalDisk: %Disk Time, PhysicalDisk: %Disk Queue Length, PhysicalDisk: Avg. Disk sec/Transfer and Avg. Disk Bytes/Transfer.

If the PhysicalDisk: %Disk Time is high, check the PhysicalDisk: Queue Length item. If the PhysicalDisk: Queue Length item is two times the number of spindles on the physical disk, then the disk is being accessed more than it can handle.

If the PhysicalDisk: Avg. Disk sec/Transfer value is high, usually greater than 0.3 seconds, the system may be retrying the disk because of failures. If the Avg. Disk Bytes/Transfer drops below 20 KB on a regular basis, then applications may be accessing the disk inefficiently.

3.2.7.3 For memory performance

Memory: Pages/sec. Servers should never regularly page memory to disk. Paging I/O should occur only occasionally, for example, when applications are initializing, but never on a continuous basis.

Average memory utilization should not exceed 70 percent of available memory.

3.2.7.4 For network performance

You should monitor the following items for detecting Network Bottlenecks:

Redirector: Bytes Total/sec. This is the rate the Redirector is processing data bytes. This includes all application and file data in addition to protocol information such as packet headers. This number should not reach the maximum rated speed of the network medium or a sustained value greater than 50 percent of its capacity. If it does, consider segmenting your network.

Redirector: Current Commands. This counts the number of requests to the Redirector that are currently queued for service.

If this value is greater than one per network adapter, then one of the following may be occurring:

- The redirector may be the bottleneck.
- The server that the redirector is communicating with is slower than the redirector.
- The network may be experiencing capacity problems.
- The redirector is busier than the adapter can keep up with.

It may be necessary to subnet the network in an attempt to partition network traffic.

Redirector: Network Errors/sec. This counts serious unexpected errors that generally indicate the Redirector and one or more servers are having serious communication difficulties. If any network errors are logged, check the Event Viewer for more details. This can be a sign of problems with the LAN adapter or LAN adapter driver, the hub or switch, or the cabling.

Redirector: Reads Denied/sec. This is the rate that the server is unable to accommodate requests for raw reads. High values may indicate that remote servers are having problems with memory allocations.

Redirector: Writes Denied/sec. This is the rate that the server is unable to accommodate requests for raw writes. High values may indicate that remote servers are having problems with memory allocations.

Server: Bytes Total/sec. The number of bytes the server has sent to, and received from, the network. This value provides an overall indication of how busy the server is. If the value is consistently at or over 50 percent of the network capacity, you may need to put the server into a faster switch or hub.

Alternatively, you can add another LAN adapter to balance the load.

To monitor other useful PerfMon objects, such as a Network Interface, you need to install the SNMP Management and Network Monitor Tools and Agent services located on the Control Panel -> Network -> Services tab. Installing the Network Monitor also installs the Network Segment object that can be used to monitor various aspects of network segment performance. After installation, you need to reboot your server.

Monitoring additional objects

Server: Work Item Shortages. The number of times STATUS_DATA_NOT_ACCEPTED was returned at receive indication time. This occurs when no work item is available or can be allocated to service the

incoming request. This could indicate a very busy server and a lack of physical memory.

To service other requests, you may want to turn off some of the services from the server.

Server: Pool Nonpaged Failures. The number of times allocations from the non-paged pool have failed. Indicates that the server's physical memory is too small. Increasing values for this counter indicate a shortage in physical memory.

Network Interface: Output Queue Length. This is the number of packets in the output queue. If this value is sustained at more than two, then the network is the bottleneck. Underlying hardware may need to be reconfigured or upgraded.

Network Interface: Bytes Total/sec. This counter reflects the number of bytes that are sent and received through the interface.

A sustained value of 80 percent indicates that the LAN adapter and media access protocol are the bottleneck.

You should consider segmenting and/or subnetting your network.

Network Interface: Current Bandwidth. This counter is an estimation of the current bandwidth on the interface in bits per second (bps).

Network Interface: %Network Utilization. This is the percentage of the network bandwidth that is in use. You can determine how much of the network capacity is being used by comparing this value to the Current Bandwidth counter.

Network Interface: %Broadcast Frames. This counts the percentage of frames that are broadcast traffic as opposed to multicast and unicast traffic. If this value reaches near 100 percent in a second, you should look for the traffic source immediately. Large number of broadcasts can be caused by a badly configured bridge, a chattering LAN adapter, or Windows NT network resolving NetBIOS names to a physical address.

You should do network segmentation or install network switches to reduce the traffic.

Network Interface: Total Bytes Received/sec. This counter indicates the total number of bytes received from a particular network. If the total is getting close to the maximum network speed, pay attention to the Output Queue Length. If

the number is too high, you need to add LAN adapters and do subnetting or add switches.

Sustained values of greater than the half network capacity indicates over utilization, which calls for segmentation and/or putting in a faster network device.

3.3 Conclusion of operating system performance

The performance of all Tivoli management applications depends on the operating system and network performance (refer to the Figure 21 on page 39). Tuning the operating system and network in your TMR are mandatory, especially in the following systems:

- TMR server
- TEC server
- Endpoint Gateway
- RIM host
- RDBMS server
- MDist repeater

In this chapter, we mainly discussed the overview of AIX and Windows NT performance tuning. If you need more detailed information about this or other operating system tuning, please refer to the appropriate manuals and documents.

Chapter 4. Improving Management Framework performance

Tivoli Management Framework provides a set of common services and facilities that enable many systems management applications. Tivoli Management core applications run on the Tivoli Management Framework and use the services and resources that are provided by the Tivoli Management Framework. The Tivoli Management Framework can affect the performance of the Tivoli Management applications that run on the framework. The following figure (Figure 28) shows the relationship between the Tivoli Management Framework and the Tivoli Management applications.

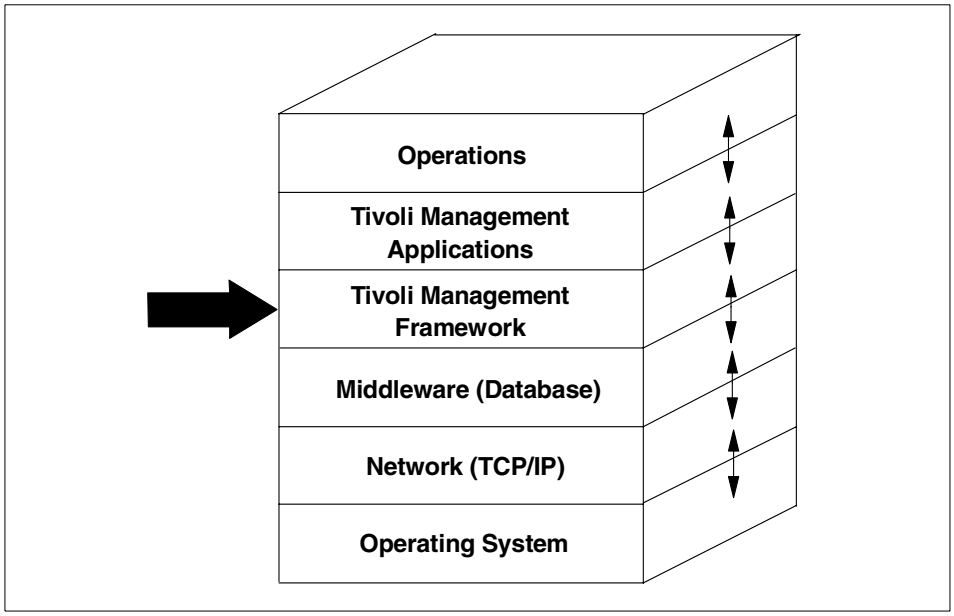


Figure 28. Tuning Tivoli Management framework

This chapter introduces the important issues for improving performance and throughput in Tivoli Management environment and how you should configure your Tivoli Management Framework for the Tivoli Management applications.

4.1 Tivoli Management Agent internals

In the three-tiered management structure, each Endpoint Gateway communicates with the Endpoints that have logged into the Endpoint Gateway by using downcalls and upcalls. When the Endpoint Gateway requests the Endpoint to perform some operations, the Endpoint Gateway

issues a downcall. On the other hand, when the Endpoint requests the Endpoint Gateway to perform some operations, the Endpoint issues an upcall.

Understanding this interaction between the Endpoint Gateway and Endpoints is important to improve their performance. In this section, we introduce how the Endpoint Gateway works when a downcall or an upcall is issued and how the Endpoint works when a downcall or an upcall is issued.

4.1.1 Downcall operation

Normally, when the Endpoint Gateway attempts to invoke an Endpoint method on the Endpoint, a downcall is issued by the Endpoint Gateway. The following figure (Figure 29) shows how each management resource works when a downcall is issued.

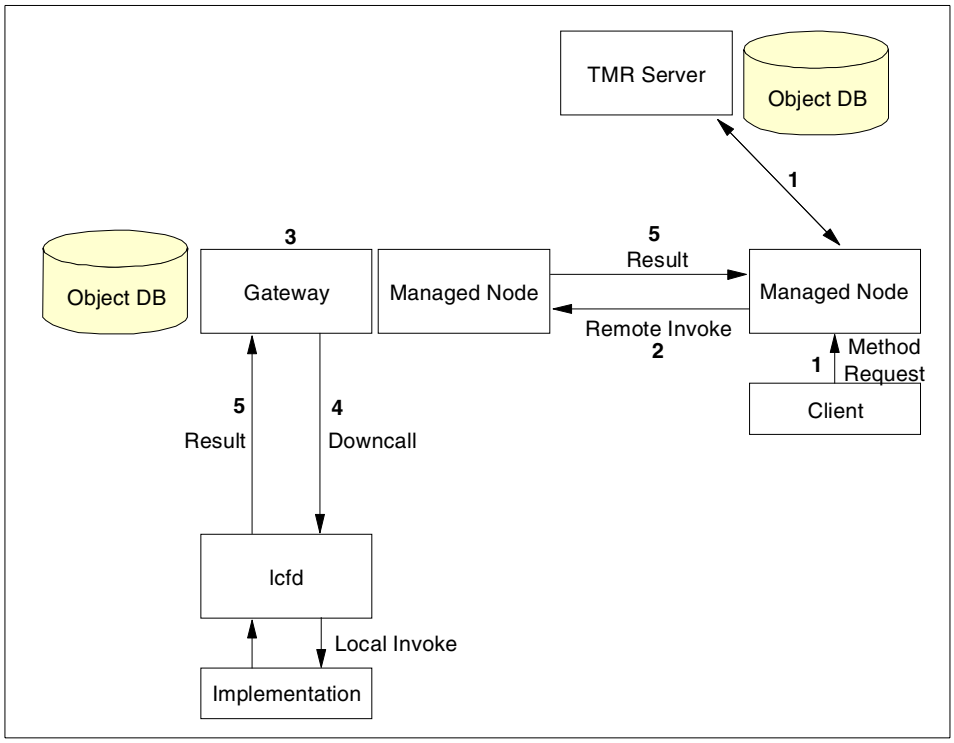


Figure 29. Method invocation with downcall

1. When an application requests the Endpoint to invoke a method, the TMR server maps the dispatcher in the object to the Endpoint Gateway object ID. The Endpoint Gateway object ID is used to determine which Managed

Node the Endpoint Gateway is running on. In this case, the plus (+) in the object reference designates that the object runs on the Endpoint.

2. The TMR server object dispatcher sends the request to the Endpoint Gateway servicing the Endpoint.
3. The Endpoint Gateway resolves the method on the corresponding behavior object and performs authorization of the invocation by invoking the Tivoli principal.
4. The method runs on the Endpoint as a downcall. The parameters are sent then the method runs, and the MDist data is transmitted from the Endpoint Gateway if it is needed.
5. The results are passed back to the Endpoint Gateway. The Endpoint Gateway then returns the results to the caller.

4.1.2 Upcall operation

Normally, when the Endpoint attempts to invoke a method on the Managed Node, such as the Endpoint Gateway, remotely, an upcall is issued by the Endpoint. The following figure (Figure 30) shows how each management resource works when an upcall is issued.

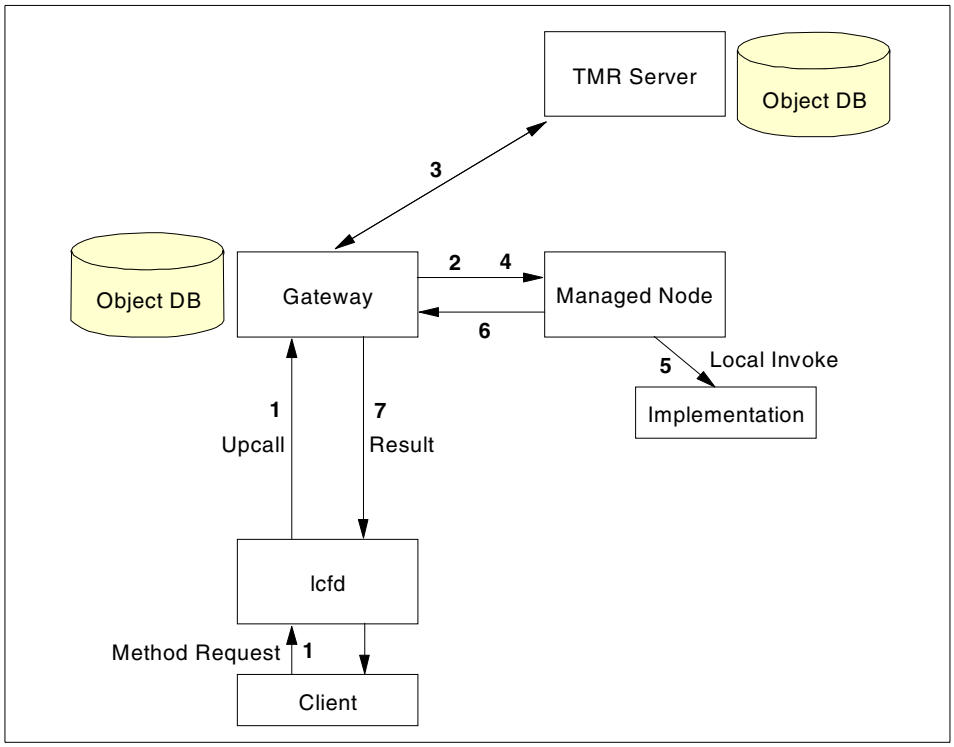


Figure 30. Method invocation with upcall

1. The Endpoint sends a request to the Endpoint Gateway with an upcall.
2. If the Endpoint Gateway can resolve the method header, it tells the object dispatcher on the Managed Node to invoke the method.
3. If it cannot resolve the method header, it goes to the TMR server to get the header information.
4. When the header is returned to the Endpoint Gateway, the Endpoint Gateway tells the Managed Node to invoke the method.
5. The method is invoked on the Managed Node.
6. The results are passed back to the Endpoint Gateway.
7. The Endpoint Gateway passes the results back to the Endpoint.

4.1.3 Bulk Data Transfer and IOM channel

Bulk Data Transfer (BDT) is a technique used when large data transfers (normally greater than 16 KB) are required between objects on separate

systems. This mechanism is provided by Tivoli Management Framework, and it works quite effectively in the Software Distribution process.

Rather than have the data go from object on system A through oserv on system A to oserv on system B to an object on system B, BDT utilizes a technique known as Inter-Object Messaging (IOM Channel). An IOM Channel establishes a direct network connection between two methods (hence, inter-object). This not only makes transferring large amounts of data more efficient, but there is also less work for an oserv. Therefore, BDT and IOM Channel:

- Are used anytime the data being transferred is greater than 16 KB in size.
- Do not use port 94 for transferring the data.
- Are not just used during multiplexed distributions.

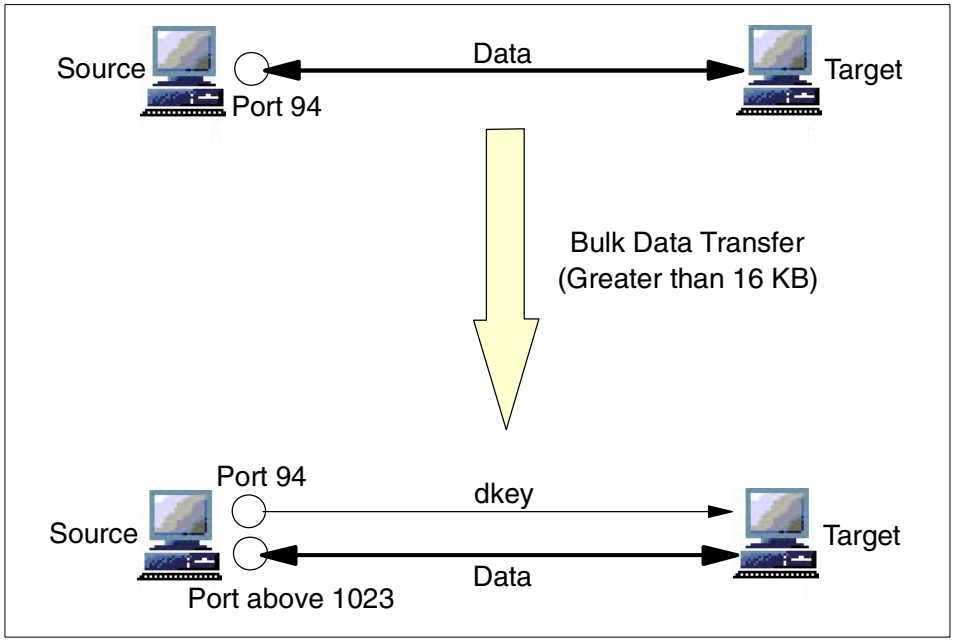


Figure 31. BDT and IOM channel

4.1.4 MDist repeater

The MDist repeater is one of the most important objects for improving performance. To improve your environment, you have to tune or customize your repeater. Fortunately, the MDist repeater is highly customizable and has many tuning parameters. This section introduces these repeater parameters and how they affect your environment.

4.1.4.1 Where is the repeater?

The TMR server is always initially a repeater. If you do not create any repeaters, the TMR server will be your repeater.

The Endpoint Gateways are configured as MDist repeaters automatically when you create them; so, you do not need to do any operation to create an MDist repeater in Tivoli management environments. In other words, the Endpoint Gateway has the MDist repeater capability built in. If you are currently using a Managed Node to repeat for MDist, and you then create an Endpoint Gateway on the Managed Node, that Endpoint Gateway becomes the MDist repeater. The Endpoint Gateway is a program that runs on the Managed Node. The Endpoint Gateway can service both Endpoints and the Managed Node.

During the Endpoint Gateway creation process, the MDist repeater object that is created on Managed Node is removed, and then the new repeater object is created on the Endpoint Gateway. However, all parameters defined on the former MDist repeater are passed to the new MDist repeater; so, your tuning configurations for the former MDist repeater will be kept.

4.1.4.2 Repeater tuning parameters

The MDist repeater provides many tuning parameters. To tune repeaters effectively, these parameters should be configured properly. If you set a parameter incorrectly, the repeater will not work, or some errors will occur. Therefore, the most important thing here is to understand the meanings of these parameters.

net_load The net_load parameter sets the kilobyte per second (Kb/sec) used in each distribution. This parameter should be tuned for the LAN and WAN links. This parameter is the reason why you should have a repeater at the central site for each WAN type. You can tune that repeater for each link. The default value is 500 (Kb/sec).

stat_inv The stat_inv parameter specifies a timeout value that times how long it will take to send each Tivoli packet (typically 16 KB). If your network is too slow or saturated, stat_inv parameter needs to be higher. The default value is 180 (sec).

max_conn The max_conn parameter sets the number of machines distributed to, at one time, from that repeater. Each open connection uses 0.5 to 2 MB of real memory. You should keep this parameter low unless you have a powerful repeater machine, such as a multiprocessor machine. The default value is 100 (connections).

mem_max The mem_max parameter specifies the value for the maximum amount of memory space that can be allocated to a repeater during a distribution. This means where data is initially spooled to the repeater. This area is real system memory. If this setting is too high, the repeater can run out of real memory. 10 MB is a good starting point. The amount can be as low as 4 MB. The default value is 10 (MB).

disk_max The disk_max parameter specifies the value for the maximum amount of disk space that can be allocated to a repeater during a distribution. This means where data spools after mem_max is full. The disk_max must be at least 10 percent or 20 percent larger than your largest distribution (for example, file package or auto pack). Software Distribution file packages are usually the largest distribution in most environments; so, check the size of those first. The default value is 50 (MB).

disk_hiwat The disk_hiwat parameter specifies the point where the repeater tells the TMR server it is at its disk_max and is about to fill up, and the TMR server should slow down sending data. It should be larger than your largest distribution but smaller than disk_max. The default value is 50 (MB).

disk_dir The disk_dir parameter specifies the directory or file system to which data spools. This file system that is specified by the disk_dir must be larger than disk_max. The default is /tmp.

These repeater tuning parameters are particularly significant for Software Distribution performance; so, please refer to Chapter 5, “Improving software distribution performance” on page 133 for more information about configuring and tuning MDist repeaters.

4.2 Detecting Tivoli Management Agent performance bottleneck

Tivoli Management Framework provides base services to Tivoli Management applications that run on the framework. Later in this redbook we will introduce the bottlenecks of each Tivoli Management application. In this section, we introduce the basic considerations when deploying the Tivoli Management environment. The following figure (Figure 32) shows the performance bottlenecks in the Tivoli Management Framework.

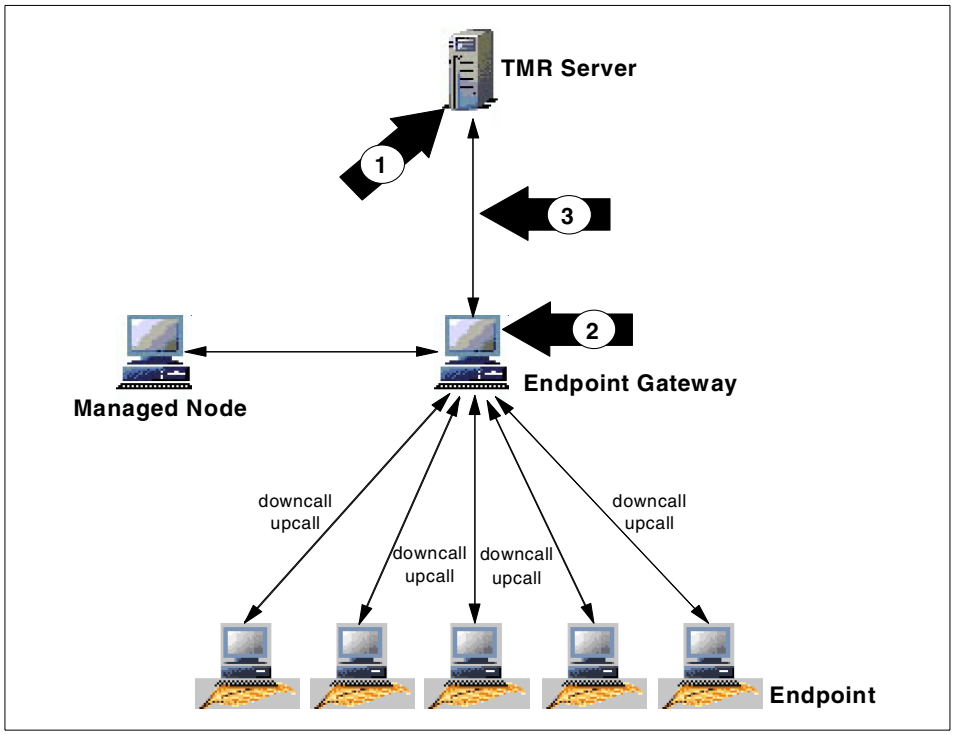


Figure 32. Tivoli Management Agent performance bottleneck

1. TMR server (Endpoint Manager)
2. Endpoint Gateway
3. Network performance

These bottlenecks can be a problem in most large scale management environments. The following are descriptions for each performance bottleneck in the Tivoli Management Framework.

TMR Server

In Version 3.6 of Tivoli, the new architecture is designed to allow Endpoint Gateways to off-load many of functions formerly performed by the TMR server. However, the TMR server still actually performs many processes for managing entire managed objects. For example, the Tivoli Name Registry (TNR) exists only on the TMR server, and the TMR server processes all lookup requests from managed resources. Therefore, the TMR

server is the biggest performance bottleneck even in the Tivoli 3.6 management environment.

Endpoint Gateway The Endpoint Gateway is a new object type that is added by the LCF architecture. The Endpoint Gateway manages Endpoints that have logged into the Endpoint Gateway. Therefore, in the three-tiered management structure, all requests from the Endpoints that are managed by the Endpoint Gateway must be received by the Endpoint Gateway, and most of them will be processed by the Endpoint Gateway instead of the TMR server. In other words, the Endpoint Gateway plays the role of the mid-level manager in the TMR. The Endpoint Gateway works not only as the manager of Endpoints but also as the MDist repeater. Therefore, the load of the Endpoint Gateway will be heavy. In the Endpoint Gateway configurations, the MDist repeater tuning is required.

Network Performance In the Tivoli Management environment, the network performance tuning is critical because most of the Tivoli Management operations are performed through the network. In other words, the network may adversely affect the performance of most of Tivoli Management operations if your network has a performance problem.

4.3 Tivoli Management Agent performance improvement strategy

Tivoli Management Framework provides the base services for managing your distributed systems environment using Tivoli Management applications. Tuning the Tivoli Management Framework is very important because the Tivoli Management Framework performance may affect all Tivoli applications that run on the Framework. In this section, we introduce how you should configure or customize your Tivoli Management Framework to improve its performance and provide useful information for Framework performance improvement.

4.3.1 Tivoli Framework performance improving process flow

The following figure (Figure 33) shows the approach for improving the performance and throughput of Tivoli Management Framework. When you

configure or tune Tivoli Management Framework in your management environment, we recommend you to follow this process flow chart.

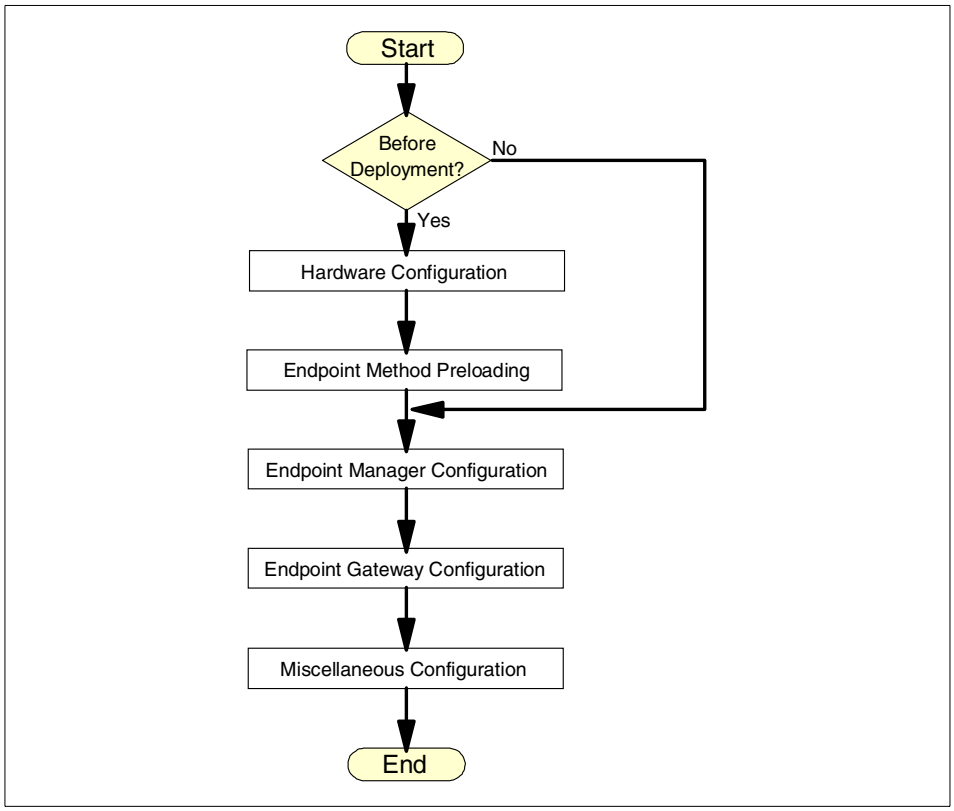


Figure 33. Tivoli Framework performance improvement process flow

In the following sections, we will introduce detailed information about how to configure and tune your framework in a large-scale management environment.

4.3.2 TMR server (Endpoint Manager) configurations

In the three-tiered management structure, the TMR server still performs many processes, even if the Endpoint Gateway has lightened the load on the TMR server. There are a few configurable parameters that affect the performance on the TMR server. In this section, we introduce and describe how these parameters work.

4.3.2.1 Endpoint policies on TMR server

You can configure the Endpoint's login behavior and communication patterns by developing scripts that execute at various times in the process. There are four such hooks. Of these, only the login_policy runs on the Endpoint Gateway; the other three run at the Endpoint Manager. These are the allow_install_policy, select_gateway_policy, and after_install_policy. The following figure (Figure 34) shows each Endpoint policy and where these Endpoint policies are executed.

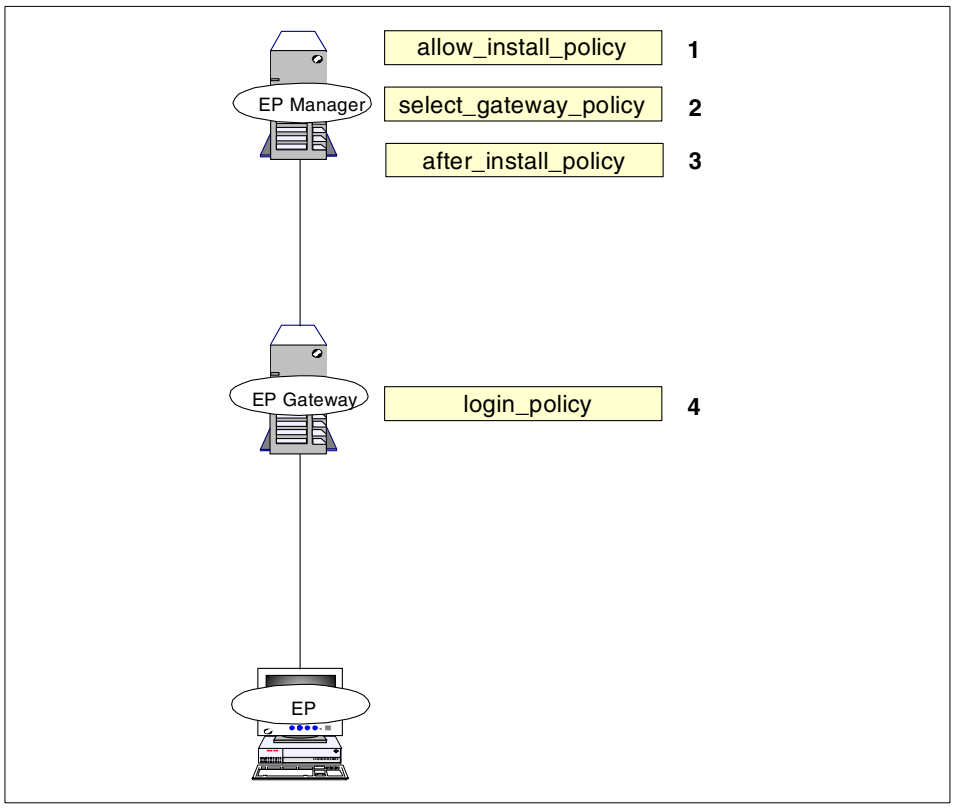


Figure 34. The Endpoint policies

As you can see, the following Endpoint policies are executed on the TMR server when the Endpoint attempts to perform initial login.

- allow_install_policy
- select_gateway_policy
- after_install_policy

In Version 3.6.1 of Tivoli, you can define the number of concurrent executions of each Endpoint policy on the TMR server instead of using the max_jobs attribute.

4.3.2.2 Controlling Endpoint policy concurrent executions

In Version 3.6.1 of the Tivoli Management Framework, the max_jobs attribute is not supported. Instead of the max_jobs, the following parameters are supported by Version 3.6.1 of the Tivoli Management Framework to control the concurrent execution of each Endpoint policy on the TMR server.

max_install Determines the number of maximum allow_install_policy scripts that are executed concurrently on the TMR server. The default value is 10.

max_sgp Governs the number of maximum select_gateway_policy scripts that are executed concurrently on the TMR server. The default value is 10.

max_after Constructs the number of maximum after_install_policy scripts that are executed concurrently on the TMR server. The default value is 10.

These parameters control the number of respective policy scripts that the Endpoint Manager (TMR server) will run at the same time. All these default to 10, and, thus, at any given time there are a maximum of 30 scripts that can run on the Endpoint Manager (TMR server).

Note

In Version 3.6.0 of Tivoli, Tivoli Management Framework provides the max_jobs attribute to control job requests on Endpoint Manager (TMR server). The max_jobs is similar to the max_concurrent_jobs on Endpoint Gateway. The difference between max_jobs on Endpoint Manager and max_concurrent_jobs on Endpoint Gateway is the type of job requests that are controlled by each attribute. This now applies to max_install, max_sgp, and max_after. Please refer to the 4.3.3.1, “Controlling Endpoint Gateway concurrent jobs” on page 86 for more information about the max_concurrent_jobs attribute on Endpoint Gateway.

4.3.2.3 Configuring max_install, max_sgp, and max_after

We recommend you use the default value for each parameter first, unless you have a large-scale environment, in which case you may need to increase them. When each parameter is exceeded, the job request goes on a wait queue. When there is a slot open, the TMR server starts to process the

request that is queued. Therefore, if the TMR server is busy, you do not get hard failures, but the quality of service suffers. If you notice that service is poor, you should upgrade the hardware configurations of your TMR server or create another TMR server to spread the load by using interconnected TMRs solution.

These parameters (max_install, max_sgp and max_after) will be used for Endpoint initial login. Therefore, if you schedule Endpoint deployment properly in your management environment, these parameters will not be exceeded. Before modifying the parameters, you should consider your deployment plan.

Note

The max_sgp is also used for isolated login. For example, if an assigned Gateway becomes unavailable, Endpoints that are managed by the assigned Gateway will be isolated and perform isolate login. In a large-scale environment, the max_sgp should be configured carefully. Please refer to the 4.3.4, “Controlling Endpoint login requests in a large environment” on page 90 for more information.

When you modify the max_install, and max_sgp or max_after, you can use the `idlattr` command as follows:

```
idlattr -t -s <epmgr_oid> <attr_name> short value
```

Note

The patch 3.6.1-TMF-0035 provides the new command (`wepmgr`) to control Endpoint Manager process and displays and sets the attributes of the Endpoint Manager object. The `wepmgr` command allows you to access the attributes of the Endpoint Manager object without the `idlattr` or `idlcall` command. The following shows the `wepmgr` command usage:

`wepmgr help`	Shows help information.
`wepmgr stop`	Stops the ep_mgr process.
`wepmgr start`	Starts the ep_mgr process.
`wepmgr restart`	Restarts the ep_mgr process.
`wepmgr ping`	Checks if the ep_mgr process is running or not.
`wepmgr get`	Displays Endpoint Manager object attributes.
`wepmgr set attribute value`	Sets Endpoint Manager object attributes.
`wepmgr update`	Refreshes the attributes in the running ep_mgr process from the attributes on the Endpoint Manager object.
`wepmgr fsck`	Rewrites the data in the Tivoli Name Registry (TNR) Endpoint resource from Endpoint data in the Endpoint Manager.

The following shows the sample output of the `wepmgr get` command:

```
# wepmgr get
max_jobs        = 20
max_install     = 10
max_sgp         = 10
max_after       = 10
login_interval  = 270
max_iom_records = 500
epmgr_flags     = 0
#
```

We recommend that you use the `wepmgr` command instead of the `idlattr` or `idlcall` command if the 3.6.1-TMF-0035 is applied in your TMR.

4.3.2.4 Miscellaneous considerations of TMR server

Since the TMR server performs many operations and maintains many object databases including the master database, the TMR server needs a lot of resources such as CPU or memory. Therefore, we strongly recommend you configure the TMR server with a fast processor machine and enough resources (for example, memory or disk) especially in a large-scale environment.

Once you configure your Tivoli Management environment, the hardware migration (for example, replace or upgrade) is complicated even if you make a backup of your environment. So, you should consider this difficulty carefully when you configure your TMR server.

4.3.3 Endpoint Gateway configurations

In the three-tiered management structure, the Endpoint Gateway plays the role of mid-level manager and performs the following operations:

- Listens for Endpoint login requests.
- Listens for upcall requests from Endpoints.
- Listens for method requests from other Managed Nodes (downcalls) and acts as a gateway for method invocation.
- Acts as a MDist repeater.

The following figure (Figure 35) shows the relationship between these operations and the Endpoint Gateway.

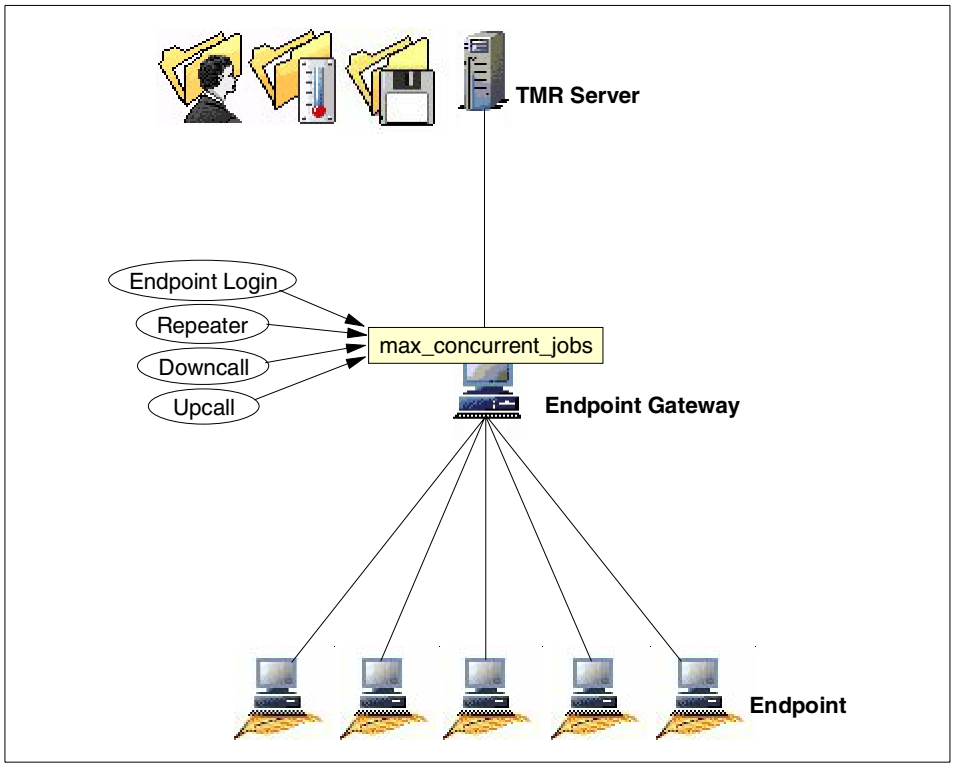


Figure 35. Endpoint Gateway and its jobs

4.3.3.1 Controlling Endpoint Gateway concurrent jobs

As we mentioned, these operations are heavy especially when a single Endpoint Gateway manages many Endpoints. Version 3.6 of the Tivoli Management Framework provides the max_concurrent_jobs parameter to control the concurrent jobs on the Endpoint Gateway. The following is the description of the max_concurrent_jobs parameter.

max_concurrent_jobs Sets the maximum number of concurrent jobs the Endpoint Gateway can run. This setting, which controls the number of threads running in the Endpoint Gateway, is the mechanism for controlling the resources (such as CPU and memory) the Endpoint Gateway uses. The default setting for max_concurrent_jobs is 200.

In this case, a job consists of the following:

- An Endpoint login

- An upcall
- A downcall

For example, a distribution operation consists of the following jobs:

- One job for the input stream.
- One job for each Endpoint being distributed to

Each job runs in its own thread. If the max_concurrent_jobs is exceeded, the job request goes on a wait queue. When there is a slot open, the Endpoint Gateway starts the job as a thread.

Normally, the upcall will always be accepted if the Endpoint Gateway is up. However, if the Endpoint Gateway is busy, the upcall job goes on the wait queue and will be serviced as soon as the Endpoint Gateway can dispatch a thread to perform it (the wait queue is processed in FIFO order). Hence, if the Endpoint Gateway is busy, you do not get hard failures, but the quality of service suffers.

4.3.3.2 Configuring max_concurrent_jobs

Basically, we recommend you to use the default value first. However, if your management environment includes the following situations, you may need to modify (increase) the max_concurrent_jobs parameter.

- A single Endpoint Gateway manages thousands of Endpoints.
- Usually, most Endpoints attempt to perform a log in to the Endpoint Gateway simultaneously.
- There are many Sentry monitors or Event Adapters in your environment, and they generate events periodically.

The worst thing that will happen is that the upcall will have to wait. If there is a problem with many upcalls (for example, TEC events) waiting a long time because the Endpoint Gateway is busy, then you can either make a new Endpoint Gateway and off-load some Endpoints or adjust the number of maximum jobs the Endpoint Gateway will perform simultaneously.

If the Endpoint Gateway is always busy and the Endpoint Gateway receives many job requests constantly, the wait queue will not be able to handle these job requests, and the Endpoints or other Managed Nodes that send the job request to the Endpoint Gateway will receive a timeout error for the request.

If you meet this situation, you need to perform one of the following:

- Modify (increase) the max_concurrent_jobs parameter.

- Create one or more additional Endpoint Gateway in order to spread the load.
- Upgrade the hardware configurations of the Endpoint Gateway.
- Migrate the Endpoint to other Endpoint Gateways.

Modify the max_concurrent_jobs parameter with caution. This may affect the overall performance of the Endpoint Gateway, and the proper value depends on your environment, for example, how many Endpoints a single Endpoint Gateway manages. Therefore, if you modify max_concurrent_jobs, start with the default max_concurrent_jobs (200) and increase the max_concurrent_jobs slowly until the requested jobs are handled without a problem.

4.3.3.3 The max_rpc_threads parameter

The Managed Node (Endpoint Gateway) provides the max_rpc_threads parameter to control the maximum number of concurrent threads. By default, max_rpc_threads is set to 250. However, in a large-scale management environment, this can cause the Endpoint login failures when many Endpoints attempt to simultaneously log in to the Endpoint Gateway even if you increase the max_concurrent_jobs parameter.

Basically, we recommend you to use the default value of the max_rpc_threads parameter. However, if you increase max_concurrent_jobs, we recommend you to increase max_rpc_threads as well. The max_rpc_threads parameter should be a little greater than max_concurrent_jobs.

```
max_rpc_threads > max_concurrent_jobs
```

You can modify the max_rpc_threads parameter by using the `odadmin` command as follows:

```
odadmin set_rpc_max_threads value
```

Note

The max_rpc_threads parameter affects not only the maximum number of the concurrent threads, but also the maximum number of concurrent RPC threads. In other words, both maximum numbers refer to the max_rpc_threads parameter definition.

This is modified by the Patch 3.6-TMF-0033. Therefore, in the lower version of Tivoli Management Framework, or before applying this patch, the maximum number of concurrent threads is hard-coded to 250; so, you cannot modify this value.

4.3.3.4 Miscellaneous considerations of Endpoint Gateway

The Endpoint Gateway needs a lot of resources to work as well as the TMR server works. Therefore, we strongly recommend you configure the Endpoint Gateway with sufficient resources, such as fast CPU and enough memory to manage Endpoints that log into the Endpoint Gateway. The essence requirements depend on your management environment, for example, the number of the Endpoints or the Tivoli Management applications that run in your environment. Resource utilization is particularly heavy if you use Software Distribution, since the Endpoint Gateway performs the MDist repeater functions.

Fortunately, the Endpoint Gateway is more configurable than the TMR server. Therefore, creating additional Endpoint Gateways is easy. In Endpoint Gateway tuning, creating additional Endpoint Gateways or migrating the Endpoints to other Endpoint Gateways, can be an alternative solution rather than modifying max_concurrent_jobs.

Note

Sometimes, in the Endpoint Gateway that runs on AIX operating system, Tivoli operation may fail because of the Maximum number of PROCESSES allowed per user parameter. By default, AIX operating system allows each user to run up to 40 processes. If the Maximum number of PROCESSES allowed per user parameter is exceeded, the next process will be failed.

You can modify this parameter by using SMIT as follows:

```
# smitty system
    -> Change / Show Characteristic of Operating System
```

Then, modify the value of the following field:

```
Maximum number of PROCESSES allowed per user
```

In this parameter, this limit does not apply to the root user.

4.3.4 Controlling Endpoint login requests in a large environment

As we mentioned, Endpoint Manager and Endpoint Gateway provide some parameters that can control or filter Endpoint login requests. These parameters should be configured carefully to control Endpoint login requests, especially in a large-scale environment. In this section we describe how these parameters work and how you should configure them.

4.3.4.1 Parameters related to Endpoint login requests

The following figure (Figure 36) shows parameters related to Endpoint login requests.

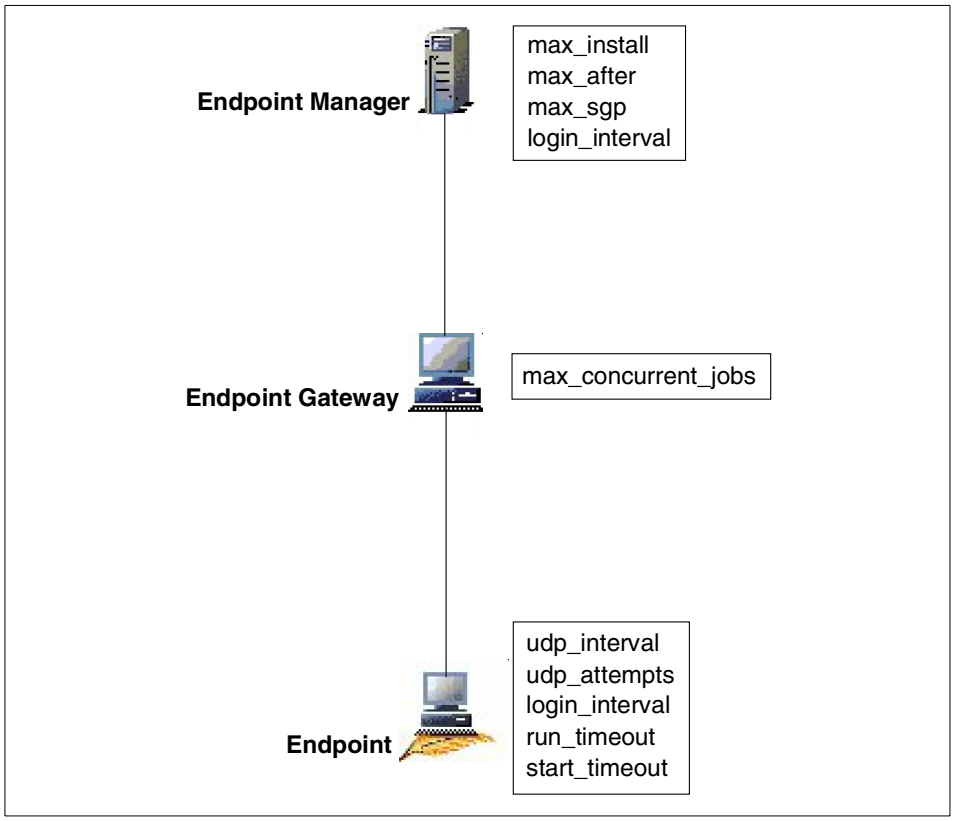


Figure 36. Filtering options for Endpoint login requests

As you can see, many parameters affect Endpoint login. Each parameter plays the following role in Endpoint login process.

Endpoint Manager and Endpoint Gateway parameters

Endpoint Manager (TMR server) and Endpoint Gateway provide the following parameters to control Endpoint login requests.

max_install Refer to the 4.3.2, “TMR server (Endpoint Manager) configurations” on page 80.

max_after Refer to the 4.3.2, “TMR server (Endpoint Manager) configurations” on page 80.

max_sgp Refer to the 4.3.2, “TMR server (Endpoint Manager) configurations” on page 80.

max_concurrent_jobs Refer to the 4.3.3, “Endpoint Gateway configurations” on page 85.

login_interval Provides login_throttling mechanism. This is governed by an attribute of EndpointManager object. Endpoint logins from the same Endpoint which appear less than login_interval seconds apart are ignored. This interval thus prevents the duplication effects of multiple initial logins from the same Endpoint (same IP address) via multiple intercepting Endpoint Gateways.

Endpoint parameters

Endpoint provides the following parameters to control Endpoint login requests.

udp_interval Specifies the number of seconds between Endpoint initial login request attempts.

udp_attempts Specifies the number of times an Endpoint will transmit an initial login request.

login_interval Specifies the waiting period before an Endpoint executes another login attempt.

run_timeout Specifies the wait time (in seconds) before a communication timeout occurs following a successful login.

start_timeout Specifies the wait time (in seconds) before a communication timeout occurs during login.

4.3.4.2 Understanding Endpoint login process and parameters

To control Endpoint login requests efficiently, understanding Endpoint login process is important. Endpoint initial login has the following three major phases:

- The Endpoint establishes communication with the TMR.
- The Endpoint Manager selects the Endpoint Gateway for the Endpoint.
- The Endpoint receives its Gateway assignment information and performs the initial login to the assigned Gateway.

The following figure (Figure 37) shows three major phases of Endpoint login process.

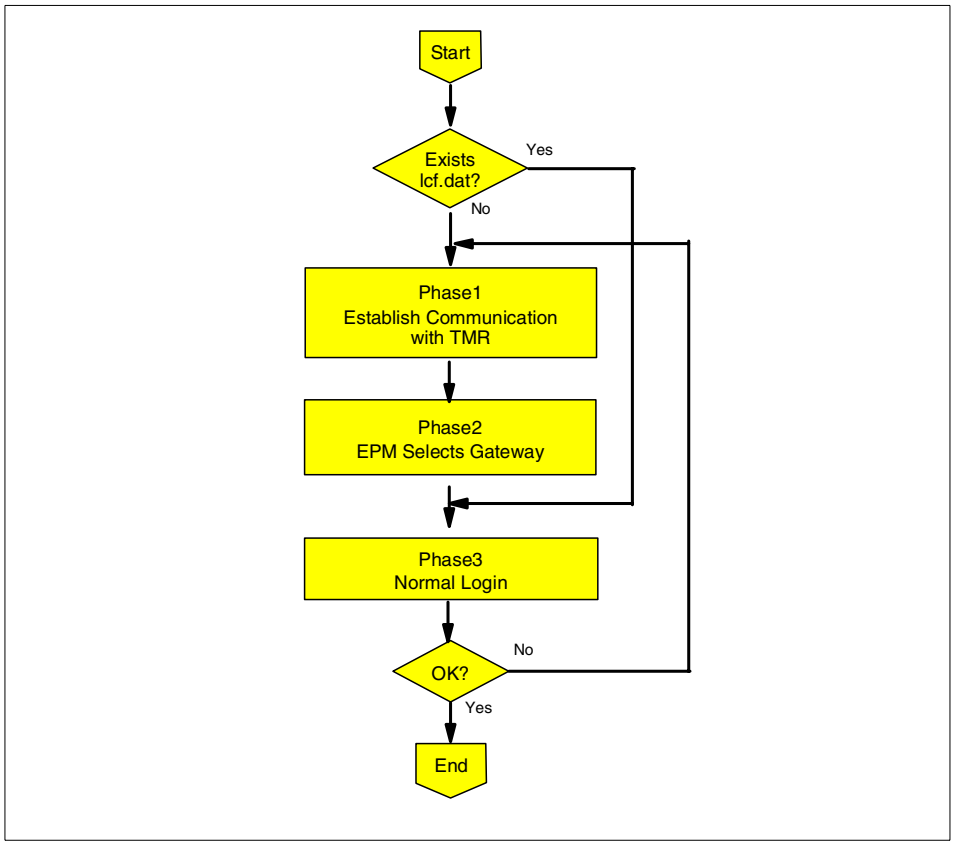


Figure 37. Endpoint initial login processes

The following figure (Figure 38) shows more detailed process flow from phase 1 to phase 3 process. The first step in the Endpoint initial login process is finding an Endpoint Gateway in the correct TMR. Finding a region means finding an Endpoint Gateway to serve the initial login request. To do this, the Endpoint uses a set of network address configured during the installation process. This set of addresses is called the Endpoint login interfaces.

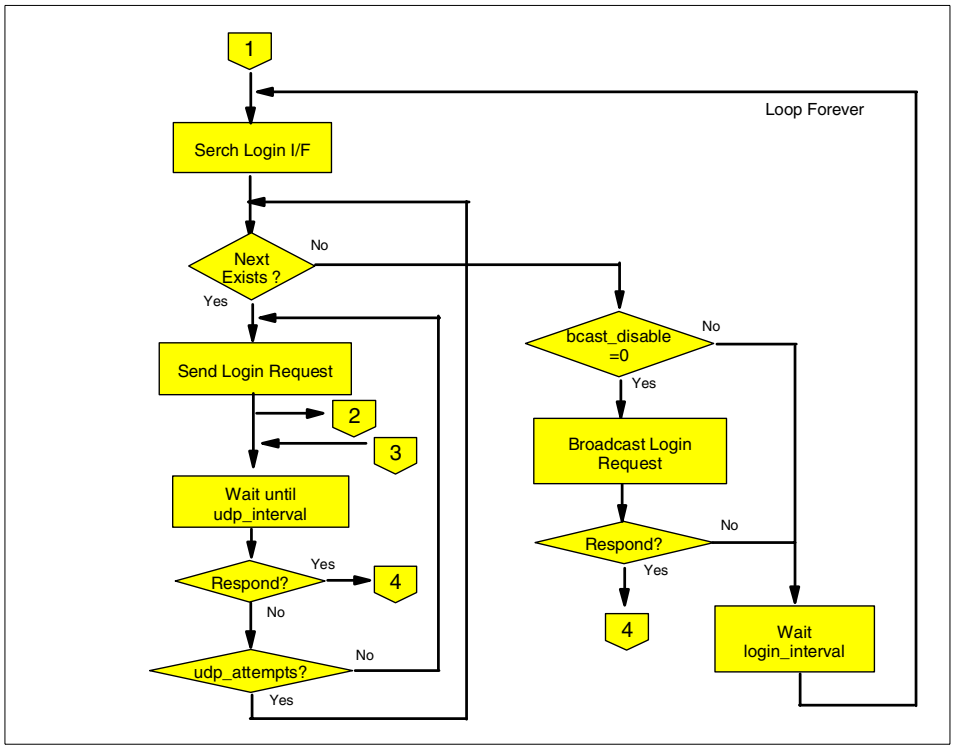


Figure 38. Finding a region

This flow chart is linked to the following flow charts shown in Figure 39 and Figure 40.

When the Endpoint Gateway receives an initial login request, it is said to be the intercepting Gateway for that request. The intercepting Gateway forwards the login request to the Endpoint Manager that is responsible for determining the best available Endpoint Gateway for that Endpoint. The selection process is user-configurable. The following figure (Figure 39) shows overview of the Gateway selection flow.

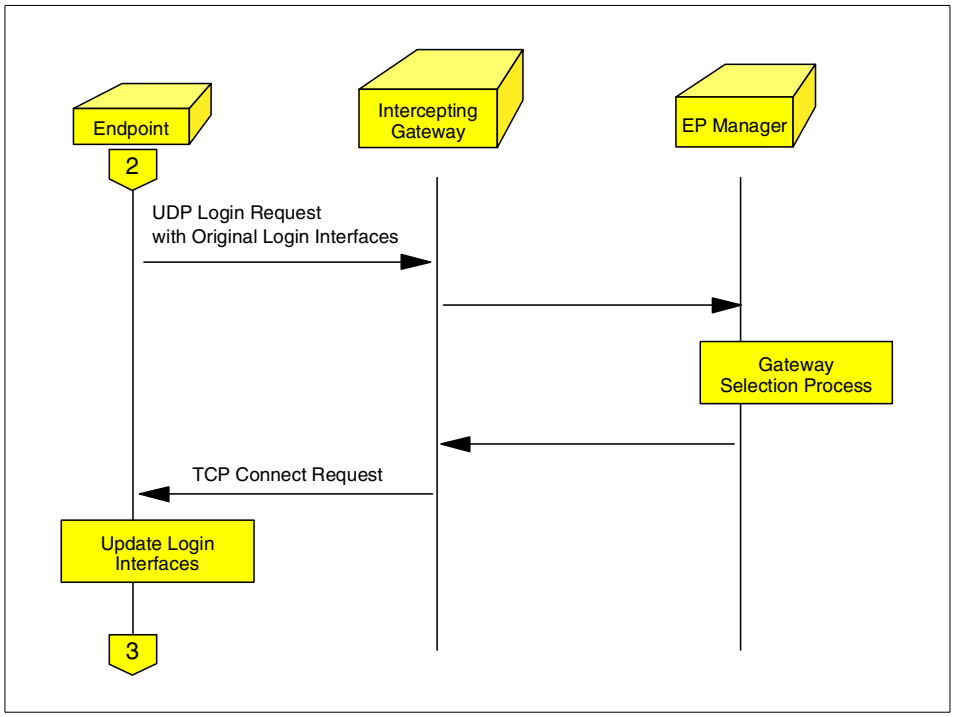


Figure 39. Gateway selection (sequence chart)

After receiving its Gateway assignment from the intercepting Gateway, the Endpoint performs the normal login to its assigned Gateway. At this time, the login_policy is run for the first time and the Endpoint codeset is downloaded to the Endpoint. The Endpoint is now ready to participate in Tivoli management operations. The next figure (Figure 40) shows the Endpoint normal login process.

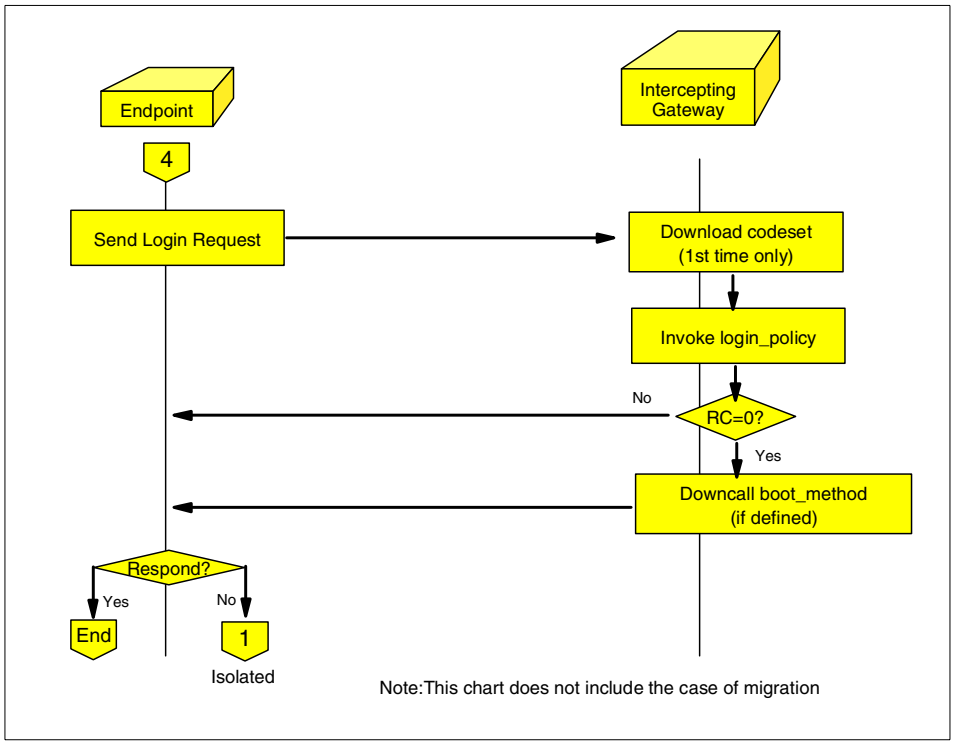


Figure 40. Endpoint normal login

Note

In this redbook we do not explain Gateway selection process in detail. Please refer to the *All About Tivoli Management Agents*, SG24-5134 for more detailed information.

4.3.4.3 How Endpoint parameters work

In this section we show an example of Endpoint login process, and also show how Endpoint login parameters work during the login process. The following table (Table 3) describes the test scenario that we performed.

Table 3. Testing Endpoint Login Parameters

Title	Description
Scenario	Perform shutdown the EP Gateway after the initial login is completed to make the Endpoint isolated. Then we restart the Endpoint and the Endpoint attempts to perform normal login.

Title	Description
Option	Refer to Figure 41 (Endpoint login interfaces list) and Table 4 (last.cfg file of the Endpoint) for more information.
EP Policy	None
Result	The Endpoint attempted to perform normal login to the assigned Gateway, however, the assigned Gateway was not available. So, the Endpoint has an isolated status. Then the Endpoint attempted to perform isolated login to the EP Gateways specified in the login interfaces list in the lcf.dat file, and sent the login request to the EP Gateway.

To display the Endpoint login interfaces list of the test machine (Endpoint), we used Endpoint web interface. The following figure (Figure 41) shows the login interfaces list.

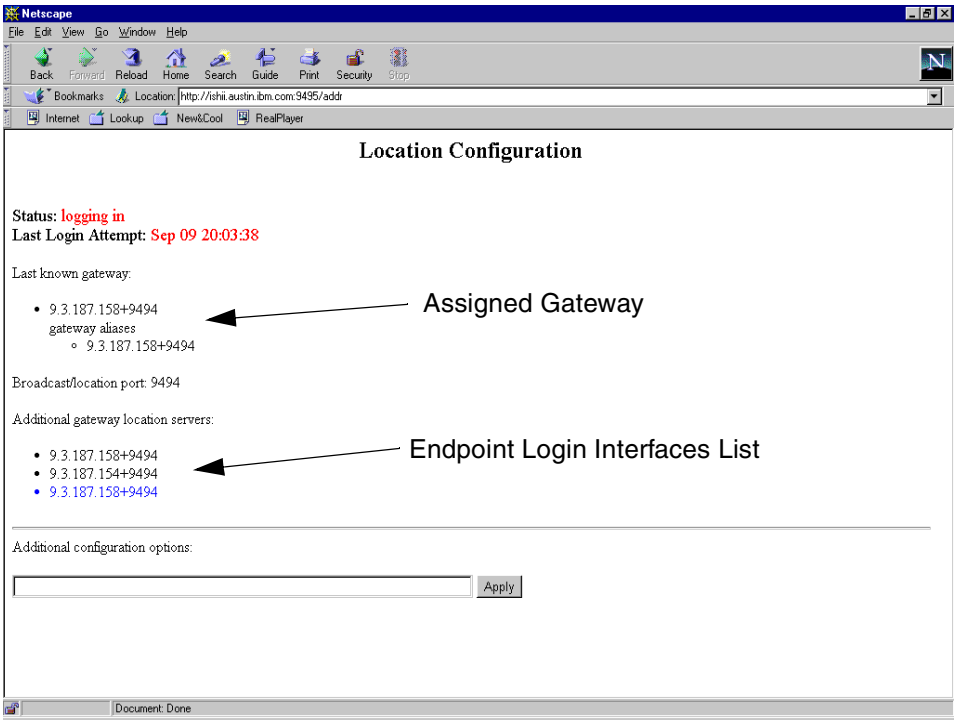


Figure 41. Endpoint login interfaces list

The following shows the contents of the last.cfg file that is located in the C:\Tivoli\lcf\dat\1 directory.

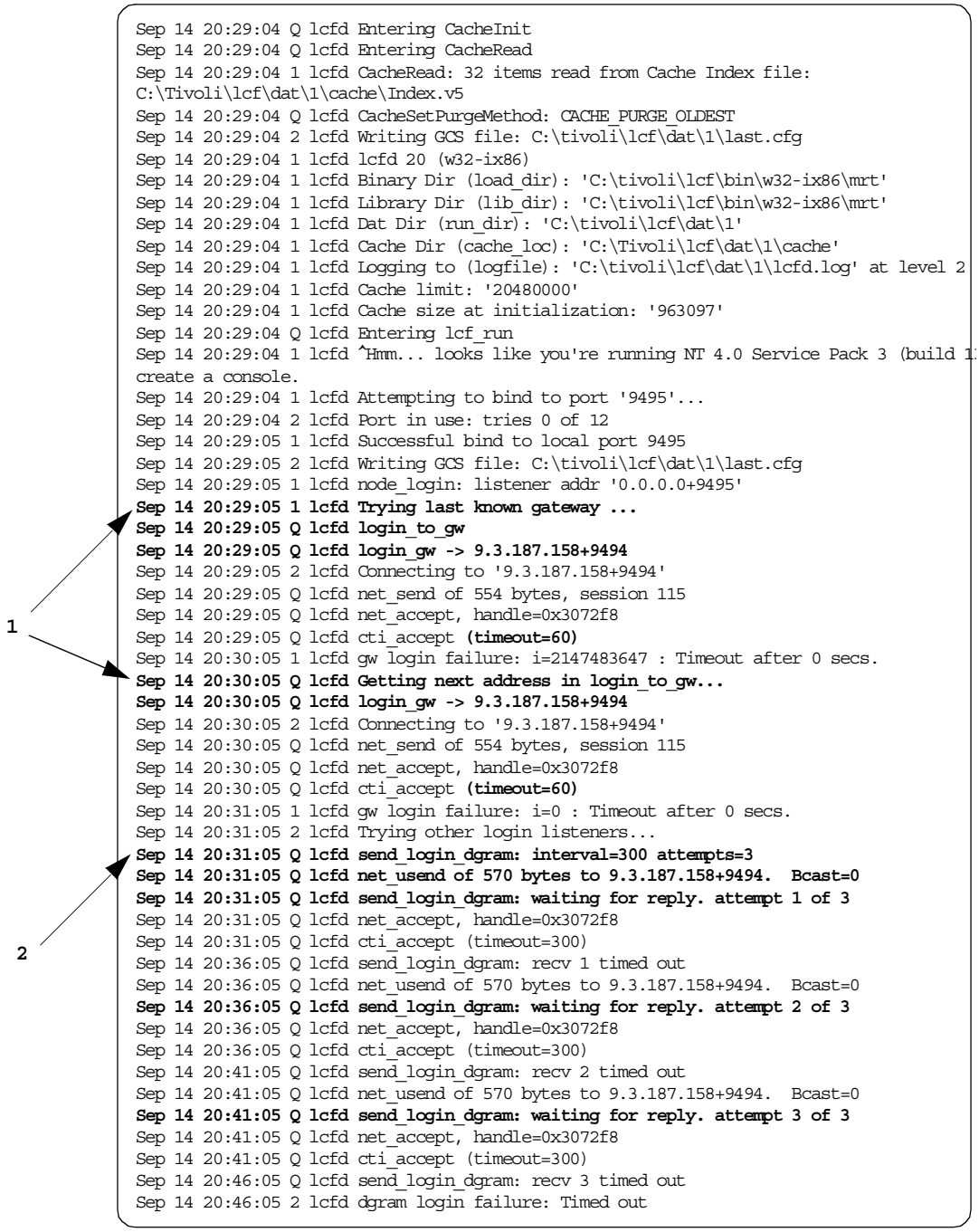
```
lcfd_port=9495
lcfd_preferred_port=9495
gateway_port=9494
log_threshold=2
start_timeout=120
run_timeout=60
lcfd_version=20
logfile=C:\tivoli\lcf\dat\1\lcfd.log
config_path=C:\tivoli\lcf\dat\1\last.cfg
run_dir=C:\tivoli\lcf\dat\1
load_dir=C:\tivoli\lcf\bin\w32-ix86\mrt
lib_dir=C:\tivoli\lcf\bin\w32-ix86\mrt
cache_loc=C:\Tivoli\lcf\dat\1\cache
cache_index=C:\Tivoli\lcf\dat\1\cache\Index.v5
cache_limit=20480000
log_queue_size=1024
log_size=1024000
udp_interval=300
udp_attempts=3
login_interval=1800
lcs.machine_name=ishii
lcs.crypt_mode=196608
lcfd_alternate_port=9496
```

As you can see, each Endpoint login parameters are set to the following values.

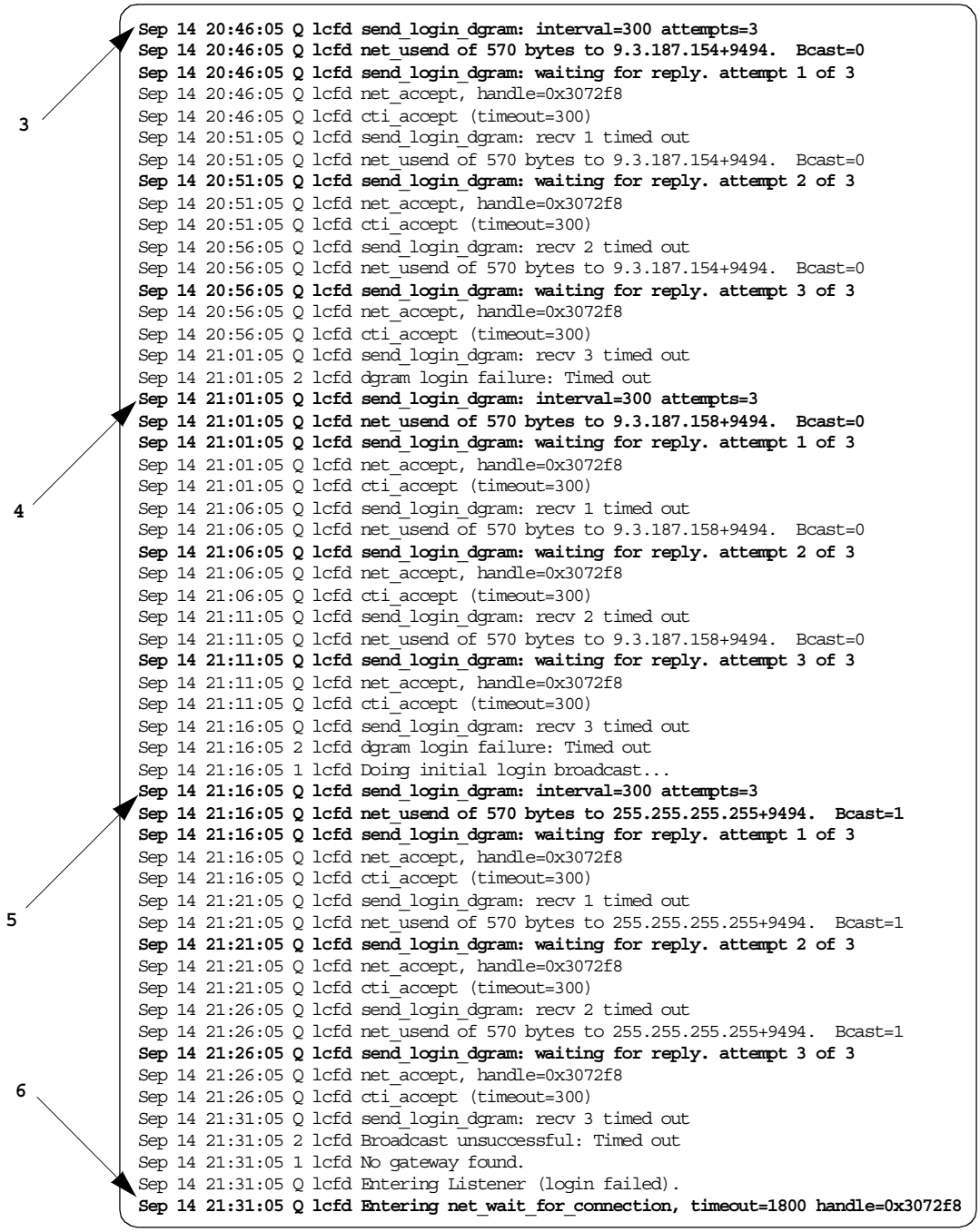
Table 4. Setting Endpoint login parameters

Parameter	Setting Value	Default Value
start_timeout	120 (Seconds)	120 (Seconds)
run_timeout	60 (Seconds)	120 (Seconds)
udp_interval	300 (Seconds)	300 (Seconds)
udp_attempts	3 (Times)	6 (Times)
login_interval	1,800 (Seconds)	1,800 (Seconds)

In this case, the following massages are logged into the Endpoint log file (lcfd.log).




```
Sep 14 20:29:04 Q lcfd Entering CacheInit
Sep 14 20:29:04 Q lcfd Entering CacheRead
Sep 14 20:29:04 1 lcfd CacheRead: 32 items read from Cache Index file:
C:\Tivoli\lcf\dat\1\cache\Index.v5
Sep 14 20:29:04 Q lcfd CacheSetPurgeMethod: CACHE_PURGE_OLDEST
Sep 14 20:29:04 2 lcfd Writing GCS file: C:\tivoli\lcf\dat\1\last.cfg
Sep 14 20:29:04 1 lcfd lcfd 20 (w32-ix86)
Sep 14 20:29:04 1 lcfd Binary Dir (load_dir): 'C:\tivoli\lcf\bin\w32-ix86\mrt'
Sep 14 20:29:04 1 lcfd Library Dir (lib_dir): 'C:\tivoli\lcf\bin\w32-ix86\mrt'
Sep 14 20:29:04 1 lcfd Dat Dir (run_dir): 'C:\tivoli\lcf\dat\1'
Sep 14 20:29:04 1 lcfd Cache Dir (cache_loc): 'C:\Tivoli\lcf\dat\1\cache'
Sep 14 20:29:04 1 lcfd Logging to (logfile): 'C:\tivoli\lcf\dat\1\lcfd.log' at level 2
Sep 14 20:29:04 1 lcfd Cache limit: '20480000'
Sep 14 20:29:04 1 lcfd Cache size at initialization: '963097'
Sep 14 20:29:04 Q lcfd Entering lcf_run
Sep 14 20:29:04 1 lcfd ^Hmm... looks like you're running NT 4.0 Service Pack 3 (build 1
create a console.
Sep 14 20:29:04 1 lcfd Attempting to bind to port '9495'...
Sep 14 20:29:04 2 lcfd Port in use: tries 0 of 12
Sep 14 20:29:05 1 lcfd Successful bind to local port 9495
Sep 14 20:29:05 2 lcfd Writing GCS file: C:\tivoli\lcf\dat\1\last.cfg
Sep 14 20:29:05 1 lcfd node_login: listener addr '0.0.0.0+9495'
Sep 14 20:29:05 1 lcfd Trying last known gateway ...
Sep 14 20:29:05 Q lcfd login_to_gw
Sep 14 20:29:05 Q lcfd login_gw -> 9.3.187.158+9494
Sep 14 20:29:05 2 lcfd Connecting to '9.3.187.158+9494'
Sep 14 20:29:05 Q lcfd net_send of 554 bytes, session 115
Sep 14 20:29:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 20:29:05 Q lcfd cti_accept (timeout=60)
Sep 14 20:30:05 1 lcfd gw login failure: i=2147483647 : Timeout after 0 secs.
Sep 14 20:30:05 Q lcfd Getting next address in login_to_gw...
Sep 14 20:30:05 Q lcfd login_gw -> 9.3.187.158+9494
Sep 14 20:30:05 2 lcfd Connecting to '9.3.187.158+9494'
Sep 14 20:30:05 Q lcfd net_send of 554 bytes, session 115
Sep 14 20:30:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 20:30:05 Q lcfd cti_accept (timeout=60)
Sep 14 20:31:05 1 lcfd gw login failure: i=0 : Timeout after 0 secs.
Sep 14 20:31:05 2 lcfd Trying other login listeners...
Sep 14 20:31:05 Q lcfd send_login_dgram: interval=300 attempts=3
Sep 14 20:31:05 Q lcfd net_usend of 570 bytes to 9.3.187.158+9494.  Bcast=0
Sep 14 20:31:05 Q lcfd send_login_dgram: waiting for reply. attempt 1 of 3
Sep 14 20:31:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 20:31:05 Q lcfd cti_accept (timeout=300)
Sep 14 20:36:05 Q lcfd send_login_dgram: recv 1 timed out
Sep 14 20:36:05 Q lcfd net_usend of 570 bytes to 9.3.187.158+9494.  Bcast=0
Sep 14 20:36:05 Q lcfd send_login_dgram: waiting for reply. attempt 2 of 3
Sep 14 20:36:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 20:36:05 Q lcfd cti_accept (timeout=300)
Sep 14 20:41:05 Q lcfd send_login_dgram: recv 2 timed out
Sep 14 20:41:05 Q lcfd net_usend of 570 bytes to 9.3.187.158+9494.  Bcast=0
Sep 14 20:41:05 Q lcfd send_login_dgram: waiting for reply. attempt 3 of 3
Sep 14 20:41:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 20:41:05 Q lcfd cti_accept (timeout=300)
Sep 14 20:46:05 Q lcfd send_login_dgram: recv 3 timed out
Sep 14 20:46:05 2 lcfd dgram login failure: Timed out
```


3
4
5
6
Sep 14 20:46:05 Q lcfd send_login_dgram: interval=300 attempts=3
Sep 14 20:46:05 Q lcfd net_usend of 570 bytes to 9.3.187.154+9494. Bcast=0
Sep 14 20:46:05 Q lcfd send_login_dgram: waiting for reply. attempt 1 of 3
Sep 14 20:46:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 20:46:05 Q lcfd cti_accept (timeout=300)
Sep 14 20:51:05 Q lcfd send_login_dgram: recv 1 timed out
Sep 14 20:51:05 Q lcfd net_usend of 570 bytes to 9.3.187.154+9494. Bcast=0
Sep 14 20:51:05 Q lcfd send_login_dgram: waiting for reply. attempt 2 of 3
Sep 14 20:51:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 20:51:05 Q lcfd cti_accept (timeout=300)
Sep 14 20:56:05 Q lcfd send_login_dgram: recv 2 timed out
Sep 14 20:56:05 Q lcfd net_usend of 570 bytes to 9.3.187.154+9494. Bcast=0
Sep 14 20:56:05 Q lcfd send_login_dgram: waiting for reply. attempt 3 of 3
Sep 14 20:56:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 20:56:05 Q lcfd cti_accept (timeout=300)
Sep 14 21:01:05 Q lcfd send_login_dgram: recv 3 timed out
Sep 14 21:01:05 2 lcfd dgram login failure: Timed out
Sep 14 21:01:05 Q lcfd send_login_dgram: interval=300 attempts=3
Sep 14 21:01:05 Q lcfd net_usend of 570 bytes to 9.3.187.158+9494. Bcast=0
Sep 14 21:01:05 Q lcfd send_login_dgram: waiting for reply. attempt 1 of 3
Sep 14 21:01:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 21:01:05 Q lcfd cti_accept (timeout=300)
Sep 14 21:06:05 Q lcfd send_login_dgram: recv 1 timed out
Sep 14 21:06:05 Q lcfd net_usend of 570 bytes to 9.3.187.158+9494. Bcast=0
Sep 14 21:06:05 Q lcfd send_login_dgram: waiting for reply. attempt 2 of 3
Sep 14 21:06:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 21:06:05 Q lcfd cti_accept (timeout=300)
Sep 14 21:11:05 Q lcfd send_login_dgram: recv 2 timed out
Sep 14 21:11:05 Q lcfd net_usend of 570 bytes to 9.3.187.158+9494. Bcast=0
Sep 14 21:11:05 Q lcfd send_login_dgram: waiting for reply. attempt 3 of 3
Sep 14 21:11:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 21:11:05 Q lcfd cti_accept (timeout=300)
Sep 14 21:16:05 Q lcfd send_login_dgram: recv 3 timed out
Sep 14 21:16:05 2 lcfd dgram login failure: Timed out
Sep 14 21:16:05 1 lcfd Doing initial login broadcast...
Sep 14 21:16:05 Q lcfd send_login_dgram: interval=300 attempts=3
Sep 14 21:16:05 Q lcfd net_usend of 570 bytes to 255.255.255.255+9494. Bcast=1
Sep 14 21:16:05 Q lcfd send_login_dgram: waiting for reply. attempt 1 of 3
Sep 14 21:16:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 21:16:05 Q lcfd cti_accept (timeout=300)
Sep 14 21:21:05 Q lcfd send_login_dgram: recv 1 timed out
Sep 14 21:21:05 Q lcfd net_usend of 570 bytes to 255.255.255.255+9494. Bcast=1
Sep 14 21:21:05 Q lcfd send_login_dgram: waiting for reply. attempt 2 of 3
Sep 14 21:21:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 21:21:05 Q lcfd cti_accept (timeout=300)
Sep 14 21:26:05 Q lcfd send_login_dgram: recv 2 timed out
Sep 14 21:26:05 Q lcfd net_usend of 570 bytes to 255.255.255.255+9494. Bcast=1
Sep 14 21:26:05 Q lcfd send_login_dgram: waiting for reply. attempt 3 of 3
Sep 14 21:26:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 21:26:05 Q lcfd cti_accept (timeout=300)
Sep 14 21:31:05 Q lcfd send_login_dgram: recv 3 timed out
Sep 14 21:31:05 2 lcfd Broadcast unsuccessful: Timed out
Sep 14 21:31:05 1 lcfd No gateway found.
Sep 14 21:31:05 Q lcfd Entering Listener (login failed).
Sep 14 21:31:05 Q lcfd Entering net_wait_for_connection, timeout=1800 handle=0x3072f8

```
Sep 14 21:31:05 Q lcfd cti_accept (timeout=1800)
Sep 14 22:01:05 Q lcfd timeout in listener loop
Sep 14 22:01:05 1 lcfd node_login: listener addr '0.0.0.0+9495'
Sep 14 22:01:05 1 lcfd Trying last known gateway ...
Sep 14 22:01:05 Q lcfd login_to_gw
Sep 14 22:01:05 Q lcfd login_gw -> 9.3.187.158+9494
Sep 14 22:01:05 2 lcfd Connecting to '9.3.187.158+9494'
Sep 14 22:01:05 Q lcfd net_send of 554 bytes, session 115
Sep 14 22:01:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 22:01:05 Q lcfd cti_accept (timeout=60)
Sep 14 22:02:05 1 lcfd gw login failure: i=2147483647 : Timeout after 0 secs.
Sep 14 22:02:05 Q lcfd Getting next address in login_to_gw...
Sep 14 22:02:05 Q lcfd login_gw -> 9.3.187.158+9494
Sep 14 22:02:05 2 lcfd Connecting to '9.3.187.158+9494'
Sep 14 22:02:05 Q lcfd net_send of 554 bytes, session 115
Sep 14 22:02:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 22:02:05 Q lcfd cti_accept (timeout=60)
Sep 14 22:03:05 1 lcfd gw login failure: i=0 : Timeout after 0 secs.
Sep 14 22:03:05 2 lcfd Trying other login listeners...
Sep 14 22:03:05 Q lcfd send_login_dgram: interval=300 attempts=3
Sep 14 22:03:05 Q lcfd net_usend of 570 bytes to 9.3.187.158+9494.  Bcast=0
Sep 14 22:03:05 Q lcfd send_login_dgram: waiting for reply. attempt 1 of 3
Sep 14 22:03:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 22:03:05 Q lcfd cti_accept (timeout=300)
Sep 14 22:08:05 Q lcfd send_login_dgram: recv 1 timed out
Sep 14 22:08:05 Q lcfd net_usend of 570 bytes to 9.3.187.158+9494.  Bcast=0
Sep 14 22:08:05 Q lcfd send_login_dgram: waiting for reply. attempt 2 of 3
Sep 14 22:08:05 Q lcfd net_accept, handle=0x3072f8
Sep 14 22:08:05 Q lcfd cti_accept (timeout=300)
```

7

You will be able to understand how each Endpoint login parameter works during Endpoint login process through the Endpoint log file. We explain some important log messages as follows (refer to the sample lcfd.log file).

1. In this case, the Endpoint performs normal login. So the Endpoint attempts to send a login request to the assigned Gateway machine (9.3.187.158). Since the assigned Gateway is unreachable, the Endpoint attempts to send a login request to the Gateway alias address (9.3.187.158). The assigned Gateway and its alias address information is stored in the lcf.dat file on the Endpoint (refer to the Figure 41 on page 97). The Gateway alias is also unreachable. As a result, the normal login fails and the Endpoint gives up performing normal login. In this process, the run_timeout (60 seconds) is used.
2. The Endpoint is isolated. The Endpoint performs isolate login to the alternate Gateways that are stored in the login interfaces list (lcf.dat file). The Endpoint attempts to send a login request to 9.3.187.158 machine first (refer to the Figure 41 on page 97). In this process, the udp_interval (300 seconds) and udp_attempts (3 times) are used (refer to the Table 4 on page 98). It means that the Endpoint will attempt to send a login

request three times every 300 seconds. Since the alternate Gateway (9.3.187.158) is unreachable, the login fails.

3. The Endpoint attempts to send a login request to the next Gateway (9.3.187.154) machine listed in the login interfaces (refer to the Figure 41 on page 97). Since the Endpoint Gateway machine that the Endpoint attempts to login is unreachable, the login fails. The Endpoint uses the udp_interval and udp_attempts as well.
4. The Endpoint attempts to send a login request to the next Gateway (9.3.187.158) machine. This process is the same as process 2.
5. Since all the alternate Gateways are unreachable, the Endpoint finally broadcasts a login request (broadcast is enabled by default). In this process, the udp_interval and udp_attempts are also used.
6. In this example, no Endpoint Gateway found. As a result, the Endpoint enters a wait loop. In this process, the login_interval (1,800 seconds) is used. The Endpoint will perform a login process (from process 2 to process 5) every 1,800 seconds until an Endpoint Gateway responds to the login request.

4.3.4.4 Understanding login_interval attribute

In Version 3.6 of Tivoli Management Framework, when broadcasting a login request to multiple Endpoint Gateways in the same region, the Endpoint establishes multiple identities for itself in the region. Version 3.6.1 of Tivoli Management Framework provides the login_interval attribute to resolve broadcast login requests that reach multiple Endpoint Gateways in the same region. The Endpoint Manager ignores any login requests sent within "login_interval" seconds of a previous one (270 seconds by default), so if an Endpoint broadcasts in a multiple Endpoint Gateways broadcast space, and if multiple intercepting Gateways relay the request to the Endpoint Manager within the "login_interval" period, all but the first relay is ignored by the Endpoint Manager. In this way, only one Endpoint identity is created.

Note

Exceptions to the login sequence are made for the initial and normal login sequence and also for Endpoints re-login immediately following an upgrade.

If an Endpoint attempts to login within "login_interval" seconds of a previous one, the Endpoint login is failed and the following messages are logged into the gatelog file.

```
1999/09/13 12:11:18 -09: process_node_login: Endpoint (38) is speaking ECP protocol ver
1999/09/13 12:11:18 -09: Ignoring login for dispatcher 38: filter failed
```

Note

When you reboot an Endpoint machine or restart lcfd daemon within the "login_interval" period, the Endpoint may not be able to login. Then, check the Endpoint Gateway log file (gatelog) first. If the above messages are logged, the login_interval filtered the login request.

Since the Endpoint Gateway reads the "login_interval" attribute on startup, and thereafter uses it in the same manner as the Endpoint Manager, the Endpoint Gateway will ignore multiple logins arriving from an Endpoint if they arrive with less than "login_interval" seconds between them.

This enables the Endpoint Manager (TMR server) to be protected from failure modes in which one or more Endpoints send login requests at unreasonably high rates. The login_interval attribute can be used to filter Endpoint initial logins at Endpoint Gateway or Endpoint Manager.

There is no specific Tivoli command for the login_interval attribute. It is settable and readable via the `idlattr` or `idlcall` command as follows.

```
# wlookup -ar EndpointManager
epmgr_1588251808.1.517 1588251808.1.517
# idlattr -tg 1588251808.1.517 login_interval short
270
# idlattr -ts 1588251808.1.517 login_interval short 300
# idlattr -tg 1588251808.1.517 login_interval short
300
#
```

In the above example, we set the login_interval attribute to 300 (seconds) by the `idlattr` command.

4.3.4.5 Configuring filtering parameters

As we mentioned, there are some filtering parameters for Endpoint login requests. This section describe how to configure these parameters to filter Endpoint login requests efficiently in a large-scale environment. The following figure (Figure 42) shows the overview of each parameter.

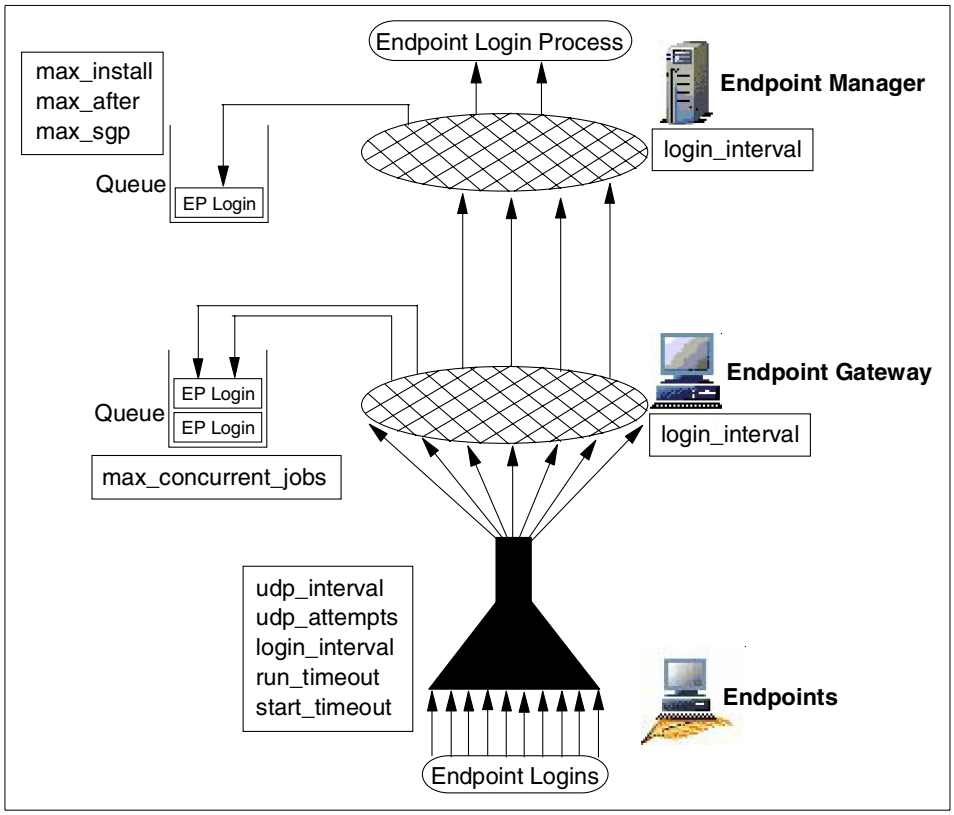


Figure 42. Filtering Endpoint login requests

The following table (Table 5) shows the default value of each parameter.

Table 5. Default values of filtering parameters

Parameter	Default Value
max_install	10
max_after	10
max_sgp	10
login_interval (EPM object attribute)	270 (seconds)
max_concurrent_jobs	200
udp_interval	300 (seconds)
udp_attempts	6 (times)

Parameter	Default Value
login_interval (lcfd option)	1800 (seconds)
run_timeout	120 (seconds)
start_timeout	120 (seconds)

The following are key issues when you configure these parameters.

max_sgp

The select_gateway_policy is executed not only at initial login but also at isolated login. In a large-scale environment, the select_gateway_policy may take a long time (for example, a few minutes) to complete the process, because Endpoint Gateway selection process is complicated (for example, filtering or key word search). In this case the default value (10) may be not enough and you should increase the max_sgp value.

To check the average number of running select_gateway_policy scripts, you can use the following way on a UNIX Managed Node.

```
odstat -c | grep sgp | wc -l
```

This command tells you how many select_gateway_policy scripts are running on the Endpoint Manager. We recommend you run this command periodically to check the trend. The following shell script logs the number of running select_gateway_policy scripts in the specified file.

```
#!/bin/ksh

. /etc/Tivoli/setup_env.sh

LOG_FILE=$1
TIME=`date`
SGP=`odstat -c | grep sgp | wc -l`
echo $TIME Running sgp is $SGP >> $LOG_FILE
```

You can use Sentry monitor (Distributed Monitoring) or UNIX cron to execute this sample script periodically. If the number of running select_gateway_policy scripts is close to 10, you should increase the max_sgp attribute.

login_interval (EP Manager object attribute)

We recommend you use the default value (270 seconds) for the login_interval attribute. Increase the login_interval attribute with caution, because it may cause an increase in rejected login requests.

We strongly recommend you follow the following formula when you modify the login_interval attribute.

```
udp_interval > login_interval
```

If the login_interval attribute value is greater than the udp_interval value and the first initial login request is failed for some reason (for example, the select_gateway_policy takes a long time to complete), the Endpoint may retry to perform initial login or isolate login many times.

If you set the login_interval attribute too low, Endpoint can retry failed logins more quickly, but you run the risk of duplicate Endpoints. If you set it too high, you reduce the risk of duplicate Endpoints but increase the length of time some Endpoints take to login. So, the default value should be fine for most environment.

udp_interval, udp_attempts and login_interval
These parameters control how long and how many times each Endpoint Gateway is attempted before trying the next. Normally, we recommend to decrease the udp_attempts (for example, udp_attempts=3 or 4) in a large-scale environment.

If you decrease the udp_interval value, Endpoint will be able to retry failed logins more quickly, but do not forget to decrease the Endpoint Manager login_interval attribute as well. The udp_interval value should be greater than the Endpoint Manager login_interval attribute.

run_timeout and start_timeout
The start_timeout parameters is used during Endpoint login process, and the run_timeout is used after login completion (including normal login). These parameters are used for timeout of TCP sessions. Normally, the default value is recommended. If your Endpoint policy script (for example, login_policy) takes a long time to complete, you had better increase the value depending on the script. Do not forget, the login_policy will be executed each time Endpoint performs normal login, so we recommend you create a simple login_policy.

Note

In the Endpoint initial login process, the Endpoint uses UDP protocol to send the initial login request to the Endpoint Gateway that is listed in the login interfaces. The udp_interval is the timeout for this UDP login request. The Endpoint sends out a UDP login packet and wait for the Endpoint Gateway to respond by establishing a TCP connection for reply delivery.

In Endpoint normal login process, the Endpoint does not use UDP login packet, so the udp_interval is not used. In this case, run_timeout is used because it is the timeout value for TCP sessions.

4.3.5 Miscellaneous considerations

In the following sections, we introduce some miscellaneous considerations for Tivoli Management Framework configurations.

4.3.5.1 Sample configurations

This section introduces a sample configuration for an Endpoint Gateway and Endpoint. It depends on the environment; so, test how this works in your specific environment before implementing.

Endpoint configurations

The following are the recommended configurations for the Endpoint:

log_threshold Sets level 0 (default). You can use the `-d` option as well.

bcast_disabled Sets disabled (`bcast_disabled=1`).

lcs.login_interfaces Specifies at least two Endpoint Gateways, hopefully, more than three.

Endpoint Gateway configurations

The configuration of the Endpoint Gateway depends a great deal on the environment. The following is a very common configuration. We recommend you to configure as follows:

set_debug_level We recommend you set the debug level to 0. The debug level of the Endpoint Gateway defines what information is logged into the $DBDIR/gatelog file. The debug level 0 means that it only logs errors of the Endpoint Gateway into the gatelog file. This improves performance and ensures that the size of the gatelog file doesn't become large quickly. To change the Endpoint Gateway's debug level from the CLI, you can use the following command:

```
wlookup -o -r Gateway -a | while read oid
do
    idlcall $oid _set_debug_level 0
done
```

You can use the `wgateway` command to change the debug level.

```
wgateway <gw_label> set_debug_level [0-9]
```

4.3.5.2 Endpoint policy considerations

The Endpoint policy is a very useful way to manage the Endpoints efficiently. However, it also expends resources at the Endpoint Manager and Endpoint Gateways. Sometimes it may cause a performance problem; so, you should take care when using the Endpoint policy.

For example, the auto upgrade function of the TMA is typical customized by using the Endpoint policy (login_policy). It allows you to upgrade the Endpoint software automatically. This is a very useful function. However, to enable this function, upgrade.sh must be configured in the login_policy. This means that the auto upgrade function will be run every time the Endpoint logs into the Endpoint Gateway even if the upgrade process does not occur. Moreover, the upgrade.sh includes the `awk` or `grep` command. The `awk` or `grep` commands are useful tools, but they expend much more resources than other commands. If many Endpoints attempt to log into the Endpoint Gateway at the same time, upgrade.sh will be executed for each Endpoint, and the upgrade process will not occur on most of the Endpoints. This is bad for performance tuning.

If you use the auto upgrade function, you should use it only when the Endpoint Gateway is upgraded because then the contents under the $DBDIR/../bin/lcf_bundle directory should be upgraded as well. To modify the Endpoint policy, you can use the `wgeteppol` and `wputeppol` commands.

Auto upgrade is a typical case. The Endpoint policy can have an adverse affect on performance. To avoid this, you should take care of the following:

- Avoid using many `awk` or `grep` commands in Endpoint policy.
- Use a C program instead of the shell script, if possible, in the Endpoint policy.
- Do not define an Endpoint policy if it is not needed (like an auto upgrade). It should be defined only when needed.

4.3.6 Endpoint method preloading

The Tivoli Management Agent has a cache that contains the Endpoint methods. When the TMA uses an Endpoint method for the first time, the Endpoint method is downloaded from the Endpoint Gateway and stored, within the TMA, in the Endpoint method cache. In a large scale deployment, you can bypass this process using the technique called Endpoint method preloading. This will simplify the TMA deployment.

The following sections introduce how TMA manages Endpoint methods and what Endpoint method preloading is.

4.3.6.1 Endpoint method cache management

Endpoint method cache management is very important in understanding the TMA implementation. The TMA gives you the base services that are necessary to perform management operations, such as those performed by Tivoli Management Applications, and these management operations are processed by calling a method on a management resource. In the TMA environment, this design has not changed; the TMA invokes an Endpoint method for performing management operations on the TMA platform.

The Endpoint method that will be used by the Endpoint is stored in the Endpoint Gateway. When the TMA performs a management operation, the TMA automatically determines what is needed to perform the given management operation. If that Endpoint method already resides on the TMA method cache, it immediately proceeds with the operation. If not, the TMA downloads the appropriate Endpoint method from the Endpoint Gateway to the TMA with no operator intervention. In addition, the TMA downloads newer versions when updates are loaded on the Endpoint Gateway. You can gain significant productivity advances with these management features because the Tivoli Management software is installed only once on the Endpoint Manager and Endpoint Gateways with updates performed automatically thereafter. The following figure (Figure 43) shows how TMA manages its Endpoint method cache.

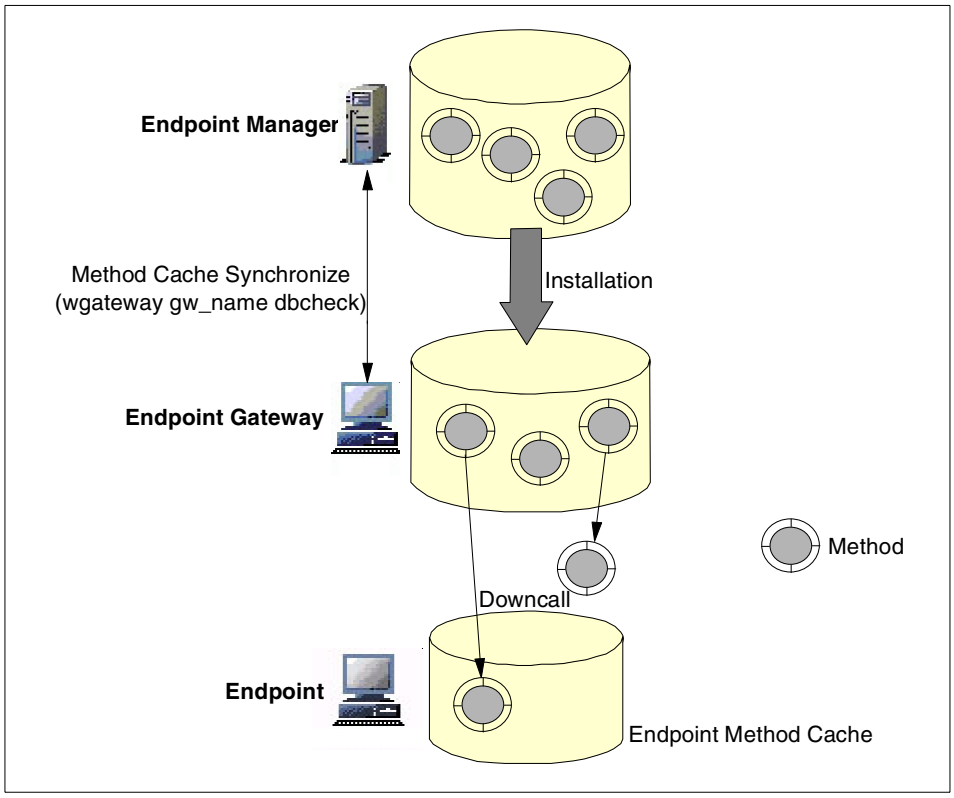


Figure 43. Endpoint method cache management

By default, the Endpoint method cache exists under the C:\Tivoli\lcf\dat\1\cache directory, and each Endpoint method actually correspond to the *.exe file in the PC platform. The Endpoint Gateway stores the Endpoint methods under the /usr/local/Tivoli/bin/lcf_bundle directory and also stores Endpoint method information in the Endpoint Gateway database ($DBDIR/gwdb.bdb file). This information can be synchronized with the information on the Endpoint Manager ($DBDIR/imdb.bdb file) using the `wgateway` command.

The following summary describes how the Endpoint manages the Endpoint method cache.

- Once the Endpoint method is stored in the Endpoint method cache, the Endpoint uses it even if the system is rebooted.
- Once the Endpoint method is downloaded to the Endpoint method cache, the Endpoint does not download the same version of the Endpoint method.

- If the Endpoint detects that the available version is greater than the current version of the Endpoint method, the Endpoint downloads the newer method automatically.
- When the Endpoint downloads the method, the related methods defined in the dependency set are also downloaded at the same time. The dependency set is a list of other files, modules, or commands that are required for the correct operation of the method.

4.3.6.2 Endpoint methods and dependency set

Tivoli Management applications invoke methods using upcall and downcall to perform their functions. Which method will be used by a Tivoli Management application? The dependency manager handles the methods for each Tivoli Management application. The dependency manager exists as a class object in the Tivoli object database and plays a very important role in implementing the LCF architecture. We can understand the relationship between the methods and the Tivoli Management applications by checking the dependency manager.

Normally, the dependency set is added to the dependency manager when you install the Tivoli Management application (but it is not mandatory; it depends on the application design). We installed some Tivoli Management applications for checking the dependency during the project, and the following table (Table 6) shows the relationship between the dependency set and the Tivoli Management applications.

Table 6. The dependency set for each Tivoli Management Application

Dependency Set	Tivoli Management Application
LCFDepList	Distributed Monitoring
courier_lcf	Software Distribution
GroupManagement	User Administration
UserManagement	
Inventory-lcf-depset	Inventory

To make sure, you can use the `wlookup` and `wdepset` commands as follows:

```
# wlookup -ar DependencyMgr
DMBootDepList   1613591617.1.1094#Depends::Mgr#
Inventory-lcf-depset    1613591617.1.953#Depends::Mgr#
LCFDepList      1613591617.1.1093#Depends::Mgr#
NtLcfInst_depset        1613591617.1.523#Depends::Mgr#
acpep-ep        1613591617.1.893#Depends::Mgr#
acpep-gateway   1613591617.1.898#Depends::Mgr#
acpep-logfile   1613591617.1.896#Depends::Mgr#
acpep-nt        1613591617.1.897#Depends::Mgr#
acpep-postemsg  1613591617.1.894#Depends::Mgr#
acpep-snmp      1613591617.1.895#Depends::Mgr#
courier_lcf     1613591617.1.711#Depends::Mgr#
msg-bind-catalogs       1613591617.1.516#Depends::Mgr#
task-lcf-base   1613591617.1.513#Depends::Mgr#
task-spawn16    1613591617.1.514#Depends::Mgr#
task-spawn32    1613591617.1.515#Depends::Mgr#
#
```

In this case, we invoked the `wlookup` command. In the TMR, we installed Distributed Monitoring, Software Distribution, Inventory, and TEC Event Adapters. If you would like to know which methods are included in the dependency set, just invoke the `wdepset` command as follows:

```
# wdepset -v dependency_set_OID
```

For example, the following example shows which methods are included in the courier_lcf dependency set (for Software Distribution).

```
# wdepset -v 1613591617.1.711
win3x:
bin/win3x/TAS/MANAGED_NODE/confirm.exe,,0
bin/win3x/TAS/MANAGED_NODE/UpdStat.exe,,0
bin/win3x/TAS/MANAGED_NODE/tivulfra.dll,,0
bin/win3x/TAS/MANAGED_NODE/tivuljpn.dll,,0
bin/win3x/TAS/MANAGED_NODE/tivulptb.dll,,0
bin/win3x/TAS/MANAGED_NODE/wsyschg.exe,,0
bin/win3x/TAS/MANAGED_NODE/WSYSUPD.BAT,,0
bin/win3x/TAS/MANAGED_NODE/WSYSUPD.PIF,,0
bin/win3x/TAS/MANAGED_NODE/COMMAND.PIF,%RUN%,8
bin/win3x/TAS/MANAGED_NODE/spawn32.dll,,0
bin/win3x/TAS/MANAGED_NODE/clisrv.exe,,0
bin/win3x/TAS/MANAGED_NODE/waddicon.exe,,0
bin/win3x/TAS/MANAGED_NODE/wclrblk.exe,,0
bin/win3x/TAS/MANAGED_NODE/wclrline.exe,,0
bin/win3x/TAS/MANAGED_NODE/wdskspc.exe,,0
bin/win3x/TAS/MANAGED_NODE/weditini.exe,,0
bin/win3x/TAS/MANAGED_NODE/winsblk.exe,,0
bin/win3x/TAS/MANAGED_NODE/winsline.exe,,0
bin/win3x/TAS/MANAGED_NODE/wmrgini.exe,,0
bin/win3x/TAS/MANAGED_NODE/wrplblk.exe,,0
bin/win3x/TAS/MANAGED_NODE/wrplline.exe,,0
bin/win3x/TAS/MANAGED_NODE/wrunprog.exe,,0
bin/win3x/TAS/MANAGED_NODE/wseterr.exe,,0
win95:
bin/win95/TAS/MANAGED_NODE/confirm.exe,,0
bin/win95/TAS/MANAGED_NODE/UpdStat.exe,,0
bin/win95/TAS/MANAGED_NODE/tivulfra.dll,,0
bin/win95/TAS/MANAGED_NODE/tivuljpn.dll,,0
bin/win95/TAS/MANAGED_NODE/tivulptb.dll,,0
bin/win95/TAS/MANAGED_NODE/wsyschg.exe,,0
bin/win95/TAS/MANAGED_NODE/wsysupd.bat,,0
bin/win95/TAS/MANAGED_NODE/wproupd.exe,cache,0
bin/win95/TAS/MANAGED_NODE/wseterr.exe,,0
w32-ix86:
bin/w32-ix86/TAS/MANAGED_NODE/wrunuiep.exe,,0
bin/w32-ix86/TAS/MANAGED_NODE/wsyschg.exe,,0
bin/w32-ix86/TAS/MANAGED_NODE/UpdStat.exe,,0
bin/w32-ix86/TAS/MANAGED_NODE/tivulfra.dll,,0
bin/w32-ix86/TAS/MANAGED_NODE/tivuljpn.dll,,0
bin/w32-ix86/TAS/MANAGED_NODE/tivulptb.dll,,0
bin/w32-ix86/TAS/MANAGED_NODE/wsysupd.bat,,0
bin/w32-ix86/TAS/MANAGED_NODE/wproupd.exe,cache,0
bin/w32-ix86/TAS/MANAGED_NODE/wseterr.exe,,0
w32-axp:
bin/w32-axp/TAS/MANAGED_NODE/wrunuiep.exe,,0
bin/w32-axp/TAS/MANAGED_NODE/wsyschg.exe,,0
bin/w32-axp/TAS/MANAGED_NODE/UpdStat.exe,,0
bin/w32-axp/TAS/MANAGED_NODE/tivulfra.dll,,0
bin/w32-axp/TAS/MANAGED_NODE/tivuljpn.dll,,0
bin/w32-axp/TAS/MANAGED_NODE/tivulptb.dll,,0
bin/w32-axp/TAS/MANAGED_NODE/wsysupd.bat,,0
#
```

The above sample introduces how to display the methods set for Software Distribution. The output of the `wdepset` command can be very large because the `wdepset` command shows all files that are defined in the dependency set. These Endpoint methods are going to be located under the /usr/local/Tivoli/bin/lcf_bundle directory and loaded to the method cache in the Endpoint where the application is called for the first time (for instance, when the Sentry profile is distributed to the Endpoint subscriber for the first time). The following figure (Figure 44) shows the relationship between the Endpoint methods and the dependency manager.

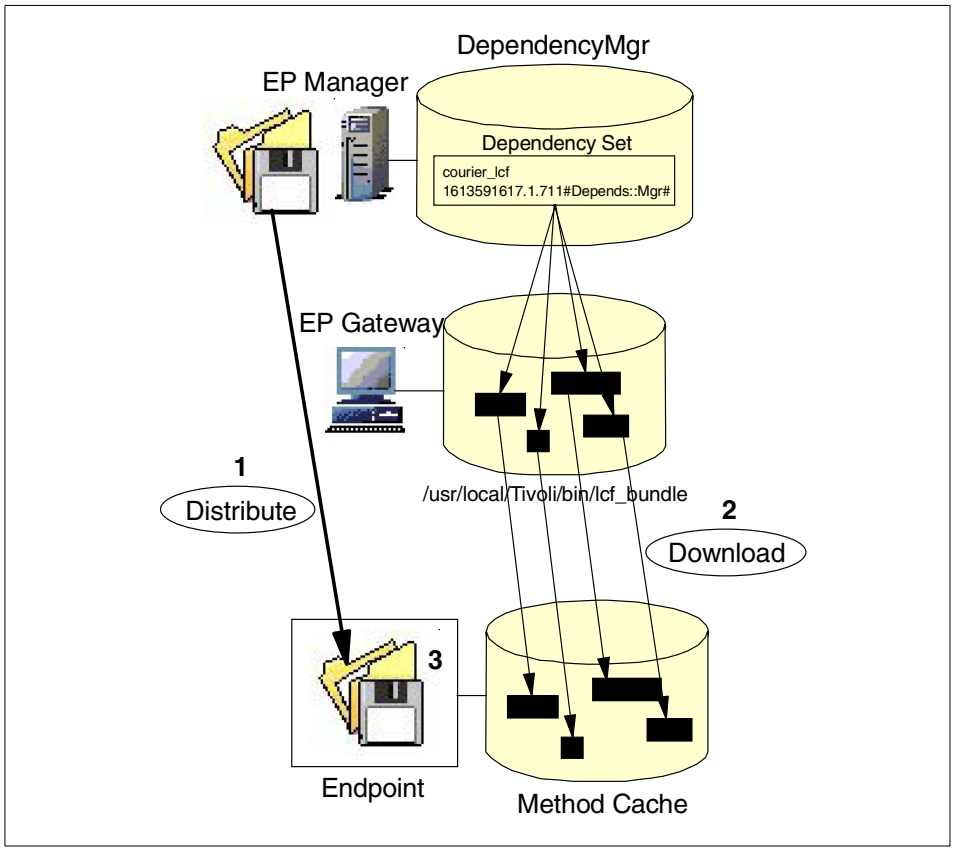


Figure 44. The dependency manager and Endpoint methods

1. Initially, the application profile is distributed to the Endpoint subscriber.
2. The Endpoint Gateway checks the Endpoint for the existence of the appropriate executables that are defined in the dependency set. Since this is the initial profile distribution, the Endpoint cannot find the executable files in the cache on its local disk. Therefore, the Endpoint Gateway loads these executables into the Endpoint method cache.
3. The application profile that is distributed to the Endpoint works with these Endpoint methods.

4.3.6.3 What is Endpoint method preloading?

In last a few sections, we introduced how Endpoint downloads the methods that are needed to perform the management operation on the Endpoint. In large scale environment, these method downloading processes may affect the performance of the Endpoint Gateway or network. For example, the total

volume of the Endpoint methods that are defined in the Distributed Monitoring dependency set is more than 3 MB (Windows NT Endpoint). The next table (Table 7) shows the approximate size of the methods that are defined in the dependency set for each Tivoli Management application (Windows NT Endpoint).

Table 7. Windows NT Endpoint method size

Application	Approximate Size
Software Distribution	1,028 KB
Inventory	696 KB
Distributed Monitoring	3,231 KB

If hundreds of Endpoints attempt to download methods for Distributed Monitoring at the same time, what happens? It is not good for the performance. To avoid this situation, we introduce the Endpoint method preloading technique.

The following figure (Figure 45) shows how the Endpoint that is preloaded with Endpoint methods works when the profile is distributed to the Endpoint for the first time.

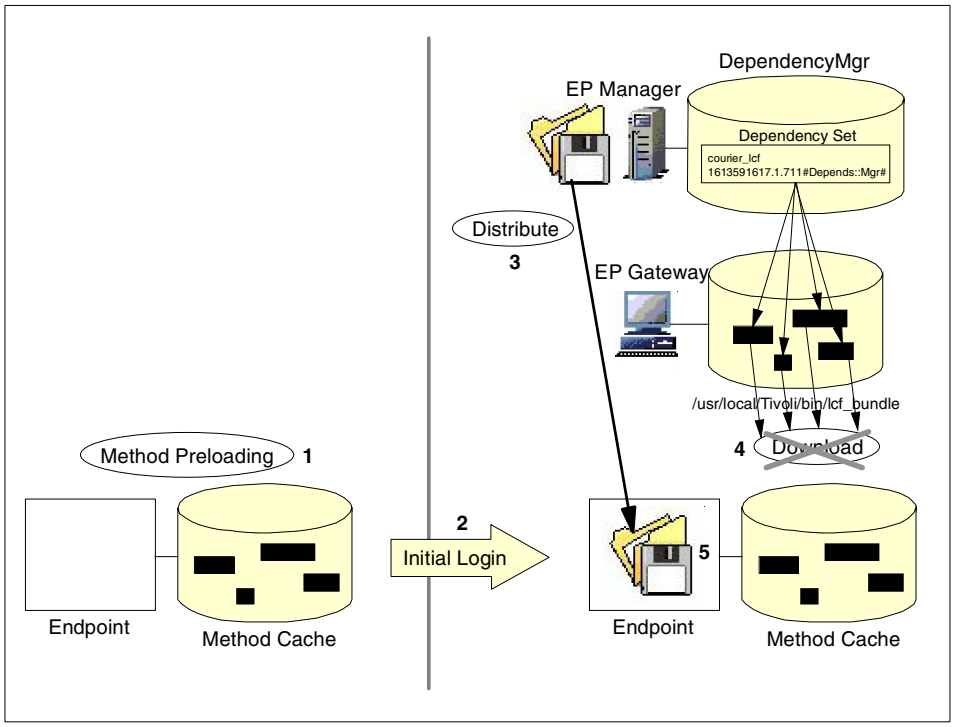


Figure 45. Endpoint method preloading

1. When you install Endpoint software, you also install the Endpoint methods that are going to be needed by the Tivoli Management applications.
2. The Endpoint performs initial login to the appropriate Endpoint Gateway and joins the TMR.
3. The profile is distributed to the Endpoint subscriber when the Tivoli Management application runs on the Endpoint for the first time.
4. The Endpoint Gateway checks the Endpoint for the existence of the appropriate executables that are defined in the dependency set. The Endpoint finds the appropriate executable files in the cache on the local disk; so, the Endpoint Gateway does not load these executables into the cache.
5. The profile that is distributed to the Endpoint works with the preloaded Endpoint methods.

In the following sections, we provide information on how to implement the method preloaded TMA and show the log files when the method preloaded TMA active.

Note

As we mentioned, once the Endpoint method is downloaded to the Endpoint method cache, the Endpoint does not download the same version of the Endpoint method. Therefore, the method preloading is effective only when the profile is distributed to the Endpoint for the first time.

4.3.6.4 How to preload Endpoint methods

The following shows an overview on how to preload the Endpoint method into the Endpoint.

1. Install the Endpoint normally, and the Endpoint is made to perform initial login to the appropriate Endpoint Gateway.
2. Distribute the profiles of the applications that you are going to use in your environment. For example, a file package (Software Distribution) or Sentry profile (Distributed Monitoring).
3. Copy the files that are located under the cache directory (C:\Tivoli\lcf\dat\1\cache by default) to another place, such as a diskette. If you are going to use Distributed Monitoring, you need to save not only the default cache directory but also the Distributed Monitoring method cache directory (normally C:\Tivoli\lcf\dat\1\lcf\sentry). The next figure (Figure 46) shows the directory structure for the Endpoint method cache.
4. Install the Endpoint into another machine normally and stop the lcfd daemon. Then copy the methods that are saved at the previous process to the appropriate directory on the system.
5. Restart the lcfd daemon. After restarting the lcfd daemon, the Endpoint is going to find the methods on the local cache when the profile is distributed for the first time so that the Endpoint is not going to download these methods from the Endpoint Gateway.

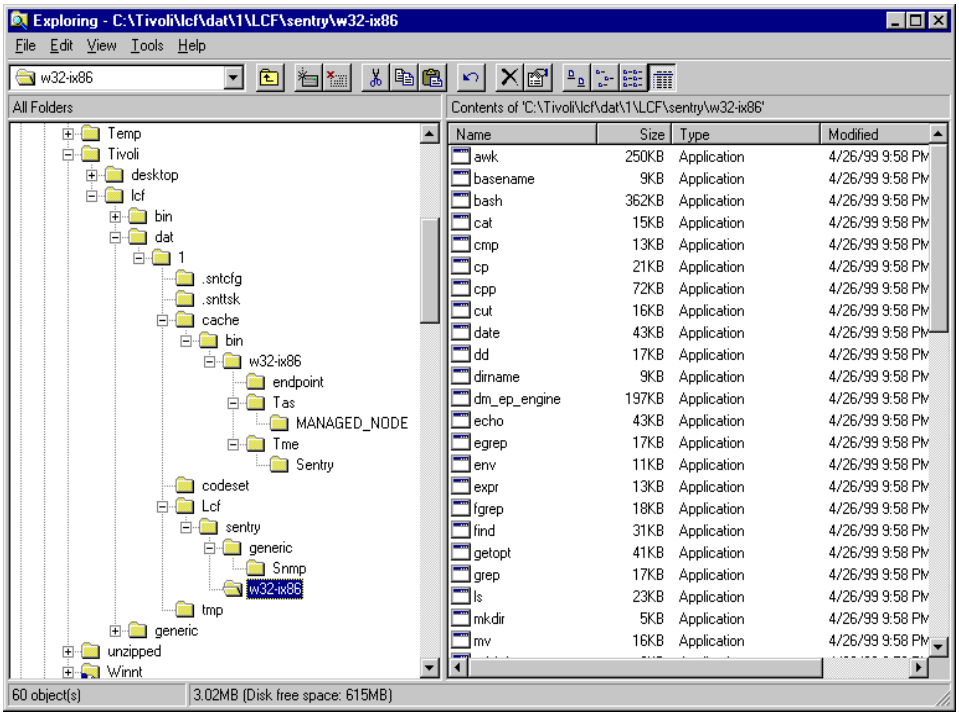


Figure 46. Endpoint method cache directories

However, in a small environment, for example less than 100 Endpoints in deployment, the method preloading might not be such an efficient way to improve its performance. On the other hand, in a large-scale environment, for example more than 1,000 Endpoints in deployment, the method preloading is going to work effectively and reduce the load of the Endpoint Gateway and network congestion. The following sections introduce how to implement the method preloading in a large-scale deployment.

4.3.6.5 Method preloading in large scale deployment

In a large-scale deployment, it is hard to copy the method files in the cache to each Endpoint machine. Therefore, you have to consider another way that can be implemented easily even if it is for more than 1,000 Endpoints in deployment.

The following figure (Figure 47) shows the overview of how to deploy the method preloaded Endpoint in a large-scale environment.

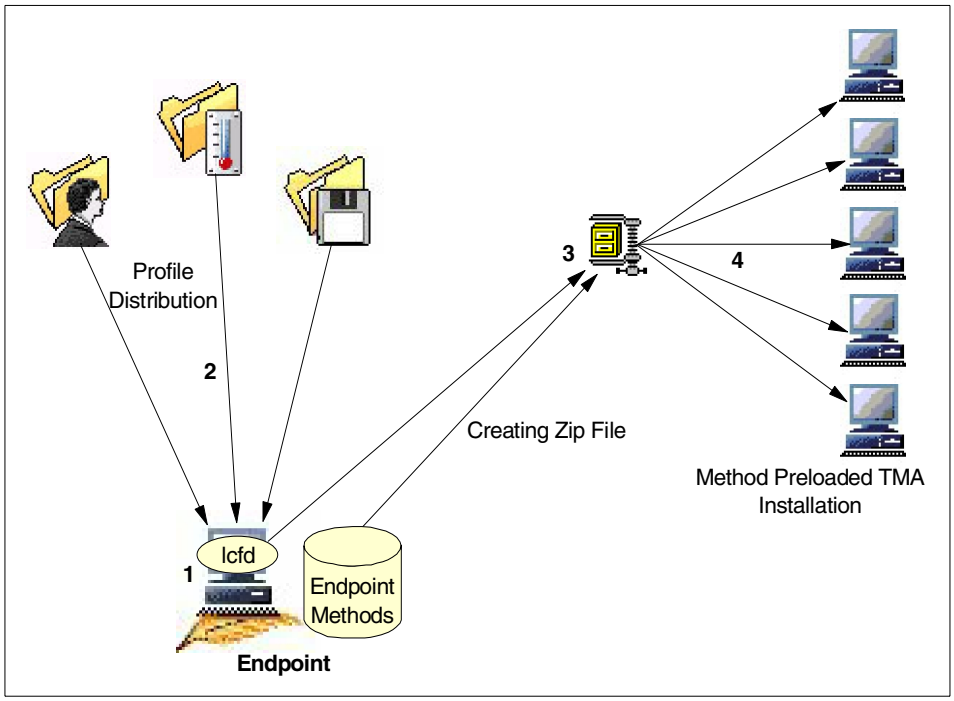


Figure 47. Deploying method preloaded Endpoint in a large-scale environment

1. Install the Endpoint normally, and the Endpoint is made to perform initial login to the appropriate Endpoint Gateway.
2. Distribute the profile of all applications that you will use in your management environment. Then the Endpoint methods that are defined in the dependency set are loaded to the Endpoint method cache.
3. Create a zip file that includes Endpoint software and Endpoint methods.
4. Install the zip file to the installation target. In this installation, the process installs not only the Endpoint software but also Endpoint methods at once.

As you can see, first of all, you need to create the master Endpoint that includes all Endpoint methods that will be needed to perform the management operations in your environment. Then you create the clone of the master Endpoint by using the zip file.

The following example makes the Endpoint software image (zip file) that has already contained all method files that will be used by the Tivoli Management Applications in your environment. We introduce two ways to install the

method preloaded Endpoint: One is manual operation, and the other is by using Windows NT logon script.

> **Note**
>
> In this example, we introduce only the Windows NT Endpoint case. However, the procedure of implementing the method preloaded Endpoint is not so different for each platform.

Step 1: Creating zipped Endpoint code

First of all, you need to create the zip file that contains Endpoint software and all methods that will be needed by the Tivoli Management applications running on the Endpoint. The following are instructions on how to create the zip file.

1. Install the Endpoint normally, and the Endpoint is made to perform the initial login to the appropriate Endpoint Gateway.
2. Distribute profile of the application that you are going to use in your environment. For example, a file package (Software Distribution) or Sentry profile (Distributed Monitoring).
3. Remove the DAT file (lcf.dat) and ID file (lcf.id) from the LCF_DATDIR directory (typically, C:\Tivoli\lcf\dat\1).
4. Copy the ntconfig.exe and libacct.dll files from the Endpoint Gateway machine (typically $BINDIR/../lcf_bundle/bin/w32-ix86/endpoint/ntconfig.exe) to the Endpoint machine (normally C:\Tivoli\lcf\bin\w32-ix86\mrt directory).
5. Create the zip file that contains all files related to the Endpoint (refer to Figure 48). Normally, the following files should be zipped:

```
C:\etc\Tivoli\C\*.*
C:\Tivoli\lcf\*.*
C:\Winnt\Tivoli\lcf\1\*.*
```

> **Note**
>
> In the above procedure, you can skip process 4 if you install the TMA with the `winstlcf` command. The `winstlcf` command will automatically install the ntconfig.exe and libacct.dll files in the C:\Tivoli\lcf\bin\w32-ix86\mrt directory.

The following example (Figure 48) shows the contents of the zipped file containing the Endpoint methods for Software Distribution.

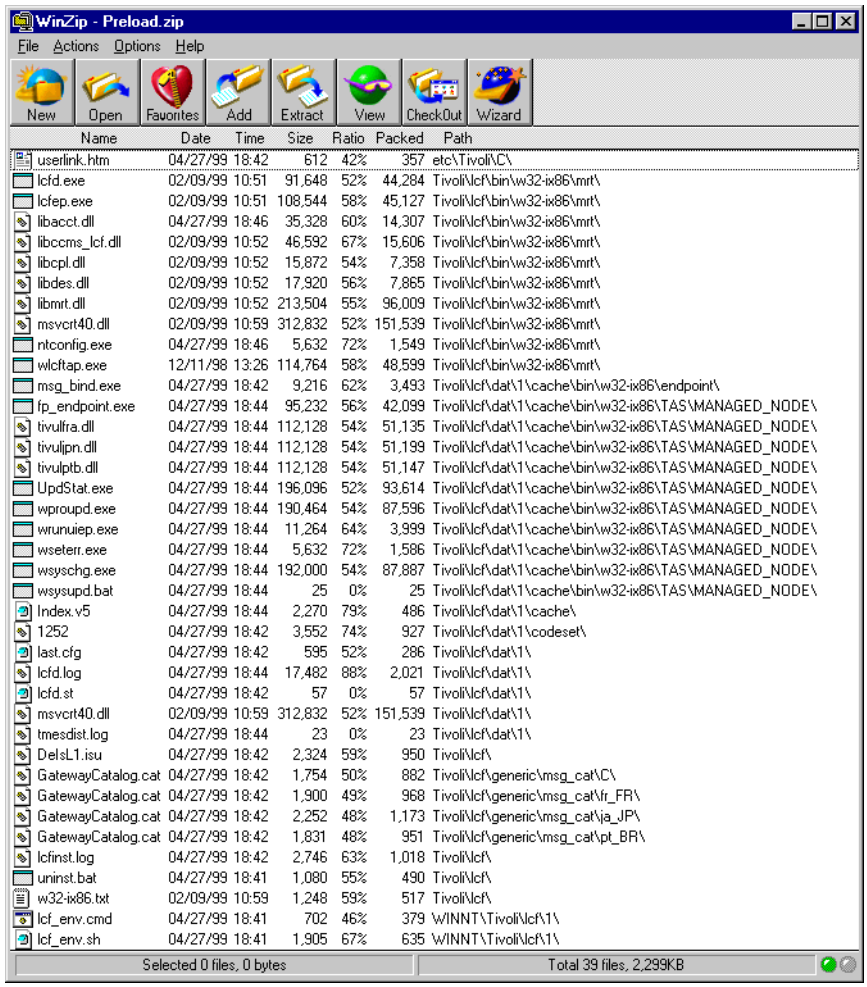


Figure 48. Zipped files for Endpoint method preloading

As you can see, all files that are needed for Endpoint installation and all method files that will be used for Software Distribution management operations are included in the zipped file.

Note

If you use the WinZip utility to create the zip file, you should create the self-extracting zip file. The self-extracting zip file simplifies the deployment process of the method preloaded Endpoints. Especially, if you plan to install these zip files using NT logon script, you should create the self-extracting zip file.

Step 2: Install method preloaded Endpoint manually

The following are instructions on how to install the method preloaded Endpoint manually on a Windows NT machine.

1. Before installation, the zip file that is created in the previous step should be copied onto the installation target machine, or the target machine should mount the network drive that contains the zip file. Make sure of this and unzip the zip file on each target machine.
2. To add an unprivileged *nobody* user and admin privilege group on the target system, execute the following command:

   ```
   C:\Tivoli\lcf\bin\w32-ix86\mrt\ntconfig.exe -e
   ```

3. To start Tivoli Authentication Package (TAP), execute the following command:

   ```
   C:\Tivoli\lcf\bin\w32-ix86\mrt\wlcftap -a
   ```

4. To install the Windows NT taskbar icon, execute the following command (this process is optional):

   ```
   C:\Tivoli\lcf\bin\w32-ix86\mrt\lcfep.exe -x -i
   ```

5. To start lcfd daemon, execute the `lcfd` command with appropriate options. The following is the sample usage of the `lcfd` command:

   ```
   C:\Tivoli\lcf\bin\w32-ix86\mrt\lcfd.exe -i -C C:\Tivoli\lcf\dat\1 -g
   9.3.187.158+9494:9.3.187.154+9494
   ```

Step 3: Install method preloaded Endpoint using NT logon script

As we mentioned, there are two ways to implement the method preloaded Endpoint, manually or by using an Windows NT logon script. This section describes how to use an Windows NT logon script to implement the method preloaded Endpoint for Windows NT. The following figure (Figure 49) shows the procedure of implementing the method preloaded Endpoint using an Windows NT logon script.

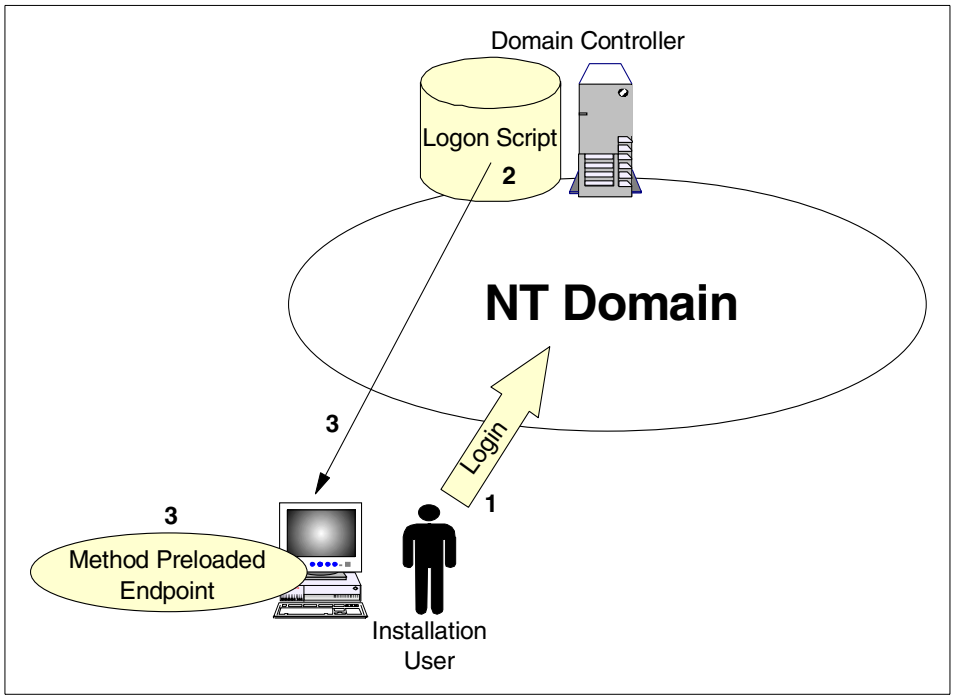


Figure 49. Install method preloaded Endpoint using Windows NT logon script

1. The user of the target machine where the method preloaded Endpoint will be installed attempts to log into the NT domain.
2. The domain controller executes the logon script defined in the user profile of the installation user.
3. The logon script performs the method preloaded Endpoint installation process. As a result, the Endpoint software and Endpoint methods are installed to the machine. Then the Endpoint attempts to perform initial login to the appropriate Endpoint Gateway.

We created a sample logon script for method preloaded Endpoint mass installation.

```
@echo off
REM Login.bat
REM Silently abort if not running Windows
if "%windir%" == "" goto end

if "%OS%" == "Windows_NT" goto NT
REM Silently abort if not running Windows NT
goto end

:NT
if not exist %SystemDrive%\Tivoli\lcf\bin\w32-ix86\mrt\lcfd.exe goto alert

set LCFOPTS=%1 %2 %3 %4 %5 %6 %7 %8 %9
set LCFROOT=%SystemDrive%\Tivoli\lcf

REM Add unprivileged 'nobody' user and admin privileges group
%LCFROOT%\bin\w32-ix86\mrt\ntconfig.exe -e

REM Start TAP
%LCFROOT%\bin\w32-ix86\mrt\wlcftap -a

REM Install Tivoli icon
%LCFROOT%\bin\w32-ix86\mrt\lcfep.exe -x -i

REM Start the TMA and install it as a service
%LCFROOT%\bin\w32-ix86\mrt\lcfd.exe -i -C %LCFROOT%\dat\1 %LCFOPTS%

set LCFOPTS=
set LCFROOT=

goto end

:alert
echo Could not find the files needed to start the Tivoli Management Agent.
echo Please contact your Tivoli Administrator.
goto end

:end
```

4.3.6.6 Using other tools

To deploy the method preloaded Endpoint in a large-scale environment, you can use other tools, such as the *ghost*. The *ghost* is a hard (physical) disk image copier. In this case, the all managed systems that you deploy must be the same configurations including hardware and software configurations.

4.3.6.7 How the method preloaded Endpoint works

In this section, we show the log file (lcfd.log) of the normal Endpoint and the Endpoint that has preloaded the methods before distributing the profile.

Normal Endpoint

The following shows the example of the lcfd.log file when the file package is distributed to the normal Endpoint for the first time. As you can see, the

Endpoint downloads all methods that are defined in the dependency set (courier_lcf) into its local cache. By default, the Endpoint works as follows:

```
Apr 26 16:42:47 Q lcfd New connection from 9.3.187.158+53024
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:47 Q lcfd Leaving net_recv: bytes=1028, (type=0 session=340329480)
Apr 26 16:42:47 Q lcfd Entering send_methstat
Apr 26 16:42:47 Q lcfd Entering send_struct
Apr 26 16:42:47 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:47 Q lcfd Leaving send_struct
Apr 26 16:42:47 Q lcfd Leaving send_methstat
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:47 Q lcfd Leaving net_recv: bytes=96, (type=7 session=340329480)
Apr 26 16:42:47 2 lcfd reading:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\fp_endpoint.exe
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:47 Q lcfd Leaving net_recv: bytes=95242, (type=11 session=340329480)
Apr 26 16:42:47 Q lcfd Entering send_methstat
Apr 26 16:42:47 Q lcfd Entering send_struct
Apr 26 16:42:47 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:47 Q lcfd Leaving send_struct
Apr 26 16:42:47 Q lcfd Leaving send_methstat
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:47 Q lcfd Leaving net_recv: bytes=97, (type=7 session=340329480)
Apr 26 16:42:47 2 lcfd reading:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\wrunuiep.exe
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:47 Q lcfd Leaving net_recv: bytes=11274, (type=11 session=340329480)
Apr 26 16:42:47 Q lcfd Entering send_methstat
Apr 26 16:42:47 Q lcfd Entering send_struct
Apr 26 16:42:47 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:47 Q lcfd Leaving send_struct
Apr 26 16:42:47 Q lcfd Leaving send_methstat
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:47 Q lcfd Leaving net_recv: bytes=96, (type=7 session=340329480)
Apr 26 16:42:47 2 lcfd reading:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\wsyschg.exe
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:47 Q lcfd Leaving net_recv: bytes=131082, (type=11 session=340329480)
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:47 Q lcfd Leaving net_recv: bytes=60938, (type=11 session=340329480)
Apr 26 16:42:47 Q lcfd Entering send_methstat
Apr 26 16:42:47 Q lcfd Entering send_struct
Apr 26 16:42:47 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:47 Q lcfd Leaving send_struct
Apr 26 16:42:47 Q lcfd Leaving send_methstat
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:47 Q lcfd Leaving net_recv: bytes=96, (type=7 session=340329480)
Apr 26 16:42:47 2 lcfd reading:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\UpdStat.exe
Apr 26 16:42:47 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=131082, (type=11 session=340329480)
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=65034, (type=11 session=340329480)
Apr 26 16:42:48 Q lcfd Entering send_methstat
Apr 26 16:42:48 Q lcfd Entering send_struct
Apr 26 16:42:48 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:48 Q lcfd Leaving send_struct
Apr 26 16:42:48 Q lcfd Leaving send_methstat
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=97, (type=7 session=340329480)
Apr 26 16:42:48 2 lcfd reading:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\tivulfra.dll
```

```
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=112138, (type=11 session=340329480)
Apr 26 16:42:48 Q lcfd Entering send_methstat
Apr 26 16:42:48 Q lcfd Entering send_struct
Apr 26 16:42:48 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:48 Q lcfd Leaving send_struct
Apr 26 16:42:48 Q lcfd Leaving send_methstat
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=97, (type=7 session=340329480)
Apr 26 16:42:48 2 lcfd reading:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\tivuljpn.dll
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=112138, (type=11 session=340329480)
Apr 26 16:42:48 Q lcfd Entering send_methstat
Apr 26 16:42:48 Q lcfd Entering send_struct
Apr 26 16:42:48 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:48 Q lcfd Leaving send_struct
Apr 26 16:42:48 Q lcfd Leaving send_methstat
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=97, (type=7 session=340329480)
Apr 26 16:42:48 2 lcfd reading:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\tivulptb.dll
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=112138, (type=11 session=340329480)
Apr 26 16:42:48 Q lcfd Entering send_methstat
Apr 26 16:42:48 Q lcfd Entering send_struct
Apr 26 16:42:48 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:48 Q lcfd Leaving send_struct
Apr 26 16:42:48 Q lcfd Leaving send_methstat
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=96, (type=7 session=340329480)
Apr 26 16:42:48 2 lcfd reading:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\wsysupd.bat
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=35, (type=11 session=340329480)
Apr 26 16:42:48 Q lcfd Entering send_methstat
Apr 26 16:42:48 Q lcfd Entering send_struct
Apr 26 16:42:48 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:48 Q lcfd Leaving send_struct
Apr 26 16:42:48 Q lcfd Leaving send_methstat
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:48 Q lcfd Leaving net_recv: bytes=103, (type=7 session=340329480)
Apr 26 16:42:48 2 lcfd reading: cache\bin\w32-ix86\TAS\MANAGED_NODE\wproupd.exe
Apr 26 16:42:48 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:49 Q lcfd Leaving net_recv: bytes=131082, (type=11 session=340329480)
Apr 26 16:42:49 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:49 Q lcfd Leaving net_recv: bytes=59402, (type=11 session=340329480)
Apr 26 16:42:49 Q lcfd Entering send_methstat
Apr 26 16:42:49 Q lcfd Entering send_struct
Apr 26 16:42:49 Q lcfd net_send of 52 bytes, session 340329480
Apr 26 16:42:49 Q lcfd Leaving send_struct
Apr 26 16:42:49 Q lcfd Leaving send_methstat
Apr 26 16:42:49 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:49 Q lcfd Leaving net_recv: bytes=96, (type=7 session=340329480)
Apr 26 16:42:49 2 lcfd reading:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\wseterr.exe
Apr 26 16:42:49 Q lcfd Entering net_recv, receive a message
Apr 26 16:42:49 Q lcfd Leaving net_recv: bytes=5642, (type=11 session=340329480)
Apr 26 16:42:49 Q lcfd setting-up inherit fd. netfd=304
Apr 26 16:42:49 1 lcfd Spawning:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\fp_endpoint.exe, ses: 14490408
Apr 26 16:42:49 Q lcfd Entering Listener (running).
Apr 26 16:42:49 Q lcfd Entering net_wait_for_connection, timeout=-1 handle=0x305b48
```

```
Apr 26 16:42:49 Q lcfd cti_accept (timeout=-1)
Apr 26 16:42:49 Q MethInit Entering mrt_run
Apr 26 16:42:49 Q MethInit argv: session_id=14490408
Apr 26 16:42:49 Q MethInit Communication timeout set: 120.
Apr 26 16:42:49 Q MethInit Entering comm_reconnect
Apr 26 16:42:49 Q MethInit inherited fd. return from net_associated_fd. ipc=3159504,
netfd=304
Apr 26 16:42:49 Q MethInit Entering run_impl
Apr 26 16:42:49 Q MethInit Entering send_methstat
Apr 26 16:42:49 Q MethInit Entering send_struct
Apr 26 16:42:49 Q MethInit net_send of 52 bytes, session 340329480
Apr 26 16:42:49 Q MethInit Leaving send_struct
Apr 26 16:42:49 Q MethInit Leaving send_methstat
Apr 26 16:42:49 Q MethInit waiting for input args
Apr 26 16:42:49 Q MethInit Entering net_recv, receive a message
Apr 26 16:42:49 Q MethInit Leaving net_recv: bytes=8537, (type=3 session=340329480)
Apr 26 16:42:49 2 MethInit Looking for method: fps_install.
Apr 26 16:42:49 Q fps_install calling method.
Apr 26 16:42:49 Q fps_install method returned.
Apr 26 16:42:49 Q fps_install send_results (max/len) 80/62
Apr 26 16:42:49 Q fps_install Entering send_methstat
Apr 26 16:42:49 Q fps_install Entering send_struct
Apr 26 16:42:49 Q fps_install net_send of 116 bytes, session 340329480
Apr 26 16:42:49 Q fps_install Leaving send_struct
Apr 26 16:42:49 Q fps_install Leaving send_methstat
Apr 26 16:42:49 2 fps_install Clean Shutdown fps_install.
```

Method preloaded Endpoint

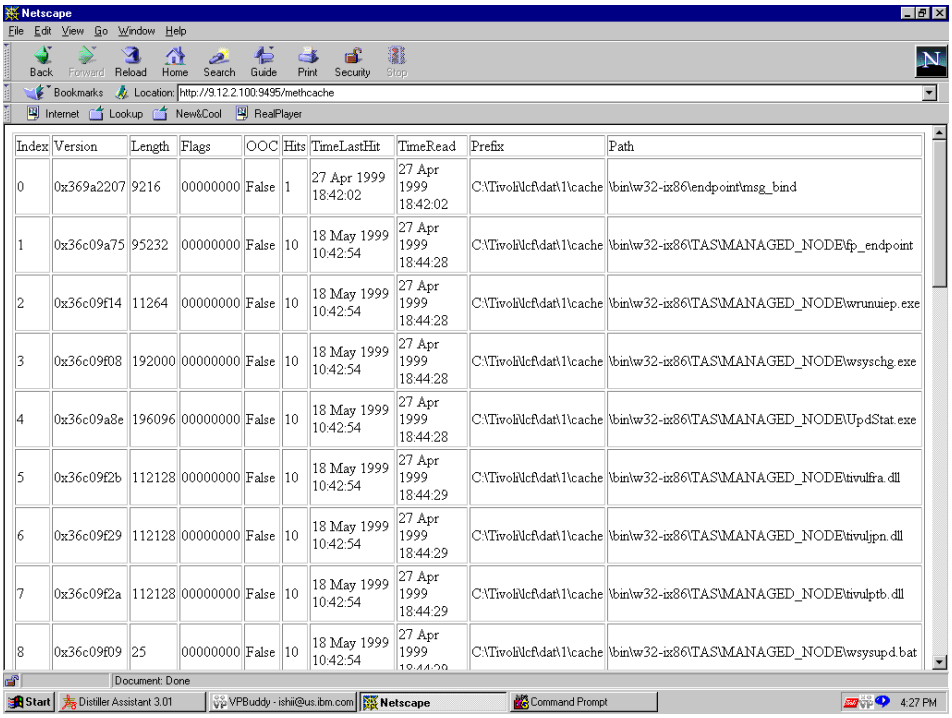
The following shows the example of the lcfd.log file when the file package is distributed to the method preloaded Endpoint for the first time. As you can see, the Endpoint finds the methods that will be needed by Software Distribution on its local cache. Therefore, the method preloaded Endpoint does not download any methods regarding Software Distribution in this case.

```
Apr 26 20:21:16 Q lcfd New connection from 9.3.187.158+54238
Apr 26 20:21:16 Q lcfd Entering net_recv, receive a message
Apr 26 20:21:16 Q lcfd Leaving net_recv: bytes=1028, (type=0 session=340329520)
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\fp_endpoint.exe
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\wrunuiep.exe
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\wsyschg.exe
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\UpdStat.exe
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\tivulfra.dll
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\tivuljpn.dll
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\tivulptb.dll
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\wsysupd.bat
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
cache\bin\w32-ix86\TAS\MANAGED_NODE\wproupd.exe
Apr 26 20:21:16 2 lcfd CacheCheckExistence: Found:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\wseterr.exe
Apr 26 20:21:16 Q lcfd setting-up inherit fd. netfd=208
Apr 26 20:21:16 1 lcfd Spawning:
C:\Tivoli\lcf\dat\1\cache\bin\w32-ix86\TAS\MANAGED_NODE\fp_endpoint.exe, ses: 14490430
Apr 26 20:21:16 Q lcfd Entering Listener (running).
```

```
Apr 26 20:21:16 Q lcfd Entering net_wait_for_connection, timeout=-1 handle=0x305b18
Apr 26 20:21:16 Q lcfd cti_accept (timeout=-1)
Apr 26 20:21:16 Q MethInit Entering mrt_run
Apr 26 20:21:16 Q MethInit argv: session_id=14490430
Apr 26 20:21:16 Q MethInit Communication timeout set: 120.
Apr 26 20:21:16 Q MethInit Entering comm_reconnect
Apr 26 20:21:16 Q MethInit inherited fd. return from net_associated_fd. ipc=3159512,
netfd=208
Apr 26 20:21:16 Q MethInit Entering run_impl
Apr 26 20:21:16 Q MethInit Entering send_methstat
Apr 26 20:21:16 Q MethInit Entering send_struct
Apr 26 20:21:16 Q MethInit net_send of 52 bytes, session 340329520
Apr 26 20:21:16 Q MethInit Leaving send_struct
Apr 26 20:21:16 Q MethInit Leaving send_methstat
Apr 26 20:21:16 Q MethInit waiting for input args
Apr 26 20:21:16 Q MethInit Entering net_recv, receive a message
Apr 26 20:21:16 Q MethInit Leaving net_recv: bytes=8537, (type=3 session=340329520)
Apr 26 20:21:16 2 MethInit Looking for method: fps_install.
Apr 26 20:21:16 Q fps_install calling method.
Apr 26 20:21:16 Q fps_install method returned.
Apr 26 20:21:16 Q fps_install send_results (max/len) 80/62
Apr 26 20:21:16 Q fps_install Entering send_methstat
Apr 26 20:21:16 Q fps_install Entering send_struct
Apr 26 20:21:16 Q fps_install net_send of 116 bytes, session 340329520
Apr 26 20:21:16 Q fps_install Leaving send_struct
Apr 26 20:21:16 Q fps_install Leaving send_methstat
Apr 26 20:21:17 2 fps_install Clean Shutdown fps_install.
```

4.3.6.8 Checking the loaded Endpoint methods

To track the status of preloaded Endpoint methods, you can use the Endpoint Web interface. Use the Web sites `http://endpoint_addr:9494` (for version 3.6 of TMA) or `http://endpoint_addr:9495` (for version 3.6.1 of TMA), then go to the List Method Cache menu. The following figure (Figure 50) shows the list of the Endpoint method cache that is provided by the Endpoint Web interface.



Index	Version	Length	Flags	OOC	Hits	TimeLastHit	TimeRead	Prefix	Path
0	0x369a2207	9216	00000000	False	1	27 Apr 1999 18:42:02	27 Apr 1999 18:42:02	C:\Tivoli\lcf\dat\1\cache	\bin\w32-ix86\endpoint\msg_bind
1	0x36c09a75	95232	00000000	False	10	18 May 1999 10:42:54	27 Apr 1999 18:44:28	C:\Tivoli\lcf\dat\1\cache	\bin\w32-ix86\TAS\MANAGED_NODE\fp_endpoint
2	0x36c09f14	11264	00000000	False	10	18 May 1999 10:42:54	27 Apr 1999 18:44:28	C:\Tivoli\lcf\dat\1\cache	\bin\w32-ix86\TAS\MANAGED_NODE\wrunuiep.exe
3	0x36c09f08	192000	00000000	False	10	18 May 1999 10:42:54	27 Apr 1999 18:44:28	C:\Tivoli\lcf\dat\1\cache	\bin\w32-ix86\TAS\MANAGED_NODE\wsyschg.exe
4	0x36c09a8e	196096	00000000	False	10	18 May 1999 10:42:54	27 Apr 1999 18:44:28	C:\Tivoli\lcf\dat\1\cache	\bin\w32-ix86\TAS\MANAGED_NODE\UpdStat.exe
5	0x36c09f2b	112128	00000000	False	10	18 May 1999 10:42:54	27 Apr 1999 18:44:29	C:\Tivoli\lcf\dat\1\cache	\bin\w32-ix86\TAS\MANAGED_NODE\tivulfra.dll
6	0x36c09f29	112128	00000000	False	10	18 May 1999 10:42:54	27 Apr 1999 18:44:29	C:\Tivoli\lcf\dat\1\cache	\bin\w32-ix86\TAS\MANAGED_NODE\tivuljpn.dll
7	0x36c09f2a	112128	00000000	False	10	18 May 1999 10:42:54	27 Apr 1999 18:44:29	C:\Tivoli\lcf\dat\1\cache	\bin\w32-ix86\TAS\MANAGED_NODE\tivulptb.dll
8	0x36c09f09	25	00000000	False	10	18 May 1999 10:42:54	27 Apr 1999	C:\Tivoli\lcf\dat\1\cache	\bin\w32-ix86\TAS\MANAGED_NODE\wsysupd.bat

Figure 50. The list of Endpoint method cache

The information that you can see in the Endpoint Web interface is the same as the contents of the Index.v5 file that is located under the C:\Tivoli\lcf\dat\1\cache directory. The Index.v5 file contains the information of the Endpoint methods that have already been downloaded into the Endpoint method cache. The following example shows the contents of the Index.v5 file.

```
Index|Version|Size|Flags|OOC|Hits|TimeLastHit|TimeRead|Prefix|Path|
0|0x369a2207|9216|00000000|False|1|27 Apr 1999 18:42:02|27 Apr 1999
18:42:02|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\endpoint\msg_bind|
1|0x36c09a75|95232|00000000|False|10|18 May 1999 10:42:54|27 Apr 1999
18:44:28|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TAS\MANAGED_NODE\fp_endpoint|
2|0x36c09f14|11264|00000000|False|10|18 May 1999 10:42:54|27 Apr 1999
18:44:28|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TAS\MANAGED_NODE\wrunuiep.exe|
3|0x36c09f08|192000|00000000|False|10|18 May 1999 10:42:54|27 Apr 1999
18:44:28|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TAS\MANAGED_NODE\wsyschg.exe|
4|0x36c09a8e|196096|00000000|False|10|18 May 1999 10:42:54|27 Apr 1999
18:44:28|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TAS\MANAGED_NODE\UpdStat.exe|
5|0x36c09f2b|112128|00000000|False|10|18 May 1999 10:42:54|27 Apr 1999
18:44:29|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TAS\MANAGED_NODE\tivulfra.dll|
6|0x36c09f29|112128|00000000|False|10|18 May 1999 10:42:54|27 Apr 1999
18:44:29|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TAS\MANAGED_NODE\tivuljpn.dll|
7|0x36c09f2a|112128|00000000|False|10|18 May 1999 10:42:54|27 Apr 1999
18:44:29|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TAS\MANAGED_NODE\tivulptb.dll|
8|0x36c09f09|25|00000000|False|10|18 May 1999 10:42:54|27 Apr 1999
18:44:29|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TAS\MANAGED_NODE\wsysupd.bat|
9|0x36c09f0a|190464|00000000|True|10|18 May 1999 10:42:54|27 Apr 1999
18:44:29|cache|\bin\w32-ix86\TAS\MANAGED_NODE\wproupd.exe|
10|0x36c09f15|5632|00000000|False|10|18 May 1999 10:42:54|27 Apr 1999
18:44:29|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TAS\MANAGED_NODE\wseterr.exe|
11|0x36c09682|117248|00000000|False|36|18 Jun 1999 18:53:04|11 May 1999
17:03:14|C:\Tivoli\lcf\dat\1\cache|\bin\w32-ix86\TME\SENTRY\dogEndpoint|
12|0x36c09643|12800|00000000|True|10|11 May 1999 20:32:44|11 May 1999 17:03:15|LCF|\sentry\w32-ix86\wntmon.exe|
13|0x36c09644|12800|00000000|True|10|11 May 1999 20:32:44|11 May 1999 17:03:15|LCF|\sentry\w32-ix86\wntmon.dll|
14|0x36c09645|10752|00000000|True|10|11 May 1999 20:32:44|11 May 1999 17:03:15|LCF|\sentry\w32-ix86\wntevlog.exe|
15|0x36c09644|7168|00000000|True|10|11 May 1999 20:32:44|11 May 1999 17:03:15|LCF|\sentry\w32-ix86\wntreg.exe|
16|0x36c0963c|44032|00000000|True|10|11 May 1999 20:32:44|11 May 1999 17:03:15|LCF|\sentry\w32-ix86\echo.exe|
17|0x36c0966f|78848|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:21|LCF|\sentry\w32-ix86\wlseng.exe|
18|0x36c09670|78848|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:21|LCF|\sentry\w32-ix86\wrunprb.exe|
19|0x36c09671|78848|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:21|LCF|\sentry\w32-ix86\wstopeng.exe|
20|0x36c09672|78848|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:22|LCF|\sentry\w32-ix86\wclreng.exe|
21|0x36c09673|78848|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:22|LCF|\sentry\w32-ix86\wasync.exe|
22|0x36c0966e|78848|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:22|LCF|\sentry\w32-ix86\wengine.exe|
23|0x36c09647|23552|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:22|LCF|\sentry\w32-ix86\wsnmpget.exe|
24|0x36c09647|5281|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:22|LCF|\sentry\w32-ix86\wsnmpget.sh|
25|0x36c09648|4450|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:22|LCF|\sentry\w32-ix86\wsnmpmon|
26|0x36c09648|24064|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:22|LCF|\sentry\w32-ix86\wsnmpnext.exe|
27|0x36c09649|24576|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:23|LCF|\sentry\w32-ix86\wsnmpset.exe|
28|0x36c09649|17920|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:23|LCF|\sentry\w32-ix86\wsnmptrap.exe|
29|0x36c0966b|201728|00000000|True|36|18 Jun 1999 18:53:04|11 May 1999 17:03:23|LCF|\sentry\w32-ix86\dm_ep_engine.exe|
31|0x36c0c815|2746|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:24|LCF|\sentry\generic\SNMP\Compaq.OID|
32|0x36c0c1aa|2006|00000000|True|10|11 May 1999 20:32:46|11 May 1999 17:03:24|LCF|\sentry\generic\SNMP\rfc1213.OID|
```

Therefore, when you implement the Endpoint method preloading technique, the preloaded Endpoint methods should be the same as the method information that is contained in the Index.v5 file and Endpoint Web interface.

4.3.6.9 When method preloaded TMA is effective

As we mentioned, the method preloaded TMA is effective only when the application profile is distributed to the Endpoints for the first time. Therefore, it may be not so efficient in a small management environment. The most effective cases are as follows:

- The network bandwidth between the Endpoint Gateway and Endpoints is slow.
- A single Endpoint Gateway is managing many Endpoints.

- An application profile is distributed to many Endpoints for the first time all at once.

You need to consider whether the method preloaded TMA is effective or not at your deployment.

4.4 Conclusion of framework performance

To improve Tivoli Enterprise performance, it is important to tune the Tivoli Management Framework because the performance of all Tivoli Management applications depend on Tivoli Management Framework performance (refer to the Figure 28 on page 71). In this chapter, we have discussed the following:

- TMA behavior
- TMR server configurations
- Endpoint Gateway configurations
- Endpoint method preloading

These considerations should be important and useful for most customer environments. We strongly recommend that you follow the information that this chapter provides.

Chapter 5. Improving software distribution performance

The deployment applications are very popular applications for most customers and Tivoli Software Distribution is one of the major deployment applications. File transfer and data transfer are the most typical subjects of systems management.

Tivoli provides Software Distribution (previously known as "Courier"). Version 3.6 of the Software Distribution application supports the TMA. The operations are almost the same as the previous version of Software Distribution, but there are some considerations for improving performance and throughput. The following figure (Figure 51) shows the relationship between Software Distribution (Tivoli Management application) and other layers.

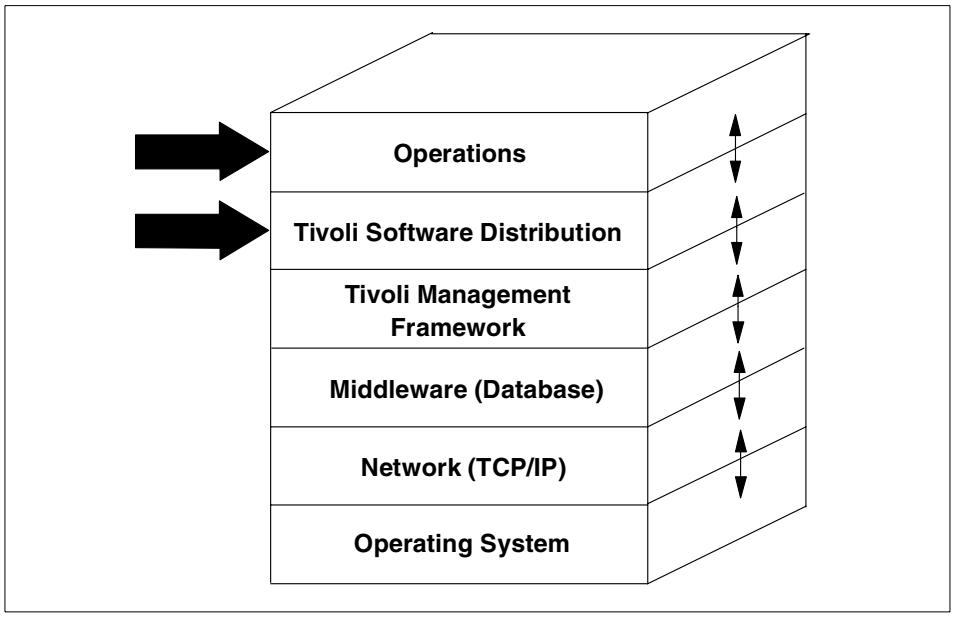


Figure 51. Tuning software distribution and its operations

In this chapter, we introduce the important issues for improving performance and throughput in a Software Distribution environment.

5.1 Software distribution internals

To improve performance and throughput in Software Distribution we need to know how Software Distribution works in a Tivoli Management environment.

Understanding Software Distribution behavior enables you to detect performance bottlenecks and improve its performance.

Software Distribution can involve the following types of machines:

- TMR server
- File package source host
- Endpoint Gateway
- MDist Repeater
- Target

A distribution operations are either from a Managed Node, including the TMR server, to another Managed Node, or from a Managed Node to an Endpoint or PC Managed Node. Between these two objects, there may be a number of intermediary steps including the use of repeater or Endpoint Gateways. However, in this redbook, we focus on the distribution process from a Managed Node to an Endpoint.

5.1.1 Software distribution and MDist repeater

The MDist repeater function plays a vital role in Software Distribution. As we mentioned in the previous chapter, Chapter 4, “Improving Management Framework performance” on page 71, the Endpoint Gateway is automatically configured as a MDist repeater. We will talk about the configuration of a MDist repeater in detail later.

5.1.2 Software distribution and methods

In Software Distribution operations, the most basic and typical operation is to distribute a file package to the subscribers. In this part of the section, we introduce the following information:

- Which methods Software Distribution invokes.
- Where Software Distribution invokes the methods.
- How Software Distribution issues a downcall.
- When Software Distribution distributes the file package to the target.

The following figure (Figure 52) shows the detailed process flow in distributing the file package.

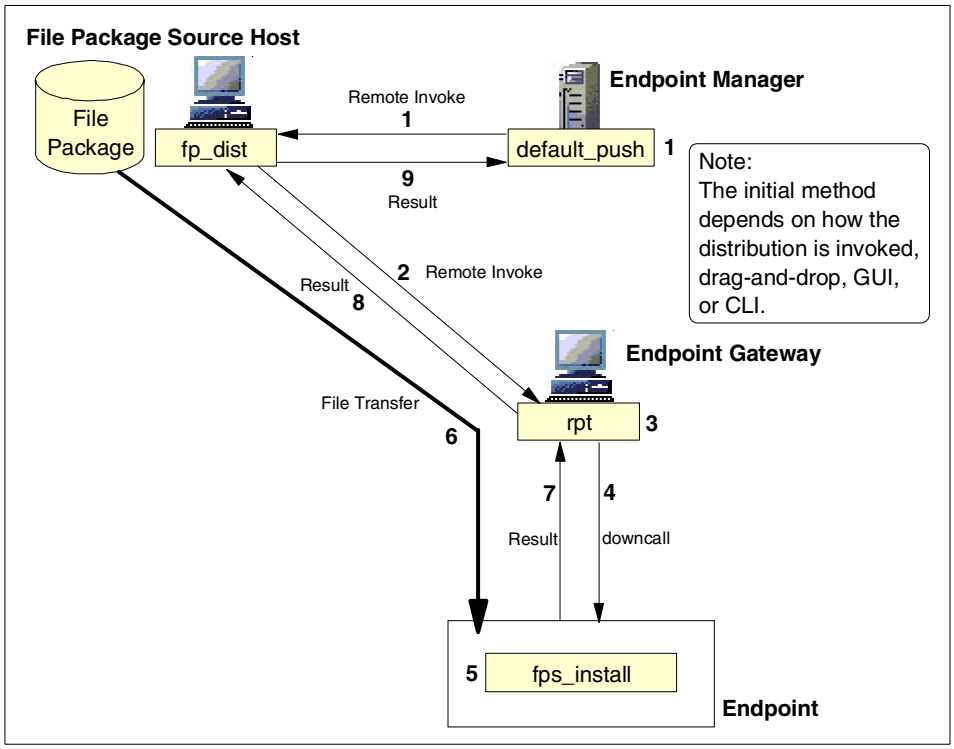


Figure 52. Software distribution and its methods

1. When the profile (file package) is distributed to the subscriber, the TMR server invokes the default_push method locally. The default_push spawns the subsequent method, fp_dist, on the file package source host. (Note that the particular push method invoked depends on how the distribution was started. The default_push is just one of the push methods.)
2. The fp_dist method calls the obj_route method to determine how the file package will be routed to each target. The fp_dist method calls the rpt method on the Endpoint Gateway.
3. The rpt method contacts the Endpoint Gateway database (gwdb.dbd) and checks the Endpoint for the existence of the Endpoint method, fps_install and pushes it to the Endpoint if necessary.
4. The Endpoint Gateway issues a downcall to invoke fps_install on the Endpoint (target).
5. The fps_install method is executed on the Endpoint (target).
6. The data in the file package is transferred to the Endpoint (target).

7. After the completion of the data transfer, the fps_install method on the Endpoint returns the result to the rpt method on the Endpoint Gateway.
8. The Endpoint Gateway returns the result of the rpt method invocation to the file package source host.
9. The file package source host returns the result of the fp_dist method invocation to the TMR Server as well. After receiving the result from the file package source host, the default_push method returns the result, and the file package distribution has been completed.

Note

The initial method depends on how the distribution is invoked, drag-and-drop, Tivoli desktop, or CLI. Actually, there are five push methods as follows.

default_push	This method is invoked by drag-and-drop or by using the profile manager Distribute (top-left pulldown on the profile manager).
fp_push	This method is invoked by the `wdistfp` or by using the profile Distribute window.
push	This method is invoked by the `wdistrib` command.
fp_sched_push	This method is invoked by scheduled filepack distribute.
fp_push_with_size	This method is invoked by the `wdistfp -u` command.

In this example, we assume the following configurations:

- File package source host is configured as a MDist repeater.
- Distribution target is Endpoint.

If your configurations are not the same as our example, the distribution interaction is slightly different.

5.2 Detecting software distribution performance bottlenecks

The performance of Software Distribution is determined by its under lying layers. Therefore, TMR design, network, and operations affect its performance and can also be the performance bottleneck. The following figure (Figure 53) shows the potential performance bottlenecks in Software Distribution operations.

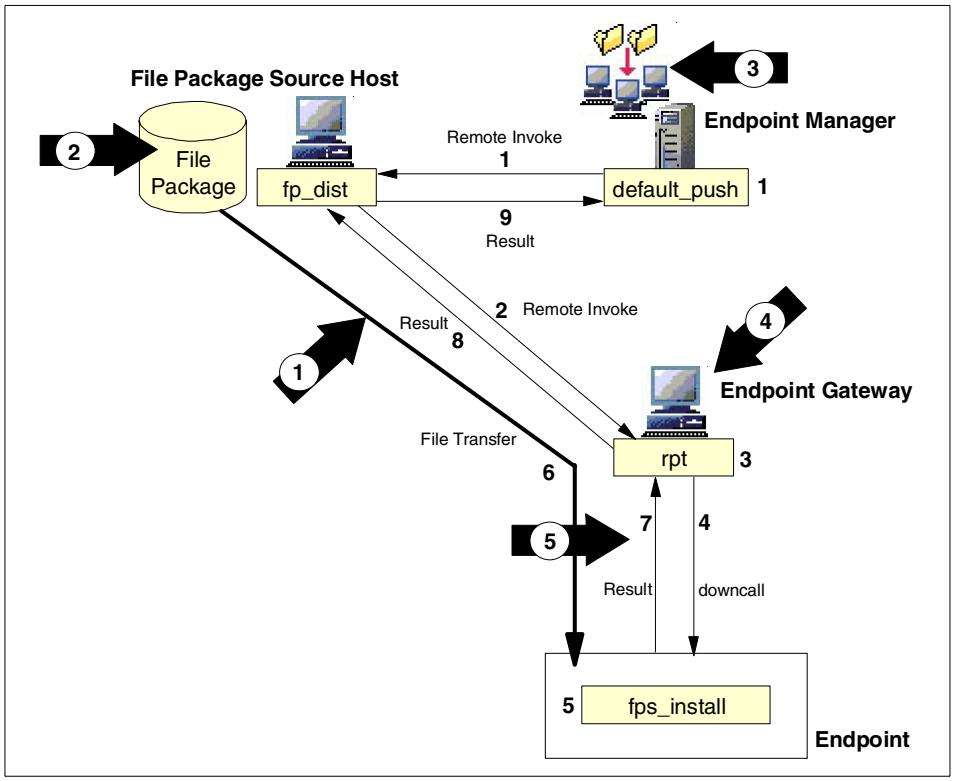


Figure 53. Software distribution performance bottleneck

1. Network Performance
2. File Package Source Host
3. Profile Manager Configuration
4. MDist Repeater
5. Unreachable Target Node

In the example process flow (Figure 53), we indicate typical performance bottlenecks of Software Distribution. These bottlenecks can be problems in most large environments. The following are the descriptions for each potential performance bottleneck in a Software Distribution environment. Later sections in this chapter will detail how to minimize the performance impact of each element.

Network Performance

The function of Software Distribution is to transfer files and data over the network. Therefore, network performance is a very

	important factor to improve transfer performance. If Software Distribution distributes a file package via WAN (Wide Area Network), you should consider not only of the performance of Software Distribution, but also the influence on the other applications that use the WAN. In large environments, it is common for the network between the source host and the Endpoint Gateway to be a low-bandwidth connection. Again, the performance of Software Distribution across this connection and the impact on other applications using the connection must be considered carefully.
File Package Source Host	The location and configuration of the file package source host can be a performance bottleneck especially in large-scale environments. It is usually a good idea for the source host to be a repeater to improve performance. Please refer to 5.3.4, “File package source host configurations” on page 141 for more detailed information.
Profile Manager	To optimize the Software Distribution environment, you should configure the subscribers hierarchy carefully. When you transfer large data, it may be the performance bottleneck. Please refer to 5.3.6, “Profile manager configurations” on page 164 for more detailed information.
MDist Repeater	Tuning the MDist repeater setting is essential to using Software Distribution efficiently. Do not use the default parameter of the MDist repeater in large-scale environments. The hardware performance and resources of the MDist repeater can be a performance bottleneck because, normally, the Endpoint Gateway plays the role of MDist repeater. This also applies to Version 3.6. Please refer to 5.3.5, “MDist repeater configurations” on page 150 for more detailed information.

Unreachable Target Nodes In large-scale management environments, handling the unreachable distribution target is one of the most important issues. Several unreachable targets may affect the whole distribution performance; however, the unreachable target must exist in large-scale environment; so, we need to consider this issue. Please refer to 5.3.9, “Handling unreachable target nodes” on page 174 for more detailed information.

In the following sections, we introduce how we configure and optimize Software Distribution environments.

5.3 Software distribution performance improving strategy

In the previous section, we introduced some possible performance bottlenecks of Software Distribution. This section introduces some approaches and directions to improve the performance and throughput of Software Distribution. We also provide information on how you should configure and optimize Software Distribution in large-scale management environments.

5.3.1 Software distribution performance improving process flow

The following figure (Figure 54) shows the approach for improving the performance and throughput of Software Distribution. When you configure or tune Software Distribution in your management environment, we recommend that you follow this process flow chart.

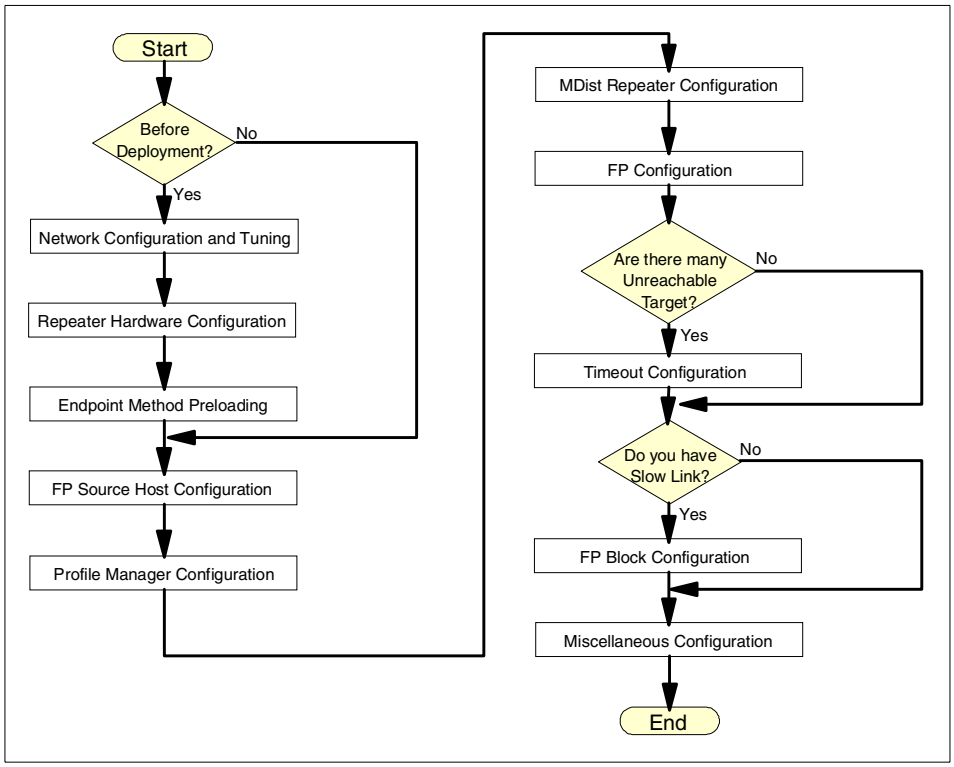


Figure 54. Software distribution performance improvement process flow

In the following sections, we will introduce detailed information about how to configure and tune your Software Distribution in a large-scale management environment.

5.3.2 Network performance tuning

If your environment includes WAN, network tuning can be a very important factor in improving Software Distribution performance. Normally, the following are typical considerations in network performance tuning.

- Network Design
- Router Configurations
- Line Speed

In this redbook, we do not discuss the network tuning in detail, therefore, please refer to the appropriate manuals or documents for more information.

5.3.3 Preloading software distribution methods

Before deploying your management environment, Endpoint method preloading can reduce network traffic at the initial file package distribution. As we mentioned, when you distribute a file package initially, the Endpoint methods that will be used for Software Distribution are also downloaded to each Endpoint. These Endpoint methods are defined in the dependency set, and the total size of the Endpoint methods for Software Distribution is greater than 1 MB.

To avoid downloading these Endpoint methods to each target at distribution time, you can use the Endpoint method preloading technique. In a large-scale deployment Endpoint method preloading can be powerful solution. Please refer to “Endpoint method preloading” on page 109 for more information about Endpoint method preloading.

5.3.4 File package source host configurations

The file package source host plays a vital role in the Software Distribution process, and there are some considerations about its performance. First, the file package source host should be configured on a different Managed Node from the TMR server. It should be configured as a repeater. These are simple changes that can significantly improve performance. The following sections introduce how to allocate the file package source host efficiently.

5.3.4.1 File package source host allocation

By default, the TMR server works as a repeater; so, it is possible to configure the TMR server as a file package source host. However, configuring the TMR server as a file package source host is not recommended because the TMR server performs many processes for managing the entire TMR. Therefore, you have to consider how you can reduce the load of the TMR server.

You should configure other Managed Nodes as the file package source hosts. In a large management environment, creating a exclusive Managed Node machine for a file package source host is common. This machine could be said to be a file server.

5.3.4.2 File package source host is a repeater

We recommend the topology shown in the next figure (Figure 55). In this example, the file package source host is configured as a MDist repeater. The file package source host will distribute the file package to Endpoints that are managed by two different Endpoint Gateways.

In this case the file package source host will distribute the file package directly to both Endpoint Gateways simultaneously. Then the Endpoint

Gateways issue the downcall to distribute the file package to each Endpoint target.

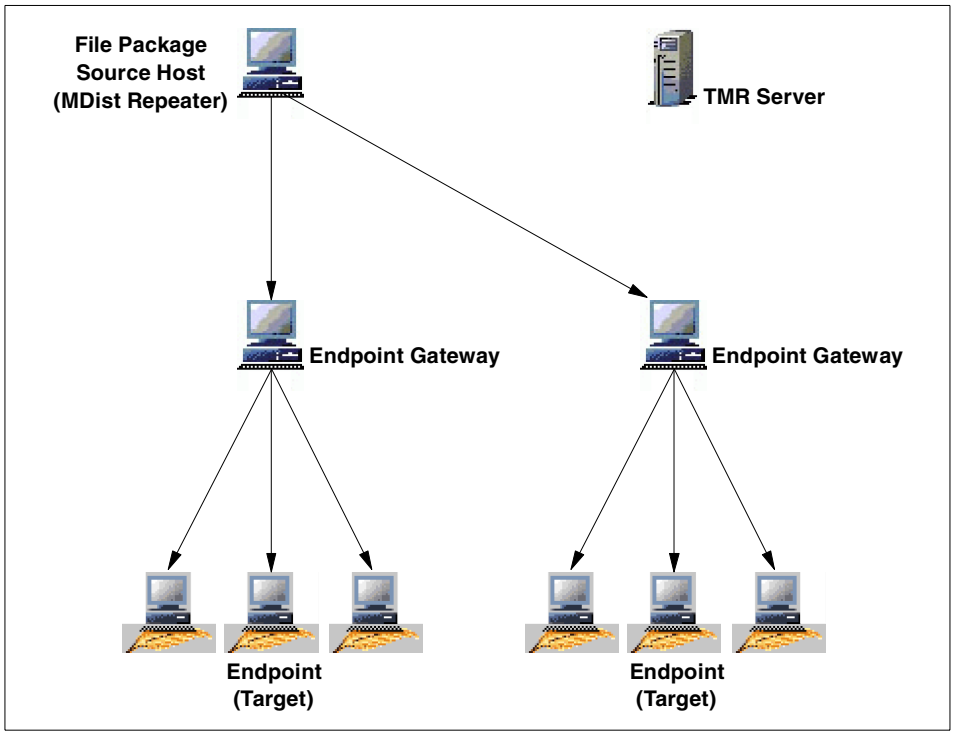


Figure 55. File package source host is a repeater

5.3.4.3 File package source host is non-repeater

In this example (Figure 49), the file package source host is not configured as a MDist repeater. The file package source host will distribute the file package to the Endpoints that are managed by two different Endpoint Gateways.

In this case the file package source host uses the TMR server as a repeater. Then the Endpoint Gateways issue the downcall to distribute the file package to each Endpoint target.

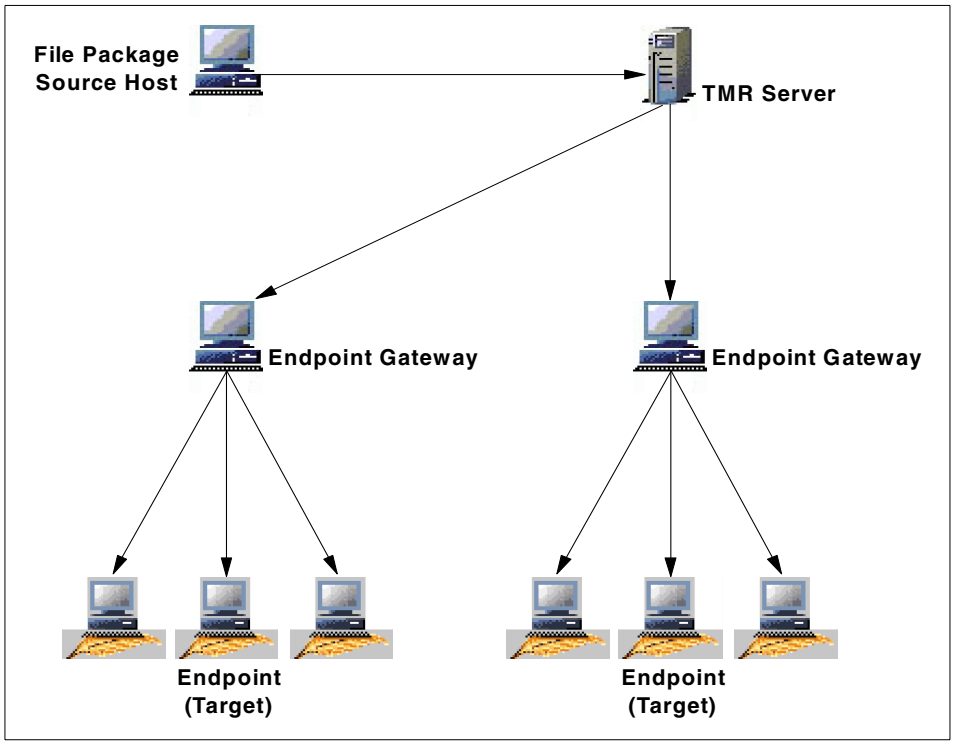


Figure 56. File package source host is non-repeater

To improve the performance of Software Distribution, the file package source host should be configured as a repeater. It is not difficult to configure the repeater using the `wrpt` command. Please refer to *TME 10 Framework 3.6 Planning and Installation Guide*, SC31-8432, for more information about the `wrpt` command.

5.3.4.4 Repeater parameters

In Version 3.6 of the Tivoli, the three-tiered management structure makes the repeater configuration simple and easy because the Endpoint Gateway is configured as a MDist repeater by default. Therefore, normally you do not need to consider repeater hierarchy configuration. To confirm the route that will be used for a distribution from the file package source host to the Endpoint, use the `wrpt` command as follows:

```
# wrpt -q @ManagedNode:trout @Endpoint:ishii
--[RPT:trout [26]]
   |--[RPT:marlin [2]]
   |   |--ishii [113]
   |

#
```

In this sample, the file package source host is `trout`, the Endpoint Gateway is `marlin` and target Endpoint is `ishii`. As you can see, distribution data is sent from the file package source host to the Endpoint Gateway directly, and does not flow through the TMR server.

Wherever you place your file package source host, it is important to tune the MDist repeater parameters. This is covered in the next section.

5.3.4.5 Checking the distribution route in each case

In this section, we introduce some distribution examples showing the difference when the file package source host is configured as a repeater or non-repeater.

When you distribute a file package, the obj_route method is invoked on the repeater manager, and it determines how the file package will be routed to each target (refer to the Figure 52). The following figure (Figure 57) shows the sample distribution environment.

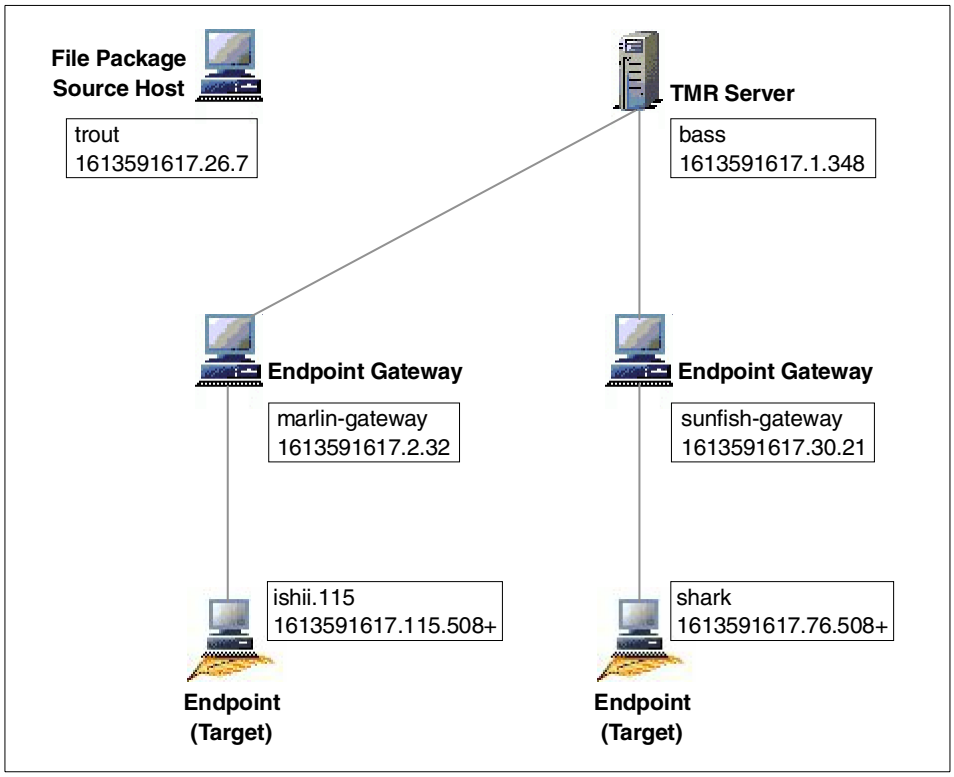


Figure 57. The sample distribution environment

In the above environment, we distribute the file package that is created on the file package source host (trout) to two Endpoint targets (ishii.115 and shark). The following shows the `wtrace` output for each case when the source host is a repeater or not.

Source host is repeater

In this case, we configured the file package source host as a repeater. Therefore, the distribution route will be as follows:

```
# wrpt -q @ManagedNode:trout @Endpoint:ishii.115 @Endpoint:shark
--[RPT:trout [26]]
   |--[RPT:marlin [2]]
   |   |--ishii.115 [115]
   |
   |--[RPT:sunfish [30]]
   |   |--shark [76]
   |

#
```

In the real distribution process that is performed by Software Distribution, the obj_route method determines how the file package will be routed to each target. The following `wtrace` output shows the invocation of the obj_route method and its result.

```
rem-ic 13454    M-H 26-13449      255
      Time run:    [Tue 18-May 10:42:43]

      Object ID:   1613591617.1.366
      Method:      obj_route
      Principal:   root@bass.itsc.austin.ibm.com (0/0)
      Path:        /aix4-r1/TAS/MANAGED_NODE/rptm
      Input Data: (encoded):
             "1613591617.26.7#TMF_ManagedNode::Managed_Node#"
             {
               2
               [
                 "1613591617.115.508+#TMF_Endpoint::Endpoint#" "1613591617.7
                 6.508+#TMF_Endpoint::Endpoint#"
               ]

             }
              "fp_distribute"  "151239580I48859I2000b4b82000b4b8 trout"

rem-oc 13454                      331
      Results: (encoded):
             2
             {
               2
               [

                 {
                   "1613591617.2.32"
                   {
                     1
                     [

                       {
                         "1613591617.115.508+#TMF_Endpoint::Endpoint#"

                         {
                           0
                         }

                       }

                     ]

                   }

                 }

                 {
                   "1613591617.30.21"
                   {
                     1
                     [

                       {
                         "1613591617.76.508+#TMF_Endpoint::Endpoint#"

                         {
                           0
                         }

                       }

                     ]

                   }

                 }

               ]

             }

             {
               10000 50000 50000 1 "NULL" 500 100 180
```

As you can see, the obj_route returns the same route as the route that is retuned by executing the `wrpt -q` command.

Source host is non-repeater

In this case, we do not configured the file package source host as a repeater. Therefore, the distribution route will be as follows:

```
# wrpt -q @ManagedNode:trout @Endpoint:ishii.115 @Endpoint:shark
--[RPT:trout [26]]
   |--[RPT:bass [1]]
   |   |--[RPT:marlin [2]]
   |   |   |--ishii.115 [115]
   |   |
   |   |--[RPT:sunfish [30]]
   |   |   |--shark [76]
   |   |
   |

#
```

The following wtrace output shows the invocation of the obj_route method and its result for the distribution in this case.

```
rem-ic 13350    M-H 26-13345      255
      Time run:    [Tue 18-May 10:36:47]

      Object ID:   1613591617.1.366
      Method:      obj_route
      Principal:   root@bass.itsc.austin.ibm.com (0/0)
      Path:        /aix4-r1/TAS/MANAGED_NODE/rptm
      Input Data: (encoded):
             "1613591617.26.7#TMF_ManagedNode::Managed_Node#"
             {
               2
               [
                 "1613591617.76.508+#TMF_Endpoint::Endpoint#" "1613591617.11
                 5.508+#TMF_Endpoint::Endpoint#"
               ]

             }
              "fp_distribute"  "151239580I48824I2000b4b82000b4b8 trout"

rem-oc 13350                      419
      Results: (encoded):
             1
             {
               1
               [

                 {
                   "1613591617.1.348#TMF_ManagedNode::Managed_Node#"

                   {
                     2
                     [

                       {
                         "1613591617.2.32"
                         {
                           1
                           [

                             {
                               "1613591617.115.508+#TMF_Endpoint::Endpoint#
                               "
                               {
                                 0
                               }

                             }

                           ]

                         }

                       }

                       {
                         "1613591617.30.21"
                         {
                           1
                           [

                             {
                               "1613591617.76.508+#TMF_Endpoint::Endpoint#"

                               {
                                 0
                               }

                             }

                           ]

                         }

                       }
```

If your distribution environment includes a complicated repeater hierarchy, you may need to consider other issues.

5.3.5 MDist repeater configurations

To improve the performance of Software Distribution, the MDist repeater must be tuned and configured properly. This section introduces MDist repeater tuning and configuration information that is useful and effective in most management environments.

5.3.5.1 Repeater range

How does a repeater know what machine it services? Repeaters service the Managed Nodes that are listed in their range. A Managed Node that is not in another repeater's range is served by the repeater with the default flag set. This repeater is usually the TMR server. The initial installation sets the TMR server as the default repeater. If you want a repeater to serve other Managed Nodes, you need to modify the range for that repeater.

In the prior versions of Tivoli, we need to consider not only repeater range but also repeater hierarchy. However, with the three-tiered management structure introduced with the Tivoli Management Agent these considerations are removed. Since a Endpoint Gateway is the repeater for all of Endpoints that have logged into the Endpoint Gateway by default, you do not need to run the `wrpt -n` command. When you create a Endpoint Gateway, the Endpoint Gateway automatically becomes a repeater.

However, the repeater tuning is mandatory to improve the throughput and performance in a large-scale environment. You can use all options for the `wrpt` command for the repeater tuning on the Endpoint Gateway. The following sections introduce each repeater parameter and how to configure the repeater.

Note

As we explained in the Chapter 1, "Introduction" on page 1, this redbook assumes the performance tuning for a large-scale three-tiered management structure (refer to Figure 2 on page 3). Therefore, we do not talk about setting the repeater range for prior TMR structures that consist of Managed Node or PC Managed Node objects. We explain only repeater range settings for the three-tiered management structure case.

5.3.5.2 Memory and disk for spooling data

The MDist repeater provides two parameters that specify values for maximum amounts of memory and disk space system resource that can be allocated to the repeater to spool data during distribution. The following are these two parameters:

- **mem_max** (10 MB by default)
- **disk_max** (50 MB by default)

The data spooling area that is defined by mem_max, and disk_max plays a vital role in the MDist fan out feature. The following figure (Figure 58) shows the interaction between the mem_max and disk_max during the distribution.

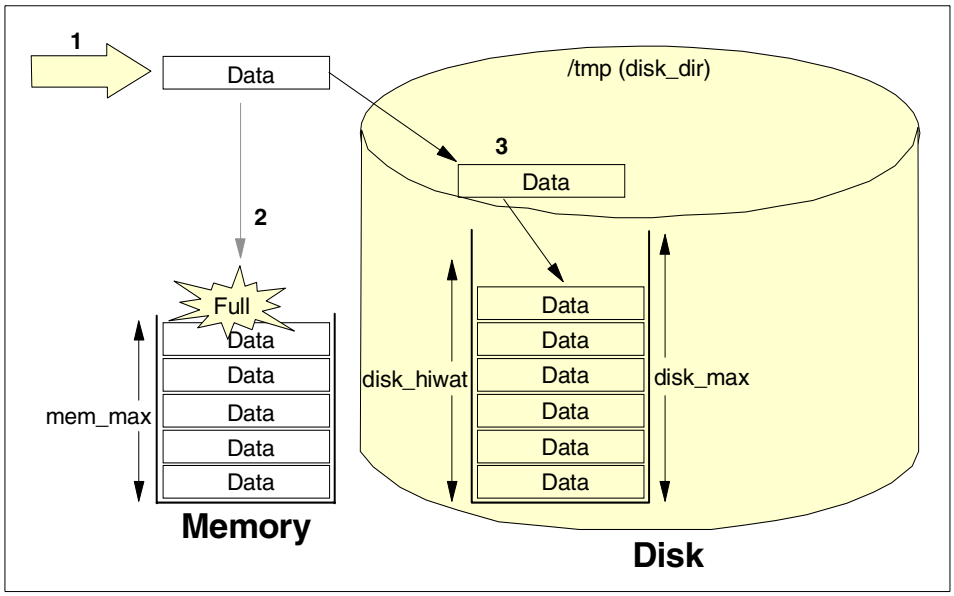


Figure 58. The mem_max and disk_max parameters

1. The file package data is distributed to the MDist repeater.
2. The data is initially spooled to the real memory on the repeater. The values for maximum amounts of the real memory that can be used for spooling data is specified by mem_max.
3. After mem_max is full, the data is spooled to disk on the repeater. The directory or file system to which data spools is specified by disk_dir. The values for maximum amounts of disk space that can be used for spooling data is specified by disk_max. In this case, disk_hiwat indicates the point

where the repeater tells the repeater manager disk_max is about to fill up, and the sender should slow down sending the data.

Note

- If the distribution data is lower than mem_max, the data is not spooled to the disk.
- When disk_max is full as well, the distribution will fail if the number of clients connected to the repeater exceeds max_conn.
- Configure disk_dir with caution. AIX's default installation does not have enough space in /tmp, and Solaris uses /tmp for swap; so, use another location.
- These parameters are global. If you have more than one distribution going through a repeater at once, tuning parameters for that repeater is used independently. If you are doing two distributions through the same repeater, then mem_max is doubled and disk_max is doubled. You must estimate them carefully if multiple distribution is performed. Basically, multiple distribution is not recommendable from a performance point of view.

Normally, memory and disk I/O speed are faster than network transmission speed; so, these two parameters should not cause a performance bottleneck. If your network is very fast, you may find a performance improvement if you make mem_max larger than 10 MB. However, set mem_max with caution because setting mem_max higher than 10 MB can cause the repeater to hang. The mem_max pins specifies areas on the real memory of the repeater machine. This means that other applications running on the repeater machine cannot use these areas on the real memory. Therefore, we strongly recommend that you allocate enough system resources (memory, disk, CPU) to the repeater machine.

From an operation point of view, if these parameters are set incorrectly, the distribution will fail. The following is a summary for the parameters regarding memory and disk on the repeater machine.

- Your repeater configuration must be:

  mem_max + disk_max > largest file package

- The relationship between disk_max and disk_hiwat should be:

  disk_max > disk_hiwat > largest file package

  where, if you set disk_max too low, it may affect the performance; so, be careful.

- The file system that is specified by the disk_max must be:

  filesystem size > disk_max
- Configure the repeater machine with enough system resources.
- The mem_max is doubled and the disk_max is doubled when two distributions are performed concurrently.
- If the distribution file package size is lower than mem_max, the distribution performance will be improved rather than having to use the disk cache (disk_dir).

5.3.5.3 Distribution connections

The MDist repeater provides the parameter that specifies the number of machines distributed to at one time from that repeater. The parameter is:

- **max_conn** (100 connections by default)

Normally, this parameter affects the distribution process between the Endpoint Gateway (repeater) and Endpoint (target). The following figure (Figure 59) shows how the max_conn parameter works in Software Distribution environment.

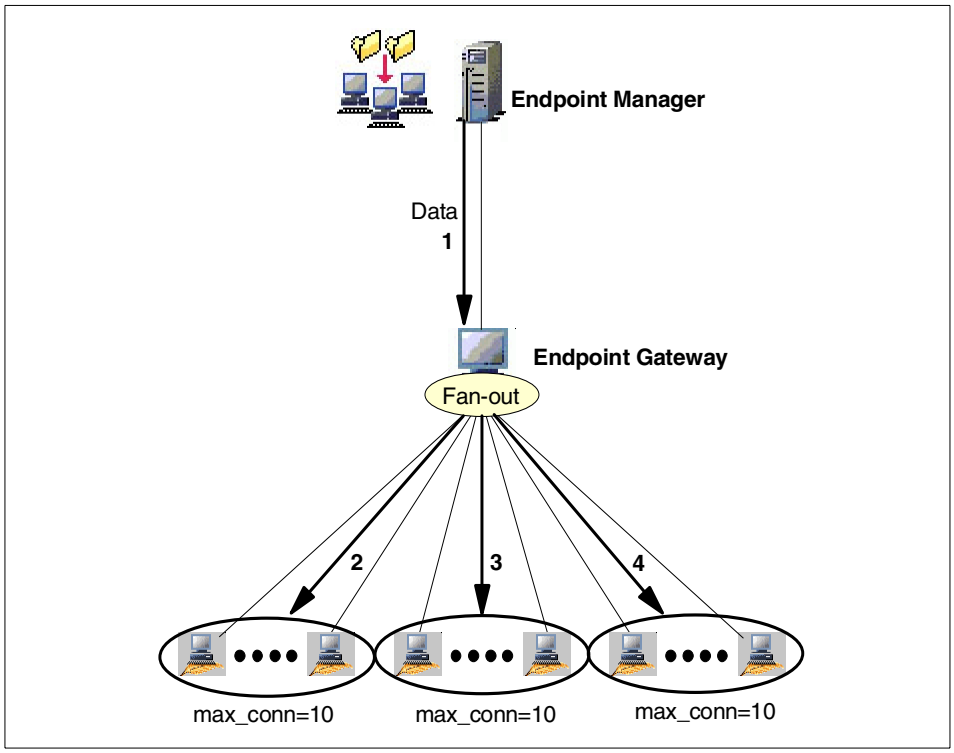


Figure 59. Parallel distribution and max_conn parameter

1. When a file package is distributed to the subscribers, the data is sent to the repeater (Endpoint Gateway) first.
2. Then the repeater fan out the data to the subscribers. In this case, the repeater attempts to open 10 TCP connections at the same time and distributes the data to every 10 machines parallel.
3. If the distribution to the first 10 machines is finished, the repeater attempts to distribute the data to the next 10 machines parallel.
4. Again, the repeater attempts to distribute the data to the next 10 machines. The repeater repeaters this process until the end of the distribution.

Note

- The max_conn parameter for that repeater is used independently. If max_conn set to 10 and two distributions are going through the repeater at once, there will be 20 TCP connections, not 10 connections.
- Since each connection in max_conn uses 0.5 MB to 2 MB per TCP connection, the repeater can run out of memory very quickly and lock up if they are not tuned properly.

Since a Windows NT machine does not seem to handle concurrent connections as well as a UNIX machine does, we recommend a UNIX machine rather than Windows NT for Endpoint Gateway (repeater) in a large environment. When tuning an Windows NT repeater machine, start with five connections (max_conn=5) and work your way up.

The following is a summary for the max_conn parameter setting.

- The default value of max_conn (100) is too much in most environments. Do not use the default value.
- Start with a low max_conn (5 for Windows NT, 10 for UNIX).
- Increase max_conn slowly until you start either having lockups on the repeater or getting high level TCP timeout errors because you are saturating your network.
- In this case, we also recommend you to set the net_load parameter so you are only using 25 percent to 50 percent of your network bandwidth. You can increase this value once your distributions are running reliably. We will introduce net_load in detail later.

Setting the max_conn parameter depends on your environment, for example, network, memory, and CPU of the repeater machine, distribution operations, and so on. To optimize your Software Distribution operations, you need to perform tests for the max_conn.

5.3.5.4 Distribution speed configuration

The MDist repeater provides the parameter that specifies the kilobytes per second used in the distribution. This is the amount of the network bandwidth used. This parameter is:

- **net_load** (500 KB by default)

The net_load parameter is used to confine the repeater to using a certain amount of the network bandwidth per distribution. The following figure (Figure

60) shows how the net_load parameter works in a Software Distribution environment.

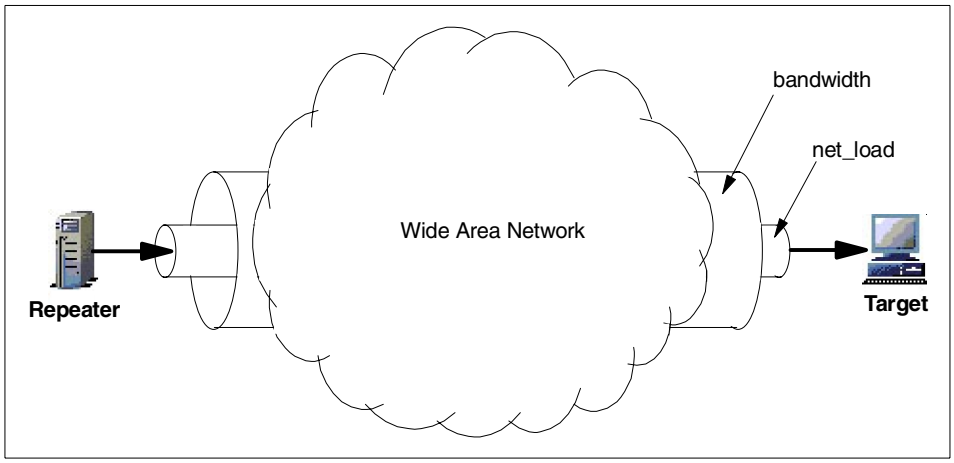


Figure 60. Bandwidth and net_load parameter

To adapt to a variety of network types and optimize each distribution environment, the net_load parameter provides two different options, the positive net_load and negative net_load.

Positive net_load

The positive net_load is to specify the net_load parameter with a positive number. This is the most common configuration. The positive net_load means that each distribution will use a set amount of bandwidth. Each connection will use the amount of the net_load divided by the number of connections. The following figure (Figure 61) shows how the positive net_load works in the distribution.

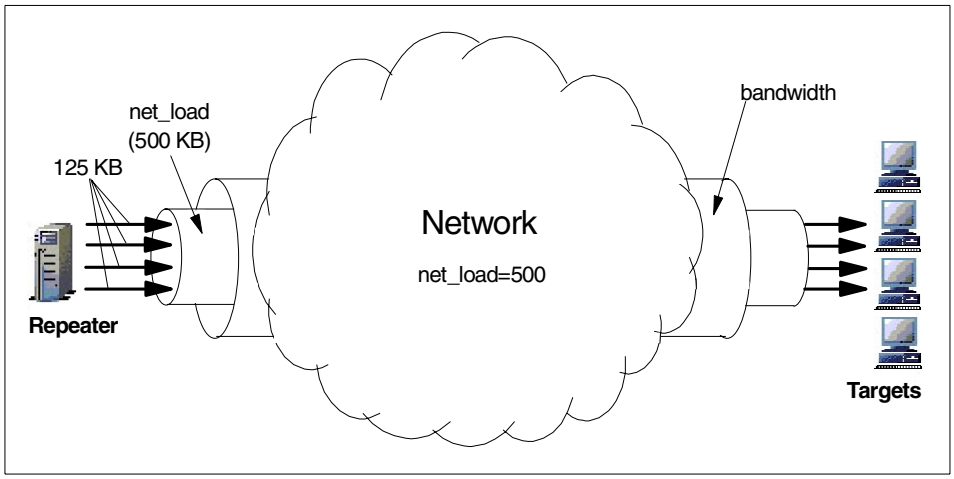


Figure 61. Positive net_load and distribution

In this example, the net_load is set to 500 (KB), and the data is distributed to four distribution targets concurrently (max_conn is set to more than 4). Therefore, each distribution connection can use up to 125 KB bandwidth.

The following table (Table 8) shows the examples of the relationship between the positive net_load and the number of the connections (where the max_conn is set to 10).

Table 8. The relationship between the positive net_load and the connections

net_load	Targets	Speed / Connection	Speed off repeater
500	4	125 KB / Sec	500 KB / Sec
500	5	100 KB / Sec	500 KB / Sec
500	10	50 KB / Sec	500 KB / Sec
500	20	50 KB / Sec	500 KB / Sec

In this example, 10 or 20 targets have the same speed per connection value. That is because the max_conn is set to 10; so, you can distribute to only 10 targets concurrently.

Note

If you have more than one distribution going through a repeater concurrently, the net_load parameter for that repeater is used independently. In other words, the net_load parameter is per distribution. If you perform four distributions to 10 targets each, each distribution will use 500 KB of the network bandwidth. Therefore, you will use 2,000 KB of the bandwidth, not 500 KB.

Negative net_load

The negative net_load is to specify the net_load parameter with a negative number. The negative is really just a flag that tells the repeater that the net_load setting is per connection, not per distribution. Therefore, each connection uses the specified network bandwidth.

The following figure (Figure 62) shows how the negative net_load works in the distribution.

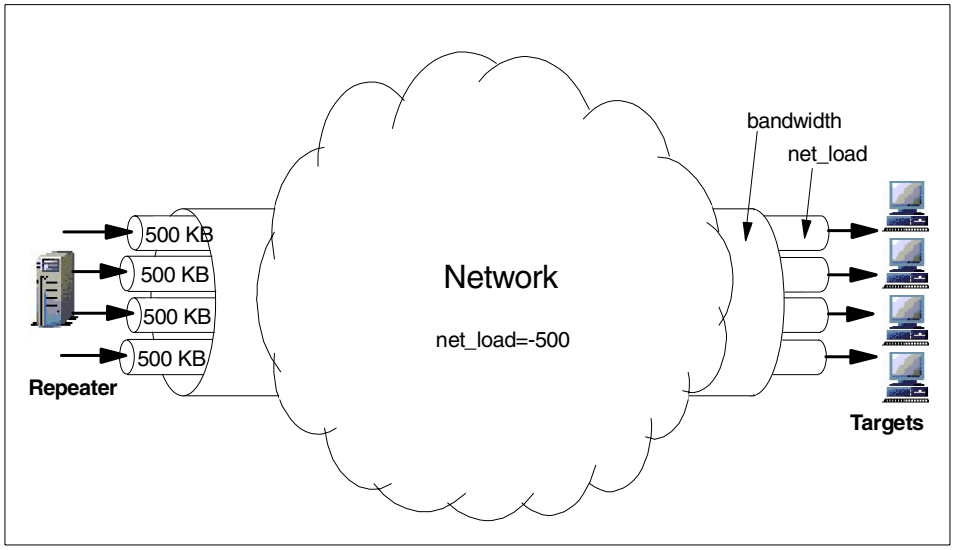


Figure 62. Negative net_load and distribution

In this example, the net_load is set to -500 (KB), and the data is distributed to four distribution targets concurrently (the max_conn is set to more than 4). Therefore, each distribution connection can use up to 500 KB bandwidth, and the total amount of the network bandwidth is 2,000KB in this case.

The following table (Table 9) shows the examples of the relationship between the negative net_load and the number of the connections (where max_conn is set to 10):

Table 9. The relationship between the negative net_load and connections

net_load	Targets	Speed / Connection	Speed off repeater
-500	4	500 KB / Sec	2,000 KB / Sec
-500	5	500 KB / Sec	2,500 KB / Sec
-500	10	500 KB / Sec	5,000 KB / Sec
-500	20	500 KB / Sec	5,000 KB / Sec

In this example, distributing to 10 or 20 targets uses the same amount of the network bandwidth because max_conn is set to 10.

5.3.5.5 Slow link configuration

Normally, a repeater sends its data in 16 KB blocks. In some environments with slow wide area network (WAN) links, 16 KB blocks are too large. In other words, the WAN link is slow enough that the net_load parameter needs to by set below 1 KB / Sec. This is accomplished by setting the SLOW_LINK to true in the oserv environment. The following figure (Figure 63) shows how the SLOW_LINK environment variable works.

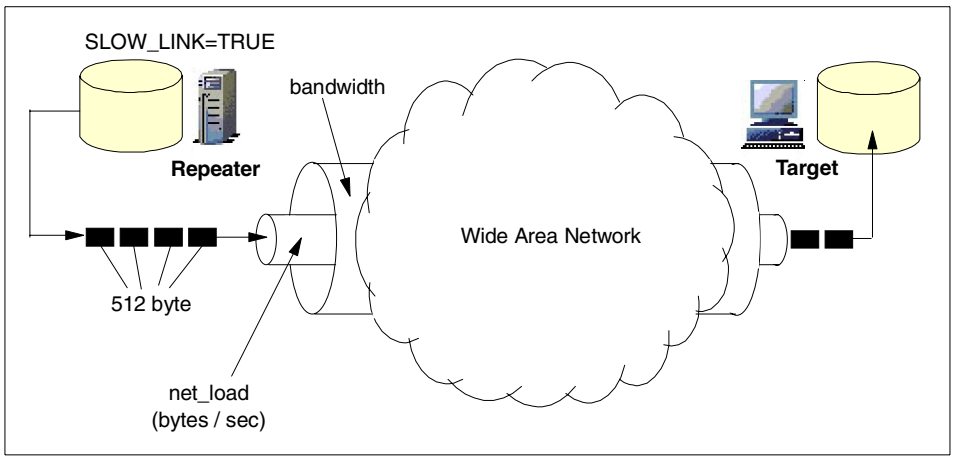


Figure 63. The SLOW_LINK environment variable and distribution

The SLOW_LINK setting is per Managed Node; so, you will need to set it for each Managed Node that needs it. If you add SLOW_LINK=TRUE, the following things happen for the Managed Node that it was set on:

- The net_load parameter is in bytes / second instead of kilobytes / second.
- Data transmission is performed in 512 byte packets instead of 16 kilobyte packets.

You can set the SLOW_LINK environment as follows:

1. Get the current environment.

   ```
   odadmin environ get > /tmp/list
   ```

2. Edit the environment list and add the following environment variable.

   ```
   SLOW_LINK=TRUE
   ```

3. Enable the environment list.

   ```
   odadmin environ set < /tmp/list
   ```

You need to perform this procedure on each Managed Node (repeater) that you want to set the SLOW_LINK.

5.3.5.6 How to optimize distribution in a WAN environment

Most wide area network (WAN) environments include a slow link. Even if you configure your backbone network by using fast digital networks, these are slower than LAN networks. The customization for the WAN environment is one of the most important repeater configurations.

In a WAN environment, the following repeater parameters may affect the performance:

- net_load
- max_conn
- SLOW_LINK

max_conn

As we mentioned, we recommend you to start with a low max_conn (5 for Windows NT, 10 for UNIX). Do not use the default value (100) especially in a WAN environment.

Note

There is no explicit limit to the number of active distributions able to be processed through a TCP/IP and MDist repeater in a UNIX operating system. This will be dictated by the available power of the machine being used as a repeater (maximum number of processes and file descriptors for each process, network buffers, memory, paging space) and the surrounding network connections.

On the other hand, inbound maximum connections for a Windows NT is 10 connections. Outbound is subject to the same constraints as UNIX.

net_load

The net_load configuration is a little complicated because it depends on the network bandwidth, network type, distribution style and so on. In general, there are two types of the networks as follow:.

- Dedicated (Switched) Type Network
- Shared Type Network

The dedicated or switched networks have a dedicated bandwidth to each connection or port on your network. In other words, the specified bandwidth is guaranteed for each connection. For example, ISDN, certain frame-relay links and switching LAN are of this type. An ISDN connection has a dedicated 64 Kb or 128 Kb connection to each node, and switching Ethernet LAN has a 10 Mb connection to each port on the network.

On the other hand, a shared network has a set amount of bandwidth that is shared between all on the network. For example, normal LAN mediums, such as standard Ethernet and Token-Ring, are of this type. The MDist repeater can handle both types of networks by specifying the proper net_load value.

We recommend that you configure net_load with a positive value, especially for a WAN that includes slow links. Tivoli recommends that you set the net_load parameter to about 25 percent of the network bandwidth; however, it depends on your environment, for example, other applications that are using the network, the file package size, and so on.

The negative net_load works efficiently for environments that have a dedicated connection to each node. The negative net_load should be configured only for the following cases:

- The network between the repeater and targets has a switched type network, for example, switching LAN medium.

- The network between the repeater and targets is a dedicated network, and no application uses this network other than Tivoli.

SLOW_LINK

Initially, we recommend that you disable the SLOW_LINK environment (by default the SLOW_LINK is disabled). In this environment, if you have a WAN link that is causing you problems, for example, high level TCP/IP timeout or strange errors, try turning on SLOW_LINK. If you set the SLOW_LINK, be sure to adjust the net_load value. Since the net_load is now bytes per second instead of kilobytes per seconds, the net_load value needs to be much higher.

Normally, the SLOW_LINK environment is used in the following cases:

- A network error (TCP/IP timeout, re-transmit, and so on) occurs frequently during the distribution.
- The WAN is shared among many applications other than Tivoli.

Basically, SLOW_LINK may improve the throughput, but it will not improve the performance. So, try the default (disable SLOW_LINK) first, then enable the SLOW_LINK if you have some problems associated with transmission speed in your distribution.

Note

If you configure the repeater in a UNIX system, you can check the interface condition by using the `netstat` command. For example, in the AIX operating system, the following commands will give you useful information:

- `netstat -v`
- `netstat -m`
- `netstat -p tcp`

Please refer to Chapter 3, “Improving operating systems and network performance” on page 39 for more information.

Example

The following figure (Figure 64) shows a typical three-tiered management structure environment. In this case, a distribution will be performed through at least two different repeaters. One is located in the source host site, and another is located in the targets site. The repeater configuration considerations are a little different between the source host site and targets site.

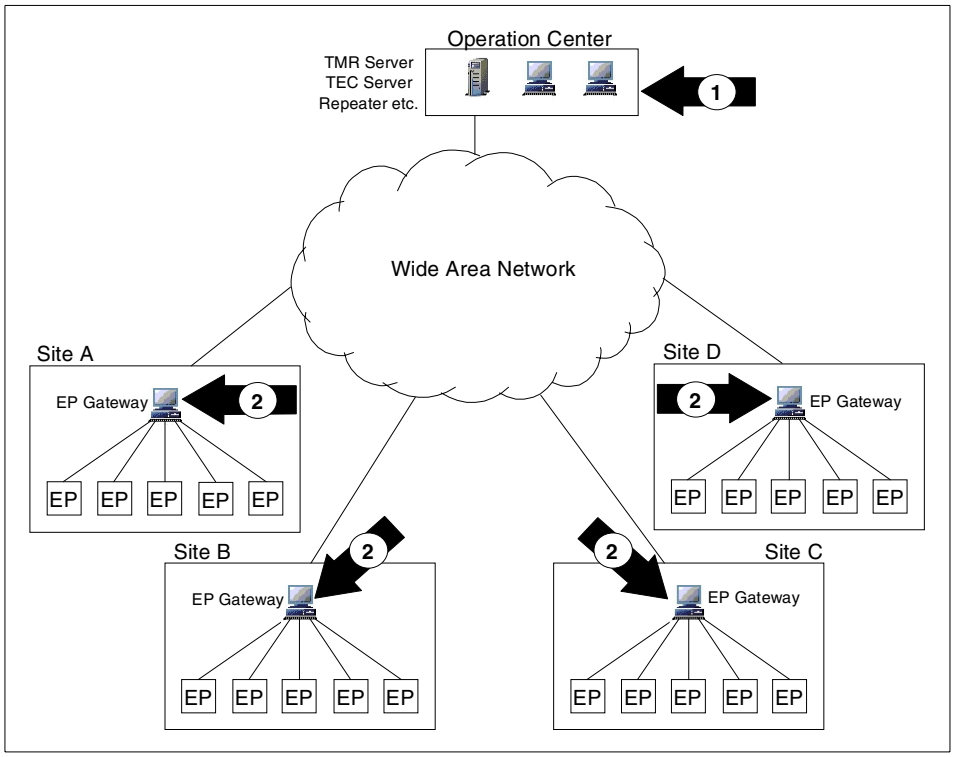


Figure 64. Wide Area Network environment

1. **Source Host Site Repeater:** max_conn, net_load, SLOW_LINK
2. **Targets Site Repeater:** max_conn, net_load

The configurations summaries for each case are as follows:

Source Host Site First of all, you need to set max_conn to the appropriate value. It is very important. If you do not modify this parameter, the repeater may hang. Then you decide the net_load. As we mentioned, it depends on your environment including your applications other than Tivoli. If you do not have any guideline, start with 25 percent of the connection bandwidth and test many cases that use a variety of the net_load value and the same file package. In this case, we recommend the positive net_load. If you have some TCP/IP timeouts or re-transmissions, then try enabling the SLOW_LINK. As we mentioned, in this case, the stat_inv and global timeout (`wrpt -T` command) are not available because

the targets are only Endpoints. The parameters related to the buffer (mem_max, disk_max, disk_hiwat) should be configured properly. Please refer to 5.3.5, “MDist repeater configurations” on page 150 for more detailed information. These parameters will not affect the performance much; so, tune for reliability first and performance second. Try to use the default mem_max first for a source host site repeater, then increase it slowly if your repeater has enough memory.

Targets Site First of all, you need to set the appropriate value to the max_conn as we described in 5.3.5, “MDist repeater configurations” on page 150. Then you decide on the net_load. As we mentioned, it depends on your environment including your applications other than Tivoli. If your LAN is a standard Ethernet (10 Mb), start with the default value (500 KB) and test as many cases that use a variety of the net_load value and the same file package. If your LAN is switching LAN, the negative net_load may improve the throughput. In this case, you do not need to consider the SLOW_LINK because the networks between the Endpoint Gateway and Endpoints are LAN. Other parameter configurations, such as mem_max or disk_max, are the same as the source host site repeater.

Note

These examples assume that you do not perform multiple distributions. Again, multiple distributions are not recommended in large-scale environments. This requires re-estimating all tuning parameters and may cause hangs in the worst case.

5.3.6 Profile manager configurations

This part of the section introduces how we should configure the subscribers for distributing the Software Distribution file package in a large environment. It may be not big issue in the small environment; however, it could be a problem when you distribute a large amount of data in the large environment.

Normally, the subscribers are configured based on the following factors:

- Platform Type (UNIX, Windows NT, Windows 95, and so on)
- Business Location (Building, Branch Office, and so on)

- Business Unit (Department, Division, and so on)
- Data Type
- User Application

The three-tiered management structure simplified the MDist repeater hierarchy. However, you should consider the relationship between the Endpoint Gateway and Endpoint (for example, the number of Endpoints per an Endpoint Gateway, network configuration or Endpoint Gateway hardware configuration) other than the above factors when you configure the profile manager. We introduce two different examples of profile manager configuration, one of such we recommend and the other we do not recommend.

One is a recommendable example, and the other is not recommendable example.

5.3.6.1 Recommended example

To optimize the MDist repeater's fan out feature, we recommend you to configure the subscribers as follows (refer to the Figure 65).

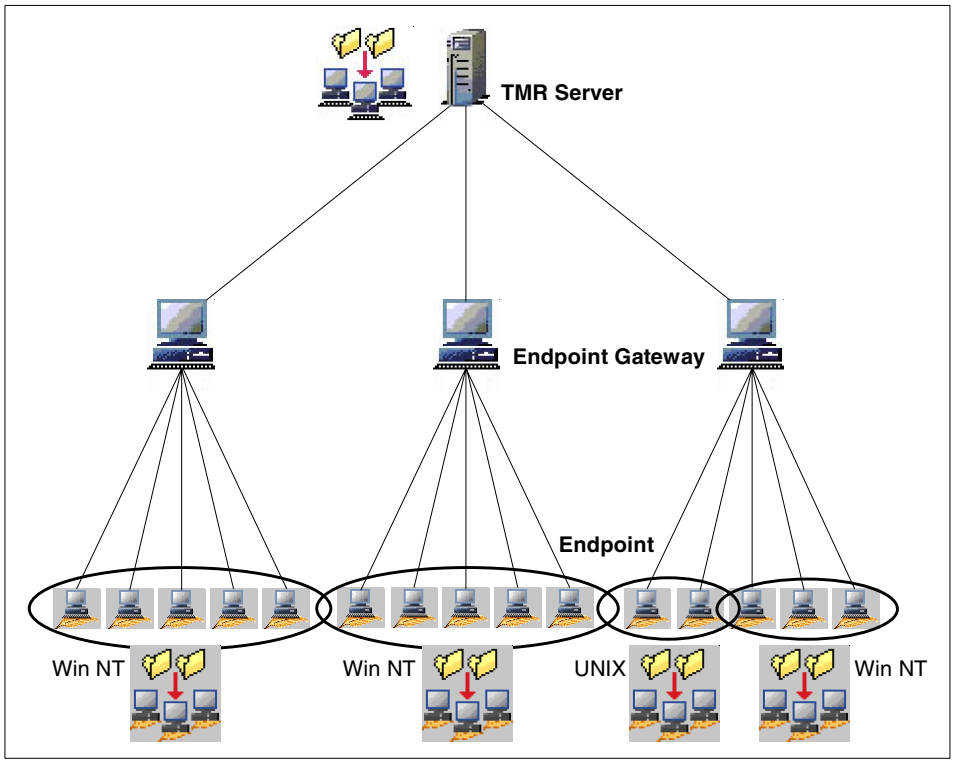


Figure 65. Profile Manager Subscriber configuration

In this example, the circle means one profile manager. As you can see, all Endpoints that are the same platform type and have logged into the same Endpoint Gateway are configured as the same subscriber group in the same profile manager. This example is our recommended configuration.

5.3.6.2 Unrecommended example

On the other hand, the next figure (Figure 66) shows the example of the profile manager configuration that we do not recommend.

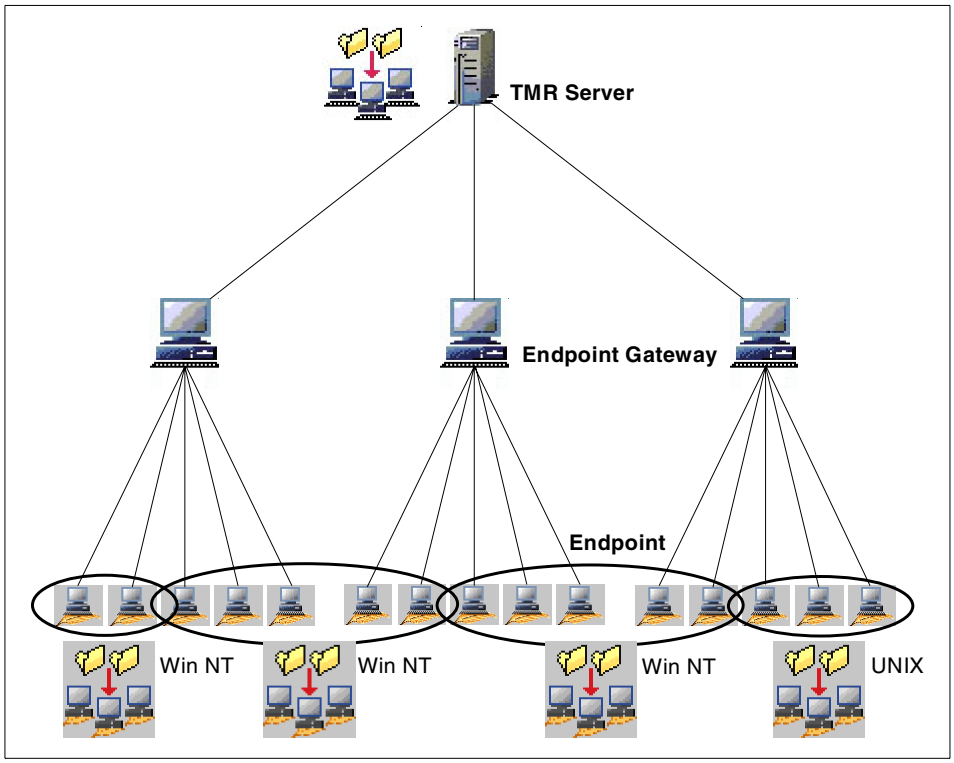


Figure 66. Profile Manager Subscriber configuration

As you can see, Endpoints that have logged into the different Endpoint Gateways are configured as the same subscriber group in the same profile manager. If you configure the profile manager like this sample, what happens? The following example (Figure 67) shows why we do not recommend this profile manager configuration.

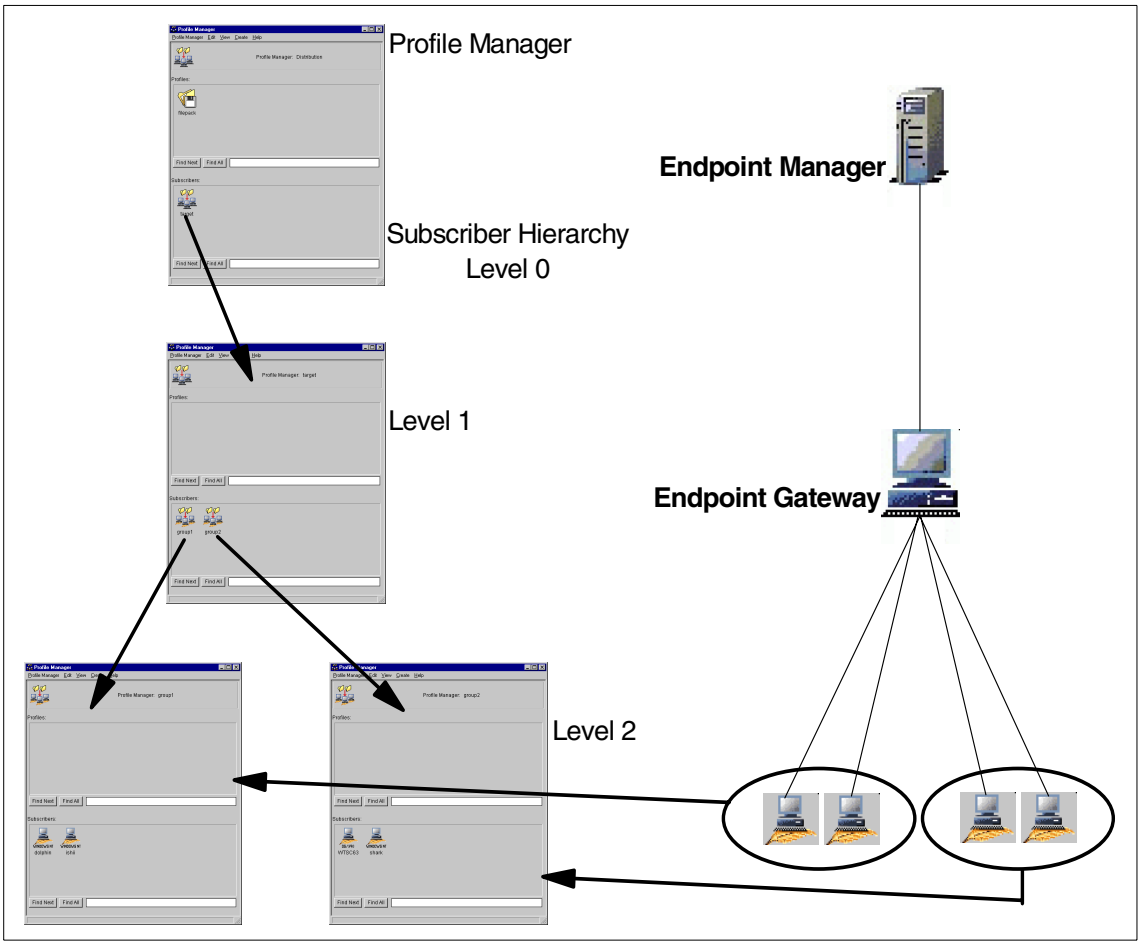


Figure 67. Profile Manager Subscribers and Endpoints

In this example, a single Endpoint Gateway is managing four different Endpoints. As you can see, two Endpoints are configured as the same subscriber group, and the other two Endpoints are configured as another subscriber group. The next figure (Figure 68) shows how the file package is distributed to the subscribers in this case.

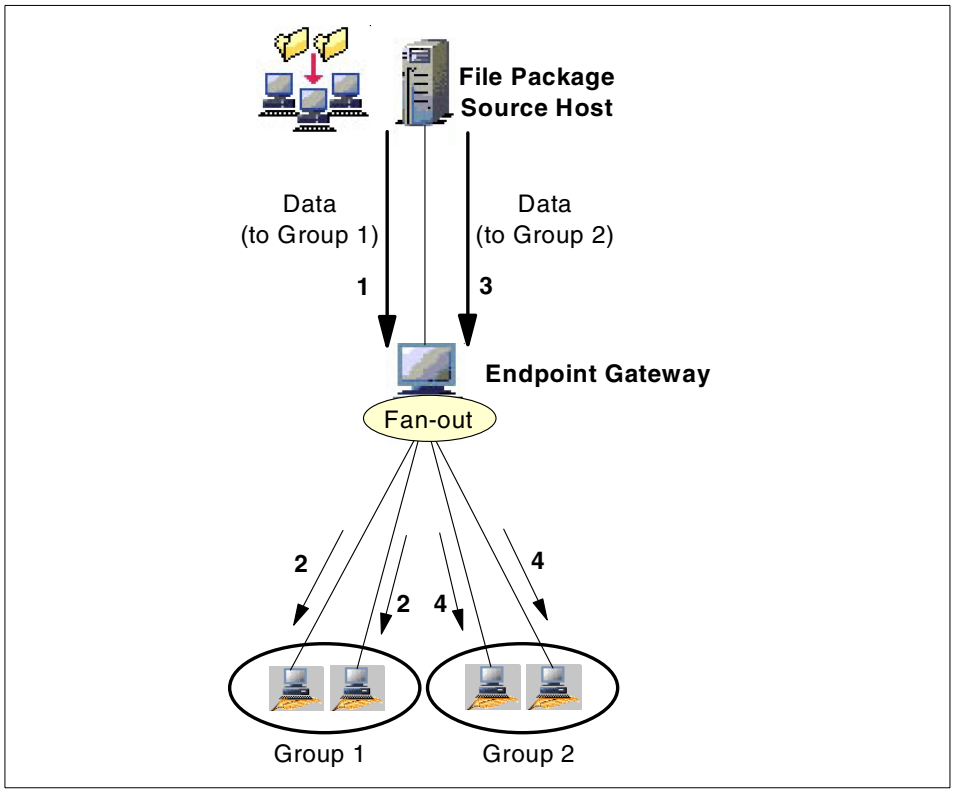


Figure 68. Subscribers and fan-out feature

1. The data is sent from the file package source host to the Endpoint Gateway for the first time.
2. The MDist repeater (Endpoint Gateway) fans out the data to the targets that are configured in the same profile manager (Group 1).
3. The same data is sent from the file package source host to the Endpoint Gateway again.
4. The repeater (Endpoint Gateway) fans out the data to the targets that are configured in another profile manager (Group 2).

In this case, the same data is sent from the file package source host to the Endpoint Gateway twice even if the repeater uses the fan-out function because the Endpoints that have logged into the same Endpoint Gateway are defined in the two different profile managers. If the file package includes large amounts of data, this data transfers may affect the performance in the network or repeater.

Therefore, to improve distribution throughput, the most important things here are:

- The Endpoints that have logged into the same Endpoint Gateway should be defined in the same profile manager.
- Do not configure Endpoints that have logged into the different Endpoint Gateways in the same subscriber group.

5.3.7 File package configurations

Software Distribution allows you to specify the options for each file package. The following figure (Figure 69) shows the File Package properties screen. To specify the file package options, you not only can use GUI, but also the `wsetfpopts` command or import its definition.

This part of the section introduces some considerations for specifying the file package options.

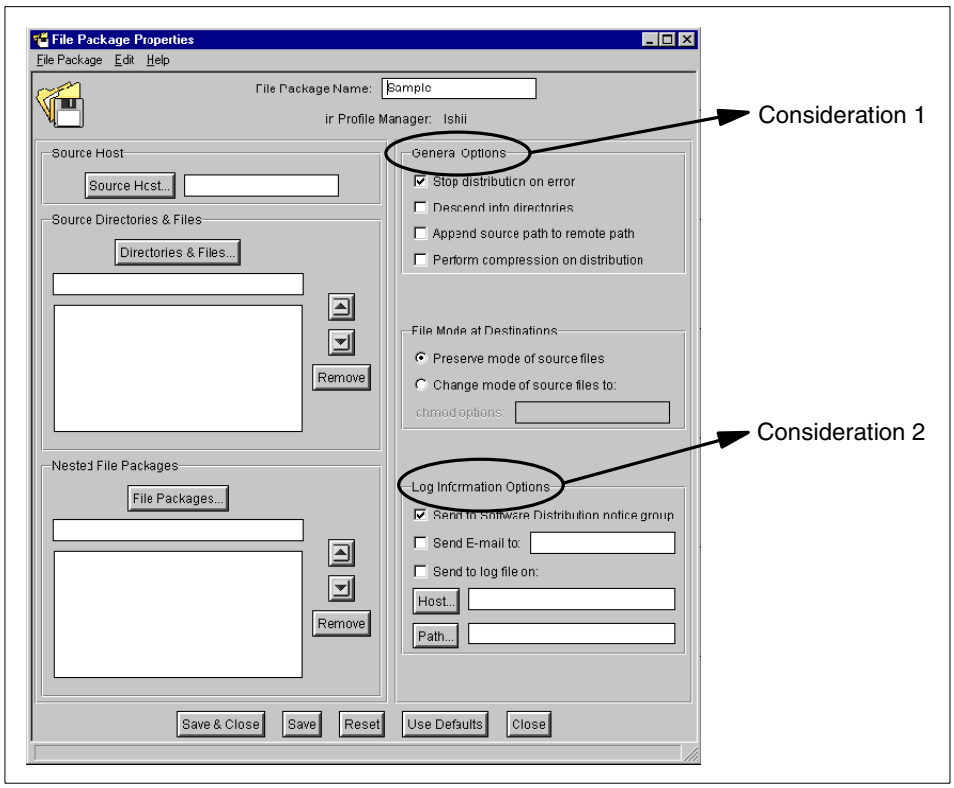


Figure 69. File package properties

5.3.7.1 Consideration 1: Compression option

Software Distribution provides a compression option. To specify this compression option, you need to check the **Perform compression on distribution** field (refer to the Consideration 1 in Figure 69). Generally, we recommend you to specify this option. However, the worst case is that the compression option makes distribution performance worse because the repeater machine does not have enough resources, for example, CPU. Therefore, we recommend the compression feature on the assumption that the repeater machine has enough resources especially if a large amount of data is distributed with the compression option.

5.3.7.2 Consideration 2: Log information option

Basically, Software Distribution's log option does not affect the performance greatly. The following figure (Figure 70) shows how the Software Distribution writes the distribution log information to the log file.

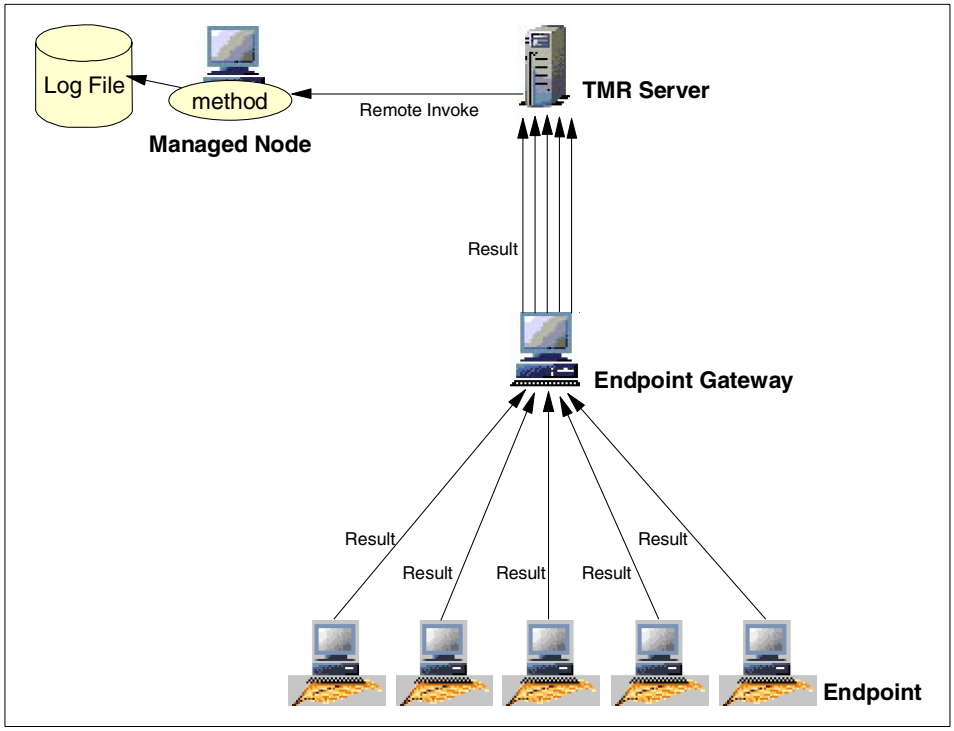


Figure 70. Distribution log information

As you can see, after the distribution completion, the TMR server invokes the appropriate methods to perform the specified actions, for example, logging to file or sending a Tivoli notice.

5.3.7.3 Consideration 3: Configuration programs

Software Distribution allows you to specify the configuration programs that run on the target before or after distribution. The following figure (Figure 71) shows the screen to set the file package options for each platform.

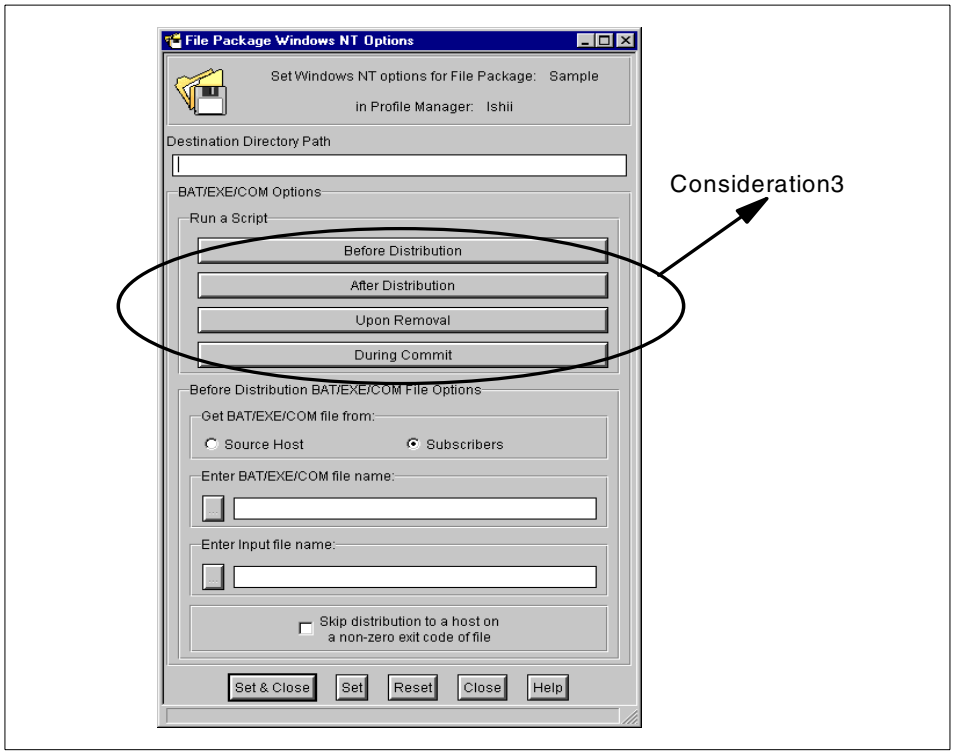


Figure 71. File package options for each platform

Usually, the configuration programs are used for setting environment variables to user applications, for example, executing the `setup.exe` program. These processes are mandatory for your distribution operations or management operations. However, a heavy configuration program must affect the throughput of the whole distribution operations in a large-scale management environment. Therefore, the simple and light scripts are recommendable for the configuration programs.

To ensure the throughput is not degraded when you specify the configuration programs, we strongly recommend you to define the progs_timeout parameter.

Please refer to the “Understanding timeout for each layer” on page 175 for more detailed information about the timeouts.

5.3.8 Software distribution operations

When you distribute the file package to the subscribers, you can specify the distribution options. The following figure (Figure 72) shows the screen for the distribution options.

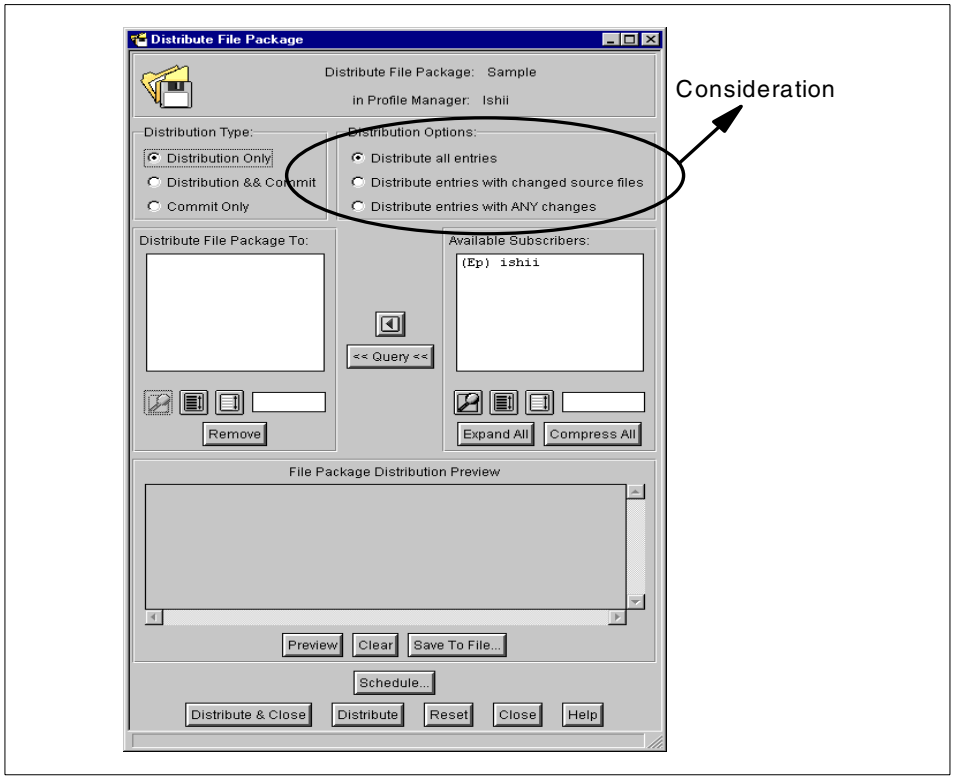


Figure 72. File package distribution

5.3.8.1 Consideration: Distribution option

Software Distribution provides the following three options for the distribution operation:

- Distribute all entries

- Distribute entries with changed source files
- Distribute entries with ANY changes

We strongly recommend that you use the Distribute all entries option in a large-scale management environment. The other two options are very convenient features; however, the load of the TMR server and file package source host is heavy, and it will take a long time if the file package size is large.

5.3.9 Handling unreachable target nodes

In a large-scale distribution, some distribution targets will be unreachable when the distribution occurs. These unreachable targets affect the throughput of the whole distribution process.

5.3.9.1 How the unreachable targets affect the distribution

Usually, when you perform a large-scale distribution, at least several distribution targets will be unreachable or unavailable. In this case, these unreachable targets affect the distribution process as follows (Figure 73).

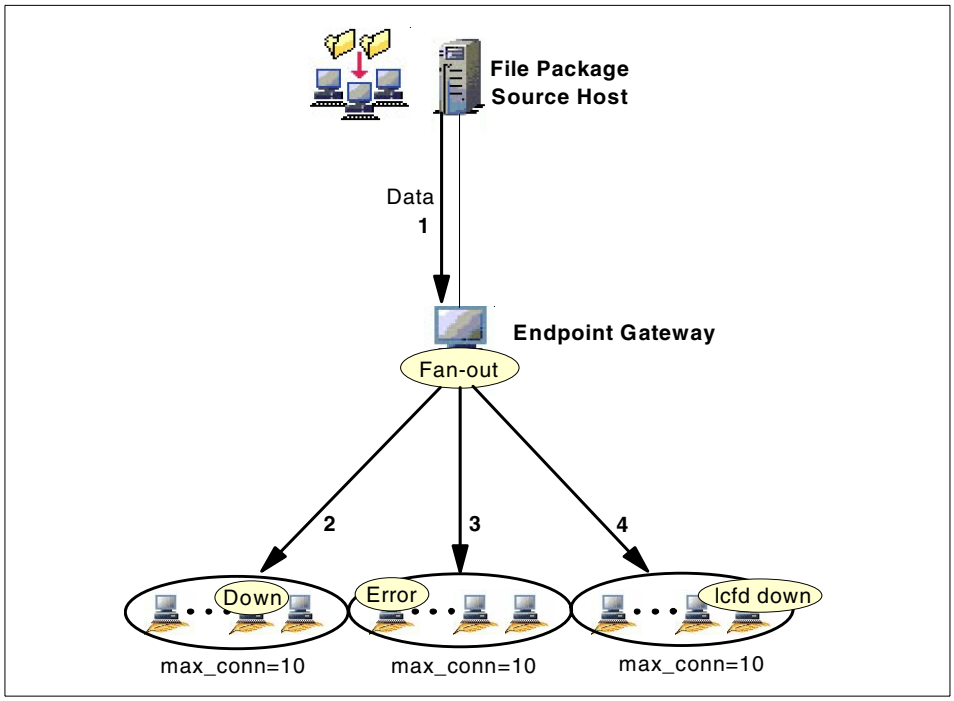


Figure 73. Distribution and unreachable target nodes

1. When a file package is distributed to the subscribers, the data is sent to the repeater (Endpoint Gateway) first.
2. Then the repeater fans out the data to the subscribers. In this case, the repeater attempts to distribute the data to 10 target nodes concurrently because max_conn is set to 10. However, one of these targets is unreachable. Then, the repeater waits for the timeout (tcp_keepinit) of the unreachable target even if the other distributions are completed. After the timeout, the repeater gives up the distribution and tries to distribute the data to the next 10 targets concurrently.
3. The repeater attempts to distribute the data to the next 10 targets. In this case, all 10 targets are reachable. However, the configuration script (after the distribution script) has been taking a long time on one of these targets. Then, the repeater waits for the timeout (progs_timeout) of that target even if other the other distributions are completed. After the timeout, the repeater gives up the distribution and tries to distribute the data to the next 10 targets concurrently.
4. Again, the repeater attempts to distribute the data to the next 10 targets. In this case, on the one of these targets, the lcfd daemon does not run. Then the repeater waits for the timeout (connection_timeout) of that target even if the other distributions are completed. However, connection_timeout is very short so that normally the lcfd daemon down does not affect the distribution throughput seriously.

5.3.9.2 Understanding timeout for each layer

As shown above, there are many timeouts in the distribution process. In this part of the section, we classify these timeouts and introduce how they work.

In a Software Distribution environment, the timeouts that are used for the distribution process depend on the type of the distribution target. Basically, there are two different patterns. One is the distribution to the Endpoint targets, and another is the distribution to the Managed Nodes or PC Managed Nodes. The following figure (Figure 74) shows the timeouts for each distribution target.

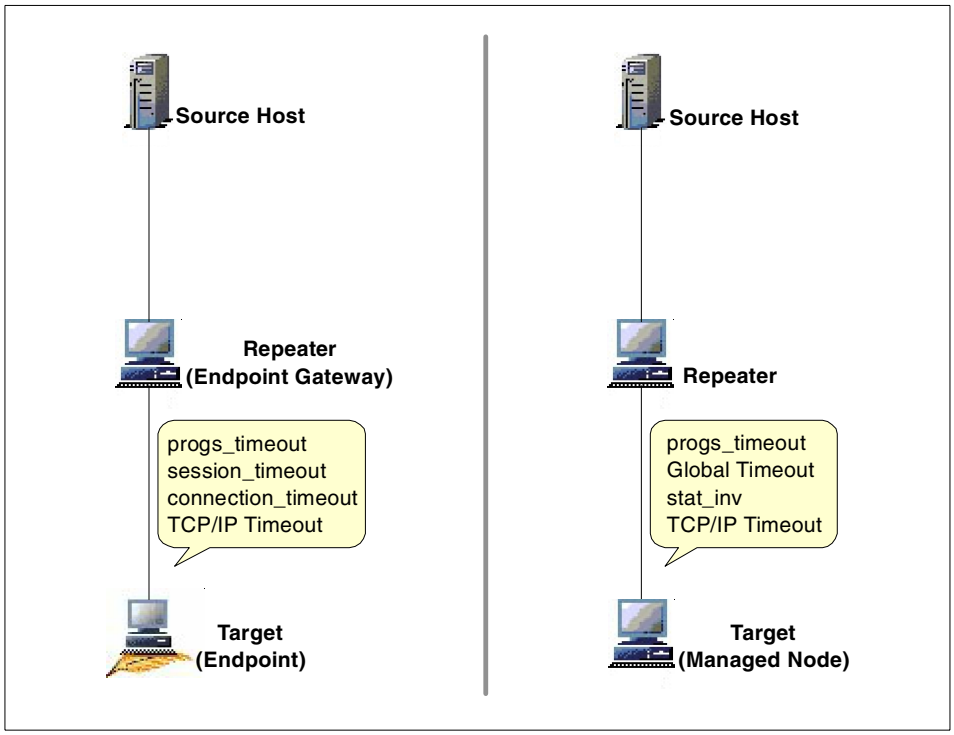


Figure 74. The timeouts value for each distribution target

As you can see, when a file package is distributed to the Endpoint target, the timeouts that are used for the distribution are different from the Managed Node target (or PC Managed Node). The next figure (Figure 75) shows detailed information about the timeouts for Endpoint targets and how these timeouts work for each layer.

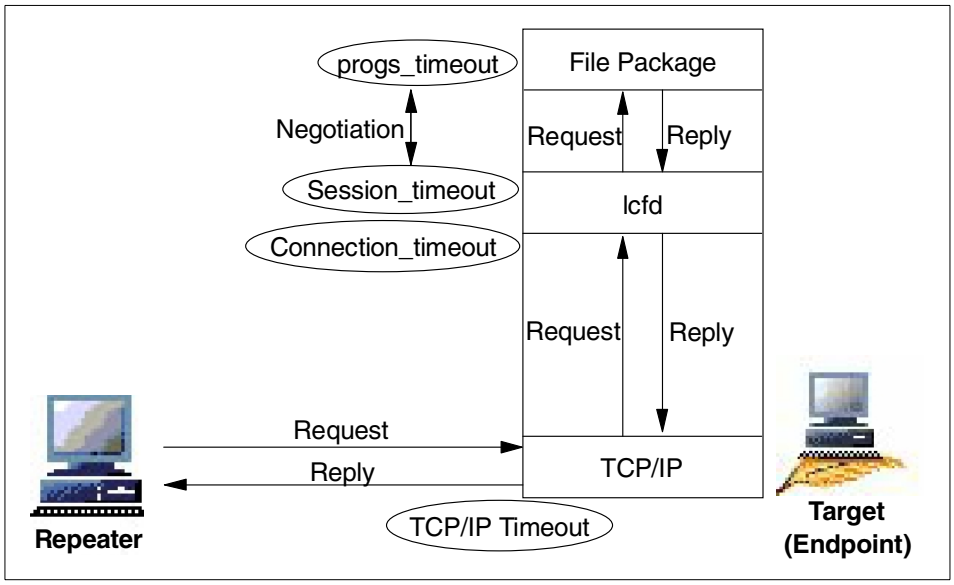


Figure 75. The timeouts for each layer

Basically, in the timeout values related to the network, there are two different types:

- Timeout waiting to establish a connection
- Timeout waiting to disconnect a connection (after establishing a connection)

Both of these types of timeouts are present in the Software Distribution environment. For example, the timeout for the situation that the target Endpoint is unreachable when you attempt to distribute a file package is different from the timeout for the situation that the target Endpoint is powered down (shutdown) in the middle of the distribution.

As you can see, the lower layer's timeout values affect the upper layer's timeout. Therefore, the following timeout setting is required:

Lower Layer's Timeout Value > Upper Layer's Timeout Value

Each timeout plays the following roles:

progs_timeout This is used by Software Distribution by the target machines of all types to time configuration programs (Before, After, Remove, and Commit scripts). You have probably experienced a problem

in a large-scale environment with hanging distributions; wherein, the failure of a single Endpoint keeps the overall distribution from proceeding. Even if the configuration (BARC) script only hangs on one of the targets, the overall distribution process never returns. To avoid this situation, the progs_timeout is provided. The progs_timeout sets a client-level timeout value for all configuration programs. The intention of this timeout is to set a time value after which a hanging BARC script that is defined in the file package is killed on the target system. The progs_timeout value does not include the distribution time. Before the introduction of this timeout, if the BARC script ran into a loop for some reason, the overall file package distribution process never returned and just waited for the timeout of the lower layer.

session_timeout This specifies the timeout value for the connection between the Endpoint Gateway and its Endpoints. This timeout provides the same functionality for connections between a Endpoint Gateway and its Endpoints as the stat_inv timeout does for connections between a repeater and its Managed Nodes and PC Managed Nodes. During the data transfer from the Endpoint Gateway to the Endpoint, if the Endpoint stops receiving the data for a period longer than the value set by the session_timeout parameter, the Endpoint Gateway terminates the distribution to that Endpoint and records the timeout error. The progs_timeout setting overrides the session_timeout setting if progs_timeout is set in the file package options. Please refer to 5.3.9.4, “Setting timeout in a large-scale environment” on page 181 for more information.

connection_timeout This timeout means the time Tivoli tries to initially communicate to a Endpoint before deciding it is off-line. This timeout parameter uses the value defined in the TCP socket program of the application, so this timeout is not configurable.

tcp_keepinit This specifies the timeout value of how long TCP/IP protocol tries to initially communicate to a

target IP host machine, such as an Endpoint, before deciding it is offline. This timeout is available on the TCP layer; so, it affects all applications (for example, Web browser or Lotus Notes client) that are running on the machine.

TCP Keep-Alive This specifies the timeout value for the connection between IP hosts. During the data transfer from the source IP host to the target IP host, if the target IP host stops receiving the data for a period longer than the value set by this timeout, the source IP host terminates the distribution to that target IP host. The TCP keep-alive timeout consists of the tcp_keepidle and tcp_keepintvl parameters. However, by default, TCP keep-alive timeout is set to 130 minutes (tcp_keepidle=14400 and tcp_keepintvl=150); so, these do not affect Tivoli applications.

Notes

TCP/IP has another timeout value for maintaining the TCP sessions. This timeout is called TCP ACK timeout. During data transfer, TCP/IP protocol is monitoring the response time of the ACK (acknowledge) segment, and if the sender cannot receive the ACK segment within the timeout value, the sender re-sends the segment to the receiver. However, this timeout value is set dynamically by the operating system; so, it is not configurable.

In this redbook, we focus on the three-tiered management structure environment, so we do not explain the timeouts (global timeout and stat_inv) that are used for only Managed Node targets and PC Managed Node targets.

5.3.9.3 Unreachable target and timeout value

In the previous section, we introduced timeout parameters for each layer. This section introduces how these timeout parameters work in the distribution operations.

As we mentioned, there are two different cases in which the target Endpoint is unreachable.

- Target Endpoint machine is powered down.
- Target Endpoint machine is powered up, but the lcfd daemon is not running.

The following figure (Figure 76) shows how these timeouts affect each case when target Endpoint is unreachable.

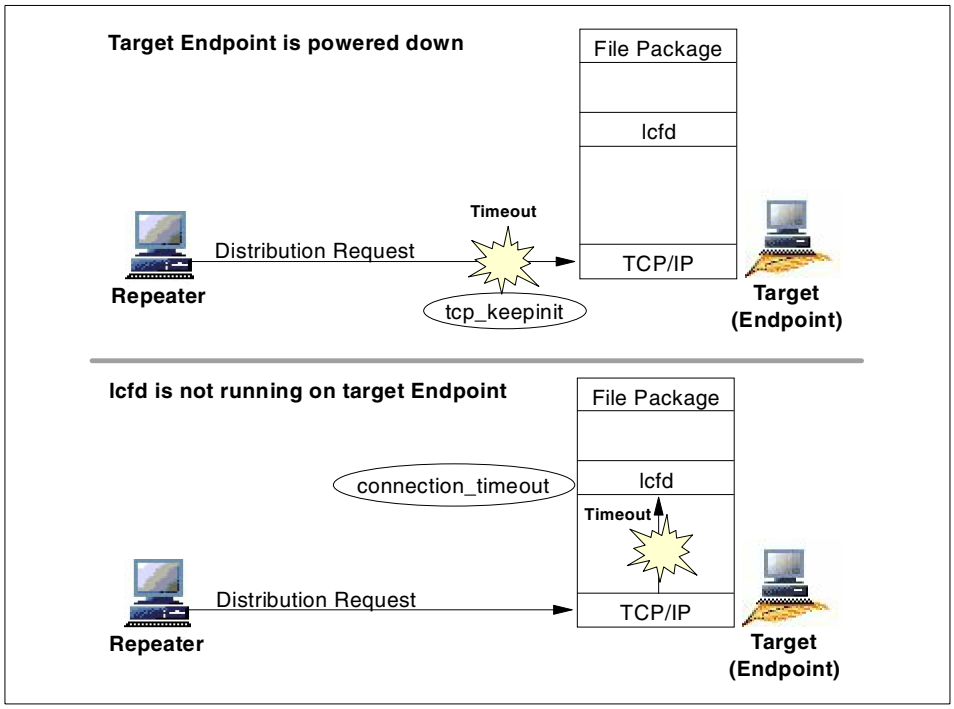


Figure 76. Unreachable target endpoint and timeout values

As you can see, the timeout value that is used when the target is powered down is different from the timeout value that is used when the lcfd daemon is down.

Target Endpoint Is powered down

In this case, the TCP layer on the target Endpoint cannot accept the distribution request because the system is powered down. Therefore, the repeater tries to connect the target until the tcp_keepinit timeout occurs. By default, this timeout is 150 seconds. It can be the performance bottleneck of the distribution process.

lcfd daemon is not running on the target Endpoint

In this case, the TCP layer on the target Endpoint accepts the distribution request and forwards it to the upper layer, the lcfd daemon. However, the lcfd daemon is not running; so, the request is not accepted and times out. Since the connection_timeout is not long, normally, it does not cause a performance bottleneck.

In the distribution operations, the worst case is that the target is powered down. However, in a large-scale management environment, these powered down targets may well exist. To improve the throughput of your distribution, you need to consider these powered down targets.

5.3.9.4 Setting timeout in a large-scale environment

As we mentioned, to handle an unreachable target efficiently, the most important timeout parameter is the tcp_keepinit parameter. However, other parameters are also very important for distributing your file package properly and efficiently. The following sections introduce some considerations and show how each timeout parameter is configured in a large-scale environment.

progs_timeout

If you define the configuration program (Before, After, Remove, and Commit scripts) in the file package, you should customize the progs_timeout for each file package.

You should set the progs_timeout by running each configuration (BARC) script in the file package separately on the slowest type of processor that will receive this file package. Take the longest time and set the progs_timeout to at least twice the amount of time that the longest script will take on the slowest system in your TMR.

For example, suppose you have a file package that has:

Before Script Checks disk space and pre-requisite software.

After Script Moves files and DLLs from a staging area and adds the registry entries and icons to the desktop.

Remove Script Moves files and DLLs form a temporary replacement hold area and deletes the registry entries and removes the icons from the desktop.

On the slowest machine in your environment, you run each of these scripts individually, and you receive the following results:

- Before Script: 10 seconds
- After Script: 150 seconds
- Remove Script: 90 seconds

In this case, the value used for the progs_timeout should be:

150 * 2 = 300 (seconds)

Therefore, the progs_timeout definition in the file package should be:

progs_timeout=300

You can change this setting by using the `wsetfpopts` command or by exporting the file package, changing the value, and re-importing the definition. You need to perform these operations for each file package that you will distribute in your management environment.

session_timeout
If you set the progs_timeout in the file package, the progs_timeout setting overrides the session_timeout value. This implementation is very flexible. Basically, we recommend you to set the progs_timeout for each file package; so, the session_timeout value should be:

session_timeout=progs_timeout (seconds)

You do not need to set the session_timeout for the distribution operation if you set the progs_timeout.

Note

The session_timeout affects each application that runs on the Endpoint and issues a downcall. For example, Inventory is the typical example. In other words, the session_timeout value is the timeout for a downcall. The Inventory profile also has its own timeout value, and this timeout setting overrides the session_timeout setting as well.

If you need to change the session_timeout value for some reason, you can change this value by using the `wgateway <gw_label> set_session_timeout value` command or the `idlattr` command.

connection_timeout
This timeout value is not configurable; so, you cannot modify this parameter.

tcp_keepinit
This timeout value affects the throughput of the distribution process greatly. However, change the tcp_keepinit with caution. The tcp_keepinit affects all applications that use TCP/IP protocol. For example, a Web browser or Lotus Notes client timeout will be changed as well. This could be a big issue.

By default, the tcp_keepinit value is 150 (75 seconds). We recommend you set the tcp_keepinit to around 40 (20 seconds) if the Endpoint Gateway machine plays only the role of Endpoint Gateway. This setting will improve the throughput of the distribution process in a large-scale environment. But, again, modify this value with caution.

You can change this value by using the `no -o` command on the Endpoint Gateway machine (AIX system) as follows:

```
no -o tcp_keepinit=40
```

If you have a Windows NT Endpoint Gateway, the TCP/IP connection establishing timeout is 169 seconds by default. Windows NT contains the definitions for TCP/IP parameters at the following directory in its registry:

```
\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
```

When you customize the TCP/IP connection establishing timeout, use the Windows NT registry editor and add the following new value on your Windows NT Endpoint Gateway machine:

Value Name TcpTimedWaitDelay

Data Type REG_DWORD

You can specify the value (seconds) for this new key. Again, this value affects all applications that use TCP/IP protocol. Modify this value with caution.

Note

The tcp_keepinit is available for AIX operating systems. Normally, each UNIX operating system has this kind of timeout value that defines how long TCP/IP protocol tries to initially communicate to the target. However, how to set this depends on each operating system. Please refer to the appropriate manuals for each operating system.

The tcp_keepinit timeout is reset after reboot of an AIX system. If you want to set the tcp_keepinit value each time the system is rebooted, you need to configure the tcp_keepinit in the /etc/rc.net file.

5.3.9.5 Solution for handling unreachable target notes

The unreachable targets affect the whole throughput of the distribution process. To handle these unreachable targets efficiently, there are two solutions:

- Modifying the tcp_keepinit timeout value
- Checking the unreachable target before the distribution

We introduced the first solution in the previous section. Another solution is to check the status of each distribution target before the file package distribution and distribute the file package only to the reachable targets. This saves the time that the distribution process has to wait when the target is unreachable,

and as a result, improves the throughput of the whole distribution process. The following figure (Figure 77) shows the scenario of this solution.

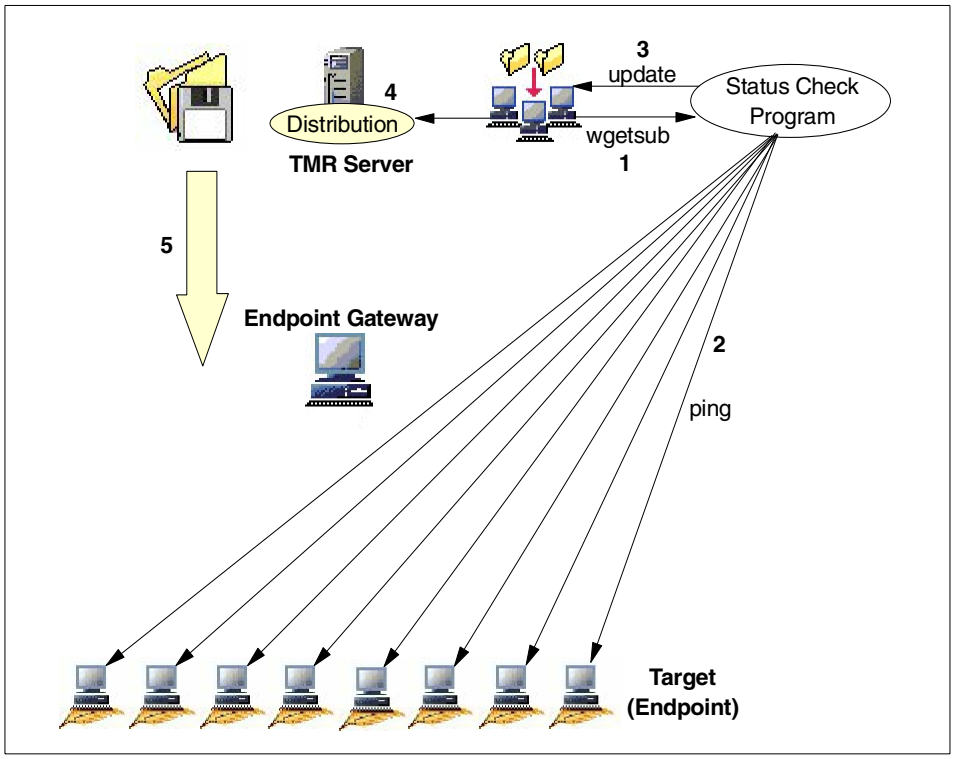


Figure 77. Checking the target status before distribution

1. The status check program gets the information of subscribers by using the `wgetsub` command. The status program also translate the subscribers information from the label into its IP address by using the `wep` command.
2. The status check program issues a `ping -c 1` command to the targets.
3. If the status check program detects an unreachable target ($? is not zero), the status check program un-subscribes the target from the profile manager and adds this target into the another profile manager that is used for the next distribution (the profile manager of the unreachable target).
4. The status check program distributes the file package only to reachable targets by using the `wdistrib` command. (You can divide the distribution process from the status checking process.)
5. The lcfd daemon may not run on some of the distribution targets. However, the connection_timeout is short; so, the throughput of the distribution will

improve. Do not forget to later distribute the file package to the targets that were unreachable at checking status time. You can use the profile manager that is created in process 3.

You can implement this solution by creating a shell script. This script depends on your environment, for example, your profile manager hierarchy, and so on; so, this section does not provide a sample.

5.3.10 File package size

Actually, there is no official guideline for the file package size because it depends on each customer environment and requirement. However, from the performance and throughput point of view, a large file package size is not recommended. Software Distribution does not support the checkpoint restart function yet. This means that if the distribution failed for some reason, the whole data in the file package is distributed again even if 99 percent of the distribution is completed when the distribution is interrupted.

In general, a large-scale management environment is less reliable than a small management environment in which each managed system is connected via LAN. Therefore, we basically recommend you to create and distribute a smaller file package other than the initial distribution. Try to distribute a file package that is less than 10 MB in a large-scale environment. This will be a reasonable size for most customer environments.

5.3.11 Using file package block

In a large-scale environment, the file package block is a very effective solution to improve the distribution throughput. The file package block consists of distribution images that are configured in the file package and file package definitions. The following figure (Figure 78) shows the components of the file package block.

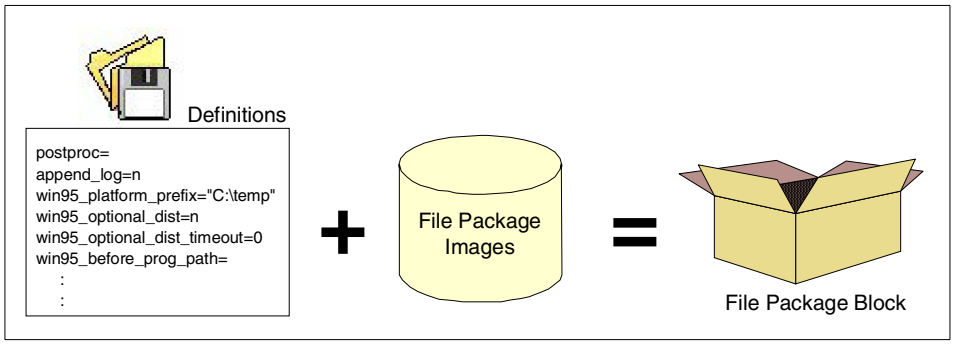


Figure 78. File package block components

As you can see, the file package block is created from existing file packages. Then, the file package block creates an archive of the file packages. As a result, the source host is moved to each location where the archive resides. The following figure (Figure 79) shows how the file package block is created and distributed to each target.

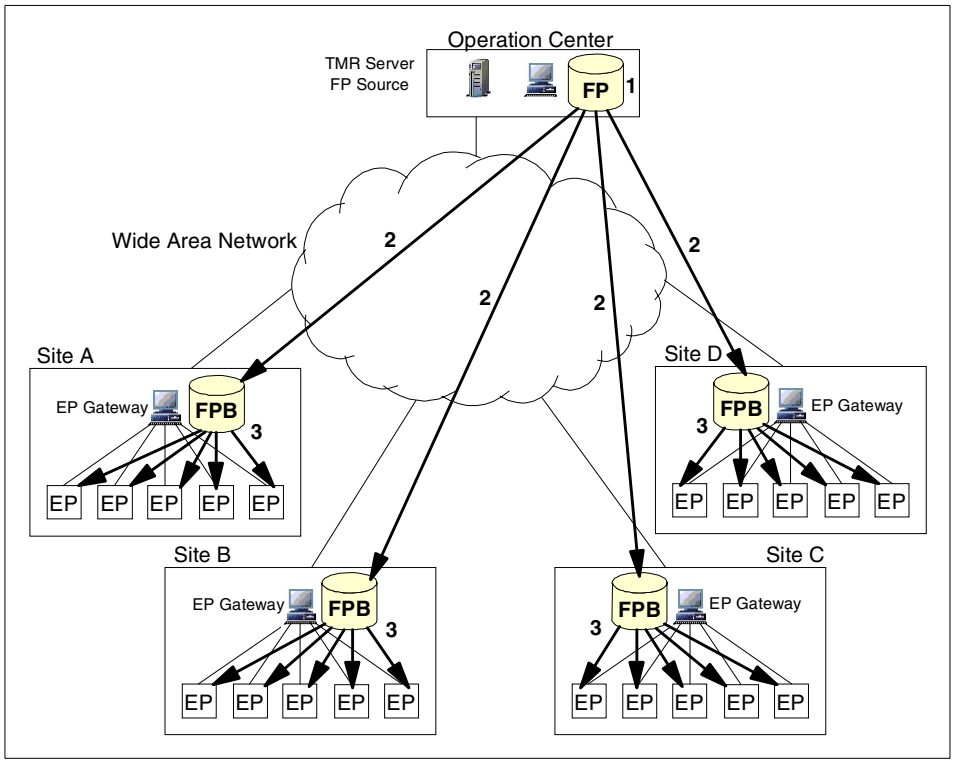


Figure 79. Using file package block in a large-scale environment

1. Create the file package as usual by using the Tivoli Desktop. Then, the file package is created on the file package source host.
2. To create the file package block, execute the `wcrtfpblock` command with the appropriate options. Then, the file package, its definitions, and its BARC programs are copied to each location that you specified. As a result, the file package block is created on each Managed Node (Endpoint Gateway) that you specified.
3. To distribute the file package block to each target (Endpoints), execute the `wdistfpblock` command with the appropriate options. Then, the file package block is distributed to each target, and BARC programs are also executed on each target.

As you can see, when you use the file package block, the distribution process is divided into the following steps:

- Distribute data from the file package source host to Endpoint Gateways (Managed Nodes).

- Distribute data from the Endpoint Gateway (Managed Node) to Endpoints (targets).

This is especially useful for performing the following operations in a large-scale management environment:

- Transferring a file package to a target on the remote side of a slow network connection, such as a WAN. This enables faster, more efficient distribution to targets over a slow network connection and improves the throughput of the distribution.
- Storing a file package whose files on the file package source host may change frequently. This enables you to distribute identical copies of the file package to all targets at any time.
- Archiving a file package for later distribution.

The file package block allows you to distribute the data asynchronously. Therefore, if the utilization of your backbone network (WAN) is low at night, to distribute the data efficiently, you should create a file package block on each Endpoint Gateway at night before distributing data to each target. Then, after all targets are booted (probably in the daytime), you can distribute the data to each target asynchronously. This is a typical scenario to use the file package block effectively.

5.3.12 Using Endpoint login_policy

Using the Endpoint login_policy can spread the load of a distribution. This solution allows you to perform a PULL type distribution by using the Endpoint login_policy. The following figure (Figure 80) shows how the Endpoint login_policy is used for distribution operations.

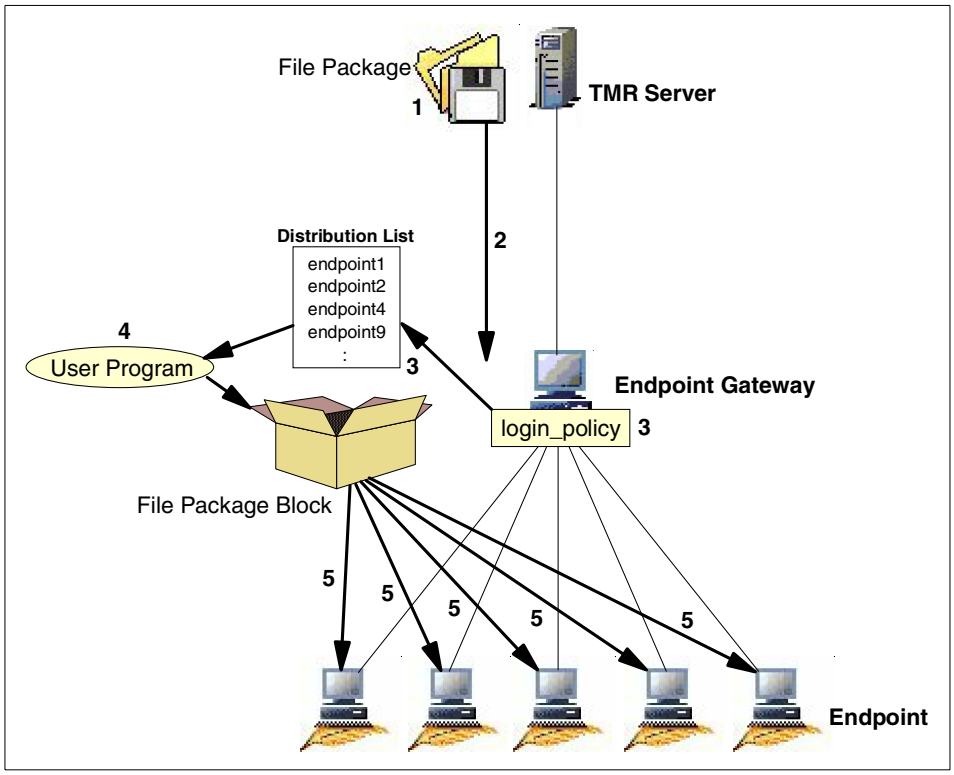


Figure 80. Using Endpoint login_policy for distribution

1. Create the file package as usual.
2. Then, create the file package block on each Endpoint Gateway when the utilization of your WAN is low. You also create and define the Endpoint login_policy. This login_policy registers the Endpoint label into the distribution list if the Endpoint that attempts to perform login is one of the correct targets.
3. When a Endpoint attempts to perform login to the Endpoint Gateway, this login_policy is executed on the Endpoint Gateway. Then, the login_policy checks whether this Endpoint is one of the correct targets or not. If this Endpoint is a correct target, the login_policy registers this Endpoint's label into the distribution list.
4. On the Endpoint Gateway, another user program is running. This user program checks the distribution list every interval time. If the distribution list contains more than 10 Endpoint labels, this user program executes the `wdistfpblock` command.

5. As a result, the file package block is distributed to the Endpoints that are registered in the distribution list. If the distribution is completed, this user program logs these distribution results and resets the distribution list.

Normally, Software Distribution performs only the PUSH type distribution. However, if you use the Endpoint login_policy and file package block, you can perform PULL type distribution. The PULL type distribution has the following advantages:

- Distribution target is known to be available.
- Load of distribution is spread out.

In this example, we use the file package block instead of the file package, because if we use the file package, the distribution must be performed from the file package source host each time the Endpoint logs into the Endpoint Gateway. This is bad in PULL type distribution.

Note

In this example, we introduced the distribution by using the distribution list. Actually, you can perform PULL type distribution without the distribution list. In this case, the login_policy will execute the `wdistfpblock` command each time the Endpoint attempts to login. However, as a result, too many `wdistfpblock` commands may be executed, and these processes may use a lot of resources on the Endpoint Gateway. Therefore, we strongly recommend you to use this kind of distribution list when you implement PULL type distribution by using the login_policy.

If you need to execute the `wdistfpblock` command from the login_policy, you also need to define the nobody user as the Tivoli Administrator login user because the login_policy is executed as a nobody user process. Of course, this is not recommended from security point of view.

In a large-scale environment, when many Endpoints attempt to login to the Endpoint Gateway at once, the load of the Endpoint Gateway will increase. If the login_policy script needs to perform complicated processes, the load of the Endpoint Gateway will increase greatly. Therefore, when you plan this way, you should perform many tests before its implementation.

5.4 Conclusion of software distribution performance

Software Distribution is the typical downcall-oriented application and handles bulk downstream data. In Software Distribution configurations, repeater tuning greatly affects Software Distribution performance, and it is mandatory. In this chapter, we discussed the following:

- Software Distribution behavior
- File package source host configurations
- MDist repeater configurations
- Profile manager configurations
- Unreachable target

There are many considerations in Software Distribution implementation. To improve Software Distribution performance, we strongly recommend you follow the information that this chapter provides.

Chapter 6. Improving distributed monitoring performance

Distributed Monitoring is one of the major Tivoli Management Applications because every system administrator would like to monitor the status and condition of their managed systems. Therefore, many Tivoli customers will use Distributed Monitoring with TMA. The following figure (Figure 81) shows the relationship between Distributed Monitoring (Tivoli Management application) and other layers.

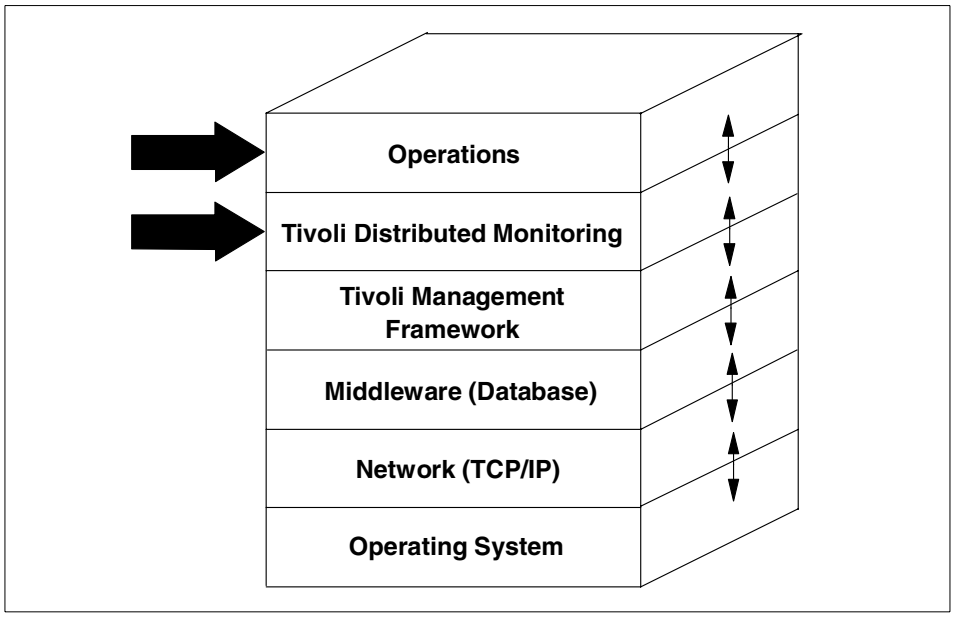


Figure 81. Tuning distributed monitoring and its operation

This chapter introduces the important issues for improving performance and throughput in Distributed Monitoring.

6.1 Distributed monitoring internals

Distributed Monitoring is an upcall-oriented application. Normally, upcall-oriented applications are more complicated than downcall-oriented applications in their implementation. In this section, we introduce how Distributed Monitoring is implemented on the TMA and how Distributed Monitoring issues an upcall, as well as the interaction between Distributed Monitoring and the TMA.

6.1.1 Distributed monitoring actions

Distributed Monitoring contains a daemon-like process called the Sentry engine. The Sentry engine (dm_ep_engine if on an Endpoint) runs on the node that we are monitoring using a Sentry monitor. Version 3.6 of Tivoli implements the LCF architecture, and the Sentry engine still exists even on the Endpoints. When the Sentry engine detects specified conditions it performs the following actions, which can be defined in the Sentry profile.

- Sending Tivoli notices
- Displaying Pop-up messages
- Changing Indicator icons
- Sending e-mail
- Logging to files
- Running programs
- Sending TEC events

The interesting thing here is that these actions are performed by the upcalls from the Sentry engine running on the Endpoint. This is very different from other Tivoli Management applications. The next section provides the information about how the Sentry engine performs the actions using the upcalls.

6.1.2 Distributed monitoring and methods

The Endpoint method that the Sentry engine issues depends on which action Distributed Monitoring performs. However, the processes to perform the action are similar to each other. As an example of an action, we describe how Distributed Monitoring processes TEC events to the TEC server. The following figure (Figure 82) shows the process of sending a TEC event to the TEC server.

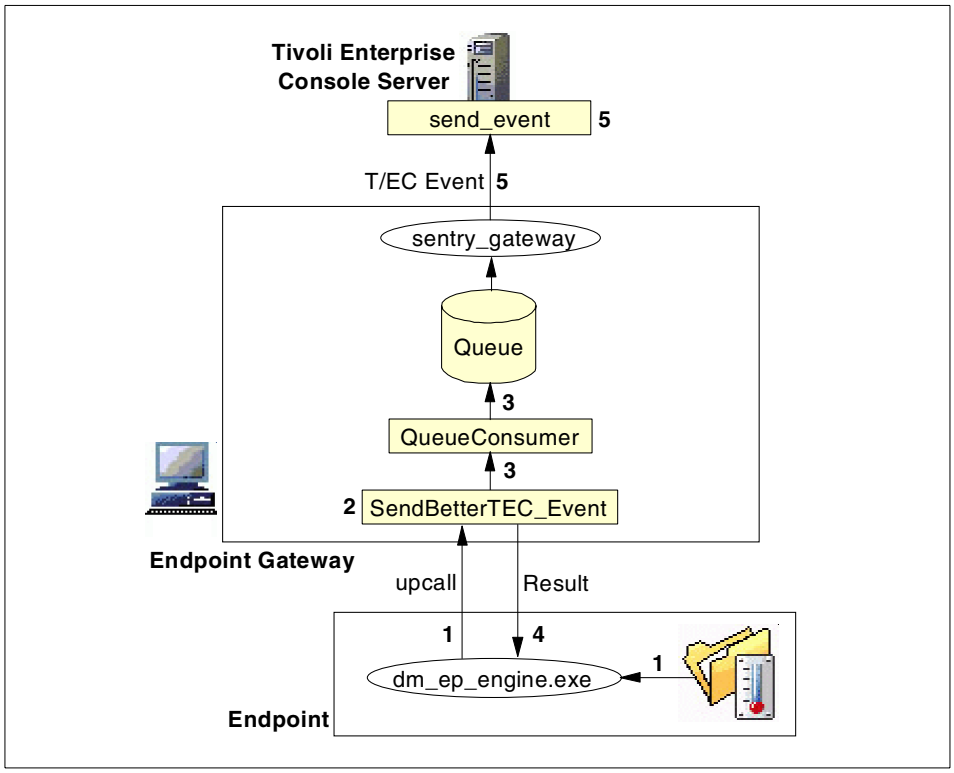


Figure 82. Sending a TEC event to TEC server

1. When the Sentry monitor detects a condition that should generate a TEC event, the Endpoint issues an upcall to invoke the SendBetterTEC_Event method on the Endpoint Gateway.
2. The SendBetterTEC_Event method is invoked by the upcall from the Endpoint. Then the SendBetterTEC_Event method spawns subsequent methods. The _get_collection method is invoked by the Endpoint many times in this sequence, and the Sentry monitor collection information is obtained from the sentry_gateway.
3. The QueueConsumer plays the role of the queue for the TEC events as well as for sending Tivoli notice and popup message.
4. After the QueueConsumer method is invoked without error, the Endpoint Gateway returns the result of the upcall to the Endpoint.
5. After returning the result of the upcall, the send_event method on the TEC server is invoked remotely from the Endpoint Gateway for sending the TEC

event. As a result, the TEC event is sent from the Endpoint Gateway to the TEC server.

Note

The _get_collection method and QueueConsumer method are invoked just once when the Sentry engine attempts to send a TEC event for the first time.

In this section, we introduced the example of sending a TEC event. When the Sentry engine performs other actions, such as sending a Tivoli notice or logging to file, other methods are invoked. However, as we mentioned, the process sequence is basically as follows:

1. Sentry monitor detects a condition.
2. Sentry engine issues an upcall to invoke the method on the Endpoint Gateway.
3. The main method that is invoked by the upcall spawns subsequent methods.
4. The Sentry gateway process (sentry_gateway) handles the request from the Sentry engine and returns its result to the Endpoint.

6.1.3 Sentry Gateway process

To support the Endpoint, Distributed Monitoring provides a new process, sentry_gateway. The sentry_gateway process runs on the Endpoint Gateway machine and is the translator between the Endpoint and Endpoint Gateway so that the sentry_gateway process handles the requests (upcalls) from the Sentry engine on the Endpoint.

When the sentry_gateway receives the request from the Endpoint by an upcall, the sentry_gateway invokes the appropriate method on the full Managed Node, for example, the TEC server or TMR server, instead of the Endpoint. After the method invocation, the sentry_gateway returns the result to the Endpoint. It acts as the proxy of the Endpoint Sentry engine. This implementation is very important for understanding how Distributed Monitoring supports the TMA. The following figure (Figure 83) shows how the sentry_gateway process works in the TMA environment.

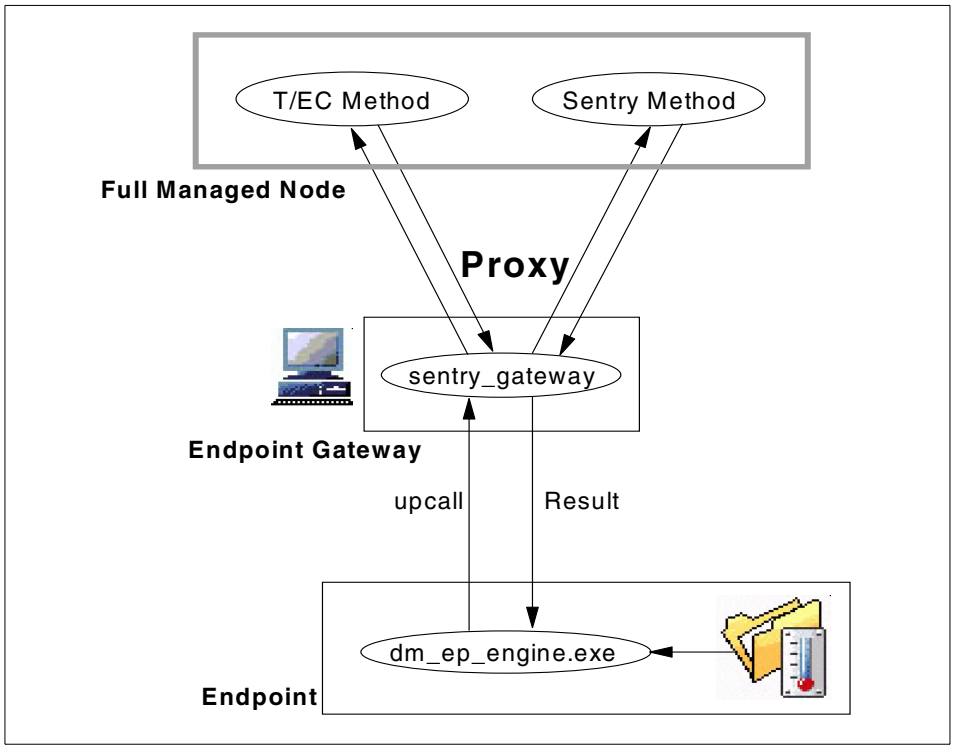


Figure 83. The role of the sentry_gateway process

6.2 Detecting distributed monitoring performance bottleneck

Unlike other applications, Distributed Monitoring does not have a list of parameters specifically for performance tuning. Distributed Monitoring performance is affected by understanding its monitoring and operation configuration. Understanding possible performance bottlenecks helps you to tune Distributed Monitoring performance efficiently. The following figure (Figure 84) shows the typical performance bottlenecks in a Distributed Monitoring environment.

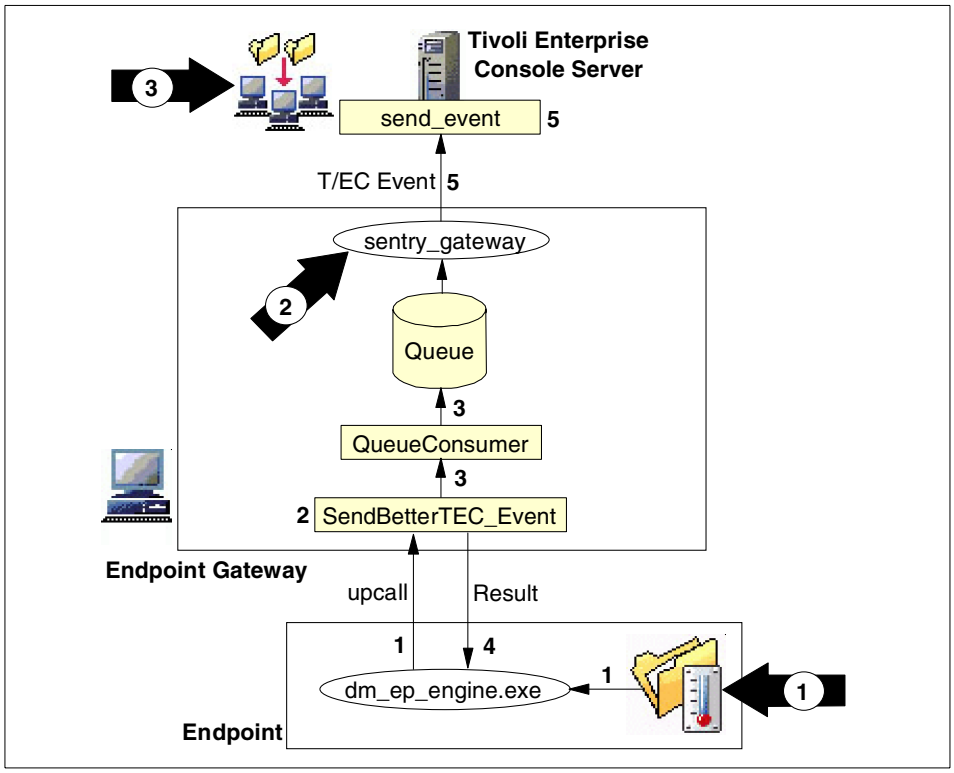


Figure 84. Sending a TEC event to the TEC server

1. Sentry Profile Configurations
2. Endpoint Gateway (sentry_engine Process)
3. Sentry Profile Distribution

In the example process flow (Figure 84), we introduced typical performance bottlenecks of Distributed Monitoring. These bottlenecks can be a problem in most large management environments. The following are the descriptions for each performance bottleneck in Distributed Monitoring environments.

Sentry Monitor Distributed Monitoring provides many Sentry monitors and configurations. To maintain the maximum throughput, you need to configure the Sentry profile carefully. We will describe considerations of Sentry monitor configuration later.

Endpoint Gateway In prior versions of Tivoli, each managed system that is monitored by Distributed Monitoring sends

information (for instance, TEC event or Tivoli notice) to the appropriate manager systems (TMR server or TEC server) directly. However, in version 3.6 of Tivoli, all information sent from Endpoints is forwarded by the Endpoint Gateway to the appropriate manager systems. In other words, the Endpoint Gateway must handle all requests regarding Distributed Monitoring from the Endpoint. Therefore, the Endpoint Gateway can be the performance bottleneck in a large management environment. We will describe considerations of Endpoint Gateway configuration later.

Profile Distribution In Distributed Monitoring environments, the repeater tuning is essential to using Distributed Monitoring successfully. Specifically, the initial distribution will cause a lot of network traffic. Therefore, you should consider both the repeater setting and network performance. Please refer to the Chapter 5, “Improving software distribution performance” on page 133 for more information about repeater configurations.

The following sections introduce how we configure and optimize Distributed Monitoring environments.

6.3 Distributed monitoring performance improving strategy

In the previous section, we introduced some major performance bottlenecks of Distributed Monitoring. Therefore, this section introduces some approaches and directions to improve the performance and throughput of Distributed Monitoring. We also provide information about how you should configure and optimize Distributed Monitoring in a large-scale management environment.

6.3.1 Distributed monitoring performance improving process flow

The following figure (Figure 85) shows the approach for improving the performance and throughput of Distributed Monitoring. When you configure or tune Distributed Monitoring in your management environment, we recommend you to follow this process flow chart.

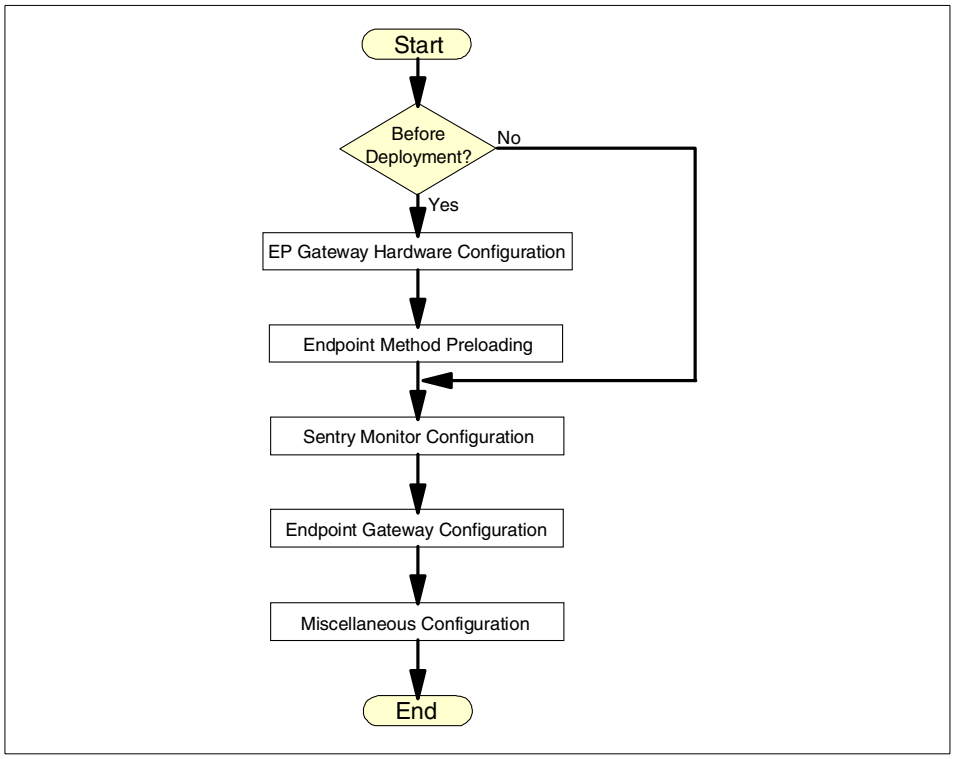


Figure 85. Distributed monitoring performance improvement process flow

In the following sections, we will introduce the detailed information about how to configure and tune your Distributed Monitoring in a large-scale management environment.

6.3.2 Preloading distributed monitoring methods

Before deploying your management environment, Endpoint method preloading can be a good solution to reduce network traffic at the initial Sentry profile distribution. When you distribute a Sentry profile initially, the Endpoint methods that will be used for Distributed Monitoring are also downloaded to each Endpoint. These Endpoint methods are defined in the dependency set, and the total size of the Endpoint methods for Distributed Monitoring is greater than 3 MB.

To avoid downloading these Endpoint methods to each target, you can use the Endpoint method preloading technique. Especially, in a large-scale deployment, Endpoint method preloading can be a very powerful solution for Distributed Monitoring environments. Please refer to “Endpoint method

preloading” on page 109 for more information about Endpoint method preloading.

6.3.3 Sentry monitor configurations

Distributed Monitoring provides many monitors to manage a variety of systems, software, hardware resources. To monitor these resources effectively, the Sentry profile provides many options. However, in a large-scale management environment, if you do not configure the proper options, these may affect adversely the monitor performance. Therefore, you need to configure the Sentry profile carefully. The following figure (Figure 86) shows the configuration panel of the Sentry monitor. In the following sections, we introduce how to configure these options to improve the performance of Distributed Monitoring.

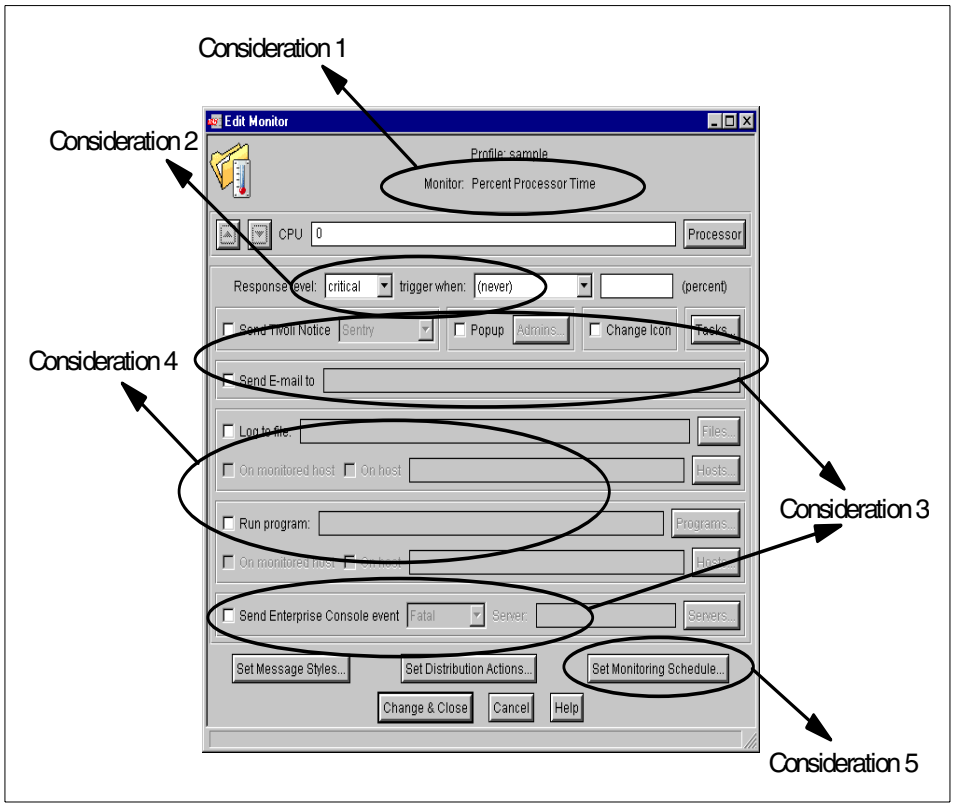


Figure 86. Sentry monitor configurations

6.3.3.1 Consideration 1: Sentry monitor selection

In the Sentry monitor, you can either use:

- An existing monitor
- A custom monitor

An existing monitor means a default monitor that is provided by Distributed Monitoring, for example, the monitor that is provided by the NT monitoring collection. It is not a customized monitor. Another monitor, custom monitor, means a monitor that is modified with a script or program by the user. A user can define some custom monitors using a shell script or program.

Normally, an existing monitor does not make a severe performance problem unless you define many monitors on each managed system. Actually, there is no guideline for how many monitors are reasonable for each managed system. How many monitors you can define on a single managed system depends on the system resources, other applications that run on the system, and so on. However, in a large management environment, we recommend you only define a minimum of monitors on the normal managed systems (client systems) and only define the monitors you need on the server or important systems.

On the other hand, use the custom monitor with caution. The heavy shell scripts or programs (which consume a lot of CPU resource and take a long time to be completed) affect adversely Sentry monitor performance. For instance, the `grep` or `awk` commands impact on the system performance may be significant. Therefore, you should consider the impact on the system performance, which is made by the script or program defined in the Sentry monitor. To reduce the load of the custom monitors, we recommend that you define the script or program as simply as you can.

Note

In most cases, the Sentry monitor that is provided by Distributed Monitoring by default (existing monitor) does not use large amounts of CPU time, disk activity, or memory at each target (Endpoint or Managed Node), Endpoint Gateway or TMR Server. The biggest resource demand is in transmitting the data over the network. As the rate of data transmission increases, the CPU utilization by Distributed Monitoring increases as well. For fast network environments that can handle megabytes per second of Sentry monitor distribution activity, CPU utilization could become a bottleneck.

6.3.3.2 Consideration 2: Response level configuration

You can configure the Sentry monitor for each response level (always, normal, warning, severe, critical). However, again, simple is the best in a large-scale management environment. Therefore, in the normal managed

systems (client systems) monitoring, we recommend you to focus on one of the following:

Threshold If you want to detect the specific condition of the managed system, define only the critical response level where you should configure the threshold.

Logging If you want to store activity data of the managed system, define only the **always** response level. However, basically, this configuration is not recommended for the normal managed systems from a performance point of view. Normally, this configuration should be implemented only on the server or important managed systems.

In the server and important systems monitoring, you should monitor as you need. To monitor a large-scale management environment efficiently, giving a priority for each managed resource is very important and mandatory.

6.3.3.3 Consideration 3: Action configurations

Distributed Monitoring provides the following monitoring actions (we will talk about logging to file and running program actions in the next section):

- Sending Tivoli Notice
- Displaying Pop-up Message
- Changing Indicator Icon
- Executing Task
- Sending e-mail
- Sending TEC Event

The important thing here is that each action issues an upcall to perform the monitoring action that is defined in the Sentry monitor when the system condition matches the trigger definition. If all monitoring actions are defined in all managed systems, what happens? Each time the trigger event occurs, many upcalls will be issued and the Endpoint Gateway has to handle all of them. As a result, your network and systems will be stressed. Therefore, we strongly recommend you define one or two actions for each normal managed system (if possible, only one action for each). It also depends on how often the managed system may perform the action in your configurations. You are able to set the response level within configuration panel.

Notes

The Sentry engine does not issue any upcall during the normal monitoring process unless the Sentry engine detects the trigger condition. Therefore, for example, the Endpoint Gateway has nothing to do for the monitored system unless the trigger condition occurs even if you create more than 100 monitors in a single monitored system. In this case, these monitors affect the performance of only the monitored system.

6.3.3.4 Consideration 4: Logging to file and running program

Logging to file or running program features are two of the monitor actions. You can specify the location (Managed Node) where the log file exists or where the program is executed.

To improve the performance, we recommend that you specify the **On monitored host** option particularly when configuring the log to file action. If you specify the **On monitored host** option, the Endpoint does not issue any upcall. The following figure (Figure 87) shows how the Sentry engine works when you specify the **On monitored host** option.

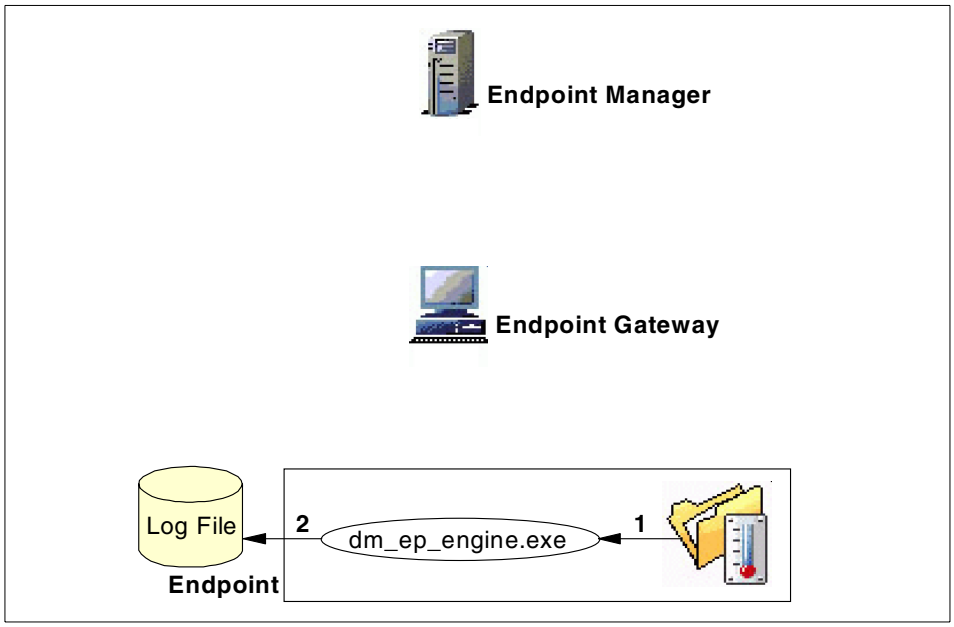


Figure 87. Logging to file action

1. When the Sentry monitor detects the condition that requires it to write a log into the log file located in the Endpoint, the Sentry engine process (dm_ep_engine.exe) on the Endpoint attempts to write a log to the log file.
2. The log is written to the local log file by the Sentry engine. No upcall is issued in this case.

As you can see, when you specify the **On monitored host**, the Endpoint Gateway does not get involved in the monitoring process. It is same for the running program action. To reduce the load of the Endpoint Gateway, we recommend this configuration.

Using network drive

As we mentioned, if you specify the **On monitored host** option, no upcall is issued in the logging to file action. In this case, if you need to gather the information that is stored in the log file, using the network drive is a very good solution. The following figure (Figure 88) shows how the Endpoint logs information to the file on the network drive.

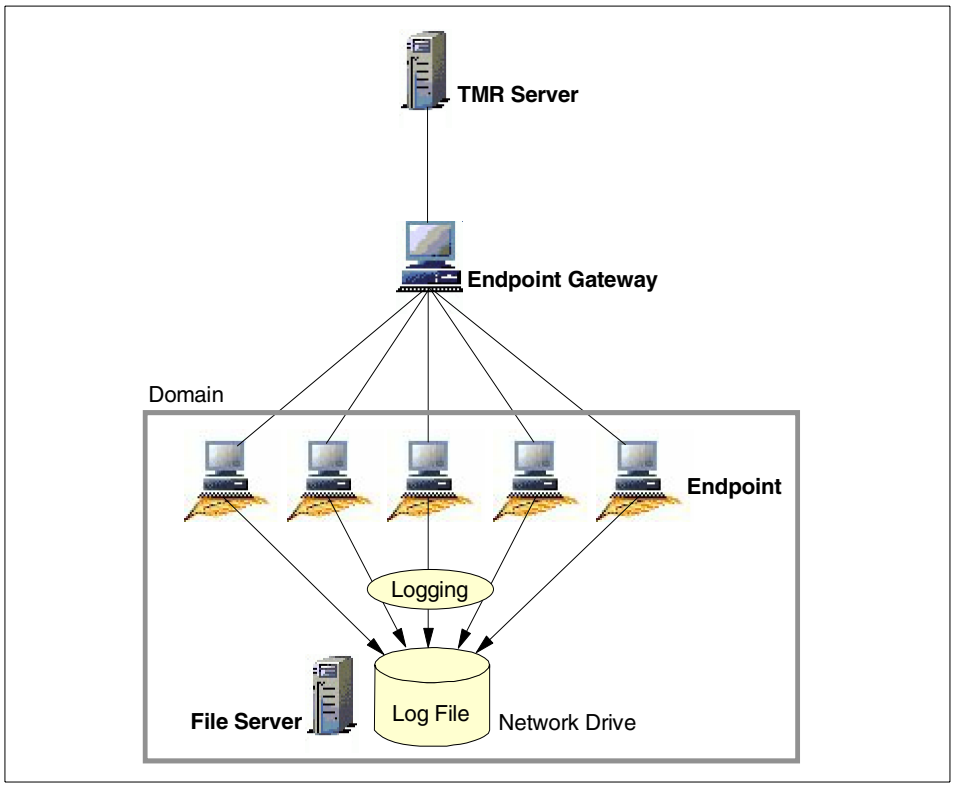


Figure 88. Logging to file on the network drive

The overview of this configuration is as follows:

1. Create the file server in your computer domain.
2. Configure each Endpoint system to mount the network drive automatically.
3. Configure the Sentry monitor that specifies the **On monitored host** option and log file that is located on the network drive.

In this configuration, each monitored Endpoint does not issue an upcall to write the log information which reduces the load of the Endpoint Gateway, and the log information is stored on a single machine.

6.3.3.5 Consideration 5: Monitor schedule

The monitor schedule affects all response levels that are defined in a single monitor; so, you should set the monitor schedule carefully. One minute is the shortest monitor interval that you can specify. The one minute granularity should be reserved for critical resources that requires detailed coverage. We recommend using the one minute interval only for critical systems.

If no trigger action occurs even if you define one minute in the **Set Monitoring Schedule** field, there is no traffic between the Sentry engine and the Endpoint Gateway. We recommend that you set the monitoring schedule to longer than five minutes.

Note

The important thing is to decide how important the monitor is in your management environment. When you configure the monitor, you should consider both of the following:

- Monitoring interval
- How often the trigger action will occur

If it is very rare for the trigger action to occur, the monitoring interval affects only the performance of the monitored system. The load of the Endpoint Gateway depends on how often the trigger action occurs.

When you define multiple monitors on a single monitored system, you need to consider the monitoring interval and the elapsed time of each monitor. The following figure (Figure 89) shows the example of efficient monitor schedule.

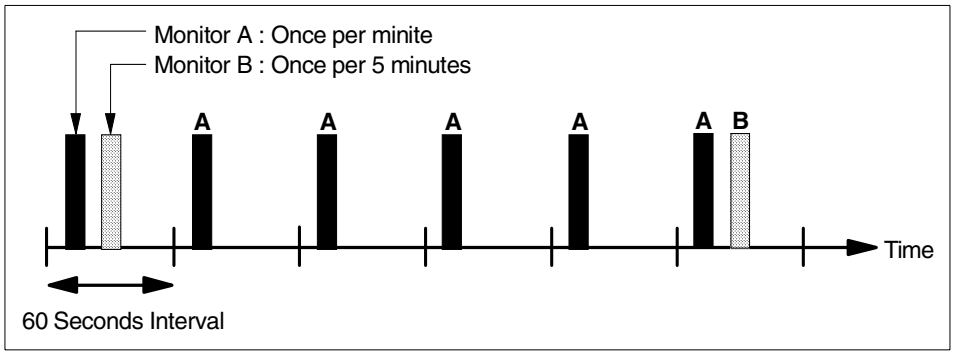


Figure 89. Efficient monitoring schedule

If you define several monitors on a single system and some of these monitors cannot be completed within the monitoring interval, the incomplete monitors will be rescheduled in the next monitoring cycle. Please refer to the following figure (Figure 90).

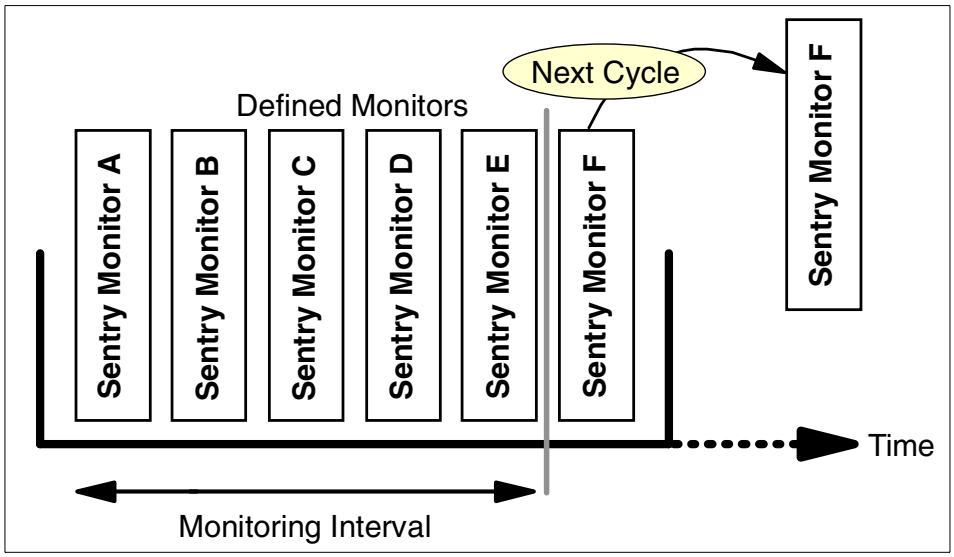


Figure 90. Sentry monitors and monitoring interval

Since the number of monitors configured exceeded monitoring interval, the next monitor is shifted into the next interval (refer to the Figure 91). As you can see, the defined monitors will be performed by the same order if all of defined monitors cannot be performed within the monitoring interval.

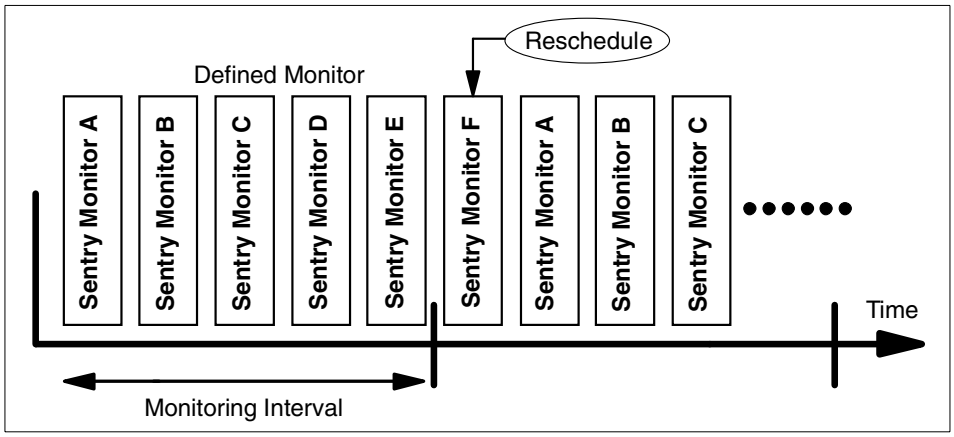


Figure 91. Rescheduled Sentry monitor

To perform efficient monitoring, we recommend that all defined monitors on a single monitored system should be completed within the monitoring interval. We also recommend that you create a monitor which completes within 60 seconds if possible. This will reduce problems in your monitoring operations.

Custom monitors, in most case, should be implemented such that they execute and return a value in the shortest amount of time to increase performance. Asynchronous monitors should be used when the data to be collected is not time dependent or time critical.

The elapsed time of the custom monitor must be shorter than monitoring interval. We strongly recommend you follow the formula below when you create and define a custom monitor.

```
Custom Monitor Elapsed Time < Monitoring Interval
```

Normally, the elapsed time of the custom monitor depends on the shell script or program that is defined in the custom monitor. The shell script or program that is defined in the custom monitor should be simple. To monitor the managed systems efficiently, the elapsed time of the shell script or program must be estimated carefully.

Asynchronous monitor is not affected by monitoring schedule, because the monitoring trigger of asynchronous monitor is time independent. Asynchronous monitor uses signal and channels. If you have to define the custom monitor that will take a long time (longer than monitoring interval) to complete, we recommend you configure asynchronous monitor.

6.3.3.6 Planning Sentry monitor

When you plan to create the Sentry monitor in your management environment, we recommend that you consider the following things:

- Define the priority in your managed systems. For example, which machine is the most important, and so on. In the important system, such as a server, you should monitor carefully.
- Decide what you really want to monitor, and create minimum monitors to satisfy your monitoring requirement.
- Decide your monitoring policy. For example, if you plan to monitor using TEC, consider only sending TEC event actions and do not use other monitoring actions.

The variety of the monitoring is one of the strong points of Distributed Monitoring. It works fine in a small management environment. However, in a large-scale management environment, you have to take care of some considerations:

- Load of the Endpoint Gateway
- Load of the TMR server
- The number of events the TEC server can handle

Again, there is no difference of the load of the monitored system between a small management environment and a large management environment. To improve performance in a very large environment, defining the monitoring priority is mandatory.

6.3.4 Endpoint Gateway configurations

First of all, in a large-scale management environment, the Endpoint Gateway must be configured using a fast CPU and enough resources. If you also use Software Distribution in the same environment, to play the role of the repeater efficiently, the hardware configuration of the Endpoint Gateway is very important. A multiprocessor machine is better than a uni-processor.

In the prior version of Tivoli, each monitored Managed Node sends TEC events to the TEC server directly when the trigger condition occurs. It is the same for sending a Tivoli notice, popup request, changing icon request, and list out all. However, Tivoli 3.6 provides a three-tiered management structure. In this environment, the sentry_gateway process that is running on the Endpoint Gateway plays the role of proxy for each Endpoint that has logged into the Endpoint Gateway. The Endpoint Gateway handles all requests of the monitoring actions and forwards the requests if needed. The following figure

(Figure 92) shows the interaction between the sentry_gateway process and monitored Endpoints.

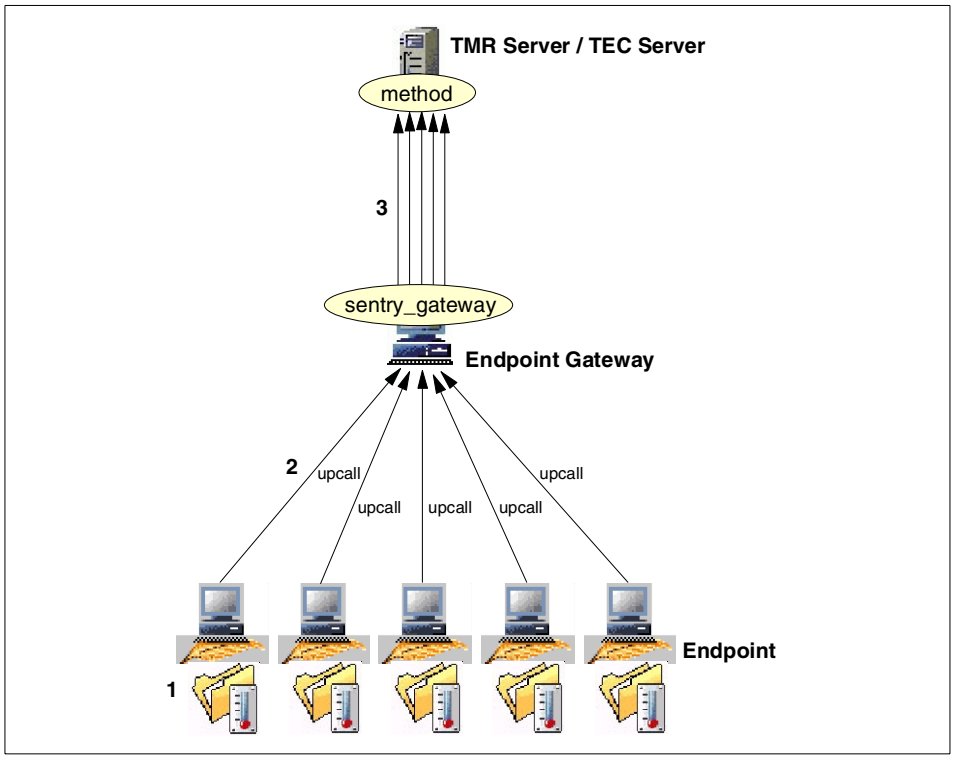


Figure 92. Sentry monitors and sentry_gateway process

1. The Sentry monitor detects the trigger condition that should perform the defined action.
2. The Endpoint issues an upcall to perform the defined action.
3. The Endpoint Gateway accepts the upcall and invokes the appropriate method remotely. Then the defined action is performed (for example, sending a TEC event, sending Tivoli notice, and so on).

As you can see, it is easy to understand how hard the sentry_gateway process is in a large-scale management environment. The Endpoint Gateway has to handle many requests from the Endpoints. The operations that are performed by the Endpoint Gateway after receiving the upcalls are almost the same as the operations that are performed by the Managed Node in the prior version of Tivoli. In other words, the operations from the Endpoint Gateway to

the TMR server or TEC server is almost the same as the operations from the Managed Node to the TMR server or TEC server.

Usually, the guideline for Endpoint Gateway configuration is that a single Endpoint Gateway can handle up to 2,000 Endpoints. However, this means that the Endpoint Gateway has to play the role of the proxy for 2,000 monitored Endpoints (refer to the “Sentry Gateway process” on page 196 for more information).

If 2,000 monitored Endpoints that have logged into the same Endpoint Gateway attempt to issue an upcall all at once, what happens? Probably the Endpoint Gateway will become a serious performance bottleneck because the Endpoint Gateway has to invoke the appropriate method 2,000 times. To avoid this situation, we recommend the following:

- Limit the monitoring events
- Optimize the max_concurrent_jobs attribute
- Multiple Endpoint Gateways when possible.

6.3.4.1 Reducing the monitoring events

As we mentioned, to manage a large-scale environment efficiently, we strongly recommend that you create a minimum number of Sentry monitors. This reduces the load of the Endpoint Gateway as well. In a large-scale environment, the simple and minimum managements are very important and are key to the success.

6.3.4.2 Optimizing the max_concurrent_jobs attribute

Normally, it is not so difficult for the downcall-oriented applications, such as Software Distribution or Inventory, to control the number of the methods that are invoked on the Endpoint Gateway concurrently because you can define the subscribers or MDist repeater parameters before the profile distribution. However, it is difficult for the upcall-oriented applications, such as Distributed Monitoring, to estimate the number of the methods that are invoked on the Endpoint Gateway concurrently because you cannot define how many trigger conditions occur per monitoring interval. Basically, the trigger condition occurrence is unpredictable.

If the upcall that is issued by the monitored Endpoint fails frequently, you should consider one of the following:

- Changing the trigger
- Changing the max_concurrent_jobs

To decrease the number of monitoring events, you should consider changing the trigger first.

The maximum number of concurrent jobs on the Endpoint Gateway can be configured using the max_concurrent_jobs attribute. If a single Endpoint Gateway is managing around 1,000 Endpoints, and the upcall fails frequently, the max_concurrent_jobs attribute may improve this situation.

By default, the max_concurrent_jobs is set to 200 (jobs). The job consists of an Endpoint login, upcall, or downcall. If the max_concurrent_jobs is exceeded, the job request goes on a wait queue. When there is a slot open, the Endpoint Gateway starts the job as a thread. However, if the max_concurrent_jobs is too low to handle all job requests, the queue will overflow, and the upcall that is issued by the Endpoint will fail. In this case, you have to increase the max_concurrent_jobs or create another Endpoint Gateway to spread the load.

Modify max_concurrent_jobs parameter with caution. This parameter is provided to avoid a hard failure. After increasing the max_concurrent_jobs, the load of the Endpoint Gateway will increase as well. It may affect other applications that are running on the Endpoint Gateway machine; so, do not increase the max_concurrent_jobs unless the hardware resources are available. When you tune the max_concurrent_jobs, to find the appropriate value, increase max_concurrent_jobs slowly until the upcall failure stops.

You can display and modify the max_concurrent_jobs as follows:

Display `idlattr -t -g <gw_oid> max_concurrent_jobs long`

Modify `idlcall <gw_oid> _set_max_concurrent_jobs value`

6.4 Conclusion of distributed monitoring performance

Distributed Monitoring is the typical upcall-oriented application. In Distributed Monitoring configurations, your monitoring definitions greatly affect Distributed Monitoring performance. In this chapter, we discussed the following:

- Distributed Monitoring behavior
- Sentry monitor configurations
- Endpoint Gateway configurations

Distributed Monitoring configurations (for example, the number of monitors on each managed resource or monitoring schedule) affect both Endpoint Gateway and TEC performance, especially in a large-scale environment. To

improve Distributed Monitoring performance, we strongly recommend you follow the information that this chapter provides.

Chapter 7. Improving inventory performance

Tivoli Inventory is a hardware and software inventory-gathering application designed to help system administrators monitor and record changes in software and hardware configurations. Inventory belongs to deployment applications. Version 3.6 of Inventory supports the TMA. The following figure (Figure 93) shows the relationship between Inventory (Tivoli Management application) and other layers.

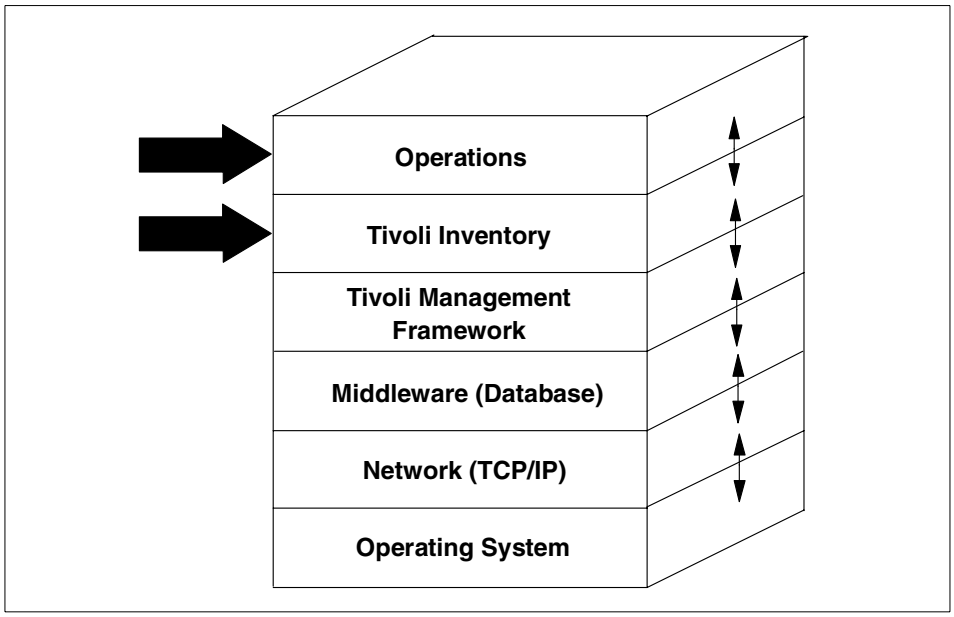


Figure 93. Tuning inventory and its operations

Inventory can generate large amounts of data that will be transported over the network and run through a single pipe into the database. If not tuned, it has the potential to take over the network. There are a few, simple, common sense steps the user can take to optimize Inventory performance:

1. Reduce the total amount of data flowing through the network by scanning only for the information needed. For example, do not perform a software scan of the entire hard drive when the only files you are interested in are in the Program Files directory.
2. Schedule CPU intensive scanning tasks, high network data flow, and heavy Inventory database usage for off-peak hours.

3. Break the number of machines to be scanned into manageable sizes so that scanning and database updates for a particular group can be completed within the off-peak time period.

In this chapter, we focus on the above issues and provide important information for improving performance and throughput in your management environment.

7.1 Inventory internals

To improve performance and throughput of Inventory, we need to understand how Inventory works in a Tivoli Management environment. Understanding Inventory behavior enables you to detect performance bottlenecks and improve performance.

Inventory can involve the following types of machines:

- TMR server
- RIM host
- RDBMS server
- Endpoint Gateway
- Target

Inventory collects and stores hardware and software information about the systems within a single TMR. Inventory uses a third-party RDBMS through the RIM component of Tivoli Management Framework. The RDBMS is used to store all the information that is gathered by the Inventory scanning process. The next section introduces the Inventory scanning procedure and which method is invoked in the operation.

7.1.1 Inventory and methods

In this section, we introduce the following information:

- Which method Inventory invokes.
- Where Inventory invokes the method.
- How Inventory issues a downcall.
- When Inventory scans and stores data of the target.

The following figure (Figure 94) shows the detailed process flow in gathering hardware and software information.

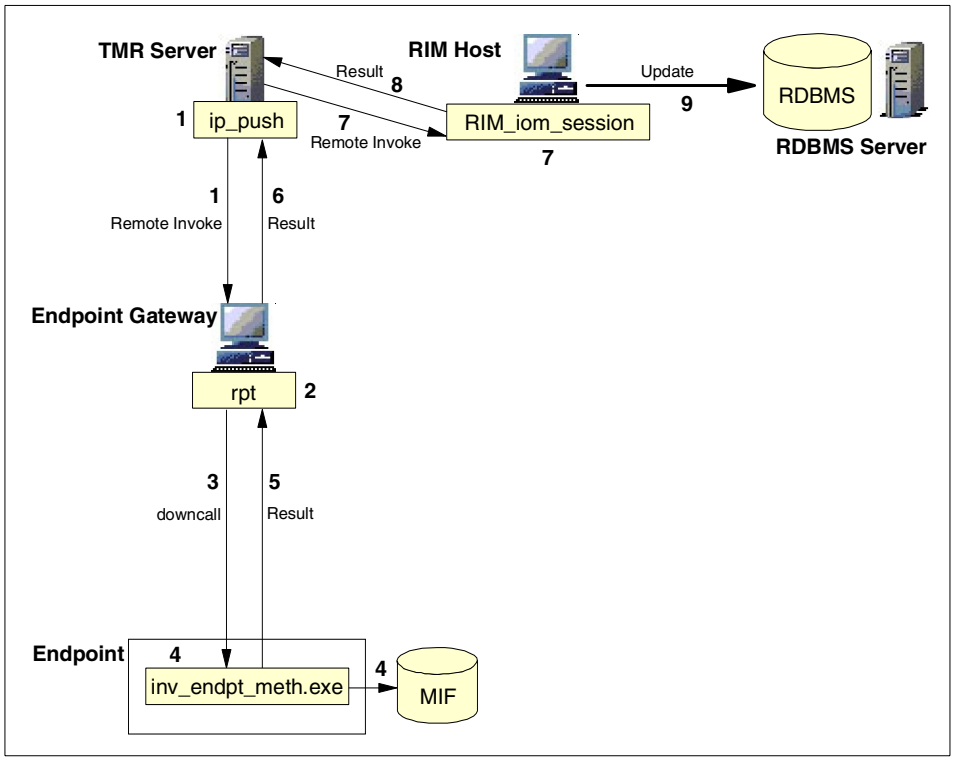


Figure 94. Inventory and its methods

1. When the Inventory profile is distributed to the subscriber, the TMR server (Endpoint Manager) invokes the ip_push method locally. The ip_push method calls subsequent method, rpt, on the Endpoint Gateway that the target Endpoint has logged into.
2. The rpt method contacts the Endpoint Gateway database (gwdb.bdb) and checks the Endpoint for the existence of the Endpoint method, ip_discover.
3. The Endpoint Gateway issues a downcall to invoke the Endpoint scanning program, inv_endpt_meths.exe, on the Endpoint target.
4. Then the scanning program, inv_endpt_meths.exe, is launched on the Endpoint target, and the scanning result is stored in the MIF file.
5. After completion of the scanning process, the Endpoint returns the result to the Endpoint Gateway.
6. The Endpoint Gateway also returns the result of the rpt method invocation to the TMR server.

7. The TMR server invokes the RIM_iom_session method on the RIM host machine. The RIM host looks up how to contact the RDBMS server and contacts the RDBMS server.
8. The RIM host returns the result of the RIM_iom_session method invocation to the TMR server.
9. The RIM host passes the data through an IOM channel.

7.2 Detecting inventory performance bottlenecks

The Inventory environment consists of many components, such as the RDBMS or the RIM host. Therefore, many factors affect its performance. The following figure (Figure 95) shows some potential performance bottlenecks in Inventory operations.

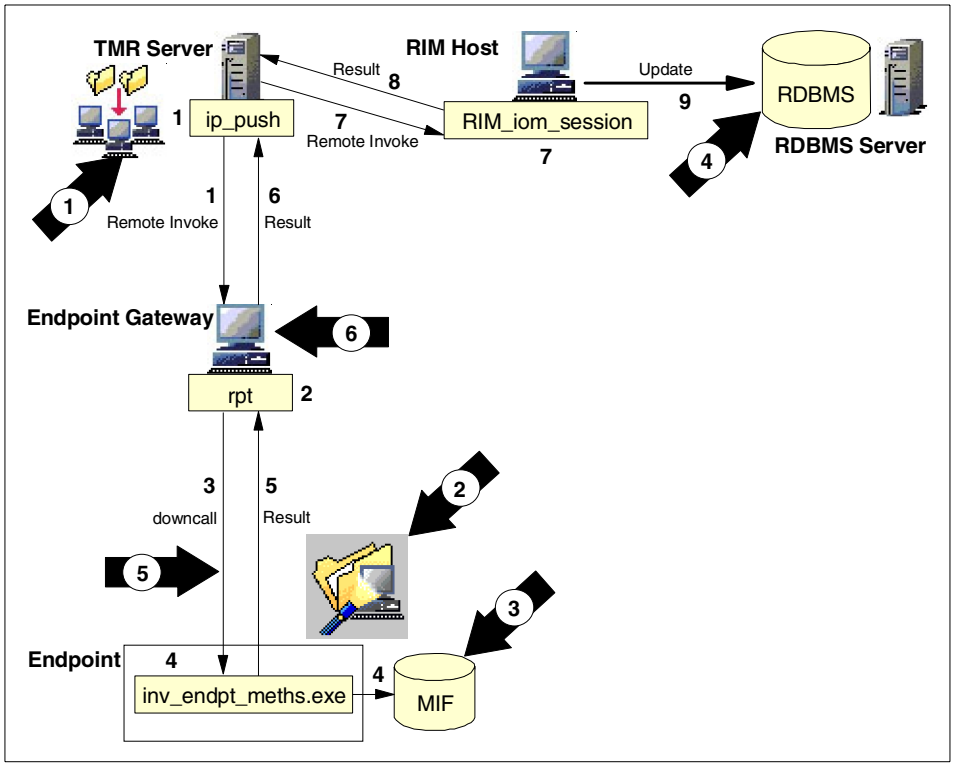


Figure 95. Inventory performance bottleneck

1. Profile Manager configurations
2. Inventory Profile configurations

3. Scanning process
4. Loading data process
5. Unreachable Target Node
6. MDist Repeater

In the example process flow (Figure 95), we indicated some of the potential performance bottlenecks of Inventory. These bottlenecks can be a problem in most large-scale management environments if the Inventory application is not tuned. The following are the descriptions for each performance bottleneck in the Inventory environment.

Profile Manager

To optimize an Inventory environment, you should configure the subscribers hierarchy carefully. Please refer to 5.3.6, “Profile manager configurations” on page 164 for more information.

Inventory Profile

Inventory profile provides some options. To improve performance, you need to configure these options properly. The important thing here is to select only the options that are really needed in your environment. Please refer to 7.3.5, “Inventory profile configurations” on page 223 for more information.

Scanning Process

The scanning process is one of the biggest performance bottlenecks. This process is a heavy user of CPU and network resources and takes a long time to be completed, especially initial scanning. To improve Inventory performance, you have to consider carefully how you should handle the scanning process. Please refer to 7.3.6, “Understanding concurrent operations” on page 226 and 7.3.8, “Dividing inventory processes into scanning and collecting” on page 235 for more information.

RDBMS Update

Version 3.6 of Inventory updates the data in the RDBMS sequentially, not concurrently; so, the Inventory database updating process can be a performance

	bottleneck. To improve Inventory throughput, the RDBMS update process should be performed during off-peak hours.
Unreachable Target Nodes	In large-scale management environment, handling the unreachable target is an issue. Several unreachable targets affect the whole performance. Please refer to 7.3.7, “Handling unreachable target nodes” on page 233 for more information.
MDist Repeater	Each repeater should be configured properly. Please refer to 7.3.6, “Understanding concurrent operations” on page 226 for more information.

Note

In the Inventory scanning process, the scanning program performs the following processes.

- Scans targets for hardware, software, custom MIF files and config files.
- Returns config files
- Parses MIF
- Translates MIF
- Calls RIM

In version 3.6 of Inventory, these processes are performed concurrently on each target. It improves the Inventory performance greatly.

However, Inventory database updating processes are performed serially because the RIM host does not support multiple connections to the RDBMS server. This may result in a potential performance bottleneck in Inventory operations.

The following sections introduce how we configure and optimize the Inventory environment.

7.3 Inventory performance improving strategy

In the previous section, we introduced some possible performance bottlenecks of Inventory. This section introduces some approaches and

directions to improve the performance and throughput of Inventory. We also provide information about how you should configure and optimize Inventory in a large-scale management environment.

7.3.1 Inventory performance improving process flow

The following figure (Figure 96) shows the approach for improving the performance and throughput of Inventory. When you configure or tune Inventory in your management environment, we recommend that you follow this process flow chart.

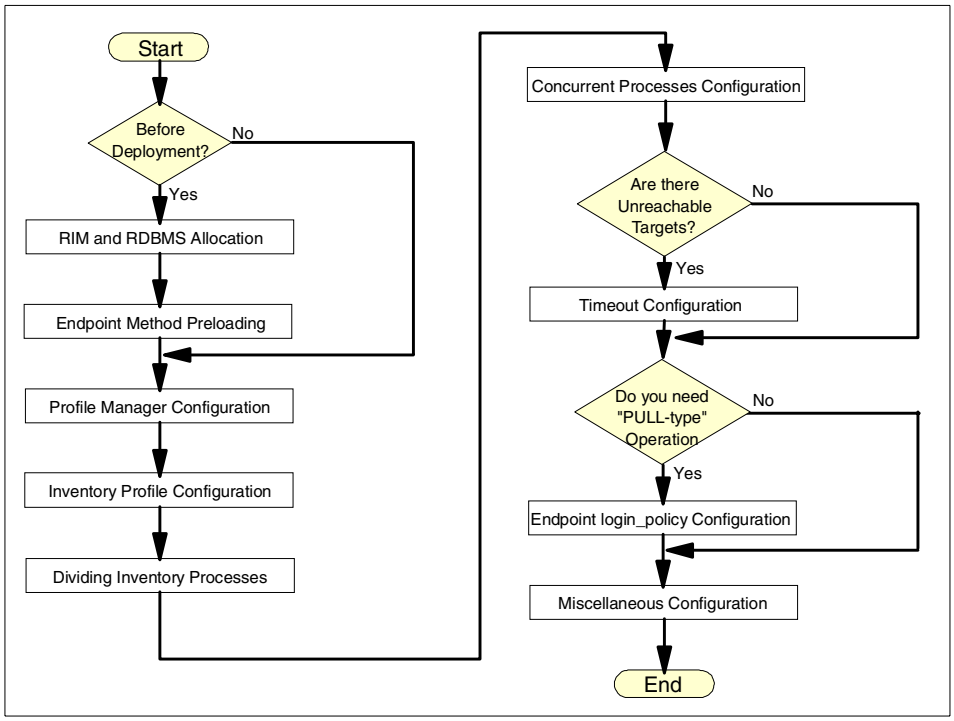


Figure 96. Inventory performance improvement process flow

The following sections describe detailed information about each subject.

7.3.2 Preloading inventory methods

Before deploying your management environment, Endpoint method preloading can be a good solution to reduce network traffic at the initial Inventory profile distribution (scanning process). When you distribute an Inventory profile initially, the Endpoint methods that will be used for Inventory are also downloaded to each Endpoint. These Endpoint methods are defined

in the dependency set, and the total size of the Endpoint methods for Inventory is greater than 0.5 MB.

To avoid downloading these Endpoint methods to each target, you can use the Endpoint method preloading technique. Especially, in a large-scale deployment, Endpoint method preloading can be a powerful solution. Please refer to “Endpoint method preloading” on page 109 for more information.

7.3.3 RIM host allocation

In Tivoli management environments, some applications will use the RDBMS server and RIM host, such as Tivoli Enterprise Console (TEC). These database operations consume a lot of CPU and network resources in Tivoli Management environments and normally can be performance bottlenecks.

The RIM host can be configured separately on a different machine. Therefore, when you use Inventory and TEC in the same TMR, we strongly recommend that you have a separate RDBMS server and RIM host for each application, especially in a large-scale management environment. It is an expensive configuration, but is a must to improve performance because Inventory and TEC are the two biggest database users in Tivoli Management applications. Of course, the RDBMS tuning is mandatory in all cases.

7.3.4 RDBMS tuning

In Inventory environments, the performance of the RDBMS is very important. Version 3.6 of Inventory does not support multiple connections between the RIM host and the RDBMS server, so Inventory updates its database serially and is a performance bottleneck. Normally, MIF data (including hardware and software scanning information) requires approximately 1.5 MB space in the database per target. When you manage thousands of Endpoints, you should estimate required database space carefully.

RDBMS performance tuning depends on which RDBMS you are using. In this book, we do not talk about how to tune the RDBMS, but we strongly recommend you to refer to appropriate manuals or documents when you install, configure, and tune the RDBMS. When you install Inventory, the installation process creates the default table for each RDBMS. Modifying some default attributes of RDBMS (Inventory database) improves the Inventory performance. If you have performance trouble in RDBMS, you should consider this.

7.3.5 Inventory profile configurations

Inventory allows you to specify the options for each Inventory profile. The following figure (Figure 97) shows the customization screen for the Inventory profile. To specify the Inventory profile options, you can use GUI or the `wsetiprf` command. This part of the section introduces some considerations for specifying the Inventory profile options.

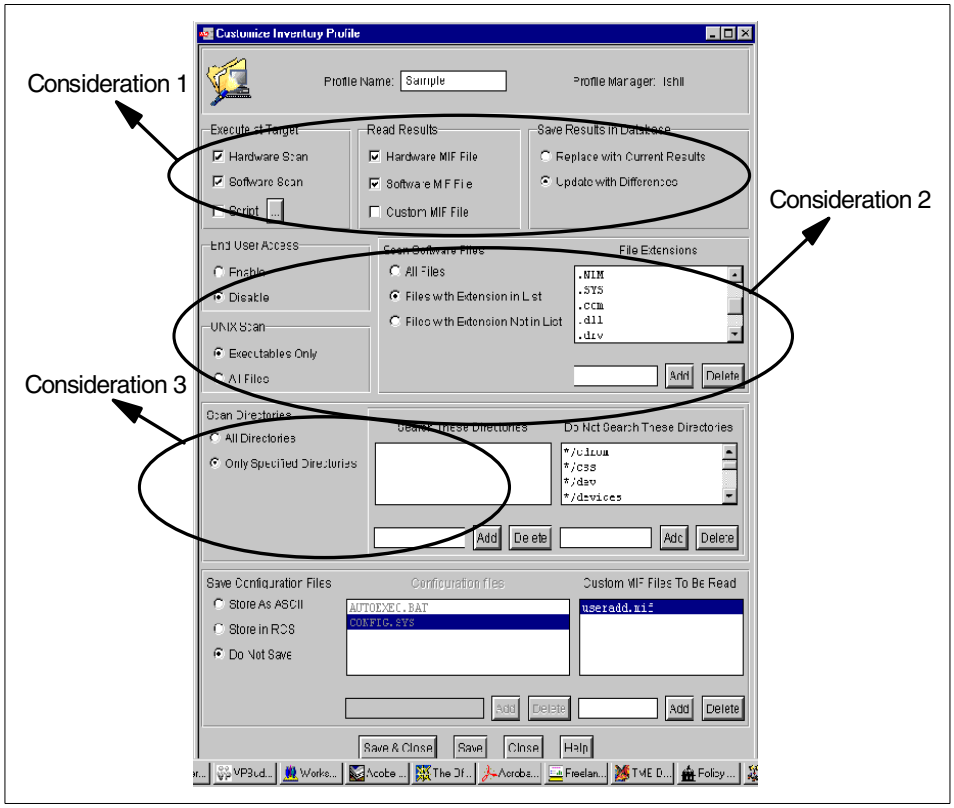


Figure 97. Customize inventory profile

7.3.5.1 Consideration 1: Scanning configurations

In the Inventory profile, there are many important options about the scanning process. These will affect the Inventory performance greatly. In this section, we introduce these options and how these options should be configured.

Scanning and Collecting MIF data

In Inventory operations, there are two types of processes, scanning and updating the Inventory database. The following figure (Figure 98) shows how

these processes work in your management environment. Both processes are heavy users of CPU and network resources, especially in the initial operation.

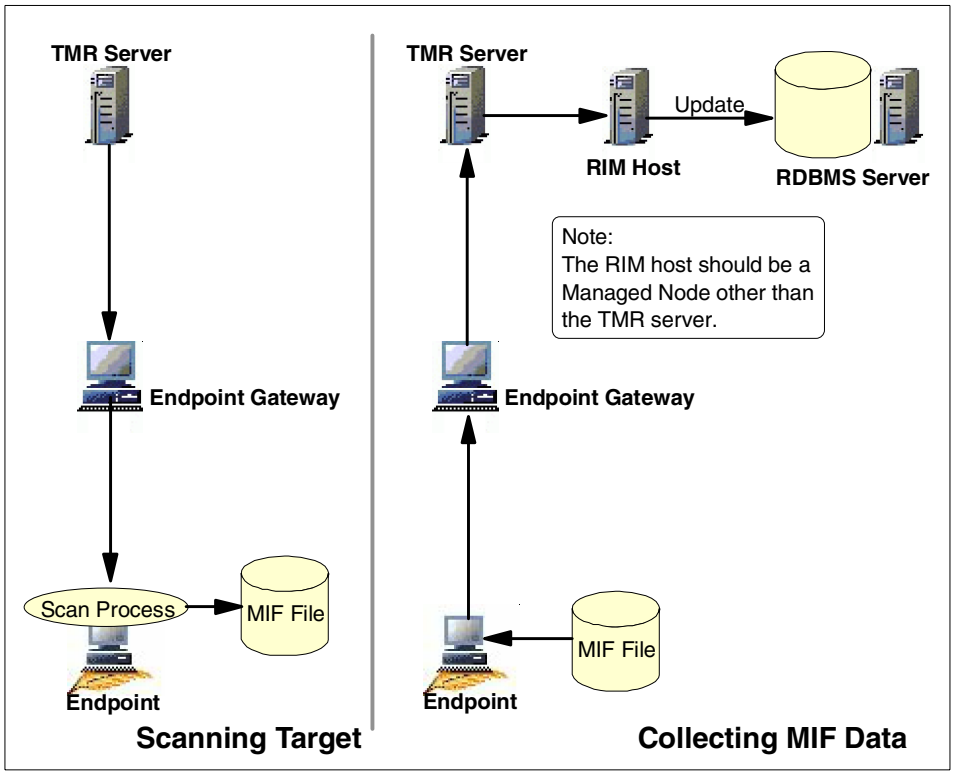


Figure 98. Inventory operations

You either define both processes in a single Inventory profile or divide both processes into two different Inventory profiles; one Inventory profile performs the scanning, and the other Inventory profile performs updating of the Inventory database. To improve the throughput in a large management environment, we recommend you to perform each process separately. We will introduce this configuration later.

Replace or update inventory database

When you perform Inventory operations, you can define one of the following options:

- Replace with Current Results
- Update with Differences

Replace with Current Results acts like an initial scan. For example, if you are doing a software scan, information on every file in the selected directories is returned and inserted into the database. If the information already exists in the database, it is overwritten. Depending on the options selected for the scan, a large amount of data could be involved. Update with Differences just returns the information that has changed since the last scan, usually involving much less data that has to be transported over the network and inserted into the database. If no information has changed, there is no data to transport and no database updates. We recommend that **Update with Differences** option be used unless there is a specific reason not to.

7.3.5.2 Consideration 2: Scanning files

Specifying scanning files affects the Inventory performance greatly and also affects the Inventory database space on the RDBMS. To improve performance, we strongly recommend that you specify the file extensions option. It depends on your management requirement, but you should carefully consider what you really want to manage. In other words, you should only scan the files that you really want to manage. This is the most certain way to improve Inventory performance.

Note

For UNIX systems, choosing the **Executable Only** option and the **All Files** option are recommended (refer to the Figure 97 on page 223). The Executable Only option allow you to scan only files having the executable permission set. This option makes the profile distribution to UNIX machines faster. If you click the **Executable Only** option, also click **Software scan** option in the Execute at Target panel. You should also click the **All Files** ratio button in the Scan Software Files panel to ensure that all executable files are scanned. By default, the Executable Only option is enabled.

Normally, Inventory requires approximately 1.5 MB of database space for each scanning target. If you scan only a few file extension types of the files, it saves database space.

7.3.5.3 Consideration 3: Scanning directories

There are two options to consider when scanning directories: All Directories and Only Specified Directories. Refer to your *Tivoli Inventory User's Guide*, GC31-8381 (Chapter 5, Creating Inventory Profiles), for specific and detailed information concerning these options. Use of the 'Only Specified Directories'

option allows you to control and reduce the amount of file information returned over the network and inserted in the database.

For example, some directories are assigned to the user data or temporary work space, and these directories are not needed by the scanning process. You should scan only directories that contain files you really want to scan.

To improve performance, we strongly recommend you specify the **Only Specified Directories** option. This option is very efficient for the management environment that consists of the machines that have almost the same directory structure.

7.3.6 Understanding concurrent operations

Inventory is a very unique application because it consists of many components, for example, RIM host, RDBMS server, and repeaters. Sometimes, Inventory is said to be a complicated application. On the other hand, Software Distribution is also a downcall-oriented application, but its distribution operation is very simple. Why is Inventory complicated? The main reason is that Inventory has two different operations as follows:

- Creating MIF file (Distributing Inventory profile)
- Updating Inventory database

One is performed from the TMR server to target, another is performed from target to TMR server. These processes are performed opposite each other.

Each process, creating the MIF file and updating the database, has the mechanism that controls the maximum number of the concurrent processes. The following parameters can manage them:

max_conn This value controls how many scanning process can be performed concurrently on the targets. This is the same as the number of targets distributed at once to Software Distribution. The default value is 100.

Onetime Max Processes This value specifies how many MIF files the TMR server can store before the TMR server transfers them to the RIM host. The default value is 100.

7.3.6.1 How inventory performs scanning process concurrently

The max_conn and Onetime Max Processes parameters play a vital role in the Inventory operations. The following figure (Figure 99) shows how these

parameters work in the Inventory scanning process. In this example, each parameter is set to the following value:

- max_conn = 10
- Onetime Max Process = 20

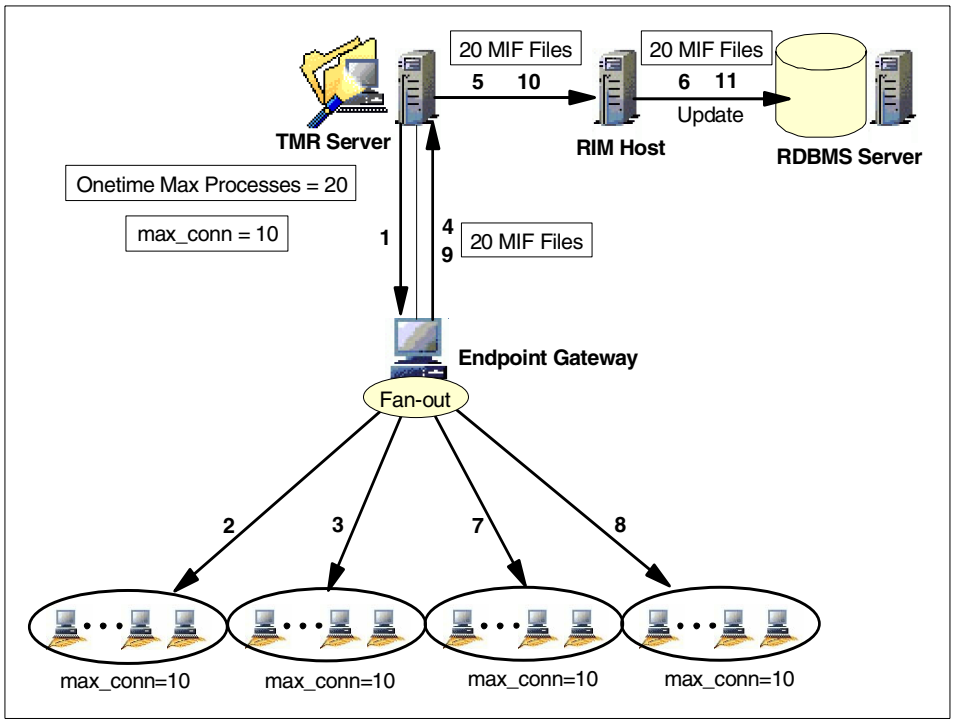


Figure 99. Inventory concurrent operations (Example 1)

1. The Inventory profile is distributed to the 40 Endpoint targets.
2. The Endpoint Gateway (repeater) distributes the Inventory profile to the first 10 targets by using the fan-out feature. Then the Inventory scanning program is executed and starts to scan software or hardware information on each target machine. The scanning information is stored in the MIF file on each target.
3. The Endpoint Gateway (repeater) distributes the profile to the next 10 targets. Then the scanning process starts to collect information on each target. The scanning information is stored in the MIF file on each target.
4. When the scanning process is completed on each target, MIF data is sent back to the TMR server and stored in memory.

5. Once the MIF data of 20 targets is stored in the TMR server, it is transferred to the RIM host.
6. Then the RIM host forwards the MIF data to the RDBMS server. The RDBMS server updates the Inventory database. Since Version 3.6 of Inventory does not support multiple connections between the RIM host and the RDBMS server, the MIF data is written to the Inventory database serially.
7. The Endpoint Gateway distributes the Inventory profile to the next 10 targets. Then the scanning process starts to collect information on each target, and the scanning information is stored in the MIF file on each target.
8. The same processes are performed on the next 10 targets.
9. When the scanning process is completed on each target, each MIF data is sent back to the TMR server and stored in memory.
10. Once the MIF data of 20 targets is stored in the TMR server, it is transferred to the RIM host.
11. Then the RIM host forwards the MIF data to the RDBMS server. The RDBMS server updates the Inventory database serially.

The following figure shows another example of the Inventory scanning process. In this case, each parameter is set to the following value:

- max_conn = 40
- Onetime Max Process = 20

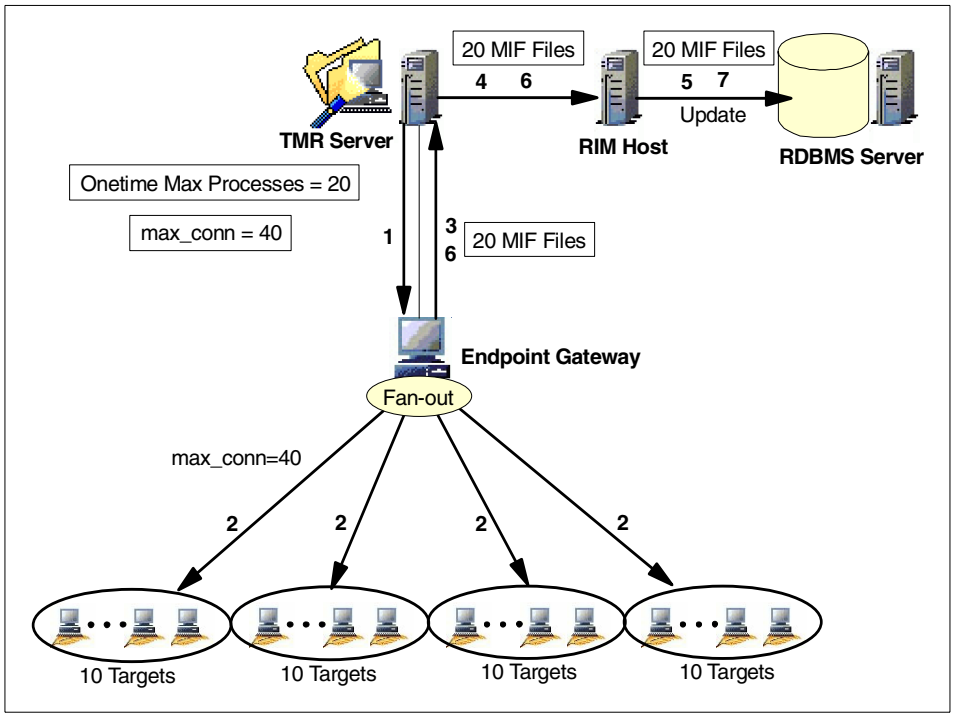


Figure 100. Inventory concurrent operations (Example 2)

1. The Inventory profile is distributed to the 40 Endpoint targets.
2. The Endpoint Gateway distributes the profile to the 40 targets by using The fan-out feature. Then the Inventory scanning program is executed and starts The scan process.
3. When the scanning process is completed on each target, each MIF data is sent back to the TMR server and stored in the memory of the TMR server.
4. If the 20 targets' MIF data is stored in the TMR server completely, the MIF data is transferred to the RIM host.
5. Then the RIM host forwards these MIF data to the RDBMS server. The RDBMS server updates the Inventory database serially.
6. When another 20 targets' MIF data is stored in the TMR server completely, the MIF data is transferred to the RIM host as well.
7. Then the RIM host forwards the MIF data to the RDBMS server, and the RDBMS server updates the database serially.

7.3.6.2 Setting the parameters

As you can see, Inventory controls the number of scanning processes and database updating processes by using the max_conn and the Onetime Max Processes parameters. When you configure these parameters, there are some considerations as follows:

- A large number of the Onetime max processes parameter affects the memory utilization of the TMR server. If you set the Onetime max processes to a large value, the TMR server may hang up because of the lack of memory.
- As we mentioned in “MDist repeater configurations” on page 150, the max_conn=10 is recommended for UNIX systems (max_conn=5 for Windows NT systems).
- Do not forget that Inventory updates its database serially. It can be a performance bottleneck even if you set a large value to the Onetime max processes parameter.

Our recommendation is to set the same value for both parameters, max_conn and Onetime max processes.

7.3.6.3 How to modify onetime max processes parameter

You can modify the Onetime max processes parameter as follows:

1. Get the InventoryProfile class object ID.

   ```
   # wlookup -r Classes InventoryProfile
   1613591617.1.940#TMF_TNR::InstanceManager#
   ```

   If you want to modify the Onetime Max Processes of the specific Inventory profile, execute the `wlookup` command as follows.

   ```
   # wlookup -r InventoryProfile Sample
   1613591617.1.122#InventoryProfile#
   ```

   Then use this object ID as the argument for the `idlattr` command in process 3.

2. Invoke the `_get_prototype` method by using the `objcall` command.

   ```
   # idlcall 1613591617.1.940 _get_prototype
   1613591617.1.941
   ```

3. Get the attributes of Inventory by using the `idlattr` command (redirect the output to the file).

   ```
   # idlattr -t -g 1613591617.1.941 chunables TMF_Types::LongList >
   file_name
   ```

4. By default, the following values are set to the attributes. The forth parameter (100) indicates the Onetime Max Processes parameter.

   ```
   { 8 5 3 100 0 0 0 1 1 }
   ```

5. Edit the list as follows (in this case, the Onetime Max Process is set to 10) and save it.

   ```
   { 8 5 3 10 0 0 0 1 1 }
   ```

6. Set the attribute by using the `idlattr` command.

   ```
   # idlattr -t -s 1613591617.1.941 chunables TMF_Types::LongList <
   file_name
   ```

7. Make sure of the modification of the Onetime Max Process attribute.

   ```
   # idlattr -t -g 1613591617.1.941 chunables TMF_Types::LongList
   { 8 5 3 10 0 0 0 1 1 }
   ```

After these operations, the modified Onetime Max Processes attribute becomes available in your Inventory environment.

7.3.6.4 MDist Repeater considerations in an inventory environment

Normally, in your Tivoli Management environment, each MDist repeater is tuned for downstream data. In other words, when you tune the repeater, you mainly consider the data transfer from the TMR server to each Endpoint. Although this is the usual customization, do not forget that the upstream data uses the same configuration of each repeater. Let us assume the following environment (refer to the Figure 101).

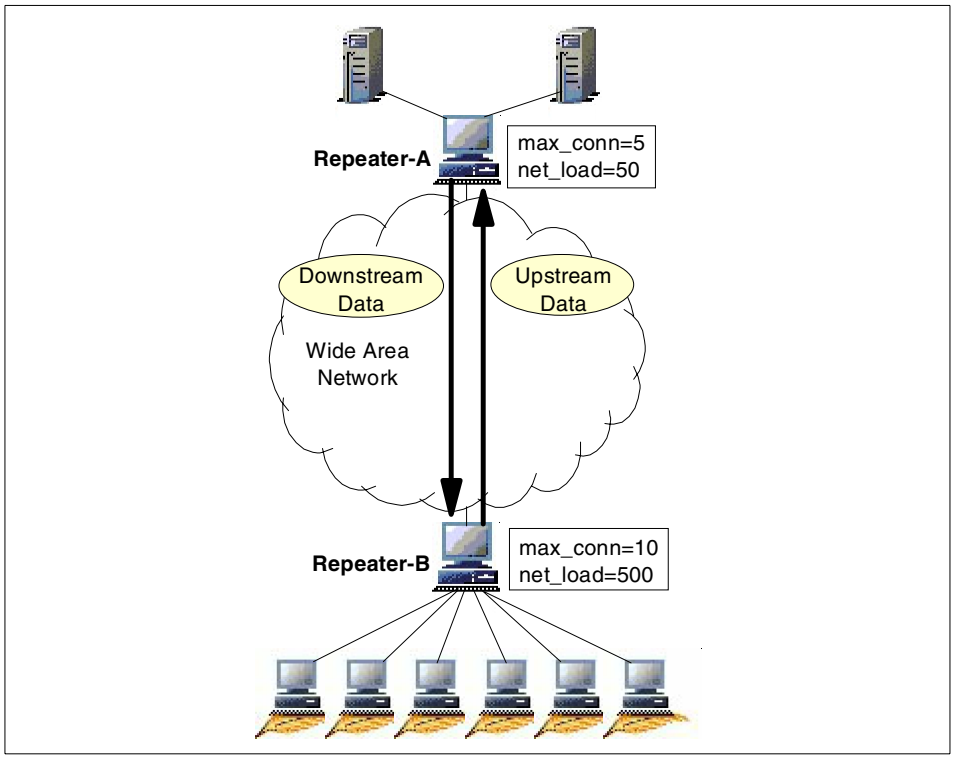


Figure 101. Downstream and upstream data in Tivoli Management Environment

In this example, each repeater has been tuned for downstream data, such as Software Distribution. Therefore, the Repeater-A is tuned for slow link (WAN), and the Repeater-B is tuned for fast link (LAN). It is optimized for the downstream data transfer.

In this example, the upstream data has to use the same parameters of each repeater. This means that the Repeater-B attempts to send the data to the Repeater-A by using net_load=500 and max_conn=10. This is very important in your environment.

Most of Tivoli operations will use downstream data, for example, profile distribution or file package distribution. So, you should tune each repeater for downstream operations even if Inventory transfers upstream data. However, you should always consider Inventory concurrent operations. Inventory provides the parameter that controls concurrent processes (Onetime Max Processes); so, you have to configure this parameter properly in a large-scale environment.

7.3.7 Handling unreachable target nodes

Inventory is one of the downcall-oriented applications, such as Software Distribution. The mechanism of the timeout is similar to Software Distribution. One difference is the Inventory scanning timeout, which corresponds to the progs_timeout of Software Distribution.

7.3.7.1 Scanning timeout

This is used by the Inventory scanning program. The scanning timeout sets a client-level timeout value for the scanning program. The intention of this timeout is to avoid the interruption of the lower layer timeout (for example, session_timeout) and to set a time value after which a hanging scanning program is killed on the target system because sometimes the scanning process encounters an error that is not returned properly. The scanning timeout setting overrides the session_timeout setting if the scanning timeout is set in the Inventory profile. It is a similar mechanism to the progs_timeout of Software Distribution. Please refer to 5.3.9.2, “Understanding timeout for each layer” on page 175 for more information.

Since Inventory uses MDist repeater features for distributing profiles to each target, the setting of the other timeouts (shown below) that affects each layer is the same as Software Distribution.

- session_timeout
- connection_timeout
- tcp_keepinit
- TCP Keep-Alive

The following figure (Figure 102) shows the information about each timeout for the Endpoint target and how these timeouts work for each layer.

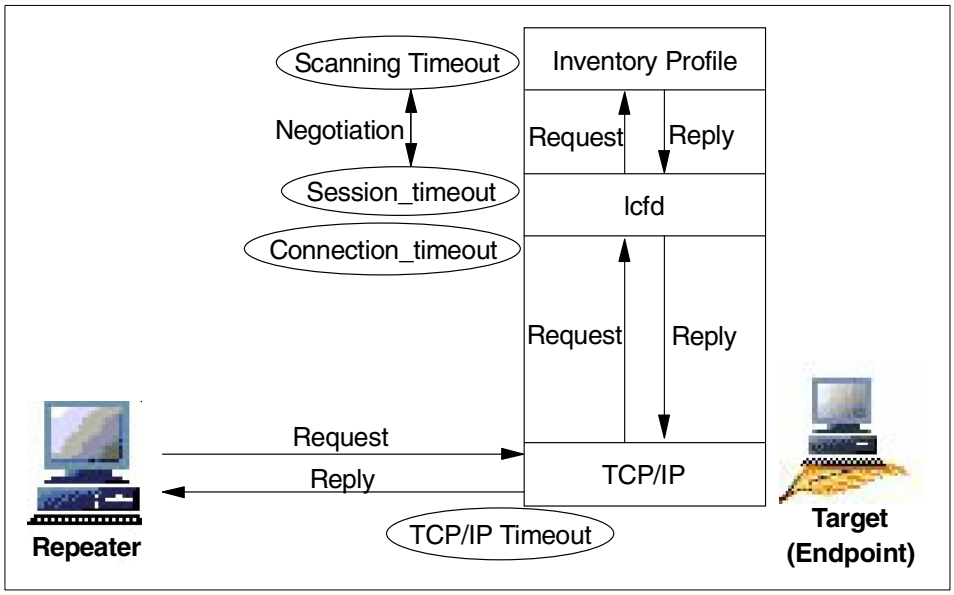


Figure 102. The timeouts for each layer

This timeout mechanism is similar to the Software Distribution timeout mechanism. Please refer to the “Handling unreachable target nodes” on page 174 for more detailed information about the timeouts.

By default, the scanning timeout is as follows:

Endpoint Target 1,800 seconds (30 minutes)

PC Managed Node Target 2,700 seconds (45 minutes)

As we mentioned, these timeout settings override the session_timeout. The session_timeout does not disconnect the session before the scanning program finishes. Normally, the scanning process will finish within the above default timeout even if the scanning target system is large. Therefore, we recommend you to use these default scanning timeout values unless a scanning process cannot finish within the timeout.

When you display the scanning timeout for the Endpoint target, you can use the `wgetiprf` command as follows:

```
# wgetiprf -t @InventoryProfile:test
Endpoint timeout period: 1800 seconds
#
```

You can use the `wsetiprf` command to set the scanning timeout for the Endpoint target as follows (the following example set the scanning timeout to 300 seconds):

```
# wsetiprf -t 300 @InventoryProfile:test
#
```

Note

When you display or modify the scanning timeout for the PC Managed Node target, you can use the `-T` argument instead of `-t` argument (`-t` argument is used for Endpoint). If you want to display all options that are configured in the Inventory profile, you can use the `wgetiprf` command with the `-a` argument

7.3.7.2 Solution for handling unreachable target nodes

To handle unreachable targets efficiently, there are two solutions as follows:

- Modifying the tcp_keepinit timeout value.
- Checking the unreachable target before the Inventory profile distribution.

You can use the same solutions to improve the Inventory scanning throughput as were used for Software Distribution. We introduced detailed information about them in a previous section; so, please refer to “Handling unreachable target nodes” on page 174 for more information.

7.3.8 Dividing inventory processes into scanning and collecting

As we mentioned, in the Inventory operations there are two different types of processes. One is to scan the software or hardware information and store this scanning information into the MIF file on each target. Another is to collect these MIF data and update the Inventory database (refer to the Figure 98 on page 224).

7.3.8.1 Dividing inventory profiles

Both processes are heavy users of CPU and network resources, especially in the initial scanning. To improve the throughput, we recommend you to divide the Inventory processes into two separate Inventory profiles. One Inventory profile performs the scanning process on each target, and another Inventory profile performs collecting MIF data from each target and updating the Inventory database. A third could collect configuration files if you define a custom MIF file because retrieving configuration files is done separately. In a large-scale management environment, the initial scanning will consume a lot

of CPU and network resources and take a long time. Therefore, we recommend dividing Inventory profiles especially for the initial Inventory operation. The following figure (Figure 103) shows how each Inventory profile works in the Inventory operations.

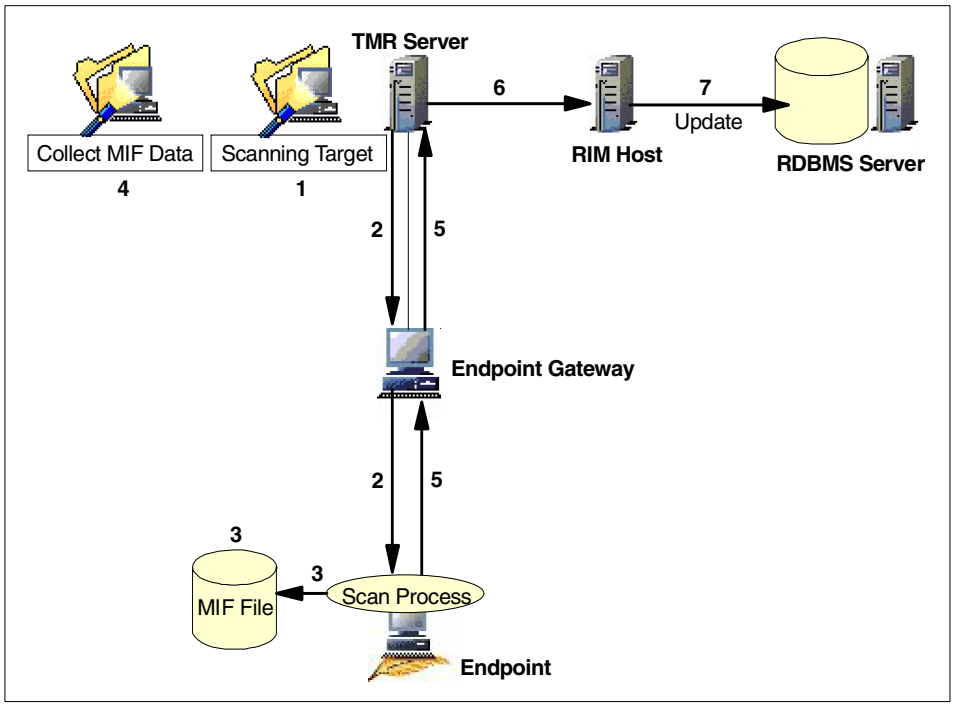


Figure 103. Dividing inventory process into scanning and collecting

1. Create the Inventory profile that performs only scanning processes.
2. Distribute the Inventory profile to the Endpoint target. If there are several Endpoint targets, the Endpoint Gateway works as a repeater and distributes the profile using the fan-out feature.
3. The scanning program is executed and starts to scan the software or hardware information on each target. The scanning program creates the MIF file and stores scanning information into the file. The scanning profile completes its scanning process after storing MIF data.
4. To collect each MIF file that contains scanning information, create another Inventory profile that performs only collecting MIF data and updating the Inventory database.
5. Distribute the Inventory profile to the Endpoint target. If there are several Endpoint targets, the Endpoint Gateway works as a repeater and

distributes the profile using the fan-out feature. After distributing the profile, Inventory starts to collect MIF data that is created by the scanning process, and this data is sent back to the TMR server.The TMR server receives the data and forwards it to the RIM host.

6. The TMR server receives the data from the target Endpoints and forwards it to the RIM host.
7. The RIM host also forwards the MIF data to the RDBMS server. The RDBMS server updates the Inventory database serially. When this process is finished, the Inventory scanning and updating database processes have been completed.

7.3.8.2 How to configure inventory profiles

In this scenario, you should create four different types of Inventory profiles. The following table (Table 10) shows how you should create each Inventory profile.

Table 10. Dividing inventory profiles

Scanning Information	Scanning Target	Collect MIF File
Hardware	Inventory Profile 1	Inventory Profile 3
Software	Inventory Profile 2	Inventory Profile 4

As you can see, each profile is created only for hardware scanning, software scanning, updating hardware MIF data, and updating software MIF data. When you configure your Inventory profile this way as shown in Table 10, you need to configure the following options in the Inventory profile (refer to the Figure 104).

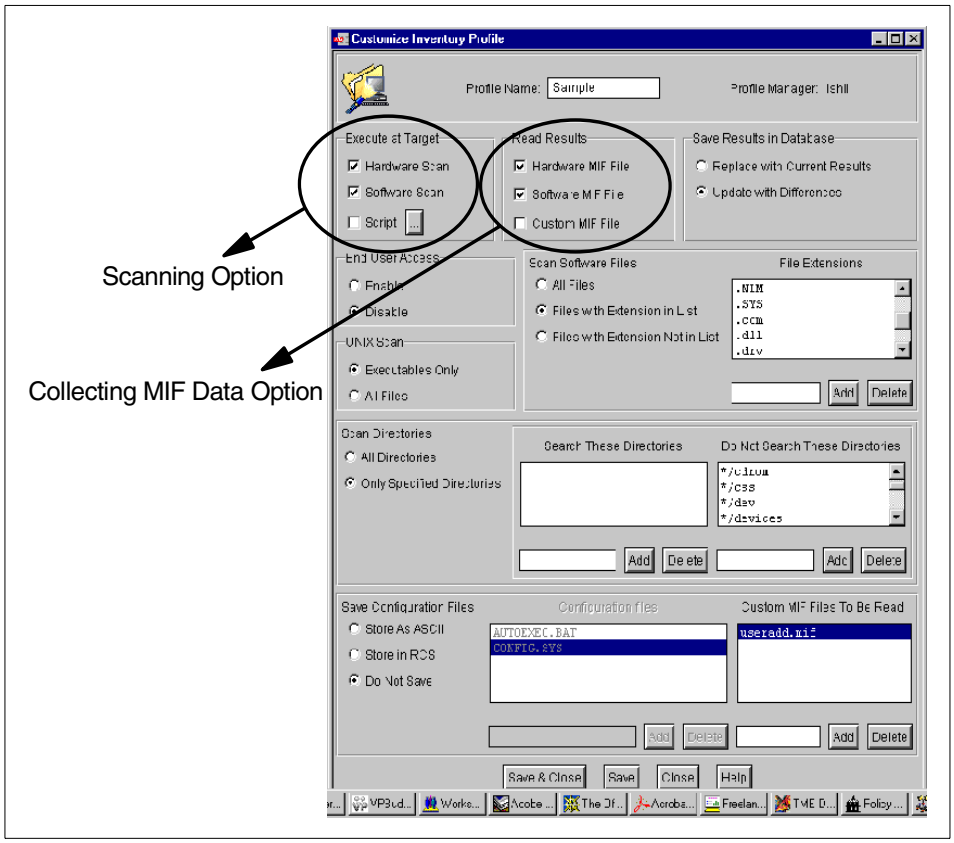


Figure 104. Setting up the scanning options in inventory profile

This configuration will improve the throughput in your Inventory management. Therefore, we strongly recommend you to configure four different types of Inventory profiles.

Note

This configuration is good not only for improving the Inventory performance but also for Inventory problem determination. If you configure a Inventory profile that performs all processes, including scanning and updating the Inventory database, it is very hard to detect the problem when the Inventory operation fails. To simplify problem determination, we strongly recommend that you divide your Inventory profile into the above four types.

7.3.9 Using Endpoint login_policy

To divide the Inventory processes into the scanning process and collecting process, you can use the Endpoint login_policy. This solution allows you to perform PULL type operation by using the Endpoint login_policy. The following figure (Figure 105) shows how the Endpoint login_policy is used for the Inventory scanning process.

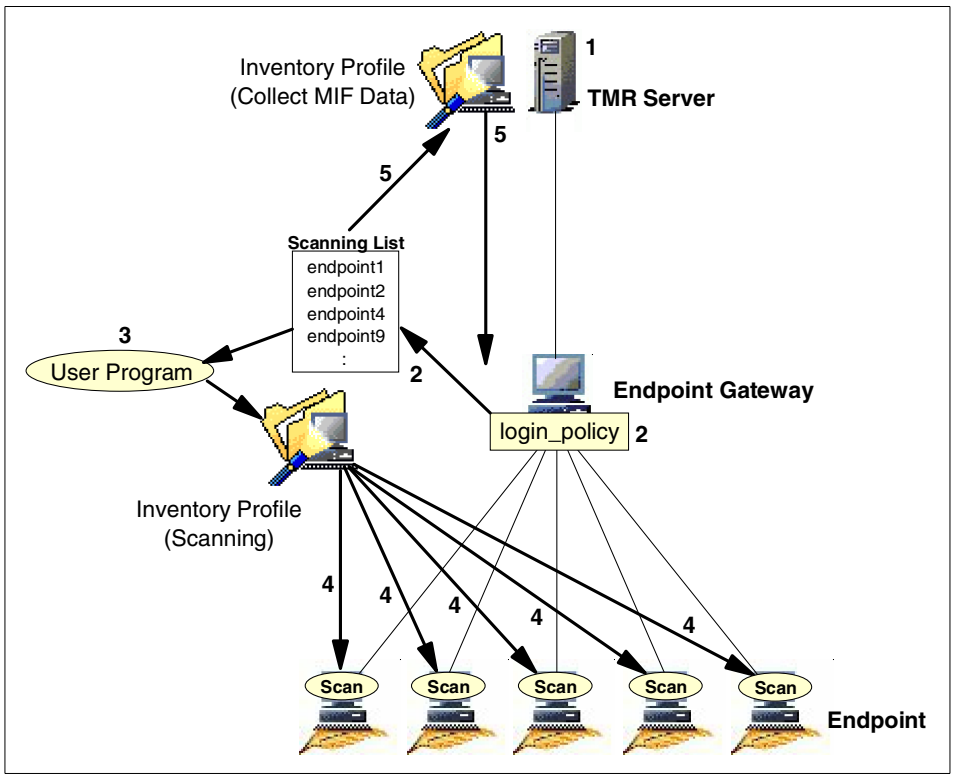


Figure 105. Using Endpoint login_policy for inventory scanning process

1. Create two Inventory profiles. One is for the scanning process and another is for collecting the MIF data process. Then, create and define the Endpoint login_policy. This login_policy registers the Endpoint label into the scanning list if the Endpoint that attempts to perform login is one of the identified scanning targets.
2. When an Endpoint attempts to perform a login to the Endpoint Gateway, the login_policy checks whether this Endpoint is one of identified scanning targets or not. If this Endpoint is an identified scanning target, the login_policy registers this Endpoint's label into the scanning list.

3. On the Endpoint Gateway, another user program is running. This user program checks the scanning list at a specified time interval. If the scanning list contains some Endpoint's labels, this user program executes the `wdistrib` command to distribute the Inventory scanning profile to the Endpoints that are listed in the scanning list.
4. As a result, the Inventory scanning profile is distributed to the Endpoints that are registered in the scanning list, and then the scanning process starts to scan the target system information and stores the information into the MIF file. If the distribution (scanning) is completed, this user program logs these scanning results and resets the scanning list.
5. When most of the scanning targets complete scanning, distribute another Inventory profile that performs collecting MIF data to the Endpoints that complete scanning (this information can be obtained by the user program and scanning list).

Normally, Inventory performs only the PUSH type operations. However, if you use the Endpoint login_policy, you can perform PULL type operations. The PULL type operations have the following advantages:

- Scanning targets are known to be available.
- The Inventory process (scanning and collecting MIF data) is spread out.
- Asynchronous operations can be performed.

As a result, the throughput of Inventory will be improved in most management environments, and the reliability of the Inventory is also improved.

In this example, we use the Endpoint login_policy, but you could also use the Windows registry if the scanning target machine is a Windows system. However, if you use the Windows registry, the Inventory scanning profile is distributed each time the Windows Endpoint machine is booted. This operation will consume a lot of CPU and network resources of the TMR server and Endpoint Gateways because the Inventory scanning profile is distributed to each target one by one. Therefore, for a large-scale environment, we do not recommend using the Windows registry to initiate a PULL type Inventory operation.

Note

In this example, we introduced the Inventory scanning process by using the scanning list. Actually, you can perform the PULL type operation without the scanning list. In this case, the login_policy will execute the `wdistrib` command each time the Endpoint attempts to log in. However, as a result, many concurrent `wdistrib` commands may be executed, and these processes may use a lot of resources of the Endpoint Gateway. Therefore, we strongly recommend you use this kind of scanning list when you implement the PULL type Inventory operations by using the login_policy especially in a large-scale management environment.

If you need to execute the `wdistrib` command from the login_policy, you also need to define the nobody user as the Tivoli Administrator login user because the login_policy is executed as nobody user process. Of course, this is not recommended from a security point of view.

In a large-scale management environment, when many Endpoints attempt to login to the Endpoint Gateway at once, the load of the Endpoint Gateway will increase. If the login_policy script needs to perform complicated processes, the load of the Endpoint Gateway may increase significantly. Thus if you plan to use the PULL type operation described in this section you should test its effect on the Endpoint Gateway before implementation.

7.3.10 Subscriber configurations

If you have a large system (such as a server) that will take a long time to complete scanning, we recommend you define these targets into the same subscriber group or create a separate Inventory profile for each of these large systems because these targets affect adversely the whole throughput of the Inventory operation.

7.4 Conclusion of inventory performance

Inventory belongs to downcall-oriented application, but it handles both downstream and upstream data. Inventory also accesses the database through RIM host; so, Inventory is a complicated application. In this chapter, we discussed the following:

- Inventory behavior
- Inventory profile configurations
- Inventory concurrent operations

- Unreachable target
- Dividing Inventory processes

There are many considerations in Inventory implementation. To improve Inventory performance, we strongly recommend you follow the information that this chapter provides.

Chapter 8. Improving Tivoli Enterprise Console performance

Tivoli Enterprise Console (TEC) is one of the availability management applications. TEC plays the role of the central collection point for alarms and events from a variety of sources. These managed sources can be Tivoli Management applications, user applications, network management applications, databases, and so on. Normally, TEC works with Tivoli Event Adapters or Distributed Monitoring in the management environment. The following figure (Figure 106) shows the relationship between Tivoli Enterprise Console (Tivoli Management application) and other layers.

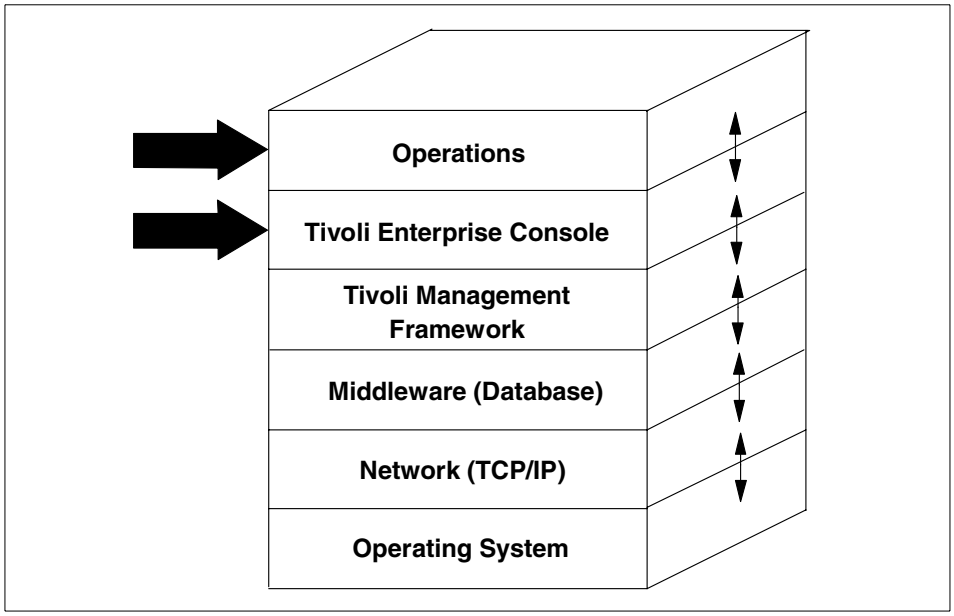


Figure 106. Tuning Tivoli Enterprise Console and its operations

This chapter introduces the important issues for improving the performance and throughput in Tivoli Enterprise Console.

8.1 Tivoli Enterprise Console internals

To improve performance and throughput in TEC, we need to know how TEC works in Tivoli Management environments. Understanding TEC behavior enables you to detect the performance bottleneck and improve its performance.

TEC is independent of the three-tiered management structure. Therefore, TEC does not issue an upcall or downcall in Tivoli Management environments. In this respect; it is different from other Tivoli Management applications, such as Software Distribution or Distributed Monitoring.

8.1.1 Tivoli Enterprise Console components

TEC can involve the following types of machines:

- TEC server
- Event console
- Event adapter
- RIM host
- RDBMS server

These components work interactively and handle each event that is sent from the managed resources. Within these components, the TEC server plays a vital role in event handling. The following processes are running on the TEC server machine, and each process has the role as follows:

tec_server The TEC server, or master process, which acts as a task master and synchronizes the other four processes in the TEC server machine.

tec_reception The reception engine, which receives events and stores them until they can be processed by the rule engine. The reception engine also performs RDBMS access.

tec_rules The rule engine, where events are evaluated against rules and actions can be performed, which are defined by the administrators.

tec_dispatch The dispatch engine, which sends events to event consoles and task requests to the task engine, also stores events in the database.

tec_task The task engine, which performs tasks as requested by the dispatch engine. These are the exec_task and exec_program in the rules.

8.1.2 Tivoli Enterprise Console process flow

In TEC operations, the most basic and typical operation is to handle a event that is sent from its managed resources. In this part of the section, we introduce the following information:

- How TEC handles a event.

- How each engine works interactively.

The following figure (Figure 107) shows the process flow in handling TEC events.

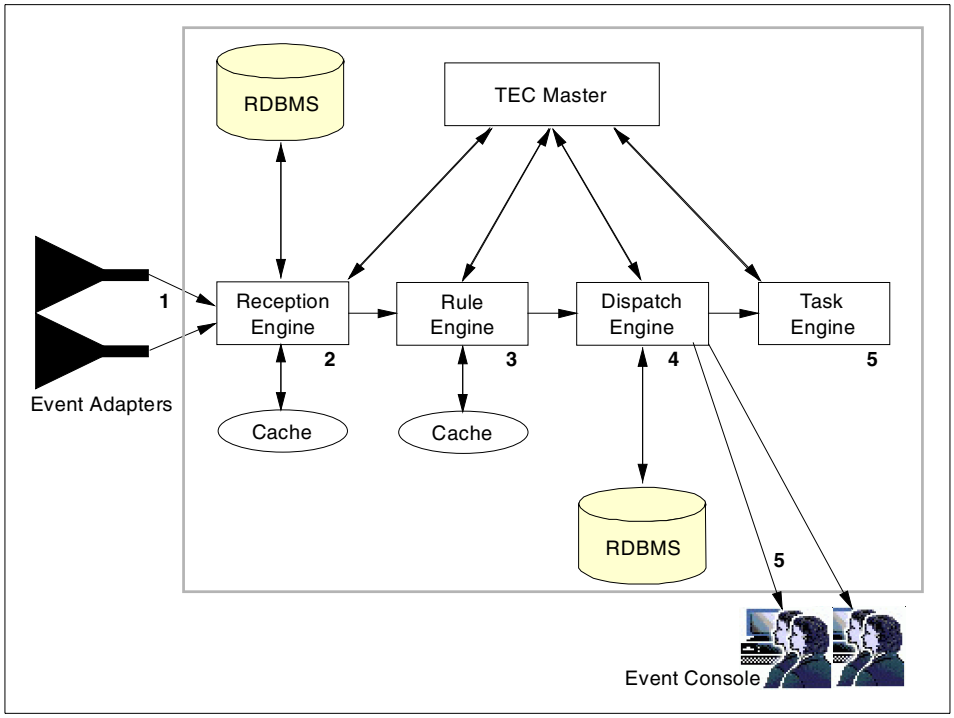


Figure 107. Tivoli Enterprise Console engines interaction

1. The Event Adapter or Distributed Monitoring (Sentry engine) emits the TEC event to the TEC server.
2. When the Reception Engine receives the TEC event, it is validated by the Reception Engine and given a unique identifier. If the Reception Engine is busy, the Reception Buffer (memory cache) maintains the event until the Reception Engine is once again available. These are called QUEUED events. If the memory also fills the events are stored in the RDBMS as WAITING events. The Reception Log (RDBMS) will record the event.
3. The event is then passed to the Rule Engine for processing. The event is processed according to the logic that is defined in the Rule Base. The Rule Engine may need access to some events that have already been processed by the Rule Engine. The Rules Cache (memory cache) holds previously processed events and is easily accessible by the Rule Engine.

The Rules Cache is useful if it becomes necessary to correlate the event, reanalyze the event, or update some slot values of the event.

4. The Rule Engine may, as a result of its analysis of an event, request the Dispatch Engine to assist in triggering several tasks or actions. The Rule Engine does not know about operators. It does not control where events go, it just sends them all to Dispatch Engine unless they are dropped. The Dispatch Engine always sends all events to all running Event Console. As the event server continues processing the event, it may create additional slots (attributes) and slot values and may also update previously defined slot values. The event is ultimately recorded in the Event Repository (RDBMS).
5. The tasks and actions that are needed to execute are handled by the Task Engine and are also displayed on the Event Console if needed.

Note

Distributed Monitoring events and other secure events (TME events) are routed via oserv, first to the TEC Master and then to the Reception Engine. Non-TME events are routed directly into the Reception Engine without going through the oserv. This means that non-TME events are faster since there is less overhead. Please refer to 8.3.8, “TME event and non-TME event” on page 264 for more information.

8.2 Detecting Tivoli Enterprise Console performance bottleneck

The Tivoli Enterprise Console consists of many components, and the interaction among these components is complicated. Therefore, several components affect the TEC performance and can be potential performance bottleneck. The following figure (Figure 108) shows the potential performance bottleneck in TEC processes.

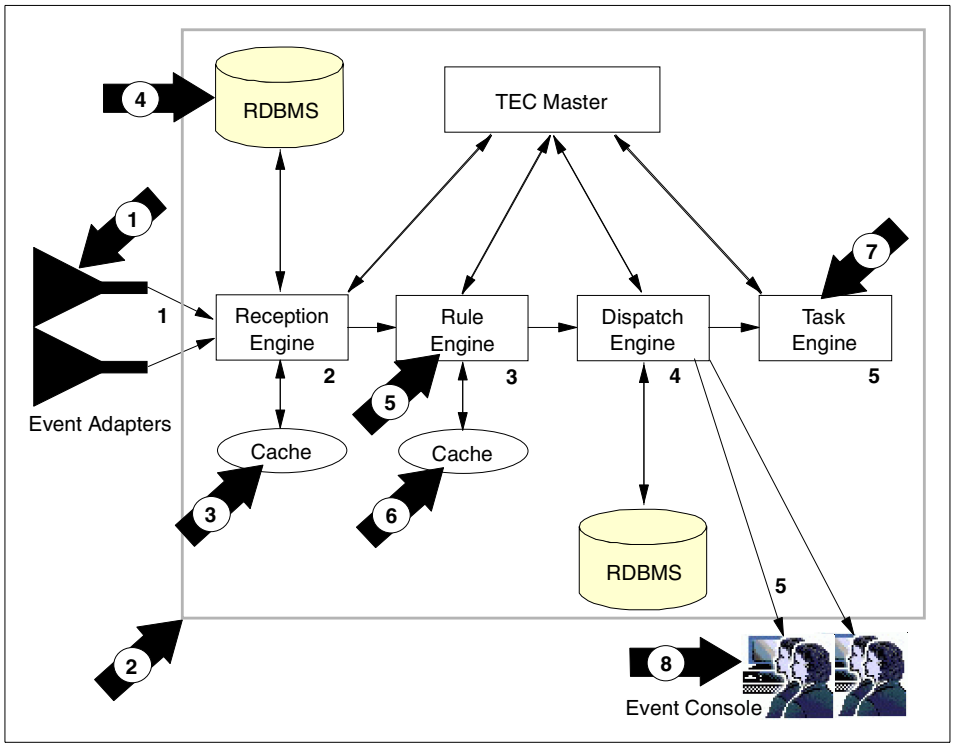


Figure 108. Tivoli Enterprise Console performance bottleneck

1. Event frequency
2. Hardware configurations and allocation (TEC server, RIM host, RDBMS server, TMR server)
3. Reception Buffer
4. Reception Log and Event Repository
5. Rule Engine (Rule Base)
6. Rules Cache
7. Task Engine (task, action)
8. Event Console

In the example process flow (Figure 108), we indicated typical performance bottlenecks of TEC. These bottlenecks can be a problem in most large-scale environments. The following are the descriptions for each performance bottleneck in TEC management environments.

Event Frequency

TEC server is a typical CPU-bound application. However, TEC cannot handle hundreds of events at once, such as Tivoli NetView for AIX, even if the TEC server is configured with high performance hardware configurations. Therefore, restricting the incoming events is the best way to improve TEC throughput. Normally, TEC can handle around 10 events per second.

TEC Component Allocation

As we mentioned, TEC consists of many components: TEC server, RIM host, RDBMS server, TMR server, and so on. The traffic among these components will be very heavy if you manage a large-scale environment.

Reception Buffer

Reception Buffer is a memory cache. Therefore, if you increase the Maximum number of events messages buffered in memory parameter, the memory utilization will also increase. This may affect the performance of the other applications on the TEC server. A large reception buffer should speed up the Reception Engine. All events must enter this buffer before being processed. If this buffer fills up, the event will have to be retrieved from the RDBMS to be put into the cache.

Reception Log

TEC logs events in the Reception Log (RDBMS). This process affects the TEC performance greatly. You can turn this option on or off. Note 8.3.6.1, “Log reception of events” on page 259 to understand the implications before turning this option off. The event repository will be concurrently updating the RDBMS, under heavy event volume, and you could get contention in the database accessing processes.

Rule Engine

The performance of the Rule Engine depends on the rules that you create on the TEC server. If you specify all_instances in your Rule Base, it may affect Rule Engine performance.

Rules Cache	Rules Cache is used by Rule Engine to store the events for future processing. It also depends on how your Rule Base works (for example, correlating or filtering) in your TEC environment. You should estimate carefully the size of the Rules Cache.
Task Engine	The heavy tasks or actions that you define may affect the TEC performance. When you define tasks or actions, you should consider this effect or performance.
Event Console	To improve TEC performance, we recommend you use the minimum number of Event Consoles at once and stagger starting them as TEC Dispatch is single threaded and temporarily stops processing events while opening a Console.

In the following sections, we introduce how you should configure and optimize TEC management environments.

8.3 Tivoli Enterprise Console performance improving strategy

In the previous section, we introduced some potential performance bottlenecks of TEC. This section introduces some approaches and directions to improve the performance and throughput of TEC. We also provide information about how you should configure and optimize TEC in a large-scale management environment.

Note

Normally, you should consider the following things when you create a Rule Base.

- Limit the use of all_instances or all_duplicates.
- Use drop_received_event when possible to reduce the number of events in the cache.
- When matching all events, use "of_class _class" instead of "of_class 'EVENT'" for a more efficiently compiled Rule Base.
- Use commit_action, commit_rule, commit_set often to only process those rules necessary for each event.

In this chapter, we will not discuss how you should create a Rule Base to improve the TEC performance. Your Rule Base may affect the TEC performance greatly; however, each Rule Base depends on your management operations or environment. Therefore, we do not take TEC Rules into account in this redbook.

8.3.1 Performance improving process flow

The following figure (Figure 109) shows the approach for improving the performance and throughput of TEC. When you configure or tune TEC in your management environment, we recommend you to use the following process flow chart.

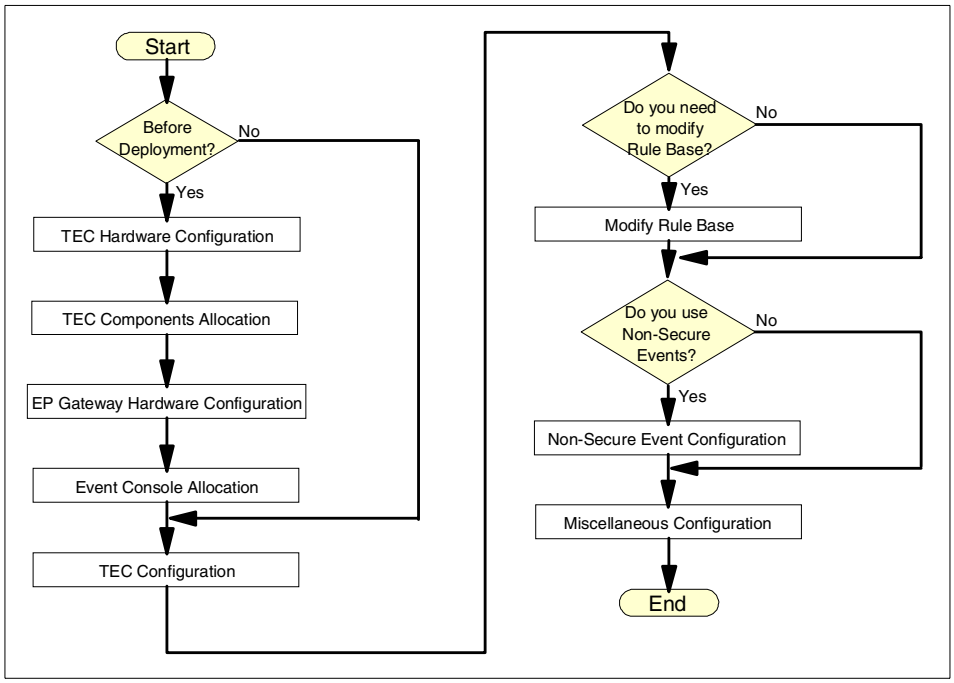


Figure 109. Tivoli Enterprise Console performance improvement process flow

In the following sections, we will introduce detailed information about how to configure and tune your TEC in a large-scale management environment.

8.3.2 TEC Components allocation

TEC consists of many components. The optimum allocation of these TEC components improve the TEC performance and throughput. In this section, we introduce some examples of TEC component allocation to improve TEC performance.

8.3.2.1 Configuring TEC server with RIM Host and RDBMS server

In most Tivoli management environments, each component of TEC is configured on a different machine. However, on a multiprocessor machine with sufficient CPU, the TEC server has higher throughput if the RDBMS is configured locally rather than configured remotely, where configuring locally means that the RDBMS server and RIM host are configured on the same machine as the TEC server. The following figure (Figure 110) shows the difference between each configuration.

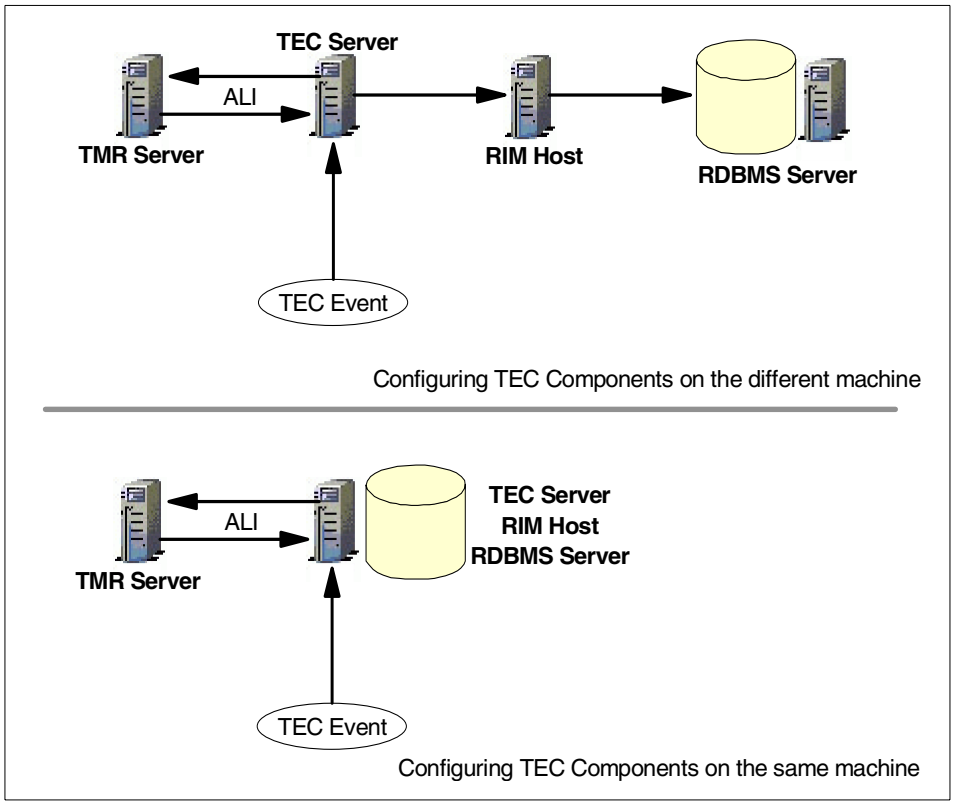


Figure 110. Tivoli Enterprise Console components allocation

As we mentioned, the TEC server is a CPU-bound application. However, if the TEC server is configured with a multiprocessor machine that has sufficient CPU performance, the throughput of event handling is improved, and then the network will be the bottleneck of its throughput.

If each component, TEC server, RIM host, and RDBMS server is configured on the same machine, then the multiprocessor will improve the overall TEC performance and throughput, because the operations related to the RDBMS are complicated (performed via the RIM host, not directly) and heavy.

8.3.2.2 Creating a separate TMR for the TEC server

To optimize a TEC management environment, a separate TMR may be the best solution. As we mentioned, TEC requires many operations that are performed by other processes than the TEC server, such as database access or authentication processes. These processes will affect the TEC performance greatly.

In the previous section, we introduced the configuration that shows that the TEC server, RIM host, and RDBMS server are installed on the same machine. However, the TMR server is still handling many requests from its managed resources other than TEC components. To avoid this situation and improve the TEC performance significantly, you can create a TMR that is used only for the TEC components. The following figure (Figure 111) shows its configurations. In this case, each TMR should be configured as an interconnected TMR.

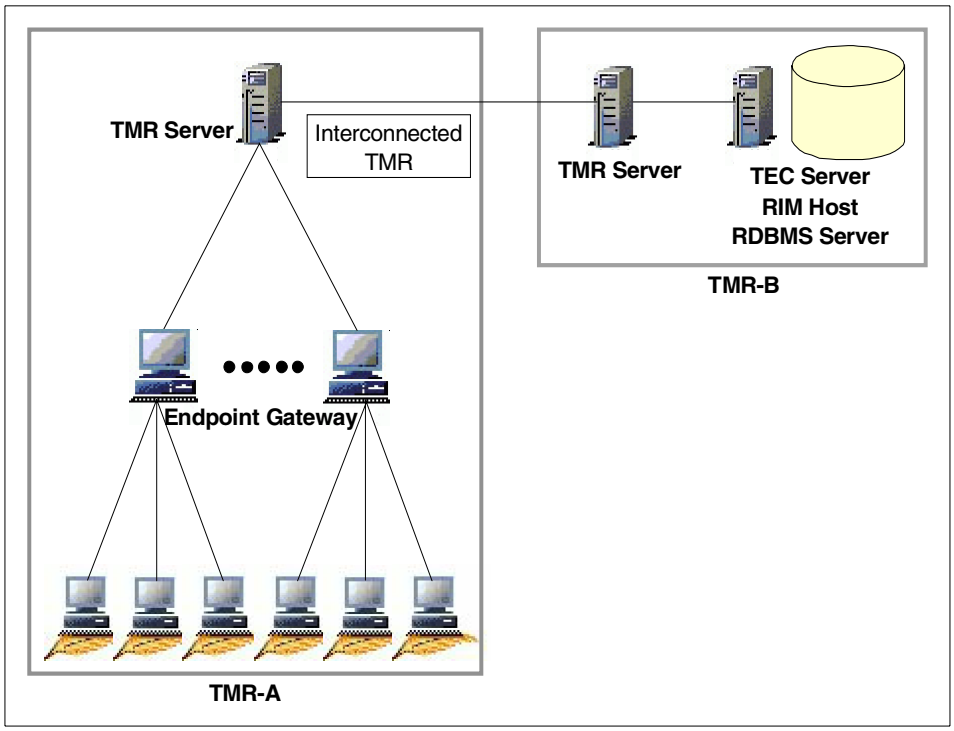


Figure 111. Creating TMR for TEC components

As you can see in this example, TMR-B is created for the TEC operations. However, all managed resources belong to TMR-A, and the TMR server in TMR-A handles requests that are sent from its managed resources. Therefore, the TMR server in TMR-B can concentrate on handling the requests from the TEC components. As a result, this configuration will improve the TEC performance and throughput in most environments.

Note

When you implement this configuration in your environment, there is another advantage. If some managed resources send non-TME (non-secure) TEC events to the TEC server by using the `postemsg` command or by using non-TME Event Adapters in your managed environment, these non-TME (non-secure) TEC events will be processed even if the TMR server in TMR-A is not available. Therefore, in this case, this configuration will improve not only the TEC performance but also its reliability. We will talk about this subject later. Please refer to “TME event and non-TME event” on page 264 for more information.

8.3.3 Hardware configurations of TEC server

As we mentioned, TEC is a CPU-bound application. In prior versions of TEC, implementing the TEC server on high performance hardware did not improve performance and throughput significantly. However, Version 3.6 of TEC is different from the prior version of TEC. If the TEC server is implemented on a high performance CPU machine, you can expect a performance improvement.

If you are in the pre-sales phase, we strongly recommend you configure the TEC server on a high performance machine. The following are considerations for each hardware configuration:

Processor It is necessary to say it is CPU-bound application again. Configuring the TEC server with a fast CPU should improve your TEC performance. If possible, a multiprocessor machine is the best for the TEC server (if it has more than a four-way processor, it is the best, because TEC consists of five processes). A multiprocessor machine will improve TEC performance dramatically.

Memory Memory usage depends on how you configure the TEC parameters, how your Rule Base works, and what other applications run on the same machine. If you have a machine that is used only for the TEC server, normally 256 MB memory size will be reasonable. However, if your Rule Base performs deep correlations, your TEC server machine may need more memory. Please refer to “TEC parameter configurations” on page 258 for more information.

8.3.4 Managed resource allocation

In the 8.3.2, “TEC Components allocation” on page 251, we described how to allocate each TEC component. Because Version 3.6 of Tivoli provides a new three-tiered management structure, you should consider the allocation of the managed resource as well when implementing a TEC management environment. The following figure (Figure 112) shows the difference between the prior TMR structure and the three-tiered structure.

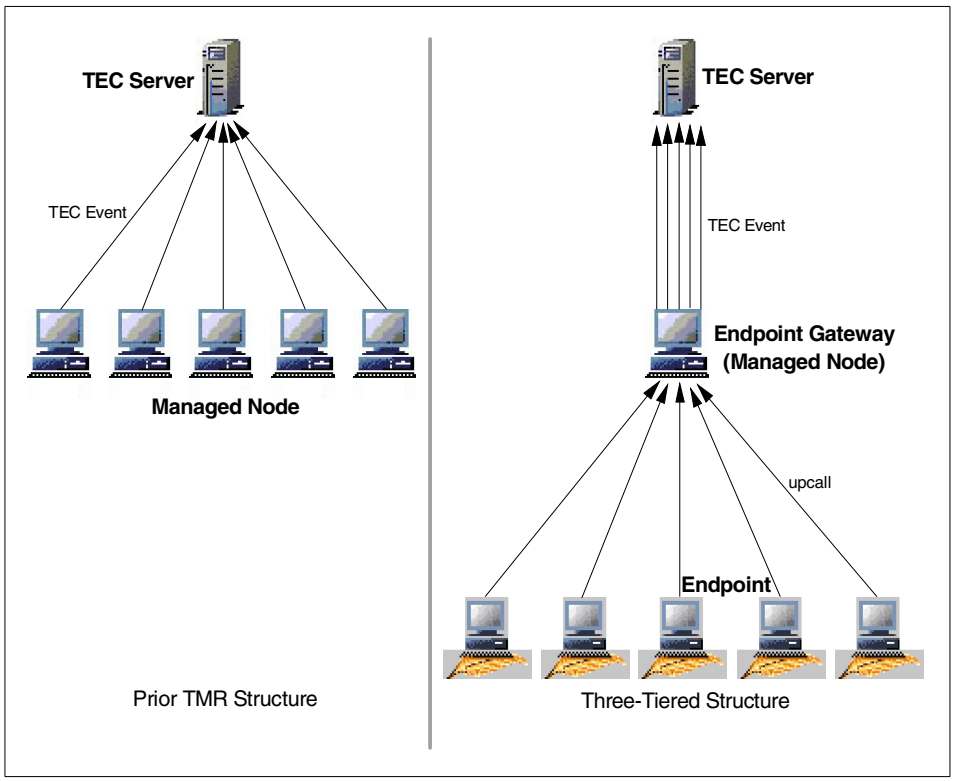


Figure 112. Managing structure and TEC events

As you can see, from a TEC point of view, there is no difference between these two management structures. TEC just receives the TEC event and processes them. However, let us look at each event source. In the prior TMR structure, each managed resource sends an event to the TEC server directly. In the three-tiered structure, the Endpoint Gateway sends all events from its Endpoints to the TEC server. In other words, the TEC server receives the events only from Endpoint Gateways in the three-tiered management structure. Each Endpoint sends a TEC event by using an upcall, then the Endpoint Gateway actually invokes the appropriate methods and sends the

TEC event to the TEC server as the proxy of all Endpoints that are managed by the Endpoint Gateway.

What does this mean? The load of each Endpoint Gateway is increased in the three-tiered management structure. Therefore, when you create an Endpoint Gateway, you should implement the Endpoint Gateway with the proper configurations. The following items should be considered carefully.

- CPU
- Memory
- How many Endpoints a single Endpoint Gateway manages.

If a single Endpoint Gateway has to manage many Endpoints, you should try to reduce the TEC events by configuring only critical events on each managed resource (Endpoint) to be sent. Please refer to Chapter 6, “Improving distributed monitoring performance” on page 193 for more information.

8.3.5 Reducing TEC events

Before tuning your TEC environment, you should consider how to reduce the number of TEC events in your management environment. This can be a good solution to improve the TEC throughput. To reduce the number of TEC events, there are two typical approaches:

- Filter events at managed resources by appropriately configuring the Sentry monitor or Event Adapter.
- Filter events at the server by using the Rule Base.

The following figure (Figure 113) shows examples for filtering your TEC events.

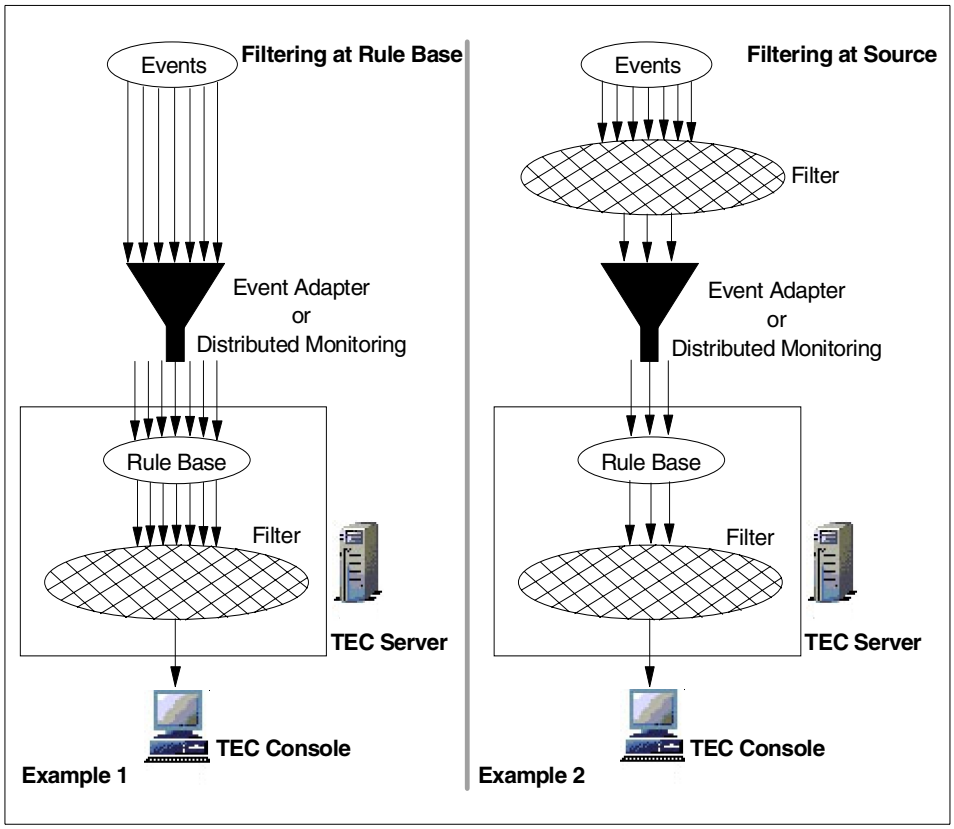


Figure 113. Filtering TEC events

As you can see, each example attempts to reduce TEC events but uses different ways. In example 1, most of events are filtered at the TEC server by the Rule Base. However, in example 2, only selected conditions generate the event, and only selected events are sent to the TEC server. These events are also filtered at the server by the Rule Base.

In a large-scale environment, the filtering process that is performed before generating TEC events is a cheaper operation than the filtering process at the TEC server because it does not use network and TEC server resources. In other words, filtering events using the Rule Engine is more expensive than filtering at the source.

In the following sections, we introduce how to configure and customize the TEC server. However, again, we strongly recommend you filter your events before generating events. This must be done to have an effective TEC server.

8.3.6 TEC parameter configurations

TEC has configuration parameters that can affect the TEC performance greatly. To improve the performance and throughput of TEC, you need to understand these parameters and configure them properly in your TEC management environment. The following figure (Figure 114) shows the configurable parameters of the TEC server.

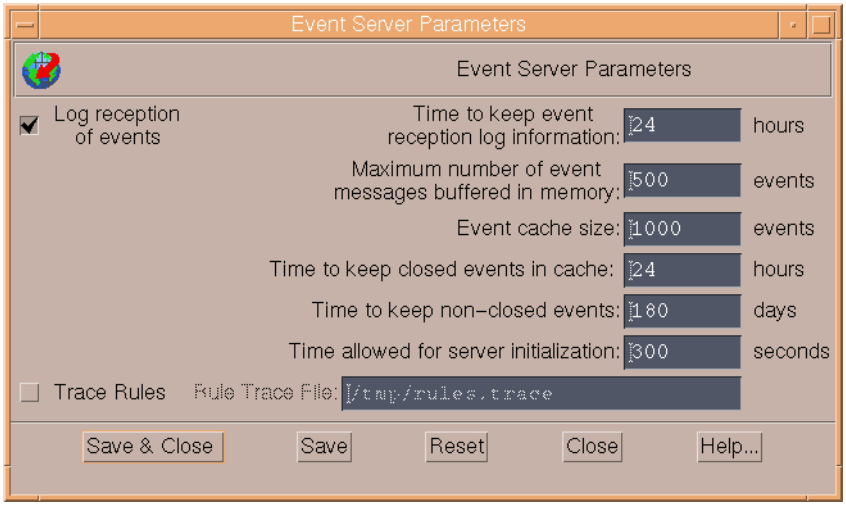


Figure 114. TEC server parameters

The following gives the meaning of each TEC server parameter.

Log reception of events
This option specifies whether the event that is received by the Reception Engine is logged into Reception Log (RDBMS) or not. By default, this option is turned on.

Time to keep reception log information
This parameter specifies the amount of time to keep Reception Log information. The events that are older than the time that is specified by this parameter will be discarded. This parameter is available only if the Log reception of events option is enabled. By default, this parameter is set to 24 hours.

Note

This setting is not automatically enforced, but rather is used by the Clean_Database task in the TEC Task library. This task may be run directly scheduled as a job.

Maximum number of events messages buffered in memory
This parameter specifies the number of incoming events to store in the Reception Buffer (memory cache) of the Reception Engine. By default, this parameter is set to 500 events.

Event cache size
This parameter specifies the maximum number of events to keep in the Rules Cache. The events that are stored in the Rules Cache can be correlated with incoming events to see if the status of any events should be updated. By default, this parameter is set to 1000 events.

Time to keep closed events in cache
This parameter specifies the amount of time to keep a closed event in the Rules Cache of the Rule Engine. By default, this parameter is set to 24 hours.

Time to keep non-closed events in cache
This parameter specifies the amount of time to keep events that are not closed yet in the Rules Cache of the Rule Engine. By default, this parameter is set to 180 days.

Time allowed for server initialization
This parameter specifies the amount of time to allow TEC to synchronize its processes. By default, this parameter is set to 300 seconds.

Trace rules
This option enables tracing in Rule Engine. By default, this option is turned off.

The following sections introduce how you should configure these TEC server parameters to improve the performance and throughput in your environment.

8.3.6.1 Log reception of events

The Reception Engine needs this option to track which events need to be recovered in case of a system crash. It will also show which events do not parse correctly and thus are not passed on to the Rule Engine nor the TEC Consoles. Updating the RDBMS with these events, especially if they need to be updated several times to change their status from WAITING to QUEUED

and then QUEUED to PROCESSED affects the RDBMS and TEC performance. You can greatly improve TEC performance by turning this option off. However, do not forget that if you turn this option off, no unprocessed events can be recovered when the TEC server system is unexpectedly stopped. Before turning this option off, you should consider these factors carefully.

8.3.6.2 Time to keep event reception log information

This parameter does not affect TEC performance much but affects the database size for Reception Log (RDBMS). Clearing of the Reception Log is not done automatically. This parameter is used by the Clean_Database task, which must be run by the user. This parameter is available only when you turn the Log reception of events option on. The Clean_Database task should be run in off-peak times to minimize database contention.

8.3.6.3 Maximum number of event messages buffered in memory

This parameter specifies the size of the Reception Buffer (memory cache) that is used by the Reception Engine. As we mentioned, when the Reception Buffer overflows, it affects not only memory utilization but also the frequency of database access. As a result, the load of the Reception Engine becomes heavy, and it makes the TEC performance worse. Increasing this parameter means the amount of memory that other applications can use will decrease. When you estimate the size of the Reception Buffer, you should at least consider the following factors:

- How often events are generated in your environment.
- TEC server memory size.
- How many events the Rule Engine can handle in your environment.

Normally, the average size of one event will be around 2 KB; so the Reception Buffer will use the following amount of memory (KB):

```
Reception Buffer = <The value of this parameter> * 2 KB
```

You should then consider the relationship between Reception Buffer size and the above factors and decide how to set this parameter.

Note

The size of each TEC event depends on which type of TEC event the managed resource generates. Normally, the size of each TEC event is approximately 2 KB, but depending on the event type it can be up to 4 KB.

We recommend you check the size of each TEC event and estimate the average size in your management environment. Understanding average event size should help you to manage your environment efficiently.

We recommend that you use the default value (500 events) unless the Reception Buffer overflows repeatedly. In a large-scale environment, before increasing this parameter, you should try to reduce the events that are sent from your managed resources again. Does each managed resource send only important events? Do you really need these events to perform your management? You should consider this again carefully and reconfigure if needed. You also need to know if buffer has overflowed when you increase this parameter. When buffer overflow happens, events will be marked as WAITING in the Reception Log.

If events are defined properly, and each managed resource sends only important events, then you can consider increasing this parameter. However, before increasing, you should check whether the TEC server has enough memory for this parameter to be increased or not.

8.3.6.4 Event cache size

This parameter specifies how many events the Rules Cache can store. This size depends on how you define your Rule Base. The events that are stored in the Rules Cache can be correlated with incoming events by the Rule Engine. However, if you increase this parameter, it will make the load of the Rule Engine heavy.

The design of the Rule Base depends on how you manage your Tivoli Management environment using TEC. In a large-scale environment, we recommend that you do not correlate against too large a history of events (the cache), especially, if you use all_instances and all_duplicates in your Rule Base. The all_instances and all_duplicates templates cause the Rule Engine to attempt to correlate all events in the Rules Cache with the incoming event. This makes the load of Rule Engine heavy and affects the performance and throughput of the TEC process. Be careful not to over-use all_instances and all_duplicates.

The default value is recommended. Before increasing this parameter, you should attempt to reduce the number of events in your management environment.

Calculating the proper size of the Rules Cache is very important. When you estimate the size of the Rules Cache in your environment, you should consider the following factors:

- How many events are generated in your environment.
- How long you need the events for correlation.

Normally, the average size of one event will be around 2 KB; so, the Rules Cache can reach the following size (KB):

```
Rules Cache = <Average number of events per day> * <Event storing days> * 2 KB
```

If most monitored applications do not generate events during the weekend, only the events that are generated on weekdays should be used for estimating the average number of events per day.

8.3.6.5 Time to keep closed events in cache

This parameter specifies how long a closed event is stored in the Rules Cache. In the normal style of problem management, the closed events are used for capacity planning or health checking of your systems. When Rules Cache size is exceeded, the oldest events are discarded first. This parameter clears out closed events and keeps the Rules Cache here. If you would like to correlate many closed events with an incoming event, you can do this by increasing this parameter.

Although this parameter setting is site dependent, we recommend you use the default value first or set this parameter to a lower value especially in a large-scale environment.

8.3.6.6 Time to keep non-closed events

This parameter specifies how long non-closed events are stored in the Rules Cache. As we mentioned, when Rules Cache size is exceeded, the oldest events are discarded first to accommodate the newer events.

Normally, in problem management, the non-closed events are very important to recognize and analyze the problem in your management environment. Therefore, you must consider how to keep these non-closed events in the Rules Cache.

The default value is recommended. If Rules Cache size is often exceeded, you should decrease the Time to keep closed events in cache parameter first.

If Rules Cache is still short, you should decrease this parameter to discard the older events.

8.3.6.7 Time allowed for server initialization

This parameter specifies the amount of time to allow TEC to synchronize its processes. If TEC cannot synchronize its processes within the time that is defined in this parameter, the initialization process will fail.

The default value is recommended. In a large-scale environment, TEC initialization may take longer. If TEC initialization sometimes fails in your environment, you should increase this parameter. These is no benefit in decreasing this parameter. Use the default value first then increase it if needed.

8.3.6.8 Trace rules

This option enables tracing in the Rule Engine. This option should be turned off unless you are performing a test or debugging. This option is useful when you have any problems in your Rule Base. If you turn this option on, TEC performance is impacted.

Note

To manage a large-scale environment efficiently, you have to try to configure your management environment simply even if you give up managing all managed resources with the same level of granularity, or very detailed event management.

You should define priority of your managed resources and make your management simple. For example, some very important managed resources, such as application servers, may send many TEC events that tell you detailed conditions, but other managed resources send only critical TEC events.

Before implementing a Tivoli Management environment, you must consider this carefully. It is the essence in managing a large-scale environment successfully. Hierarchical TEC servers are also sometimes implemented to balance the event load.

8.3.7 TEC Event Console

Initializing an event console is a heavy operation for the Dispatch Engine because it makes many database access calls. The events that are sent to that Event Console are stored in the Event Repository (RDBMS). The TEC Event Console has configuration options to limit the number of events by time

for closed events (Display Closed Messages) and open events (Display Open Messages). These two parameters affect the Event Console process and initialization performance, and are a way to tune the Event Console.

The Dispatch Engine sends only the events that the Event Console wants based on the Display Open Messages and Display Closed Messages parameters. We strongly recommend you initialize only minimum consoles and limit the number of events by using the Display Open Messages and Display Closed Messages parameters. In addition, we also recommend reducing the events stored in the Event Repository, so there will be fewer events to read from the RDBMS at Event Console initialization.

You should consider allocation (how many and where) of Event Console as well. TEC Event Console software can be installed on different Managed Nodes in your TMR. To spread this workload, we recommend you install your Event Consoles on separate Managed Nodes. You can use a Managed Node as a TEC Event Console machine. This off-loads some of the work from the TEC server and improves the TEC performance.

Also, event groups affect Event Console performance. The more groups and filters a Event Console has, the more work it must do to filter each event.

8.3.8 TME event and non-TME event

TEC events are typically generated by the Event Adapter or Distributed Monitoring and sent to the TEC server. In a TEC management environment, there are two types of the TEC events. One is the TME (secure) event, and the other is a non-TME (non-secure) event. The easiest way to generate these events is to use the commands `wpostemsg` and `postemsg`. This way is especially recommended when you want to generate a large amount of events, for example, in your testing. The following are descriptions about each type of event.

wpostemsg Generates TME events. This means events that use the Tivoli Framework services. This command is only available on machines that belong to the TMR. For example, TMR server, Managed Nodes, or Endpoints.

postemsg Generates non-TME events. These events do not use the Tivoli Framework services. These events can be generated from any machine that has a TCP/IP protocol interface and the `postemsg` command installed.

8.3.8.1 Performance of TME events and non-TME events

A TME event is generated by using Tivoli Framework services including authentication or object request broker (ORB), and goes through TEC Master to Reception Engine. On the other hand, the non-TME event does not use the authentication process, so they go directly to the Reception Engine. As a result, the speed of the non-TME event processing is faster than the speed of TME event processing.

Note

In the Event Adapter configurations, if you configure the `ConnectionMode` keyword as follows:

```
ConnectionMode=connection_oriented
```

for TME (secure) Event Adapter, the authentication process is only done once, not for every event. In this case, TME Event Adapters may be faster than non-TME (non-secure), although non-TME Event Adapters may also be configured as `connection_oriented`.

Whether you use TME events or non-TME events depends on your management environment. The next section introduces an example of using non-TME events.

8.3.8.2 Example of using non-TME events

If you use temporary notification mechanism, the non-TME events are useful. The following figure (Figure 115) shows an example of using the non-TME events.

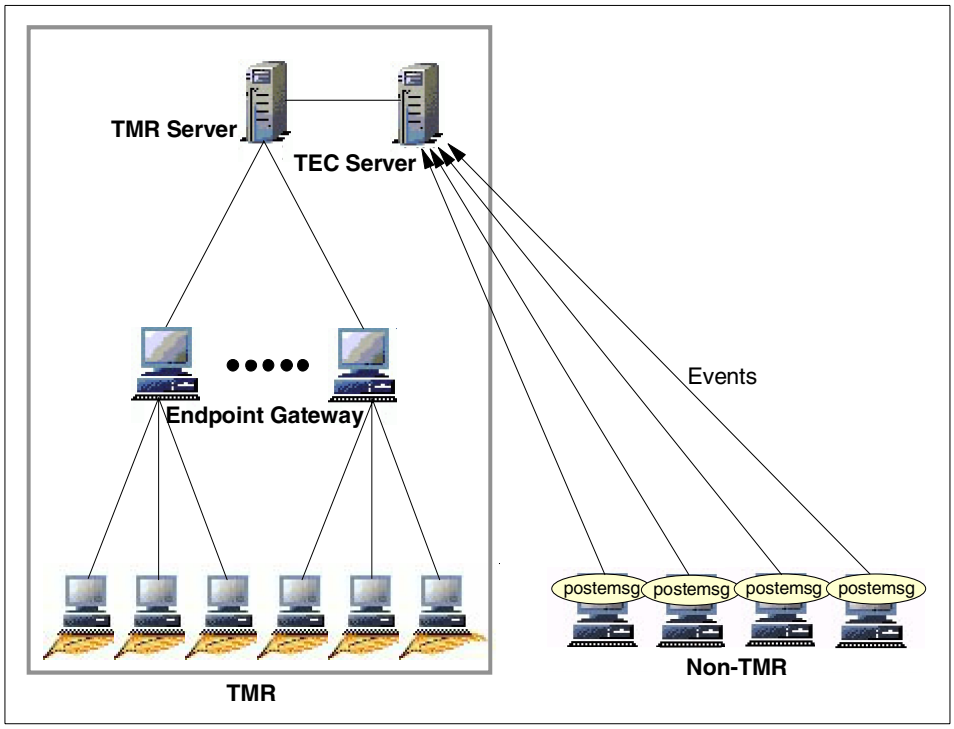


Figure 115. Using non-TME event in your management environment

As you can see, each machine that emits a TEC event by using the `postemsg` command is not part of the TMR. Please refer to the *TME 10 TEC Reference Guide 3.6*, SC31-8505, for more information about the `postemsg` command. Below is a useful scenario of using the non-TME events.

Example scenario

When you plan to install a software package on a machine that is not part of any TMR and need to check its installation status, the `postemsg` command will be useful. In this case, if you put the `postemsg` command into the installation batch program, you can detect the machines that fail to install the software and can also find which machines complete the installation by checking the TEC events that are sent by the `postemsg` command in the installation batch program.

The non-TME events are very useful when you use the event to check on one specific process, such as in this scenario. You can use TEC events from any batch programs or shell scripts.

8.3.9 Nagle algorithm

The Nagle algorithm is defined as RFC 896 in 1984. The Nagle algorithm is designed to resolve network congestion especially in Wide Area Network (WAN) environments. For example, it is famous as the optimum algorithm for the telnet session in TCP/IP protocol.

In the prior version of TEC, the Nagle algorithm is implemented, and each TEC operation is processed based on the Nagle algorithm. However, as a result, the following problems occurred in the TEC operations:

- Event handling speed is always slow even if the CPU utilization is low.
- TEC can handle up to around five events per second even if the TEC server is configured on a fast processor machine.
- Throughput of event handling does not depend on whether the Rule Base is complicated or not. The throughput is almost always the same.

This algorithm is excellent for some operations, such as telnet; however, it was not good for TEC operations. For example, the Nagle algorithm affects the following TEC operations:

- RDBMS sessions.
- Interprocess communication within the TEC server (especially, tec_dispatch, tec_reception).

In Version 3.6 of TEC implementation, this algorithm is disabled (`NO_DELAY`). It automatically improved the TEC performance and throughput and now event handling speeds will be improved if events are processed by a fast processor.

8.4 Conclusion of TEC performance

TEC plays the role of event collector in the TMR and it can receive TME events from Distributed Monitoring and TEC Event Adapters. TEC has many components and each process is complicated. In this chapter, we discussed the following:

- TEC behavior
- TEC components allocation
- TEC parameters
- TEC event configurations

TEC has many tunable parameters, and these should be considered and configured carefully. To improve TEC performance, we strongly recommend you follow the information this chapter provides.

Chapter 9. Improving Remote Control performance

Tivoli Remote Control is used to take control of any Intel-based system in the Tivoli Management environment from another Intel-based machine in the Tivoli Management environment.

Normally, Remote Control is used as a help desk support tool. Version 3.6 of Remote Control supports the TMA; so, Remote Control enables the help desk operators to control and monitor the Intel-based Endpoint system remotely including a mouse operation. The following figure (Figure 116) shows the relationship between Remote Control (Tivoli Management application) and other layers.

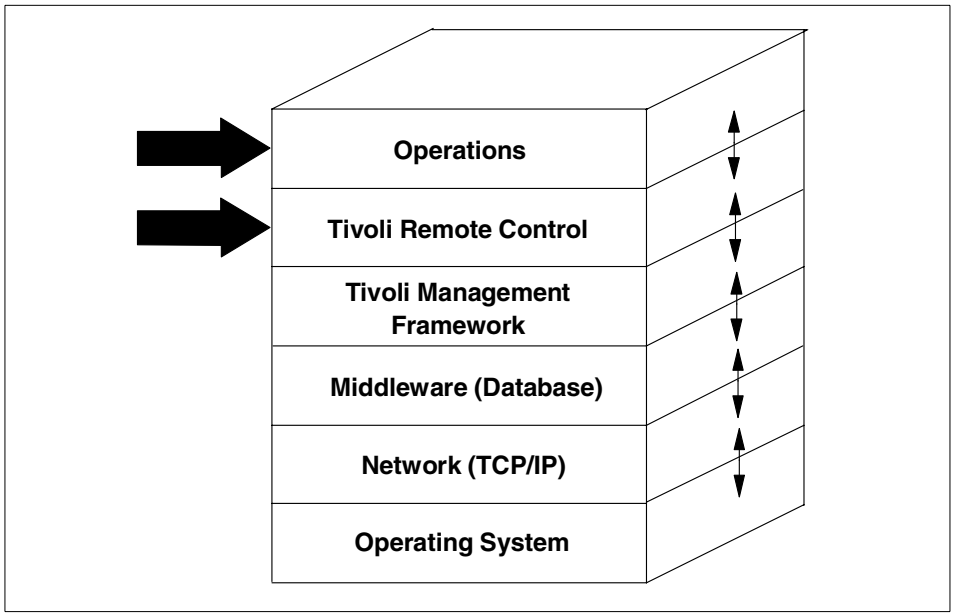


Figure 116. Tuning Remote Control and its operations

In this chapter, we introduce what are the important issues for improving performance and throughput in a Remote Control environment.

Note

The information provided in this chapter is based on Version 3.6 of Remote Control and Patch 3.6-RCL-0002. Version 3.6.5 of Remote Control has improved Remote Control performance greatly. We will introduce the information about Version 3.6.5 of Remote Control in 9.4, “Version 3.6.5 of Remote Control performance improvement” on page 296. To pay attention to the Version 3.6.5 important information, we have also put some notes into this chapter.

9.1 Remote Control internals

Remote Control is not profile based application. Remote Control consists of the following four components:

- Remote Control Server
- Remote Control Gateway
- Remote Control Controller
- Remote Control Target

The Remote Control Server allows you to start the Remote Control session from the Tivoli Desktop. It must be installed on a Managed Node.

The Remote Control Gateway is an optional machine that handles the communications between the Remote Control Controller and Targets. It must be installed on a Managed Node and work with the Remote Control Server.

Remote Control allows you not only to monitor the target PC system but also operate the target PC system. Remote Control Target is a machine that the Remote Control Controller monitors or acts upon. Remote Control Target supports the following platforms:

- Windows NT
- Windows 3.1x, Windows for Workgroup 3.11, Windows 95
- OS/2

9.1.1 Remote Control and methods

Monitoring or operating the target system is performed by invoking some methods in your management environment. In this part of the section, we introduce the following information:

- Which methods Remote Control invokes.

- Where Remote Control invokes each method.
- How Remote Control issues a downcall.

In the Remote Control environment, the Remote Control Gateway component is optional. Therefore, there are two major configurations for a Remote Control environment. One is to configure the Remote Control Gateway and another is not to configure the Remote Control Gateway. In this part of the section, we also introduce the typical configurations of a Remote Control environment and how it works in each case.

9.1.1.1 Remote Control process flow (without Gateway)

The following figure (Figure 117) shows the detailed process flow in a Remote Control environment. In this case, the Remote Control Gateway is not configured. In the next section, “Remote Control process flow (with Gateway)” on page 272, we introduce how Remote Control works when the Remote Control Gateway is configured.

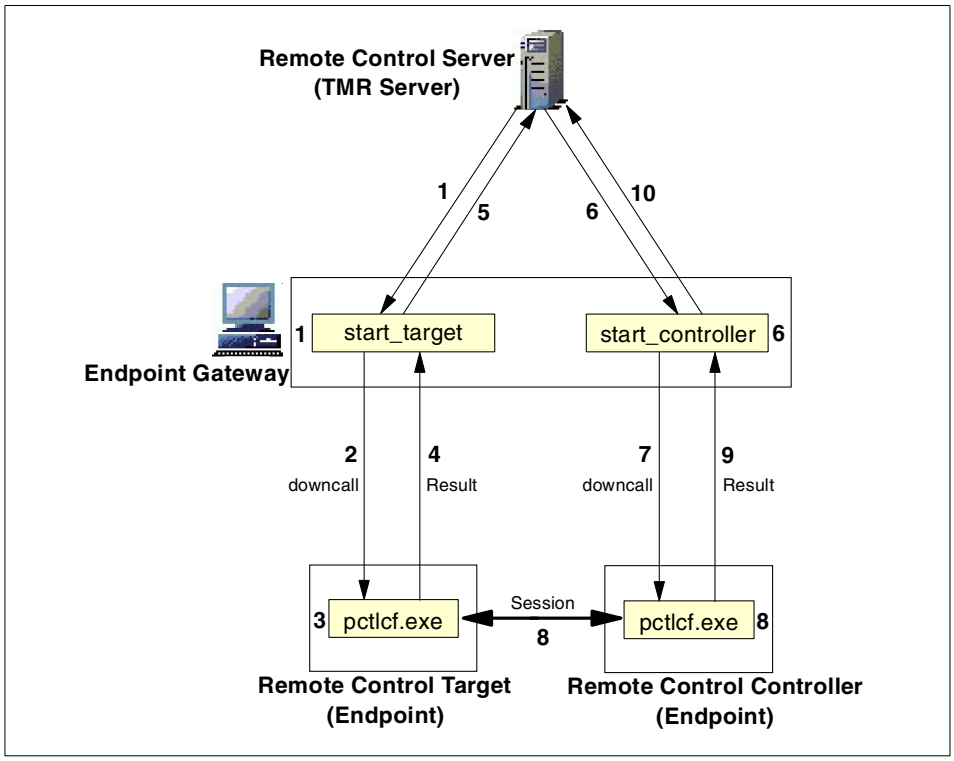


Figure 117. Remote Control and its methods

1. When the Remote Control session starts, the Remote Control server contacts the Endpoint Gateway that the Endpoint Controller has logged into and retrieves its properties information. The Remote Control server also contacts the Endpoint Gateway that the Endpoint Target has logged into and then retrieves its properties information. After the completion of the retrieving the properties information, the Remote Control Server invokes the `start_target` method on the Endpoint Gateway remotely.
2. The Endpoint Gateway issues a downcall to invoke `pctlcf.exe` on the Endpoint Target.
3. Then `pctlcf.exe` is executed on the Endpoint Target.
4. The Endpoint Target returns the result to the Endpoint Gateway.
5. The Endpoint Gateway also returns the result of the `start_target` method invocation to the Remote Control server.
6. Then the Remote Control server invokes the `start_controller` method on the Endpoint Gateway remotely.
7. The Endpoint Gateway issues a downcall to invoke `pctlcf.exe` on the Endpoint Controller.
8. Then `pctlcf.exe` is executed on the Endpoint Controller, and the communication between the Remote Control Target and Remote Control Controller is established.
9. The Endpoint Controller returns the result to the Endpoint Gateway.
10. The Endpoint Gateway also returns the result of the `start_controller` method invocation to the Remote Control server.

9.1.1.2 Remote Control process flow (with Gateway)

Remote Control Gateway is optional in a Remote Control management environment. If you use Remote Control in the firewall environment, you may need the Remote Control Gateway. The Remote Control Gateway is required in the firewall environment.

When you configure the Remote Control Gateway in your environment, the behavior of Remote Control is a little different from when the Remote Control Gateway is not configured in your environment. The following figure (Figure 118) shows the detailed process flow when you configure the Remote Control Gateway in your environment.

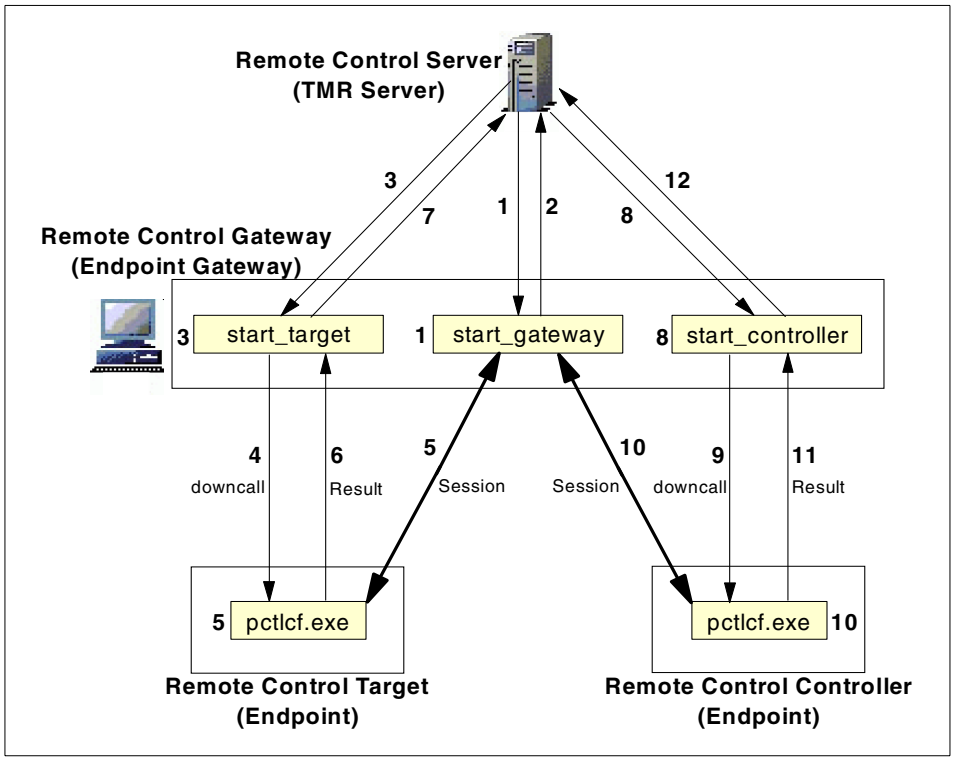


Figure 118. Remote Control and its methods (with Remote Control Gateway)

1. When the Remote Control session starts, the Remote Control server contacts the Endpoint Gateway that the Endpoint Controller has logged into and retrieves its properties information. The Remote Control Server also contacts the Endpoint Gateway that the Endpoint Target has logged into and then retrieves its properties information. After completion of retrieving the properties information, the Remote Control Server invokes the `start_gateway` method on the Endpoint Gateway remotely.
2. The Endpoint Gateway returns the result of the `start_gateway` method invocation to the Remote Control Server.
3. The Remote Control Server invokes the `start_target` method on the Endpoint Gateway remotely.
4. The Endpoint Gateway issues a downcall to invoke `pctlcf.exe` on the Endpoint Target.

5. The `pctlcf.exe` is executed on the Endpoint Target, and then communication between the Remote Control Target and Remote Control Gateway is established.
6. The Endpoint Target returns the result to the Endpoint Gateway.
7. The Endpoint Gateway also returns the result of the `start_target` method invocation to the Remote Control Server.
8. The Remote Control Server invokes the `start_controller` method on the Endpoint Gateway remotely.
9. The Endpoint Gateway issues a downcall to invoke `pctlcf.exe` on the Endpoint Controller.
10. The `pctlcf.exe` is executed on the Endpoint Controller, and then the communication between the Remote Control Controller and Remote Control Gateway is established.
11. The Endpoint Controller returns the result to the Endpoint Gateway.
12. The Endpoint Gateway also returns the result of the `start_controller` method invocation to the Remote Control Server.

Please refer to 9.3.4.2, “Remote Control Gateway” on page 294 for further discussion of where to use a Remote Control Gateway.

Note

To run Remote Control Gateway on an UNIX Managed Node, you must install the Patch 3.6-RCL-0002 on both the TMR server and the Managed Node.

9.2 Detecting Remote Control performance bottleneck

Remote Control consists of many components, such as Server or Controller. Therefore, many factors may affect its performance and throughput, and allocation of these components also can be the performance bottleneck. The following figure (Figure 119) shows the potential performance bottlenecks in a Remote Control environment.

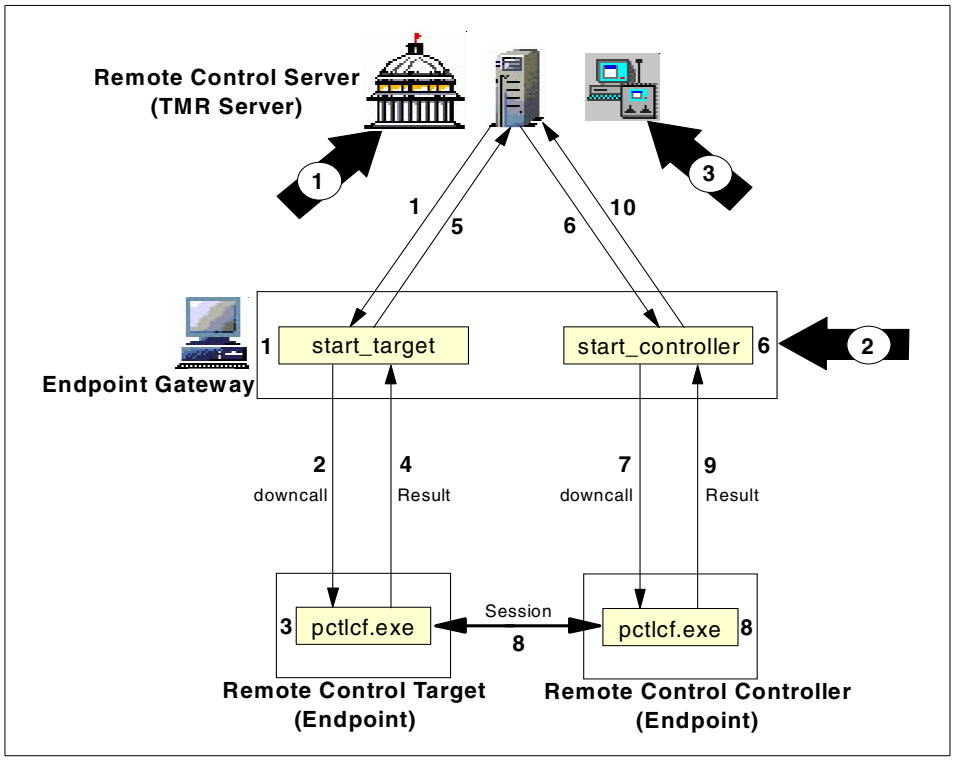


Figure 119. Remote Control performance bottleneck

1. Remote Control policy definitions
2. Remote Control components allocation
3. Remote Control operations

In this example process flow (Figure 119), we introduced typical performance bottlenecks of Remote Control. These bottlenecks can be a problem in most large management environments. The following are descriptions for each performance bottleneck in a Remote Control environment.

Policy Definitions

In a Remote Control environment, you can define some Remote Control GUI options or Remote Control session options by using the policy methods. To optimize your environment, you should define these policies carefully, since these policy definitions significantly affect Remote Control performance.

Remote Control Gateway When you configure the Remote Control Gateway in your environment, the operations that are handled by Remote Control will increase. It affects its performance as well. Therefore, the Remote Control Gateway should be configured only when your environment really needs to create the Remote Control Gateway, for example in the firewall environment.

Operations In a Remote Control management environment, your operations have a significant impact on performance. For example, when you take control of the target from the controller or if you move the target's mouse a little, this information is sent to the target and makes network traffic immediately. To improve performance, you should consider Remote Control operations.

In the following sections, we introduce how to configure and optimize a Remote Control environment.

9.3 Remote Control performance improving strategy

In the previous section, we introduced some major performance bottlenecks of Remote Control. This section introduces some approaches and directions to improve the performance and throughput of Remote Control. We also provide information about how you should configure and optimize Remote Control in a large-scale management environment.

9.3.1 Remote Control performance improving process flow

The following figure (Figure 120) shows the approach for improving the performance and throughput of Remote Control. When you configure or tune Remote Control in your management environment, we recommend you follow this process flow chart.

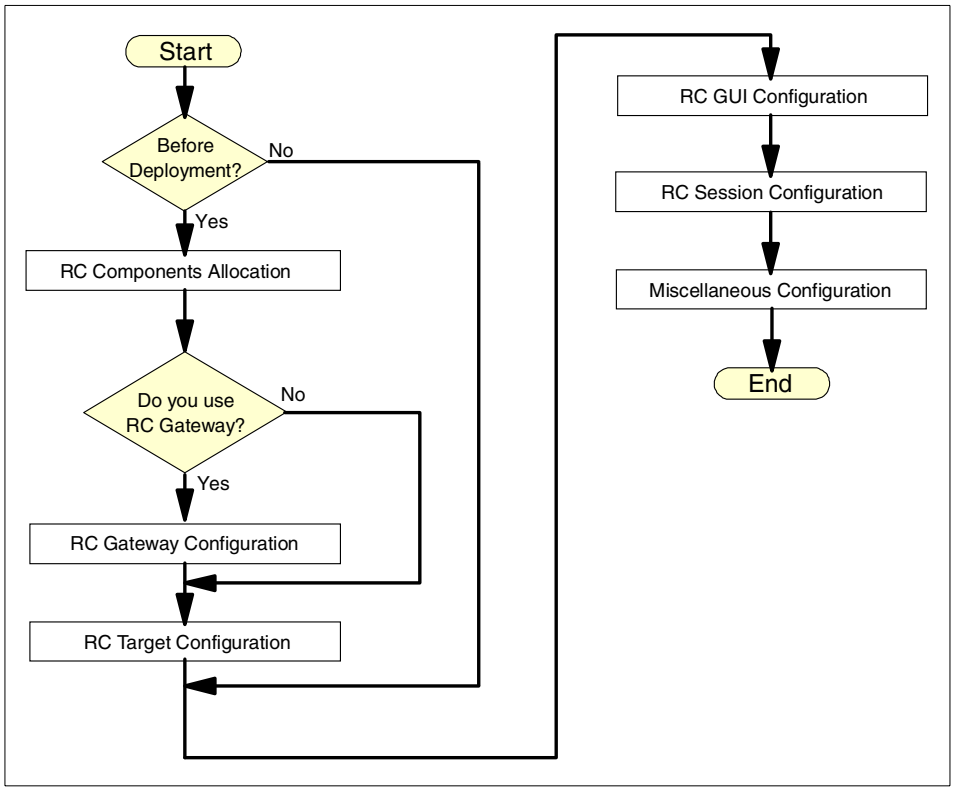


Figure 120. Remote Control performance improvement process flow

In the following sections, we will introduce detailed information about how to configure and tune your Remote Control in a large-scale management environment.

9.3.2 Remote Control policy definitions

Remote Control policy allows you to customize many options in your Remote Control environment. In other words, to optimize the Remote Control environment, you need to define the Remote Control policy carefully. In this section, we introduce which Remote Control policy definitions affect the performance, and how to configure Remote Control.

9.3.2.1 Improving Remote Control GUI performance

Remote Control provides some policy methods and values that are related to the Remote Control GUI. These policy methods and values also affect the performance of the Remote Control.

Patch 3.6-RCL-0002

Patch 3.6-RCL-0002 contains binaries for the Remote Control Server, Controller, and Target components. The following figure (Figure 121) shows the contents of Patch 3.6-RCL-0002 for a Managed Node and PC Managed Node.

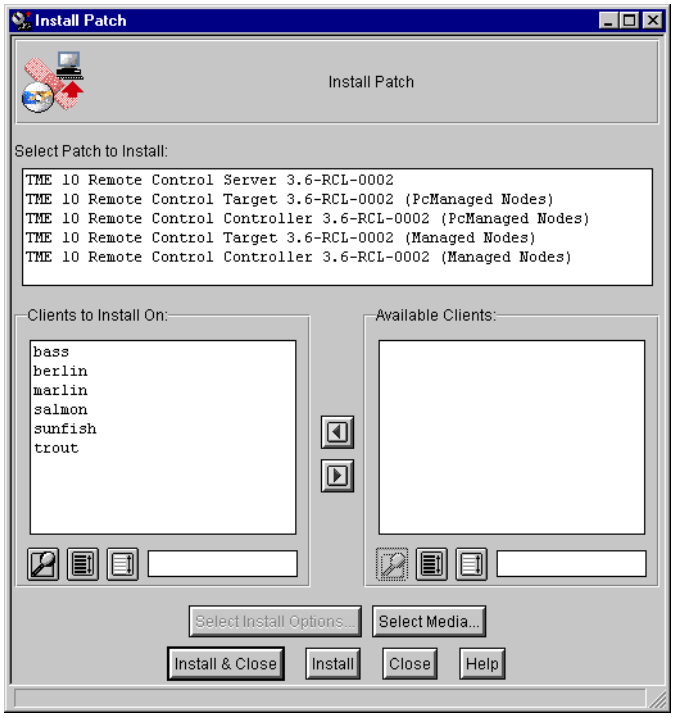


Figure 121. The contents of the Patch 3.6-RCL-0002

Patch 3.6-RCL-0002 adds new policy values and methods to improve Remote Control GUI performance. The following are new policy values and methods that are provided by Patch 3.6-RCL-0002:

- UncheckedList (New value for the rc_def_define policy)
- rc_def_uncheckedlist (Policy)
- rc_def_checkinterp (Policy)

In this part of the section, we introduce how to apply these new policy values and methods in your environment. After applying Patch 3.6-RCL-0002 in the Remote Control server, the rc_def_define policy can be set to one of the following values:

DefinableTargetList

The target list can be built by the user by using the rc_def_targets policy method. If you set the rc_def_define to this value, the GUI should look like the following figure (Figure 122). This is the default value.

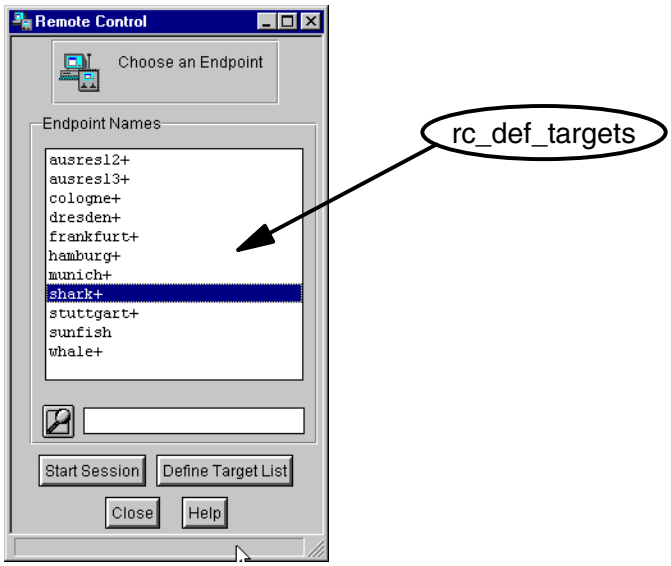


Figure 122. The first Remote Control dialog (The rc_def_define=DefinableList)

Note

Version 3.6.5 of Remote Control provides the new GUI. Please refer to the Figure 130 on page 298 for more information.

FilteredList

The target list cannot be built by the user. In this case, you can use the rc_def_filter_mode or rc_def_polfilter_mode policies to customize the GUI. If you set rc_def_define to this value, the GUI should look like the following figure (Figure 123).

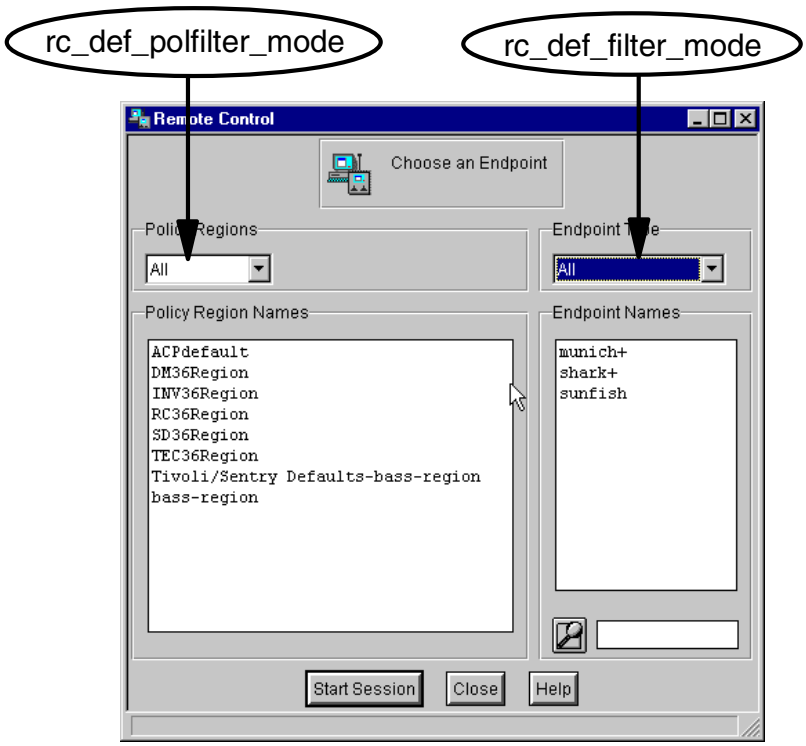


Figure 123. The first Remote Control dialog (The rc_def_define=FilteredList)

Note

Version 3.6.5 of Remote Control provides the new GUI. Please refer to the Figure 131 on page 299 for more information.

UncheckedList

The target list can be built by the user by using the rc_def_uncheckedlist policy method and the target list will be displayed without any validations. All validations will be performed only after the target has been chosen. If you set rc_def_define to this value, the GUI should look like the following figure (Figure 124).

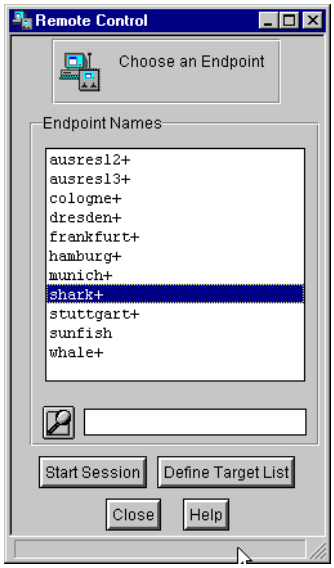


Figure 124. The first Remote Control dialog (The rc_def_define=UncheckedList)

If you set rc_def_define to UncheckedList, Remote Control does not invoke the `region_get_all` and interpreter methods. This will improve the response time that Remote Control displays the first Remote Control dialog. Therefore, we recommend you to set rc_def_define to the UncheckedList value.

If you manage thousands of targets from the central management site, it may be difficult to select the appropriate target from the target list that includes all targets in your management environment. If each target has a special naming convention in your environment, it may not be so difficult. However, if there is no naming convention, it will be very complicated to select the appropriate target from the target list, and you may need to use the GUI that is organized by the appropriate policies. In this case, the FilteredList value is useful (refer to the Figure 123).

The following policy method is also added by Patch 3.6-RCL-0002 to improve Remote Control GUI performance. The rc_def_checkinterp policy method works as follows:

rc_def_checkinterp This policy method optionally excludes the UNIX machine that is not supported as the Remote Control Target from the target list. To perform interp checking, set rc_def_checkinterp to Yes.

This policy is applicable when rc_def_define is set to DefinableTargetList or FilteredList.

Recommendable policy configurations

As introduced, Patch 3.6-RCL-0002 provides some new policies, and these affect Remote Control GUI performance. The following table (Table 11) shows the recommended configurations for these new policy values or methods.

Table 11. Recommendations to improve Remote Control GUI performance

Remote Control GUI	Recommendations
GUI simply shows the list of targets (**Case1**).	`rc_def_define=UncheckedList` `rc_def_uncheckedlist=<Input the list of your preferred targets>`

GUI shows two fields: The left field is for the policy region, and right field is for the its targets (**Case 2**).	`rc_def_define=FilteredList`
	`rc_def_polfilter_mode=region`
	`rc_def_checkinterp=No`

The following sections introduce examples of how to configure the policies.

Examples of policy configurations (Case 1)

To implement our recommended configurations for the GUI that simply shows the list of targets, we introduce the following examples. In this case, you must set rc_def_define to the UncheckedList value.

1. Create a new default policy object in the Remote Control class.

   ```
   wcrtpol -d RemoteControl RC_PDO_1 RemoteControl_PDO
   ```

2. Get the current rc_def_define values.

   ```
   wgetpolm -d RemoteControl RC_PDO_1 rc_def_define > rc_def_define.1
   ```

3. Edit the rc_def_define.1 file by using the appropriate editor as shown in the following example.

```
#!/bin/sh
#
#  Default policy method for Remote Control Define Target list
#
#  This method authorizes the user to define a new targets list.
#
#  Possible modes are:
#       DefinableTargetList   (can filter the list)
#       FilteredList          (cannot filter the list)
#       UncheckedList         (the list is displayed w/o any validation)
#
#       DefinableTargetList-locked      (the user cannot change the setting)
#

echo "UncheckedList"

exit 0
```

4. Set the rc_def_define=UncheckedList definition.

   ```
   wputpolm -d RemoteControl RC_PDO_1 rc_def_define < rc_def_define.1
   ```

   Then, we customize the rc_def_uncheckedlist policy method definition by performing the following steps.

   1. Get the current rc_def_uncheckedlist value.

      ```
      wgetpolm -d RemoteControl RC_PDO_1 rc_def_uncheckedlist >
      rc_def_uncheckedlist.1
      ```

      The default output of the rc_def_uncheckedlist.1 is shown in the following example.

```
!/bin/sh
# This method is to set a list of the possible RemoteControl targets
# Input values follow this naming convention :
# ManagedNode           echo "label"
# PCMangedNode:         echo "label"
# NetWare Clients      echo "NetWareClientName@NetWareManagedSiteName"
# Endpoint              echo "label+"
# Default value: wgetallinst ManagedNode wgetallinst PCMangedNode

wgetallinst ManagedNode
wgetallinst PCMangedNode

exit 0
```

   2. Edit the rc_def_uncheckedlist.1 file by using the appropriate editor to meet your requirements, otherwise, create the shell script to perform target selection. For example, if you need the list of all Endpoints in your TMR, you can use the following shell script shown in the following example.

```
##This script writes in rc_def_uncheckedlist.1
#echo "label+"
#for all the Endpoints installed in the TMR

echo "#!/bin/sh" > /tmp/rc_def_uncheckedlist.1
echo "# This method is to set a list of the possible RemoteControl targets" >> /tmp/rc_def_unchecked
wlookup -ar Endpoint | awk '{print $1}' | while read row
do
echo echo \"$row+\" >> /tmp/rc_def_uncheckedlist.1
done
echo "exit 0" >> /tmp/rc_def_uncheckedlist.1
```

You can easily modify this sample shell script to meet your environment.

   3. Set the rc_def_uncheckedlist value.

      ```
      wputpolm -d RemoteControl RC_PDO_1 rc_def_uncheckedlist <
      rc_def_uncheckedlist.1
      ```

Examples of policy configurations (Case 2)

To implement our recommended configurations for the GUI that shows two fields, we introduce the following examples (refer to theTable 11 on page 282). In this case, you must set rc_def_define to the FilteredList value.

1. Create the new default policy object in the Remote Control class.

   ```
   wcrtpol -d RemoteControl RC_PDO_2 RemoteControl_PDO
   ```

2. Get the current rc_def_define values.

   ```
   wgetpolm -d RemoteControl RC_PDO_2 rc_def_define > rc_def_define.2
   ```

3. Edit the rc_def_define.2 file by using appropriate editor as shown in the following example.

```
#!/bin/sh
#
#  Default policy method for Remote Control Define Target list
#
#  This method authorizes the user to define a new targets list.
#
#  Possible modes are:
#       DefinableTargetList   (can filter the list)
#       FilteredList          (cannot filter the list)
#       UncheckedList         (the list is displayed w/o any validation)
#
#       DefinableTargetList-locked      (the user cannot change the setting)
#

echo "FilteredList"

exit 0
```

4. Set the rc_def_define=FilteredList definition.

   ```
   wputpolm -d RemoteControl RC_PDO_2 rc_def_define < rc_def_define.2
   ```

   Then, customize the rc_def_polfilter_mode policy method definition by performing the following steps.

   1. Get the current `rc_def_polfilter_mode` value.

      ```
      wgetpolm -d RemoteControl RC_PDO_2 rc_def_polfilter_mode >
      rc_def_polfilter_mode.2
      ```

   2. Edit the rc_def_polfilter_mode.2 file by using the appropriate editor as shown in the following example.

```
#!/bin/sh
#
#  Default policy method for Remote Control Policy Region
#
#  This method prints the filtering mode for the list of possible targets.
#
#  Possible modes are:
#       region          (show according the policy chosen)
#       myregion                (show according the policy region where is the tool)
#       all             (show all)
#
#       XXX-locked      (show XXX only, and the user cannot change the setting)
#                       (e.g. "pcnode-locked")

echo "region"

exit 0
```

3. Set the rc_def_polfilter_mode=region definition.

```
wputpolm -d RemoteControl RC_PDO_2 rc_def_polfilter_mode <
rc_def_polfilter_mode.2
```

Then, we customize the rc_def_checkinterp policy method definition by performing the following steps.

4. Get the current rc_def_checkinterp values.

```
wgetpolm -d RemoteControl RC_PDO_2 rc_def_checkinterp >
rc_def_checkinterp.2
```

5. Edit the rc_def_checkinterp.2 file by using the appropriate editor as shown in the following example.

```
#!/bin/sh
#
#  Default policy method for Remote Control
#
#  This method allows to set another GUI Performance tuning :
#  you can choose if avoid or not from the Endpoint list the UNIX ManagedNode
#  Possible modes are:
#          Yes  the check on each ManagedNode interp is done
#          No   no check is executed
#

echo "No"
```

6. Set the rc_def_checkinterp=No definition.

```
wputpolm -d RemoteControl RC_PDO_2 rc_def_checkinterp <
rc_def_checkinterp.2
```

Note

To compare the response time between the different policy settings, you can use the `wrc` command. Patch 3.6-RCL-0002 enhanced the CLI support by supplying the `wrc` command:

```
wrc remote_control_resource_label action RCtarget RCcontroller [options]
```

where:

- The `action` specifies the requested action:

`monitor`	Monitor session
`controla`	Control session (initial state active)
`controlm`	Control session (initial state monitor)
`reboot`	Reboot target system

- The `RCtarget` and `RCcontroller` must be specified as follows:

  ```
  @(ManagedNode | PCManagedNode):managednode_label
  @NetwareManagedSiteClient:client_name@NetwareManagedSiteName
  @Endpoint:ep_label
  ```

- The `options` can be configured by using the following values:

`-B:(Y|N)`	Disable background
`-C:(Y|N)`	Limit to 16 colors
`-G:(0|5|10|15|30|60)`	Grace period
`-P:(Y|N)`	Proceed if timeout
`-R:(100|.....)`	Refresh rate
`-T:(Y|N)`	Enable change state on target
`-Z:(Y|N)`	Enable compression

The following is the example usage of the `wrc` command.

```
wrc RC@bass controla @Endpoint:shark @Endpoint:whale -G:0
```

9.3.2.2 Improving Remote Control session performance

Remote Control provides some policy methods to control Remote Control sessions. Understanding these policy methods is very important to improve the performance of Remote Control. In this part of the section, we introduce the policy methods for a Remote Control session and provide information about how you should define these policy methods in a large-scale management environment.

Understanding policy methods for a Remote Control session

Remote Control provides four policy methods to improve the performance of your Remote Control session.

rc_def_backgrnd	Defines whether to disable the desktop background of the target workstation when the session starts.
rc_def_color	Defines whether to limit to 16 colors the target screen displayed on the controller.
rc_def_comp	Defines whether to compress the data transmitted to the controller.
rc_def_rate	Defines the amount of time between the refreshes of the screen displayed on the controller.

The following table (Table 12) summarizes some guidelines that you should consider when you configure or customize your Remote Control environment.

Table 12. The Guidelines for Improving Remote Control Session Performance

Policy Method	Guidelines
rc_def_backgrnd	Disable the background always.
rc_def_color	**CPU Consideration:** The color reduction (16 colors) improves execution time only if the target machine is not CPU bound. The color reduction is very CPU intensive and should not be used if the system consumes more than 25 percent of the CPU without remote execution. **Target Platform:** Generally, it is recommended to use this option only on Windows 95 and Windows 3.x targets.
rc_def_comp	**Slow Link:** Enabling the compression of the transmitted data is effective if you work with slow link (lower than 57.6 Kb). **Fast Link:** Enabling the compression of the transmitted data does not improve the performance significantly if the connection is fast (LAN environment). **Target Platform:** Disabling the compression of the transmitted data is recommended with a Windows NT target.
rc_def_rate	**General Rule:** The slower connection needs the higher refresh rate. Start conservatively, around 300 ms, and increase by 100 ms intervals until you get the best response time. **Slow Link:** If the connection is slow, probably 500 ms will be your best choice. **Fast Link:** If the connection is fast, probably 100 ms will be your best choice.

You can set these attributes by using the GUI. The following figure (Figure 125) shows the GUI to set these attributes.

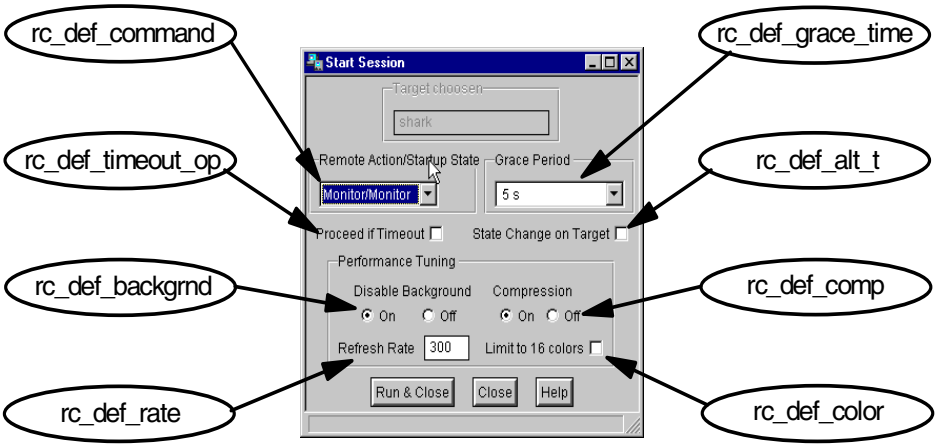


Figure 125. Configuring Remote Control session definitions

Note

Version 3.6.5 of Remote Control improved Remote Control session performance greatly. Version 3.6.5 or Remote Control provides a new panel for Remote Control session setting (it corresponds to the Figure 125). The new session setting panel allows you to configure the following options (refer to the Figure 129 on page 297).

- Grace Period
- Proceed if Timeout
- Startup State
- State Change on Target
- Desktop Optimization
- Reduce Number of Colors
- Refresh Rate
- Inactivity Timeout
- Compression

Please refer to the *Tivoli Remote Control User's Guide*, GC31-8437, for more information about Version 3.6.5 of Remote Control.

Understanding Remote Control session

As we mentioned, Remote Control provides some policy methods (options) that control Remote Control sessions. The following figure (Figure 126) shows the relationship between each policy method (tuning option) and Remote Control operations when the screen data is sent from the Remote Control Target to the Remote Control Controller.

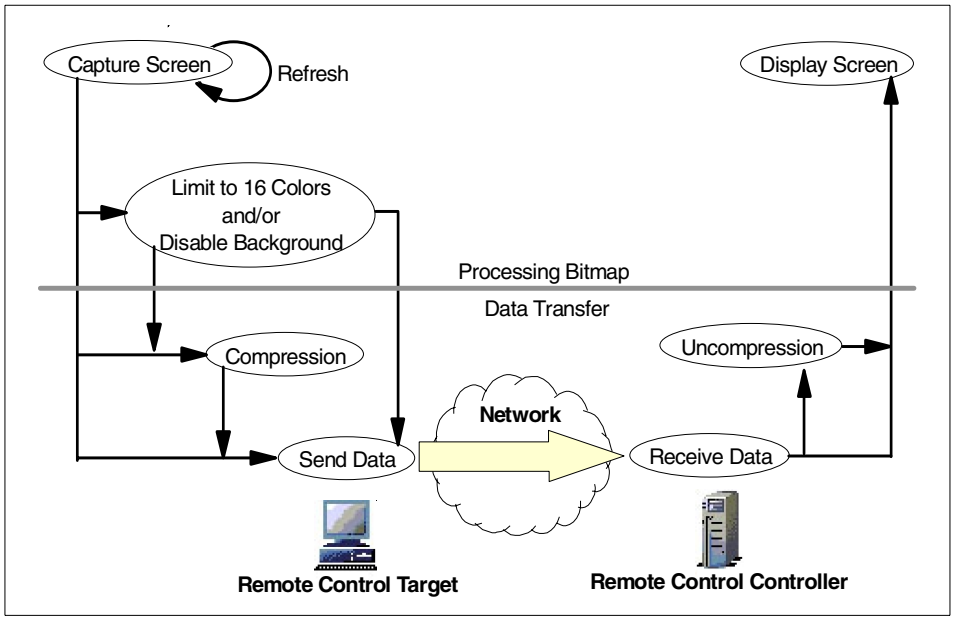


Figure 126. Remote Control Session and policy methods

As you can see, the load of the processes that are performed at the Remote Control Target before sending the screen data to the Remote Control Controller will increase if you specify many tuning options. The next section introduces how each tuning option affects the Remote Control performance.

Tuning options and Remote Control session performance

In the Remote Control session configurations, the following options are configurable:

- No Option
- Disable Background
- Limit to 16 Colors
- Compression
- Refresh Rate

- Combination of the Above Options

The following list introduces when you specify each tuning option.

No Option This case means there is no filter in the Remote Control session. Therefore, a lot of data will be sent from the Remote Control Target to the Remote Control Controller. This means the maximum amount of data is transmitted.

Disable Background In this case, actually, the amount of data that is sent from the Target to the Controller will be almost the same as when you specify the no option. The elapsed time may be a little improved. However, this option will not improve the Remote Control session performance dynamically.

Limit to 16 Colors In this case, the amount of data that is sent from the Target to the Controller will be almost half of the no option case. The elapsed time will also be improved by specifying this option.

Compression In this case, the amount of data that is sent from the Target to the Controller will be reduced greatly. Therefore, it reduces the network traffic. However, the elapsed time will not be improved dynamically because the compression process will need many resources and take time.

Refresh Rate In this case, the amount of data that is sent from the Target to the Controller will be almost the same as the no option case.

All Options This case means that the Disable Background, Limit to 16 Colors, and Compression options are specified at once. In this case, the amount of data that is sent from the Target to the Controller will be reduced dynamically. The elapsed time will be improved as well.

As you can see, when you set up the Remote Control environment, these tuning options should be configured properly. The next section introduces the considerations when you set up these tuning options and considerations of the Remote Control operations.

9.3.3 Remote Control operations considerations

When you customize these tuning options for the Remote Control session, there are some considerations. These considerations affect the performance of Remote Control.

9.3.3.1 Reducing available color on each target machine

This is not a Remote Control option, however, reducing available color on each Remote Control Target is very effective tuning to improve the Remote Control performance. We strongly recommend you set the Color Palette field to 256 colors. This should be done to improve the Remote Control session performance.

If the Remote Control Target machine is the Windows machine, select the menu, **Start** -> **Settings** -> **Control Panel** -> **Display** -> **Settings**, then you can configure the Color Palette on the following panel as shown in the Figure 127.

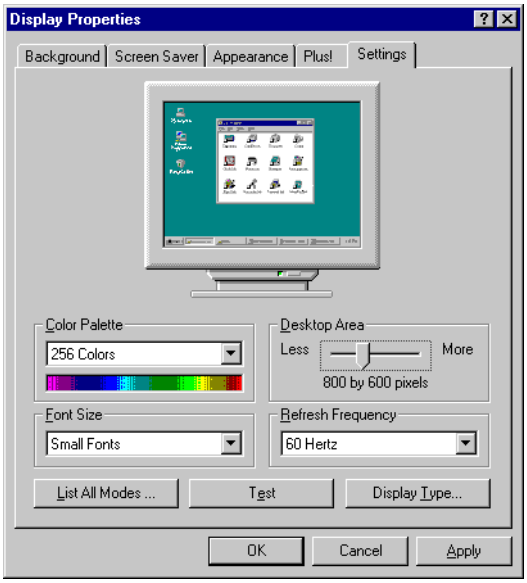


Figure 127. Configuring display properties of Windows machine

9.3.3.2 Setting up tuning options

In this part of the section, we classify the information about setting the Remote Control tuning options and introduce them. To set up each tuning option, there are some considerations which are detailed below. When you

configure your Remote Control environment, you should consider them carefully.

- Reduce the available colors at each Remote Control Target machine if you can.
- Disable background always.
- If your network is fast, and the Remote Control Target machine's CPU is slow, the compression option may affect adversely the Remote Control performance.
- Limit to 16 colors option needs a lot of resources, especially CPU.
- Increase the refresh rate if your network is slow.

The following table (Table 13) shows the summary of setting the Remote Control tuning options.

Table 13. Remote Control tuning options

Network Speed	**Fast Network**		**Slow Link Network**	
Configurations and CPU Speed	Fast CPU Target	Low Speed CPU Target	Fast CPU Target	Low Speed CPU Target
Setting Target to 256 Colors	Very Good	Very Good	Very Good	Very Good
Disable Background	Good	Good	Good	Good
Limit to 16 Colors	Good	None	Very Good	Good
Compression	Good	None	Very Good	Good
Increasing Refresh Rate	None	None	Good	Good
16 Colors and Compression	Very Good	Good	Very Good	Good

Very Good means it can improve the performance, Good means it can improve performance a little (it depends on your environment), and None means it may not improve the performance.

9.3.3.3 Operations considerations

Your Remote Control operations may affect the Remote Control performance. When you use Remote Control, you should remember that Remote Control will send the data in the following cases:

- Blinking cursor

- Scroll operation
- Moving mouse in active mode session
- Opening, closing, or resizing window

Especially, the operations to open a window, close a window, or resize a window create a lot of network traffic, and may adversely affect the Remote Control session performance.

When you use the application, such as the command prompt or notepad, on the Remote Control Target from the Remote Control Controller, you should not use the blinking cursor.

When you use Remote Control, you also should not perform useless operations. These operations make network traffic and may adversely affect the Remote Control session performance. To reduce mouse operations, we strongly recommend you to customize your Remote Control environment and perform Remote Control operations as follows:

- The programs that are executed frequently should be defined as the shortcut on the desktop.
- Make a batch program that includes processes frequently used.
- Do not move the windows.
- Do not resize the windows.
- Maximize the command prompt window if you perform only command line operations.

In summary, operations in a Remote Control environment have a significant impact on performance, so the suggestions listed above will greatly improve performance.

9.3.4 Remote Control resource allocation

The following are some considerations when you allocate the Remote Control resources in your management environment.

9.3.4.1 Remote Control Controller

In “Remote Control internals” on page 270, we introduced that the Remote Control server invokes the `start_controller` method remotely and spawns the controller process on the Remote Control Controller (refer to theFigure 117 on page 271).

If the Remote Control Controller is configured on the Endpoint, the following will be happened:

- The `start_controller` method is invoked remotely on the Endpoint Gateway that the Controller Endpoint has logged into by the Remote Control server.
- The Endpoint Gateway issues a downcall to execute the controller process (`pctlcf.exe`). If it is the initial execution, the Endpoint downloads this executable into its method cache.

If you configure the Remote Control Controller on the Endpoint Gateway, the downcall operations do not occur. This does not mean that all your Remote Control Controller must be Managed Nodes. However, if you have a Endpoint Gateway that has enough resources (CPU, memory, and so on), you can consider that you should configure the Remote Control Controller on the Endpoint Gateway, and it may improve the Remote Control performance.

Again, in Version 3.6 of Tivoli Management environment, the load of the Endpoint Gateway is heavy. Therefore, you should consider resources when you configure the Remote Control Controller on the Endpoint Gateway. For example, it depends on the following factors:

- How many Endpoints are logging into the Endpoint Gateway
- Endpoint Gateway resources (CPU, memory, and so on)

9.3.4.2 Remote Control Gateway

Remote Control provides the Remote Control Gateway to handle the communication between the Controller and Targets. When the Remote Control Gateway is configured in your environment, there is no direct communication between the Controller and Targets. All the transmitted data go through the Remote Control Gateway.

The Remote Control Gateway is used to minimize port exposure across your firewall, or used for protocol switch. Hence, security conscious environments may wish to implement the Remote Control Gateway. However, be aware that implementing the Remote Control Gateway will dramatically increase the response times from the Remote Control product. Therefore, it is recommended that the firewall itself is configured to resolve this issue. Some application proxies (Checkpoint FW-1, IBM SNG, Eagle Raptory, and so on) that are provided by the firewall vendors can be used to minimize the security threat without affecting performance.

If you are not configuring a firewall in your management environment, the Remote Control Gateway is not recommended. All operations of Remote Control can be performed without the Remote Control Gateway. From the

Remote Control performance point of view, there is no advantage in the Remote Control Gateway environment.

Therefore, we do not recommend to configure the Remote Control Gateway in your environment unless you have a firewall.

9.3.4.3 Remote Control resource

The Remote Control managed resource can be created on a Managed Node other than the TMR server that the Remote Control Server component is installed on (refer to the Figure 128).

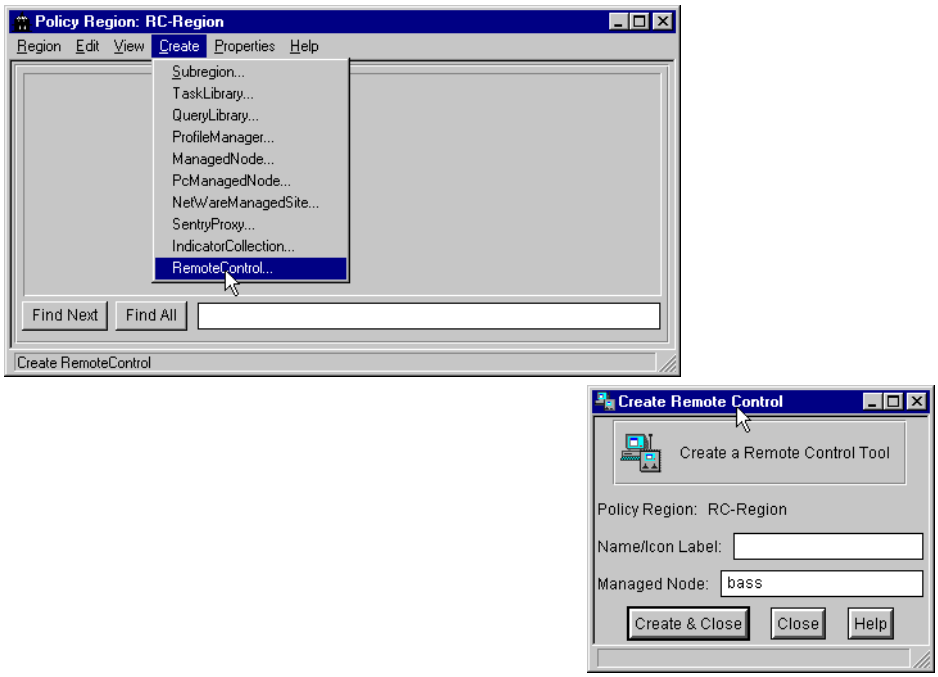


Figure 128. Creating a Remote Control resource

In some environments, it could be useful to create the resource on a Managed Node other than the TMR server.

Table 14. Creating Remote Control resource on managed node

Performance	Description
Advantage	The time that the Remote Control Server takes to connect to the selected target can be decreased.

Performance	Description
Disadvantage	The time to load the target list can be increased when you separate the TMR Server and Remote Control server component.s

To resolve this performance disadvantage, you can perform the following:

- Create one Remote Control resource for each LAN environment in a different policy region.
- Allocate each Remote Control resource on a Managed Node that is attached to the LAN environment.
- Customize the Remote Control policy definition in order to retrieve only the target list that includes targets in the LAN environment.

In each case, you have to test in your environment and figure out which solution gives you the best response time.

9.4 Version 3.6.5 of Remote Control performance improvement

Performance improvement have been the key requirement in Remote Control operations. In Version 3.6.5 of Remote Control, there are two performance aspects that need to be discussed when implementing Remote Control.

- Target Selection Performance
- Screen Refresh Performance

With the term target selection performance we refer to the time necessary to start a remote control session. Ideally the help desk analyst, once called by an end-user, needs to be able to very quickly (normally during the telephone call) connect to the end-user's machine to resolve or troubleshoot the end-user's problem.

With the term screen refresh performance we refer to the time required to replay graphical-screen-updates of the remote workstation (target) on the local administrative machine (controller).

Version 3.6.5 of Remote Control improves these performance greatly and can be an answer to the above requirements. Target selection performance has been addressed by a web interface that has been released as a technology preview (the code is located in \PREVIEW directory on the Version 3.6.5 of Remote Control CD-ROM), and screen refresh performance has been greatly improved as well.

9.4.1 Configuring Version 3.6.5 of Remote Control

Version 3.6.5 of Remote Control provides new panels for Remote Control session settings. The following figure (Figure 129) shows the Remote Control session definition screen that is provided by Version 3.6.5.

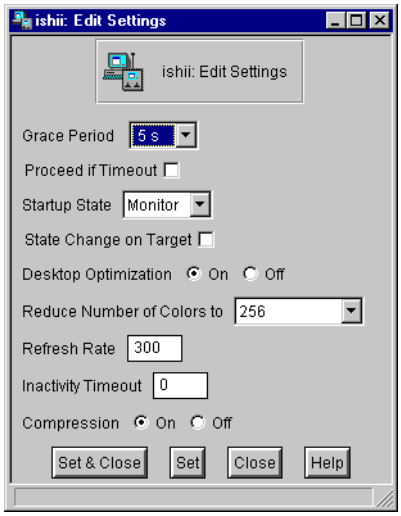


Figure 129. Configuring Remote Control session definition in Version 3.6.5

As you can see, the new GUI allows you to specify more detailed parameters than the previous version of Remote Control. The following figure shows the first Remote Control dialog.

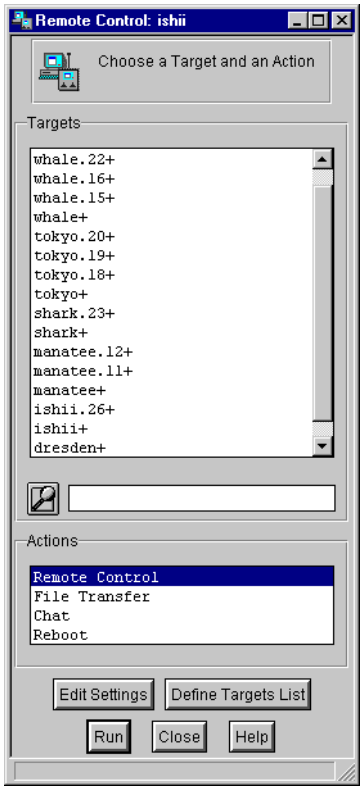


Figure 130. The first Remote Control dialog in Version 3.6.5

As you can see, the action field is added and you can choose an action that you would like to perform. The File Transfer and Chat are new functions that are provided by Version 3.6.5 of Remote Control. We will introduce the overview of these new functions in 9.4.4, “New features” on page 302.

The next figure (Figure 131) shows the Define Target List screen.

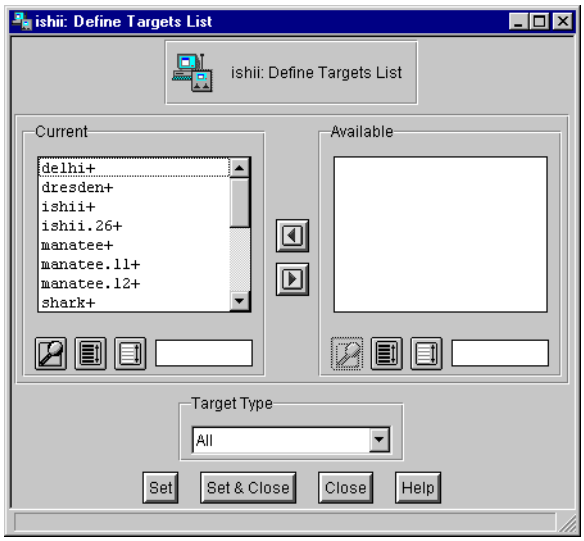


Figure 131. The Define Target List screen

These new GUIs allow you to perform more detailed setting and efficient operations.

9.4.2 Target selection performance

In large environments, when an help desk analyst has to support more than 5,000 users from a central location, you may run into some scalability problems if you are not careful during the Tivoli Management Framework and Remote Control implementation.

In these large-scale environment you normally have many centralized analysts or administrators that connect to the central Tivoli Server. Since Tivoli Enterprise products and Remote control are designed for centralized management, administrators must connect to the only machine (Central Tivoli Server) that has visibility of all the enterprise machines. The central Tivoli Server is normally already overloaded by other activities like software distribution and application monitoring and your objective is to minimize resources of this vital machine.

Here an example of the 300 Remote Control administrators and 32,000 nodes centralized service desk implemented at a customer as follows (refer to the Figure 132).

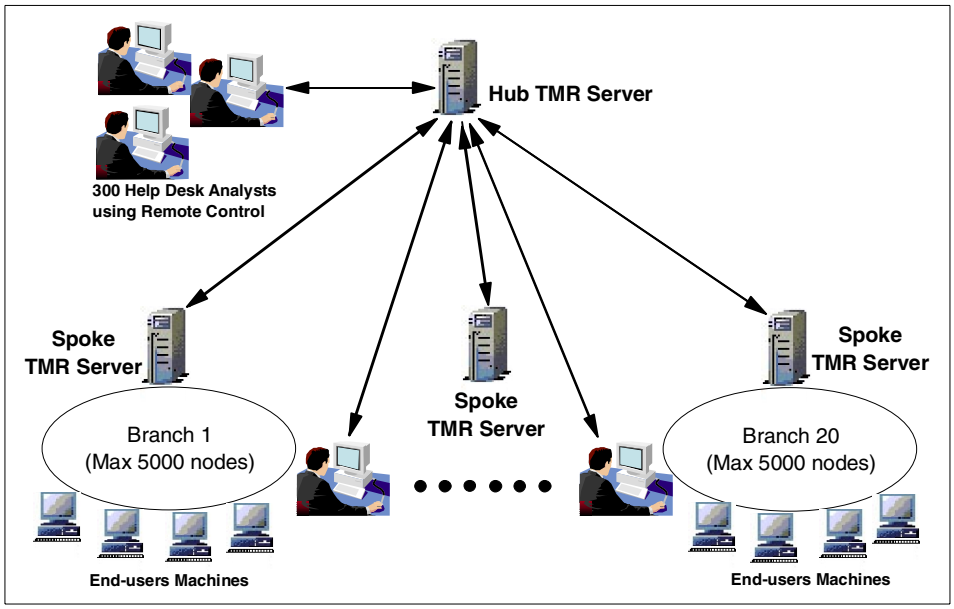


Figure 132. An Example of Remote Control environment

This example has a single (Hub) TMR server that has visibility of the 32,000 nodes and 20 secondary (Spoke) TMR servers that are interconnect to Hub and may see up to 5,000 nodes.

The 300 help desk analysts need to be able to connect to all the 32,000 nodes, so they all have to login to the Hub TMR and start the Remote Control session from that machine.

The problem to solve is how to size the Hub server and to minimize resources used on the server. In addition in several cases using the Tivoli Desktop is not the more natural way to start a Remote Control session. For example some of these administrators prefer launching Remote Control directly from the Tivoli Service Desk GUI and not using two different user interfaces.

Whenever you open a Tivoli Desktop to connect to the Hub server you will use about 7 KB of Hub server's memory and a dedicated TCP/IP session. In the case where you have 100 simultaneous administrators, in idle mode, you end-up already using 700 KB of memory and 100 TCP/IP sessions just for keeping these unused desktops open. So the Hub server's memory is critical. You need to have at least 1 GB of memory. Also CPU resources are never enough.

Strategies to off-load Hub server at the customer has been:

- Use static target lists
- Launch Remote Control from web-browser
- Launch Remote Control from Tivoli Service Desk

9.4.2.1 Using static target lists

As we explained in the previous section, loading a huge (greater than 10 KB) target list using dynamic lists (for example, querying in real time the machines with certain properties currently defined on the central server) uses server resources and may not be efficient.

One of solutions is to generate every day a set of static lists (for example, run a script that queries the Hub database during the night and produces lists of machines with certain common hardware and software information). These lists can be loaded quickly by the Tivoli Desktop. To define quickly load-able target lists use and set policy method as follows.

```
rc_def_define=UncheckedList
```

9.4.2.2 Launching Remote Control from Web browser

Tivoli Framework server provides a secure HTTP demon. Applications like User Administration have default secure (for example, can only be accessed by a Tivoli administrator) pages.

One of solutions is to add a secure Remote Control page to satisfy their help-desk analysts that prefer to use a web-browser interface to start Remote Control.

The web-browse interface uses less CPU and does not use memory and network resources in idle mode on the Hub server. In addition it helps productivity since a web-browser interface may be integrated with other browser based user interfaces. This implementation has been provided as a technology preview in V 3.6.5 of Remote Control.

9.4.2.3 Launching Remote Control from Tivoli Service Desk

Launching Remote Control from Tivoli Service Desk provides seamless management ability. Technically is always accomplished though the `wrc` command line that is started on the Hub server.

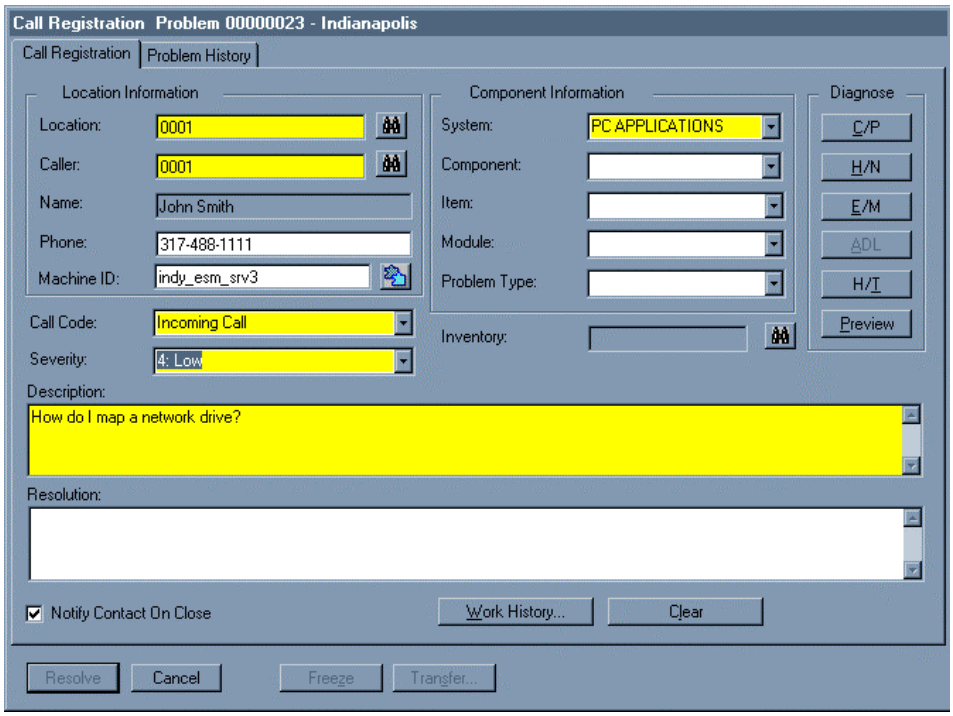


Figure 133. Tivoli Service desk

9.4.3 Screen refresh performance

After releasing Version 3.6 of Remote Control it was still necessary to improve performance on lines with bandwidth lower than 64 KB bps. For example on slow WAN environments the product is not fast as we would like. It is also hard to use it on dial-up environments for remote access (although we know that some customers are using it for remote access).

Version 3.6.5 of Remote Control meets these objectives in this area (for example, remote access) when using Windows 95 and Windows NT targets. Screen refresh performance has been greatly improved in Version 3.6.5 of Remote Control.

9.4.4 New features

Version 3.6.5 of Remote Control provides some new features. This section introduces the overview of these new features.

File transfer feature
The file transfer tool allows an administrator at the Remote Control Controller workstation to browse the file system of a remote Target workstation and to perform on it the following operations:

- Transfer files or directories from and to the remote file system.
- Make directories on local and remote file systems.
- Delete directories or files on both local and remote file systems.

The transfer of files is managed by the administrator and performed between Controller and Target Endpoints.

Chat feature
The chat facility enables written communication between the administrator at the Controller workstation and the remote user at the Target machine. The chat graphical user interface is newly provided.

Enhanced file protection using an external product
Trusted File Commander Plus for Tivoli V2.03 is an optional software product from Pinnacle Technology, Inc. It allows you to disable access to selected files, directories or disk drives. If installed, Trusted File Commander Plus V2.03 integrates with Remote Control and file transfer features.

The rc_def_tfc policy method returns one or more configuration filenames that are passed to the Remote Control or file transfer Target. The configuration files specify which files cannot be accessed on the Target. Note that during the Remote Control session, file protection is enabled only when the state is active.

Please refer to the *Tivoli Remote Control User's Guide*, GC31-8437, for more information about the new features of Remote Control Version 3.6.5.

9.5 Conclusion of Remote Control performance

Remote Control belongs to downcall-oriented applications. Remote Control consists of many components, and they work interactively. Remote Control provides many tunable options, and they greatly affect Remote Control performance. In this chapter, we discussed the following:

- Remote Control behavior
- Remote Control policy definitions
- Remote Control operations
- Remote Control components allocation

As we mentioned, Remote Control performance depends on not only tuning options but also operations. To improve Remote Control performance, we strongly recommend you follow the information that this chapter provides.

Appendix A. Special notices

This publication is intended to help system administrators, technical users and customers of Tivoli Enterprise products to understand more about the Tivoli Enterprise performance tuning. The information in this publication is not intended as the specification of any programming interfaces that are provided by Tivoli Enterprise application suite. See the PUBLICATIONS section of the IBM Programming Announcement for Tivoli Management Framework and core applications for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

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, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

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

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate

them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

AIX	AS/400
AT	IBM ®
NetView	OS/2
OS/390	RISC System/6000
Tivoli	Tivoli Enterprise
Tivoli Ready	TME

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

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

countries.

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

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

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Appendix B. Related publications

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

B.1 IBM Redbooks publications

For information on ordering these publications see “How to get IBM Redbooks” on page 311.

- *All About Tivoli Management Agents*, SG24-5134
- *Tivoli Enterprise Internals and Problem Determination*, SG24-2034
- *An Introduction to Tivoli Enterprise*, SG24-5494

B.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

B.3 Other resources

These publications are also relevant as further information sources:

- *TME 10 Framework 3.6 Planning and Installation Guide*, SC31-8432
- *TME 10 3.6 Reference Manual*, SC31-8434
- *TME 10 TEC Reference Guide 3.6*, SC31-8505

- *TME 10 Inventory 3.6 User's Guide*, GC31-8381
- *Tivoli Remote Control User's Guide*, GC31-8437
- *Accelerating AIX*, ISBN 0-2016-3382-5

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

AIX. Advanced Interactive Executive.

BARC. Before After Remove Commit Script.

CLI. Command Line Interface.

DB2. Database 2.

DLL. Dynamic Link Library.

DNS. Domain Name Service.

FTP. File Transfer Protocol.

GUI. Graphical User Interface.

IBM. International Business Machines Corporation.

ITSO. International Technical Support Organization.

LCF. Lightweight Client Framework.

MDist. Multiplexed Distribution.

MIF. Management Information File.

RDBMS. Relational Database Management System.

RIM. RDBMS Interface Module.

TAP. Tivoli Authentication Package.

TCP/IP. Transmission Control Protocol/Internet Protocol.

TEC. Tivoli Enterprise Console.

TMA. Tivoli Management Agent.

TME. Tivoli Management Environment.

TMF. Tivoli Management Framework.

TMR. Tivoli Management Region.

TNR. Tivoli Name Registry.

Index

E

F

N

O

P

Q

R

S

T

U

V

W

X

Z

IBM Redbooks evaluation

Tivoli Enterprise Performance Tuning Guide
SG24-5392-00

Your feedback is very important to help us maintain the quality of IBM Redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

_____

_____

_____

_____

What other Redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

SG24-5392-00
Printed in the U.S.A.

Tivoli Enterprise Performance Tuning Guide

SG24-5392-00